Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 7 № 4

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2017

Тематика номера:
Математические модели в физике, математическое моделирование, численные методы и визуализация, информационные технологии, суперкомпьютерные вычисления, математические исследования.

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, вопросы численного анализа, история науки и техники

The topic of the issue:
Mathematical models in physics, mathematical modeling, numerical methods and visualization, information technologies, supercomputer calculations, mathematical research.

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, problems of numerical analysis, history of science and of technique

Заведующий редакцией: Ю.Н.Штейников

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

СОДЕРЖАНИЕ

IV. СУПЕРКОМПЬЮТЕРНЫЕ ВЫЧИСЛЕНИЯ

V. МАТЕМАТИЧЕСКИЕ ИССЛЕДОВАНИЯ И ВОПРОСЫ ЧИСЛЕННОГО АНАЛИЗА

Сложные системы, методы статистической физики и вычисление свободной энергии

Б.В. Крыжановский [1], Л.Б. Литинский [2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] *kryzhanov@mail.ru,* [2] *litin@mail.ru*

Аннотация: Опираясь на методы статистической физики излагается подход к описанию большой системы взаимодействующих бинарных объектов, объясняется значение свободной энергии. Описан метод n-окрестностей, претендующий на роль универсального метода вычисления свободной энергии. В общем виде получено уравнение состояния, включающее внешнее магнитное поле. Приближение среднего поля является частным случаем метода n-окрестностей. Уравнение состояния решено для модели Изинга на гиперкубической решетке. Получено неизвестное ранее аналитическое выражение для критической температуры, установлены границы применимости метода n-окрестностей для модели Изинга.

Ключевые слова: свободная энергия, метод n-окрестностей, модель Изинга, приближение среднего поля.

1. Ведение

Методы статистической физики находят применение не только в физике, но и в задачах комбинаторной оптимизации [1], в теории нейронных сетей [2] и машинного обучения [3], в компьютерной обработке изображений [4]. Одна из центральных проблем здесь - вычисление свободной энергии, поскольку через нее выражаются многие макро-характеристики системы - намагниченность, теплоемкость, восприимчивость и т.д. Нами разрабатывается метод n-окрестностей вычисления свободной энергии. Этот подход не зависит от специфики матрицы связи и позволяет включить в рассмотрение внешнее магнитное поле. Применение метода n-окрестностей к модели Изинга на D-мерной кубической решетке дало простое аналитическое выражение для критических значений обратной температуры. Эта формула хорошо описывает компьютерный эксперимент для целого ряда размерностей: D = 3, 4, 5, 6 и 7. Поясним сказанное подробнее.

Будем изучать систему, состоящую из большого числа N взаимодействующих *спинов* s_i: $s_i = \{\pm 1\}$, $i = 1,2,...,N$, $N >> 1$. Взаимодействия между спинами задаются симметричной матрицей связи $\mathbf{T} = (T_{ij})$, самодействие отсутствует: $T_{ii} = 0$. Состояние системы как целого описывается конфигурационным вектором $\mathbf{s} = (s_1,...,s_N)$. Если H - однородное магнитное поле, то энергия состояния в расчете на один спин равна:

$$E(\mathbf{s},H) = -\frac{1}{2N}\sum_{i,j=1}^{N} T_{ij}s_i s_j - \frac{H}{N}\sum_{i=1}^{N} s_i \ . \qquad (1)$$

Эффективный подход к изучению таких систем доставляет статистическая физика. Центральная роль при этом принадлежит вычислению статистической суммы:

$$Z_N(\beta,H) = \sum_{\text{all }\mathbf{s}} \exp\left[-\beta E(\mathbf{s},H)\right],$$

где $\beta = 1/T$ - обратная температура, а суммирование ведется по всем 2^N состояниям $\mathbf{s}$. Статистическая сумма есть функция обратной температуры и внешнего поля. Поскольку значение $Z_N(\beta,H)$ даже при $\beta = 0$ равно 2^N (что при $N >> 1$ есть очень большое число), а интересуются *термодинамическим пределом* $Z(\beta,H) = \lim_{N\to\infty} Z_N(\beta,H)$, удобнее работать со *свободной энергией*:

$$F(\beta,H) = -\frac{\ln Z(\beta,H)}{\beta N}.$$

Вслед за большинством исследователей мы будем использовать *безразмерную свободную энергию*

$$f(\beta,H) = \beta \cdot F(\beta,H). \qquad (2)$$

Функции $F(\beta,H)$ и $f(\beta,H)$ при $N \to \infty$ принимают конечные значения (за исключением особых точек).

Роль свободной энергии определяется тем, что многие макро-характеристики системы являются ее производными [5].

Например, среднее значение энергии при данной температуре – эту характеристику называют *внутренней энергией* U - есть

$$U(\beta,H)=\frac{\partial f(\beta,H)}{\partial\beta}, \qquad (3)$$

теплоемкость $C(\beta,H)=-\beta^2\frac{\partial^2 f(\beta,H)}{\partial\beta^2}$, намагниченность состояния вычисляется как

$$M(\beta,H)=-\frac{\partial F(\beta,H)}{\partial H}, \qquad (4)$$

а магнитная восприимчивость равна $\chi(\beta,H)=-\frac{\partial^2 F(\beta,H)}{\partial H^2}$. Чтобы выяснить зависимость этих макро-характеристик от внешних параметров задачи, необходимо уметь вычислять свободную энергию (2).

Важное место в статистической физике принадлежит фазовым переходам. Поясним, что под этим понимается. Из самых общих соображений свободная энергия является непрерывной функцией своих аргументов. Но производные свободной энергии могут иметь особенности - скачки и/или бесконечные разрывы. Пусть, например, первая производная свободной энергии при некотором значении β_c испытывает скачок. (Для этого достаточно, чтобы на графике функции $F(\beta,H)$ при $\beta=\beta_c$ имелся излом.) Тогда внутренняя энергия $U(\beta,H)$ в точке β_c меняется скачком – см. (3). Иными словами, для β, меньших β_c, бесконечно малому приращению β отвечает бесконечно малое изменение внутренней энергии $U(\beta)$. А в точке β_c бесконечно малое приращение аргумента приводит к конечному скачку $U(\beta)$. Ясно, при $\beta=\beta_c$ происходит какая-то перестройка спиновой системы: либо меняется тип решетки, в узлах которой сидят спины, либо меняется характер распределения спинов и тому подобное. Говорят, что при переходе через критическое значение β_c система переходит из одного фазового состояния в другое. Отыскание всех фазовых состояний, их классификация и описание условий существования составляет предмет изучения большой спиновой системы. На формальном языке речь идет об исследовании особенностей свободной энергии. Если особенность имеется у первой производной $F(\beta,H)$ (как в рассмотренном случае) – говорят о фазовом *переходе первого рода*; если особенность имеет вторая производная $F(\beta,H)$ фазовый переход называют переходом *второго рода*. Фазовые переходы более высокого рода не наблюдались.

Вычисление свободной энергии, описание ее зависимости от внешних параметров - β, H и др. - задача, которую удается решить точно лишь для нескольких типов матрицы $\mathbf{T}$ [1, 2, 5, 6]. Как правило, при этом использовалась специфика конкретной матрицы связи. Мы развиваем подход к вычислению свободной энергии, который не зависит от конкретного вида матрицы связи [7]–[14]. Еще одно достоинство нашего подхода – возможность учета магнитного поля H (что обычно составляет целую проблему). Кратко существо нашего подхода состоит в следующем.

Зафиксируем начальную конфигурацию (начальное состояние) $\mathbf{s}_0\in\mathbf{R}^N$, и определим ее *$n$-окрестность* Ω_n как множество состояний, отличающихся от $\mathbf{s}_0$ противоположными значениями n координат. Параметр n принимает значения в интервале от 0 до N: $n\in[0,N]$. Ясно, что Ω_n включает в себя C_N^n конфигураций. Распределение энергий состояний, принадлежащих Ω_n, как правило неизвестно, однако можно получить точные выражения для среднего значения E_n и дисперсии σ_n^2 этого распределения – см. формулы (П1) в Приложении. Эти моменты выражаются через характеристики матрицы связи и начальную конфигурацию $\mathbf{s}_0$. Далее, эмпирический факт состоит в том, что гистограмма распределения энергий состояний из Ω_n является унимодальной, и хорошо аппроксимируется гауссовой плотностью вероятности с параметрами E_n и σ_n^2 (детальное обсуждение можно найти в [8]). Тогда в выражении для статистической суммы

$$Z_N=\sum_{n=0}^{N}\sum_{\mathbf{s}\in\Omega_n}\exp\bigl(-\beta E(\mathbf{s})\bigr)$$

суммирование по состояниям из Ω_n можно заменить интегрированием по гауссовой мере, а суммирование по n-окрестностям – интегрированием по непрерывной переменной $x=n/N$: $x\in[0,1]$. После преобразований статистическая сумма принимает вид двойного интеграла от экспоненты, в показателе которой стоит большой множитель N:

$$Z_N \approx \int dx \int dE \exp\left[-N \cdot \varphi(x,E)\right].$$

Такие интегралы вычисляются стандартным образом методом перевала (см. следующий раздел).

Функция в экспоненте $\varphi(x,E)$ зависит как от параметров от обратной температуры β, внешнего магнитного поля H (если имеется) и характеристик матрицы связи.

В общем виде выражение для $\varphi(x,E)$ получено в [8], [9].

Чтобы определить границы применимости метода n-окрестностей, мы исследовали с его помощью модель Изинга на D-мерной кубической решетке. За почти 100 лет изучения модели Изинга получено много точных результатов, на которых можно тестировать новые методы и подходы.

Основные полученные нами результаты таковы.

1. В предположении, что взаимодействия вдоль различных направлений решетки может быть различным, получено уравнение состояния, которое естественным образом обобщает известное уравнение Брэгга-Вильямса для модели среднего поля [5].

2. Для решеток больших размерностей $D > 3$ удается воспроизвести все известные для модели Изинга результаты: наличие фазового перехода второго рода и классические значения критических показателей [5]. Для критического значения обратной температуры получена неизвестная ранее аналитическая формула $\beta_c = \left(1-\sqrt{1-2/D}\right)/2$,хорошо описывающая результаты компьютерного эксперимента для всех размерностей $3 \le D \le 7$, для которых моделирование проводилось [15]-[18].

3. Для двумерной модели Изинга ($D = 2$) наш метод хоть и дает приемлемое критическое значение обратной температуры, но предсказывает скачок намагниченности в критической точке, что является признаком фазового перехода первого рода. Это противоречит известному точному решению Онсагера, которым предсказывается фазовый переход второго рода [5]. Для одномерной модели Изинга ($D = 1$) наш подход неприменим.

2. Основные соотношения и D-мерная модель Изинга

Поначалу все рассмотрения проводятся для матрицы связи $\mathbf{T} = \left(T_{ij}\right)_1^N$ общего вида, а затем переходим к D-мерной модели Изинга с взаимодействием между ближайшими спинами и периодическими граничными условиями [5].

Метод n-окрестностей основан на том, что распределение энергий состояний из n-окрестности начальной конфигурации аппроксимируется гауссовой плотностью.

Детальное обсуждение этого вопроса можно найти в [8]. Там же приводятся точные выражения для среднего значения энергии E_n и дисперсии σ_n^2.

Для удобства мы помещаем эти выражения в Приложении 1 - см. формулы (П1).

В качестве начальной конфигурации возьмем $\mathbf{s}_0 = (1,1,...,1) \in \mathbf{R}^N$. Ее n-окрестность Ω_n состоит из конфигураций, у которых точно n спинов равны «-1»: $\Omega_n = \left\{\mathbf{s} : \sum_{i=1}^N s_i = N - 2n\right\}$, $n = 0,1,...,N$. Через $E(\mathbf{s})$ обозначим энергию конфигурации $\mathbf{s}$ без учета внешнего поля, а через E_0 - энергию начальной конфигурации $\mathbf{s}_0$:

$$E(\mathbf{s}) = -\frac{1}{2N}\sum_{i \ne j}^{N} T_{ij} s_i s_j, \qquad E_0 = E(\mathbf{s}_0) = -\frac{1}{2N}\sum_{i \ne j}^{N} T_{ij}. \tag{5}$$

Если H - однородное внешнее поле, то полная энергия состояния $\mathbf{s} \in \Omega_n$ есть

$$E(\mathbf{s},H) = E(\mathbf{s}) - H\left(1 - \frac{2n}{N}\right).$$

В [8,13,14] показано, что асимптотическое выражение для статистической суммы приводится к виду:

$$Z_N \sim \int_0^1 dx \int_{E_x(\min)}^{E_x(\max)} \exp\left[-N \cdot \varphi(x,E)\right] dE, \tag{6}$$

где $N >> 1$, а функция в показателе экспоненты равна:

$$\varphi(x,E) = S(x) + \beta\left[E - H\cdot(1-2x)\right] + \frac{1}{2N}\left(\frac{E-E_x}{\sigma_x}\right)^2$$

Здесь $x = n/N \in [0,1]$,
$S(x) = x\ln x + (1-x)\ln(1-x)$, а $E_x = \lim\limits_{N\to\infty} E_n$ и

$\sigma_x^2 = \lim\limits_{N\to\infty}\sigma_n^2$ суть асимптотические выражения для среднего E_n и дисперсии σ_n^2 (П1):

$$E_x = E_0(1-2x)^2, \tag{7}$$

$$\sigma_x^2 = \frac{\mathbf{Tr}\,\mathbf{T}^2}{N^2}2x(1-x)\times\Big[4x(1-x)(1-\varepsilon_0^2) + (1-2x)^2 2N(d_0-\varepsilon_0^2)\Big], \tag{8}$$

где $\mathbf{Tr}\,\mathbf{T}^2 = \sum\limits_{i,j=1}^{N} T_{ij}^2$ есть след матрицы $\mathbf{T}^2$,

$\varepsilon_0^2 = \dfrac{4E_0^2}{\mathbf{Tr}\,\mathbf{T}^2}$, $d_0 = \dfrac{\|\mathbf{T}\mathbf{s}_0\|^2}{N\cdot\mathbf{Tr}\,\mathbf{T}^2}$.

Для D-мерной модели Изинга выражение для дисперсии сильно упрощается, так как в этом случае $d_0 \equiv \varepsilon_0^2 = \dfrac{q}{N} \to 0$, и тогда имеем:

$$\sigma_x^2 = \frac{8\cdot\mathbf{Tr}\,\mathbf{T}^2}{N^2}x^2(1-x)^2.$$

Интеграл в выражении (6) вычисляется методом перевала, для чего необходимо минимизировать функцию $\varphi(x,E)$ по переменным x и E для каждой пары значений внешних параметров β и H. После некоторых преобразований задача сводится к минимизации функции одной переменной

$$f(x) = S(x) + \beta E_x - \frac{\beta^2 N\sigma_x^2}{2} - \beta H(1-2x) \tag{9}$$

при условии $\beta N\sigma_x^2 + E_0 - E_x \le 0$. Введем новые обозначения, дающие более удобную нормировку:

$$b = \beta\frac{\mathbf{Tr}\,\mathbf{T}^2}{\sum_{ij}T_{ij}},\ q = \frac{\left(\sum_{ij}T_{ij}\right)^2}{N\cdot\mathbf{Tr}\,\mathbf{T}^2},\ h = \frac{\sum_{ij}T_{ij}}{\mathbf{Tr}\,\mathbf{T}^2}H, \tag{10}$$

и перейдем к D-мерной модели Изинга. Окончательно задача принимает вид: для каждого значения безразмерной обратной температуры b необходимо отыскать глобальный минимум функции

$$f(x) = S(x) - \frac{qb}{2}\Big[(1-2x)^2 + 8b(x(1-x))^2\Big] - bh(1-2x) \tag{11}$$

интервале $[0,x_b]$, где $x_b = 1/2$, когда $b \le 1$, и $x_b = \left(1-\sqrt{1-1/b}\right)/2$, когда $b \ge 1$. Тогда свободная энергия $f(b,h)$ (2) есть:

$$f(b,h) = \min_{x\in(0,x_b)} f(x,b,h). \tag{12}$$

Сделаем несколько замечаний. Для D-мерной модели Изинга начальная конфигурация $\mathbf{s}_0 = (1,1,...,1)$ является глобальным минимумом по энергии (*основным состоянием*). Такой выбор $\mathbf{s}_0$ позволяет указать точные границы интегрирования по энергии $E_x(\min)$ и $E_x(\max)$ в выражении (6) для Z_N. В простейшем случае, когда взаимодействия между ближайшими спинами одинаковы: $T_{ij} = (1-\delta_{ij})J$, выражения (10) заметно упрощаются:

$$\sum_{ij}T_{ij} = 2DNJ,\ \sum_{ij}T_{ij}^2 = 2DNJ^2 \Rightarrow$$
$$\Rightarrow\ b = \beta J,\ q = 2D,\ h = H/J.$$

Большую роль в дальнейшем играет параметр q, который в данном простейшем случае равен числу ближайших соседей у каждого спина: $q = 2D$.

Если взаимодействия вдоль различных направлений D-мерной решетки различны, значение параметра q не обязательно будет целым числом.

Например, для $2D$ модели Изинга с различными константами взаимодействия по вертикали (J) и горизонтали (K) имеем: $q = 2\left(1 + \dfrac{2}{K/J + J/K}\right)$. Такое q может равняться любому числу из интервала $2 < q < 4$.

Аналогично, для $3D$ модели Изинга с тремя различными константами взаимодействия $q = 2\dfrac{(J+K+L)^2}{J^2+K^2+L^2} \in (2,6)$. В общем случае действительное число q есть эффективное координационное число, характеризующее взаимодействие спина с его ближайшим окружением. Наконец, положим $H = 0$ и будем решать задачу в отсутствие внешнего поля.

3. Уравнение состояния

1) Уравнение состояния.

Для каждого значения b надо минимизировать на интервале $[0, x_b]$ функцию

$$f(x) = L(x) - \frac{qb}{2}\left[(1-2x)^2 + 8b(x(1-x))^2\right],$$

для чего необходимо решить уравнение

$$\frac{\partial f}{\partial x} = \ln\frac{x}{1-x} + \\ +2qb(1-2x)\left[1-4bx(1-x)\right] = 0 \quad . \quad (13)$$

Очевидно, что $x_0 = 1/2$ всегда является решением уравнения (13). Назовем решение x_0 *тривиальным*. Чтобы найти остальные решения, избавимся от тривиального решения, преобразовав уравнение (13) к виду:

$$\frac{\ln\frac{1-x}{x}}{2(1-2x)} = qb\left[1-4bx(1-x)\right]. \quad (14)$$

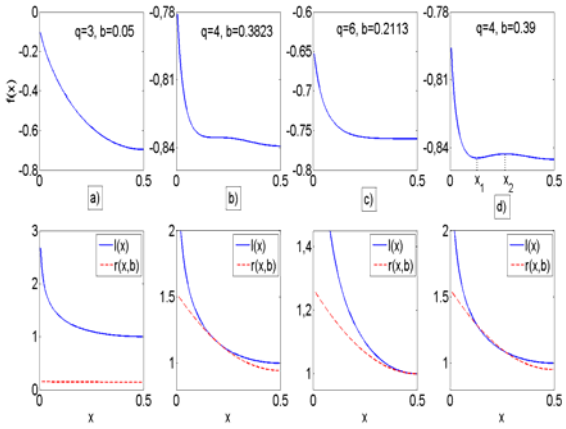
Уравнение (14) называется уравнением состояния. Обозначим через $l(x)$ его левую часть, а через $r(x,b)$ - правую:

$$l(x) = \frac{\ln\frac{1-x}{x}}{2(1-2x)}, \quad r(x,b) = qb\left[1-4bx(1-x)\right].$$

Функции $l(x)$ и $r(x,b)$ ведут себя симметрично относительно точки $x_0 = 1/2$, поэтому будем изучать их поведение на интервале [0, 1/2].

Функция $l(x)$ монотонно убывает от $+\infty$ на левом краю интервала до значения, равного 1 на правом краю: $l(x_0) = 1$. Ее первая и вторая производные на правом краю равны $l'(x_0) = 0$ и $l''(x_0) = 8/3$ соответственно. Квадратный трехчлен $r(x,b)$ монотонно убывает от значения $r(0) = qb$ на левом краю до значения $r(x_0) = qb(1-b)$ на правом – см. нижние панели на рис.1. По мере возрастания параметра b у функции $f(x)$ будут возникать точки перегиба и/или локальные экстремумы – см. графики на верхних панелях рис. 1. Эти трансформации удобно изучать, анализируя взаимное расположение кривых $l(x)$ и $r(x,b)$.

Пока значение b невелико ($b \sim 0$), кривые $l(x)$ и $r(x,b)$ не имеют общих точек – см. нижнюю панель рис.1a. Для таких b единственным решением уравнения состояния остается тривиальное решение $x_0 = 1/2$, а свободная энергия (12) равна:



$$f(b) = f(x_0, b) = -\ln 2 - \frac{qb^2}{4}, \quad b << 1.$$

Рис. 1. Для различных значений q и b графики функции $f(x)$ (верхние панели) и кривые $l(x)$ и $r(x,b)$ (нижние панели).

С увеличением значения параметра b кривая $r(x,b)$ как целое поднимается вверх, приближаясь к кривой $l(x)$. При некотором значении b кривые $l(x)$ и $r(x,b)$ коснутся друг друга. Касание кривых может произойти либо во внутренней точке интервала $(0, x_0)$ (нижняя панель рис.1b), либо на правом конце интервала, в точке $x_0 = 1/2$ (см. нижнюю панель рис. 1c).

Касание кривых $l(x)$ и $r(x,b)$ во внутренней точке интервала означает, что у функции $f(x)$ образовалась точка перегиба, в которой первая производная функции

$f(x)$ равна нулю – см. верхнюю панель рис.1b. Такая точка не является минимумом функции $f(x)$, так что данное решение уравнения (14) интереса не представляет. Однако стоит слегка увеличить значение параметра b, и вместо точки перегиба у функции $f(x)$ появятся два экстремума: локальный минимум x_1 и локальный максимум x_2 – ср. верхние панели рис. 1b и рис. 1d. Новый локальный минимум является нетривиальным решением уравнения (13), и его необходимо сравнивать по глубине с минимумом в точке $x_0 = 1/2$.

В работах [13,14] исчерпывающим образом проанализирована зависимость решения уравнения состояния от значения параметра q. Оказывается, имеются два критических значения $q_1 = 4\ln 2 \approx 2.77$ и $q_2 = 16/3$, и свойства решения уравнения (14) кардинально зависят от того, к какому из трех интервалов принадлежит значение q: $q < q_1$, $q_1 \le q < q_2$ или $q_2 \le q$. Опуская детали, опишем полученные результаты.

2) Решение уравнения для $q_2 \le q$.

В этом случае кривые $l(x)$ и $r(x,b)$ соприкасаются только на правом конце интервала, в точке $x_0 = 1/2$. Касание произойдет, когда параметр b станет равным критическому значению

$$b_c = \frac{1-\sqrt{1-4/q}}{2}. \qquad (15)$$

После касания минимум функции $f(x)$ покидает точку $x_0 = 1/2$ и переходит в бесконечно близкую к ней точку $x_\varepsilon(b) = x_0 - \varepsilon(b)$, а x_0 становится точкой максимума. Это перемещение глобального минимума сопровождается фазовым переходом второго рода. Дальнейшее увеличение параметра b сдвигает точку минимума x_ε к левому концу интервала, в направлении 0. При $b \to \infty$ точка глобального минимума $x_\varepsilon(b)$ асимптотически стремится к 0, никогда его не достигая. Так и должно происходить: спиновая система асимптотически стремится к основному состоянию, когда температура $T = 1/b$ стремится к 0.

Для модели Изинга больших размерностей $D = 3$, 4, 5, 6 и 7 точные критические значения обратной температуры неизвестны, их оценки получены в результате компьютерного моделирования [15]-[18]. На рис.2 сплошной линией показаны результаты компьютерного моделирования, штриховой линией – наша оценка (15). Точечной линией показана оценка $b_c = 1/2D$, получающаяся в приближении среднего поля (уравнение Брэгга-Вильямса [5]). Видно, что наша формула описывает результаты компьютерного эксперимента много лучше.

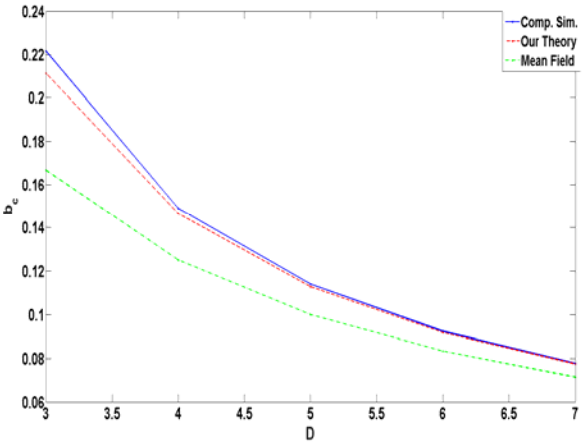


Рис.2. Зависимость b_c от размерности решетки D: компьютерное моделирование (сплошная линия), формула (15) (штриховая), приближение среднего поля (точечная).

С ростом размерности D согласие между теорией и компьютерным экспериментом улучшается. Напомним, что параметр q в выражении (15) может принимать любые значения, не только целочисленные – см. замечание в конце предыдущего раздела.

Для $q \ge 16/3$ можно получить значения критических показателей [5]. Для таких q критические показатели принимают классические значения, характерные для

модели среднего поля: $\alpha = 0$, $\beta = 1/2$, $\gamma = \gamma' = 1$, $\delta = 3$. Для $D > 3$ это совпадает с тем, что ранее было получено другими авторами. Для $D = 3$ общепринятыми являются неклассические значения критических показателей, приближенные значения которых получены методом ренорм-группы.

3) Решение уравнения для $q_1 \le q < q_2$.

В этом случае касание кривых $l(x)$ и $r(x,b)$ при некотором значении $b = b_t$ происходит не на конце интервала, а во внутренней точке $x_t \in (0,1/2)$. При дальнейшем увеличении параметра b, за значение b_t, у функции $f(x)$ появляется локальный минимум в точке $x_1(b)$ - см. выше. С увеличением b глубина нового минимума растет, а сама точка $x_1(b)$ сдвигается в направлении 0. При достижении некоторого критического значения $b = b_j$ глубина нового минимума станет равна глубине минимума в «тривиальной» точке $x_0 = 1/2$. В этот момент произойдет перескок глобального минимума из x_0 в точку x_j, отвечающую критическому значению b_j (“j” от jump=прыжок). Данный фазовый переход сопровождается скачкообразным изменением намагниченности $m = 1 - 2x$, и является фазовым переходом первого рода - см. (4).

В [13,14] получено трансцендентное уравнение, численное решение которого дает критическое значение b_j. Для $q = 4$ получаем $b_j \approx 0.3912$. В то же время, для 2D модели Изинга Онсагером получено точное решение, которое дает фазовый переход второго рода при $b_c \approx 0.4407$. Таким образом, для $q_1 \le q < q_2$ к результатам, полученным методом n-окрестностей надо относиться с осторожностью. С одной стороны, количественные результаты находятся на приемлемом уровне точности: различие между оценкой b_j и точным значением b_c составляет около 11%. На рис. 3 приведены графики свободной энергии $f(\beta)$, построенные для различных значений константы взаимодействия по горизонтали K при фиксированном значении взаимодействия по вертикали ($J = 1$). Сплошной линией показана $f_{Ons}(\beta)$, вычисленная по теории Онсагера, а кружочками - то, что получается в нашем подходе: $f_{n-vic}(\beta)$. Видно, что отличие наших результатов от точной теории становится заметным только в районе критического значения обратной температуры, а относительная ошибка находится на уровне 1%. С другой стороны, смущает скачок намагниченности в момент фазового перехода. Этот скачок не предсказывается никакими теориями. Применение метода n-окрестностей для таких значений q требует дополнительных исследований.

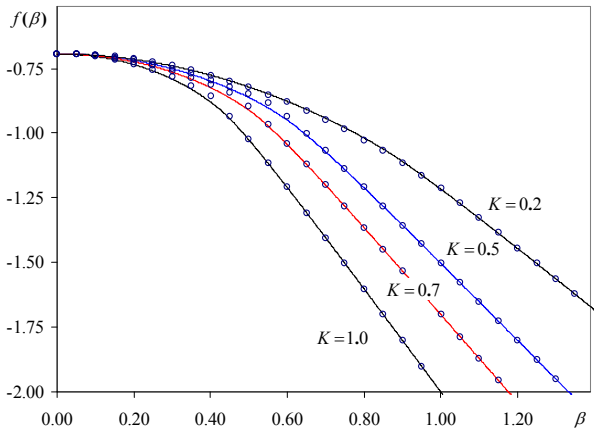


Рис. 3. Для 2D модели Изинга графики свободной энергии $f(\beta)$ при $K = 0.2, 0.5, 0.7, 1.0$: сплошные кривые – теория Онсагера, кружки – наша теория.

4) Наконец отметим, что для $q < q_1$ метод n-окрестностей неприменим вовсе. Это следует из результатов работы [19], посвященной вычислению спектральной плотности для модели Изинга. Доказательство данного утверждения отложим до следующей публикации [20]. Здесь лишь констатируем, что

невозможность использовать метод n-окрестностей в области $q < q_1$ является ограничением нашего метода.

4. Модель среднего поля

Применим метод n-окрестностей для анализа модели среднего поля [5], которая сыграла заметную роль в статистической физике. В рамках этой модели впервые удалось описать фазовый переход второго рода (Брэгг-Вильямс, 1934).

В модели среднего поля взаимодействие спина с ближайшим окружением заменяется величиной, усредненной по всем спинам. Если спин имеет q ближайших соседей, а константа взаимодействия равна J, то суммарное воздействие на i-й спин со стороны остальных спинов записывается как $h_i = \frac{qJ}{N-1}\sum_{j\neq i}^{N} s_j$, а энергия (5) в расчете на один спин есть

$$E(\mathbf{s}) = -\frac{\sum_{i=1}^{N} s_i h_i}{2N} = -\frac{(\mathbf{Ts},\mathbf{s})}{2N},$$

где $T_{ij} = (1-\delta_{ij})\frac{qJ}{N-1}$.

В качестве начальной конфигурации по-прежнему выберем $\mathbf{s}_0 = (1,1,...,1)$. Тогда $E_0 = E(\mathbf{s}_0) = -\frac{qJ}{2}$. Нетрудно видеть, что для любой конфигурации $\mathbf{s} \in \Omega_n$ выполняется равенство:

$$E(\mathbf{s}) = -\frac{qJ}{2}\cdot\frac{(N-2n)^2 - N}{N(N-1)} = E_0\cdot\frac{(N-2n)^2 - N}{N(N-1)}. \quad (16)$$

Сравнивая (16) с E_n из (П1), видим, что энергия каждого состояния из Ω_n равна среднему значению энергии E_n. Следовательно, дисперсия σ_n^2 будет равна 0 (в этом можно убедиться и прямым вычислением по формуле (8)). Тогда асимптотические значения этих моментов равны

$$E_x = E_0(1-2x)^2, \quad \sigma_x^2 \equiv 0. \quad (17)$$

Подставляя их в (9), получим выражение для функции, которую необходимо минимизировать:

$$f(x,H) = S(x) - q\frac{\beta J}{2}(1-2x)^2 - \beta H(1-2x).$$

Аналог уравнения состояния (14) принимает вид

$$\frac{1}{2}\ln\frac{1-x}{x} = qb(1-2x) + bh, \quad b = \beta J, \quad h = H/J.$$

Если вместо переменной x использовать намагниченность $m = 1-2x$, уравнение состояния приводится к классическому виду, в котором его обычно и анализируют [5]:

$$\frac{1}{2}\ln\frac{1+m}{1-m} = qbm + bh. \quad (18)$$

Наше уравнение состояния (14) в этих терминах и с учетом внешнего магнитного поля (10), (11) имеет вид:

$$\frac{1}{2}\ln\frac{1+m}{1-m} = qbm\left[1 - qb(1-m^2)\right] + bh. \quad (19)$$

Фактически, приближение среднего поля можно трактовать как такой вариант метода n-окрестностей, когда распределение энергий состояний из Ω_n аппроксимируется средним значением энергии по n-окрестности (16), а дисперсию распределения полагают равной 0 – см. (17). Иными словами, для аппроксимации истинной плотности распределения $P_n(E)$ используется δ-образное распределение с центром в E_x. Конечно, это очень грубая аппроксимация, и получающиеся результаты могут служить только качественными оценками - см. нижнюю кривую на рис. 2. В методе n-окрестностей мы, во-первых учитываем дисперсию распределения σ_x^2 (8), а, во-вторых, аппроксимируем $P_n(E)$ гауссовой плотностью. В результате, уравнение состояния (19) приобретает

дополнительный член по сравнению с выражением (18). Для модели Изинга больших размерностей это позволяет получить значения критических температур, которые хорошо описывают результаты компьютерного эксперимента.

5. Обсуждение и выводы

Основное приближение метода n-окрестностей состоит в том, что при переходе от суммирования по Ω_n к интегрированию по Ω_x истинную плотность распределения энергий $P_n(E)$ заменяют гауссовой плотностью с известными E_x и σ_x^2. Центральная часть $P_n(E)$ аппроксимируется при этом достаточно хорошо, но хвосты $P_n(E)$ всегда не гауссовы: для $D=2$ и $D=3$ они «толще» гауссовой плотности, для $D=1$ напротив, тоньше. Эти различия на хвостах и являются источником ошибки.

В работах [13, 14] показано, что с ростом координационного числа $q=2D$ величина ошибки должна уменьшаться - это мы и наблюдаем для модели Изинга. Заметим также, что использование для аппроксимации $P_n(E)$ гауссовой плотности на всем интервале значений n, верно передает какие-то существенные характеристики истинного распределения $P_n(E)$, и дает разумные результаты по крайней мере, для значений q выше границы отсечения: $q > 4\ln 2$.

Если говорить о физических приложениях метода n-окрестностей, отметим, что учет взаимодействия спинов со следующими соседями – или даже с более дальними соседями, - может привести к увеличению эффективного координационного числа q. Можно ожидать, что это улучшит согласие между результатами, полученными методом n-окрестностей и точным решением (либо компьютерным экспериментом).

Одной из областей применения нашего метода могут оказаться современные методики анализа сцен и компьютерной обработки изображений [4, 21-23]. Здесь происходит многократное перевычисление так называемой *нормировочной константы*, которая и представляет собой значение статистической суммы при фиксированных значениях внешних параметров. Затем значения внешних параметров меняются. Востребованным здесь оказывается быстрое – пусть даже и приближенное - вычисление статистической суммы, а критическая температура и характер фазового перехода никого не интересуют.

Перспективным для всех приложений является также включение внешнего магнитного поля. Наши выражения (9), (11) позволяют сделать это достаточно эффективно.

Работа выполнялась при финансовой поддержке Российского Фонда Фундаментальных Исследований (гранты №15-07-04861 и 16-01-00626).

Приложение

Приведем выражения для среднего E_n и дисперсии σ_n^2 распределения энергий состояний из n-окрестности Ω_n, впервые полученные в [24,25]. Если использовать нормировки как в (1) и положить $E_0 = E(\mathbf{s}_0)$, то :

$$E_n = E_0 \frac{(N-2n)^2 - N}{N(N-1)},$$

$$\sigma_n^2 = \frac{\mathbf{Tr}\,\mathbf{T}^2 \cdot 2n(N-n)/N^2}{N(N-1)(N-2)(N-3)} \times \qquad , \quad (\Pi 1)$$

$$\times \left[A\left(1 - \frac{4E_0^2 N}{(N-1)\mathbf{Tr}\,\mathbf{T}^2}\right) + B\frac{\|\mathbf{T}\mathbf{s}_0\|^2 - 4E_0^2 N}{\mathbf{Tr}\,\mathbf{T}^2}\right]$$

где $\mathbf{Tr}\,\mathbf{T}^2$ есть след матрицы $\mathbf{T}^2$, $A = 4(n-1)(N-n-1), \quad B = 2\left[(N-2n)^2 - N + 2\right]$. Формулы являются точными для конечных значений N и n, и годятся для любой матрицы $\mathbf{T}$ и произвольной начальной конфигурации $\mathbf{s}_0$.

Наконец, приведем выражения для среднего $E_n(\mathbf{H})$ и дисперсии $\sigma_n^2(\mathbf{H})$ распределения энергий для Ω_n, когда в системе присутствует неоднородное магнитное поле $\mathbf{H} = (H_1, H_2, ..., H_N)$:

$$E_n(\mathbf{H}) = E_n - \left(1 - \frac{2n}{N}\right) \cdot \frac{(\mathbf{s}_0, \mathbf{H})}{N},$$

$$\sigma_n^2(\mathbf{H}) = \sigma_n^2 + \frac{4n(N-n)}{N^3(N-1)} \left[\|\mathbf{H}\|^2 - \frac{(\mathbf{s}_0, \mathbf{H})^2}{N} + \right.$$

$$\left. + 4 \frac{N-2n}{N-2} \left(\frac{(\mathbf{T}\mathbf{s}_0, \mathbf{H})}{2} + \frac{E_0 \cdot (\mathbf{s}_0, \mathbf{H})}{N} \right) \right].$$

Впервые выражения для $E_n(\mathbf{H})$ и $\sigma_n^2(\mathbf{H})$ были получены в [7]. Формулы заметно упрощаются, если $\mathbf{H}$ коллинеарно начальной конфигурации $\mathbf{s}_0$: $\mathbf{H} = H\mathbf{s}_0$. В этом случае, $E_n(\mathbf{H}) = E_n - (1 - 2n/N)H$, $\sigma_n^2(\mathbf{H}) = \sigma_n^2$.

Complex systems, statistical physics methods and free energy calculation

B.V. Kryzhanovskiy and L.B. Litinskii

Abstract: Basing on the methods of statistical physics, we discuss an approach allowing us to analyze large systems of interacting binary objects. We propose a method of n-vicinities, which from our point of view is a universal method of calculation of the free energy. In the general form, we obtained the state equation in the presence of a magnetic field. The mean field approximation is a special case of the n-vicinity method. We solved the state equation for an Ising model on a hypercube lattice. With the aid of analytical methods, we obtained a previously unknown expression for the critical temperature and defined the boundaries of the application of our n-vicinity method for the Ising model.

Keywords: the free energy, the method of n-vicinities, the Ising model, mean field approximation.

Литература

[1] O.C. Martin, R. Monasson, R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems. «Theoretical Computer Science», v. 265 (2001), 3-67.
[2] D. Amit, H. Gutfreund and H. Sompolinsky. Statistical Mechanics of Neural Networks Near Saturation. «Annals of Physics», v.173 (1987), 30-67.
[3] G.E. Hinton, S. Osindero, Y. The. A fast learning algorithm for deep belief nets. «Neural Computation», v. 18 (2006), 1527-1554.
[4] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. «IEEE Trans. on Information Theory», v.51 (2005), 2313–2335.
[5] Р. Бэкстер. Точно решаемые модели статистической физики, М., Мир, 1985.
[6] В.С. Доценко. Универсальная случайность. «Успехи физических наук», т.181 (2011), 269–292.
[7] B. Kryzhanovsky, L. Litinskii. Approximate method of free energy calculation for spin system with arbitrary connection matrix. «Journal of Physics: Conference Series», v. 574 (2015), 012017.
[8] Boris Kryzhanovsky and Leonid Litinskii. Generalized approach to description of energy distribution of spin system. «Optical Memory & Neural Networks (Information Optics)», v.24 (2015), 165-185.
[9] Б.В. Крыжановский, Л.Б. Литинский. Обобщенное уравнение Брэгга-Вильямса для систем с произвольным дальнодействием. ДАН, т.459 (2014), 1-5.
[10] Б.В. Крыжановский, Л.Б. Литинский. Общий метод вычисления статистической суммы**.** «Нейроинформатика-2015. XVII Всероссийская научно-техническая конференция. - Москва, 21-25 сентября 2015 (труды)», М., изд-во МИФИ, 2015, т. 2, стр. 81-92.

[11] Б.В. Крыжановский, Л.Б. Литинский. Обобщенный подход к описанию распределения энергий спиновой системы. «Труды НИИСИ», т.5 (2015), №2, 5-21.
[12] Б.В.Крыжановский, Л.Б.Литинский. Приближенное вычисление статистической суммы для модели Изинга на гиперкубе. Труды НИИСИ РАН, т.6 (2016), №2, 5-10.
[13] B. Kryzhanovsky and L. Litinskii. Applicability of n-vicinity method for calculation of free energy of Ising model. «Physica A: Statistical mechanics and its applications», v. 468 (2017), 493–507.
[14] Б.В.Крыжановский, Л.Б.Литинский. Метод n-окрестностей для вычисления статистической суммы в модели Изинга: границы применимости. «Труды НИИСИ РАН», т.7 (2017), №1, 4-17.
[15] Blote H W J, Shchur L N, Talapov A L. The cluster processor: new results. 1999 «Int. J. Mod. Phys. C», v. 10 (1999), 1137.
[16] R.Häggkvist , A. Rosengren, P.H. Lundow , K. Markström, D. Andren, P. Kundrotas. On the Ising model for the simple cubic lattice. «Advances in Physics», v. 56 (2007), №5, 653-755.
[17] P.H. Lundow, K. Markstrom. The critical behaviour of the Ising model on the 4-dimensional lattice. «Phys. Rev. E», v.80 (2009), 031104. *Preprint* arXiv: 1202.3031v1.
[18] Lundow P H, Markstrom K The discontinuity of the specific heat for the 5D Ising model. «Nuclear Physics B» v.895 (2015), 305-318.
[19] B.V. Kryzhanovsky. The Spectral Density of a Spin System Calculated for Solvable Models, arXiv: 1704.01351 (2017).
[20] Б.В.Крыжановский, Л.Б.Литинский. Метод n-окрестностей и аппроксимация спектральной плотности (готовится к печати).
[21] C. Wang, N. Komodakis, N. Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. Preprint to Elsevier (2013).
[22] J. Liu, R. De Lamare. Low-latency reweighted belief propagation decoding for LDPC codes. «IEEE Communications Letters», 2012.
[23] A. Novikov, A. Rodomanov, A. Osokin, D. Vetrov. Putting MRFs on a Tensor Train. «Proc. of the 31st Intern. Conf. on Machine Learning, Beijing, China, 2014. Journal of Machine Learning Research: W&CP volume 32».
[24] B.V. Kryzhanovsky, V.M. Kryzhanovsky. «Lecture Notes in Electrical Engineering» v.24 (2009), 51-61.
[25] Ya.M. Karandashev, B.V. Kryzhanovsky. «Optical Memory & Neural Networks», v.19 (2010), 110.

Обобщение решения Онсагера для двумерной модели Изинга конечных размеров

М.Ю.Мальсагов [1], Я.М.Карандашев [2], Б.В.Крыжановский [3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1] malsagov@niisi.ras.ru, [2] karandashev@niisi.ras.ru, [3] kryzhanov@mail.ru

Аннотация: Численными методами исследована зависимость термодинамических характеристик двумерной модели Изинга от числа спинов N. Экспериментальные данные, полученные алгоритмом Кастелейна-Фишера для модели на квадратной решетке $N=L\times L$, сравниваются с асимптотическим решением Онсагера ($N\to\infty$). Получены эмпирические выражения, описывающие зависимость критических параметров от N. Произведено обобщение решения Онсагера на случай решетки конечных размеров и получены аналитические выражения для свободной энергии и её производных (внутренней энергии, дисперсии энергии, теплоёмкости), хорошо описывающие результаты численного эксперимента. Показано, что с ростом N теплоемкость в критической точке возрастает логарифмически. Указаны ограничения на точность определения критической температуры, обусловленные конечным размером системы.

Ключевые слова: статистическая сумма, нормировочная константа, модель Изинга.

1. Введение

Вычисление статистической суммы является центральной задачей статистической физики. К сожалению, решить эту задачу точно удалось только для небольшого ряда моделей, описанных в классических монографиях [1,2]. Наиболее известным является решение Онсагера для двумерной квадратной решетки [3], полученное много лет тому назад. Более сложные модели, в том числе и модели учитывающие наличие магнитного поля, до сих пор не имеют точного аналитического решения. Для их описания используются приближенные методы, такие как приближение Брэгга-Вильямса, приближение молекулярного поля Кюри-Вейсса, приближение Бете-Пайерлса [4,5] и т.д. Несмотря на сложность подхода, очень продуктивным для изучения свойств модели Изинга оказался метод трансфер-матрицы [6,7]. Кроме того, большой ряд аналитических и численных методов был использован для изучения модели с помощью конечно-цепочечных экстраполяций [8] (finite chain extrapolations), высокотемпературных рядов [9], двухвременных функций Грина [10], отображения Сузуки-Троттера [11] , методов Монте-Карло [12,13], приближения Бете [14], комбинаторного подхода [15], низко-температурных рядов с разложением по степенному многочлену [16,17], а также метода n-окрестностей [18,19]. Развитые статфизические подходы использованы для исследования свойств ассоциативной памяти [20]-[23] и разработки методов обучения нейронных сетей [24]-[26] конечных размеров.

В последнее время большой прогресс достигнут в численных методах, направленных на исследование критических показателей [27]-[30] и спектра энергий спиновых систем [31]. Основным алгоритмом в такого рода исследованиях является метод Монте-Карло, позволяющий делать приближенные оценки. Однако, появились алгоритмы, позволяющие точно вычислять свободную энергию конечномерной планарной спиновой решетки. Исследователи стараются проводить расчеты с возможно большим числом спинов N, чтобы наилучшим образом аппроксимировать физические результаты, соответствующие пределу $N\to\infty$. Однако возможности численного счета ограничены и всегда остается открытым вопрос «достаточно ли велика размерность задачи»?

Целью настоящей работы является исследование зависимости параметров системы от размерности задачи N и получение аналитических выражений, пригодных при конечных значениях N. Полученные ниже результаты позволяют оценить, насколько велика должна быть размерность задачи, чтобы результаты численных расчетов удовлетворительно описывали свойства физических моделей. Кроме того, наличие аналитических выражений позволит использовать их для глубокого обучения нейронных сетей конечного размера и алгоритмов обработки изображений.

В настоящей работе мы интенсивно использовали алгоритм Кастелейна-Фишера [32-33] для подсчёта свободной энергии спиновой системы на квадратной двумерной решётке. Этот алгоритм является точным,

поскольку вычисление статсуммы в нем сводится к вычислению детерминанта некоторой матрицы, строящейся в соответствии с рассматриваемой моделью. Алгоритм позволяет за полиномиальное время точно вычислять свободную энергию спиновой системы для произвольного планарного графа с произвольно заданными связями. Подробнее об алгоритме можно прочитать в работе [34]. В настоящей работе мы использовали имплементацию алгоритма, предложенную в [35], которая давала совпадающие с [34] результаты, но оказалась значительно быстрее.

2. Исходные выражения

Пусть задана система спинов, описываемая гамильтонианом:

$$E = -\frac{1}{2N}\sum_{i,j=1}^{N} J_{ij} s_i s_j \ , \ s_i = \pm 1 . \qquad (1)$$

Нас интересует свободная энергия системы:

$$f = -\frac{\ln Z}{N} , \qquad (2)$$

где статистическая сумма $Z = \sum_S e^{-N\beta E(S)}$ определена как сумма по всем возможным конфигурациям $S = (s_1, s_2, ..., s_N)$, а β - обратная температура. Знание свободной энергии позволяет вычислять основные измеряемые параметры системы:

$$U = \frac{\partial f}{\partial \beta}, \ \sigma_E^2 = -\frac{\partial^2 f}{\partial \beta^2}, \ C = -\beta^2 \frac{\partial^2 f}{\partial \beta^2}, \qquad (3)$$

где внутренняя энергия $U = \overline{E}$ есть среднее по ансамблю при заданном β, $\sigma_E^2 = \overline{E^2} - \overline{E}^2$ - дисперсия энергии, а $C = \beta^2 \sigma_E^2$ - теплоемкость.

Эксперимент мы проводили для планарной модели, в которой спины расположены на квадратной решетке, а взаимодействие имеется только с ближайшими четырьмя соседями. Ограничимся сначала случаем, когда взаимодействие с ближайшими соседями задается константой $J_{ij} = J$, причем для простоты выражений будем полагать $J = 1$. Обобщение на случай разных констант связей $J \neq J'$ вдоль разных осей решетки будет очевидным.

Решение Онсагера, полученное в пределе $N \to \infty$ для модели с периодическими граничными условиями, имеет вид:

$$f(\beta) = -\frac{\ln 2}{2} - \ln\left(\cosh 2\beta J\right) - \frac{1}{2\pi}\int_0^{\pi} \ln\left(1 + \sqrt{1 - k^2 \cos^2\theta}\right) d\theta \ , \qquad (4)$$

где

$$k = \frac{2\sinh 2\beta J}{\cosh^2 2\beta J} . \qquad (5)$$

Известно, что решение Онсагера описывает логарифмическую расходимость теплоёмкости при $\beta \to \beta_{ONS}$, где критическое значение температуры определяется из условия $k = 1$ в виде:

$$\beta_{ONS} = \frac{1}{2}\ln\left(1 + \sqrt{2}\right). \qquad (6)$$

В следующем разделе мы проведем сравнение численных результатов, полученных при помощи алгоритма Кастелейна и Фишера для решёток различных размеров, с аналитическим решением Онсагера. В частности, мы проведем анализ поведения теплоёмкости вблизи критической температуры $\beta_c = \beta_c(N)$, проанализируем изменения β_c с ростом размерности и приведем выражение, позволяющее естественным образом обобщить формулу Онсагера на случай конечных размерностей.

3. Условия эксперимента

Алгоритм Кастелейна-Фишера позволяет с машинной точностью вычислять величину свободной энергии. При помощи этого алгоритма нам удалось проанализировать поведение свободной энергии и её производных (внутренняя энергия U и теплоёмкость C) для ряда решёток различной размерности $N = L \times L$. Линейный размер решетки варьировался от $L = 25$ до $L = 10^3$. В принципе, работа алгоритма не ограничена только квадратными решётками, но в данной работе мы использовали только их.

Подчеркнем, что используемый нами алгоритм применим только к планарным решеткам. Это означает, что мы рассматривали только случай решеток со свободными граничными условиями, поскольку решетка с периодическими граничными условиями не является планарным графом. Соответственно, энергия основного состояния имеет вид:

$$E_0 = -2\left(1 - \frac{1}{\sqrt{N}}\right). \qquad (7)$$

В данной работе мы ограничились линейными размерами $L \le 10^3$. Значения температур в основном варьировались в пределах $\beta \in [0, 1]$, поскольку при $\beta > 1$ величина свободной энергии практически не отличается от асимптотического значения $f \approx \beta E_0$. Для каждого значения N на указанном отрезке $\beta \in [0, 1]$ с шагом

$d\beta = 10^{-4}$ вычислялись значений функции $f = f(\beta)$ и ее производных. Вне этого отрезка делалось всего лишь 10^2 контрольных точек. Для подсчёта первой и второй производных функции $f = f(\beta)$ в экспериментах мы вычисляли значения свободной энергии f в трёх точках, равноотстоящих друг от друга на расстоянии $d\beta = 10^{-5}$, и вычисляли производные методом конечных разностей. Хотя алгоритм позволяет считать величину f для решёток любой размерности, однако при больших линейных размерах ($L > 400$) точности вычислений (в формате чисел типа double) уже не хватало и вычисление вторых производных производилось только с точностью только до 4-го знака после запятой.

4. Экспериментальные результаты

В ходе эксперимента вычислялась свободная энергия $f = f(\beta)$ и ее производные. Как и ожидалось, пик кривой $C = C(\beta)$ сдвинут вправо от пика кривой $\sigma_E^2 = \sigma_E^2(\beta)$. По положению пика теплоемкости определялась критическая температура β_c и критические значения $f_c = f(\beta_c)$, $U_c = U(\beta_c)$ и $C_c = C(\beta_c)$. По положению пика дисперсии энергии определялась вторая критическая точка β_c^* и соответствующие критические значения $f_c^* = f(\beta_c^*)$, $U_c^* = U(\beta_c^*)$ и $C_c^* = C(\beta_c^*)$. Результаты эксперимента и анализа данных представлены ниже в графическом виде на рисунках 1-5. Значения величин β_c, f_c, U_c и C_c приведены в Таблице 1. Через косую черту в этой же таблице приведены значения величин β_c^*, f_c^* U_c^* и C_c^*.

Как видно из рис. 1, экспериментально полученные значения свободной энергии и внутренней энергии при увеличении размерности приближаются к решению Онсагера. Показаны кривые только для небольших линейных размеров $L = 25, 50, 100$. При $L > 100$ графики уже фактически сливаются с решением Онсагера и потому не отображены на рисунке.

Поскольку при больших значениях β значение статистической суммы определяется главным образом состояниями с наименьшей энергией, то свободная энергия при больших β начинает линейно убывать (см. рис.1). В соответствии с (7) асимптотическое поведение свободной энергии при больших β описывается выражением

$$f \approx -2\beta\left(1 - \frac{1}{\sqrt{N}}\right). \qquad (8)$$

Именно наличие члена $\sim 1/\sqrt{N}$ приводит к тому, что кривые для малых линейных размеров не сливаются с решением Онсагера.

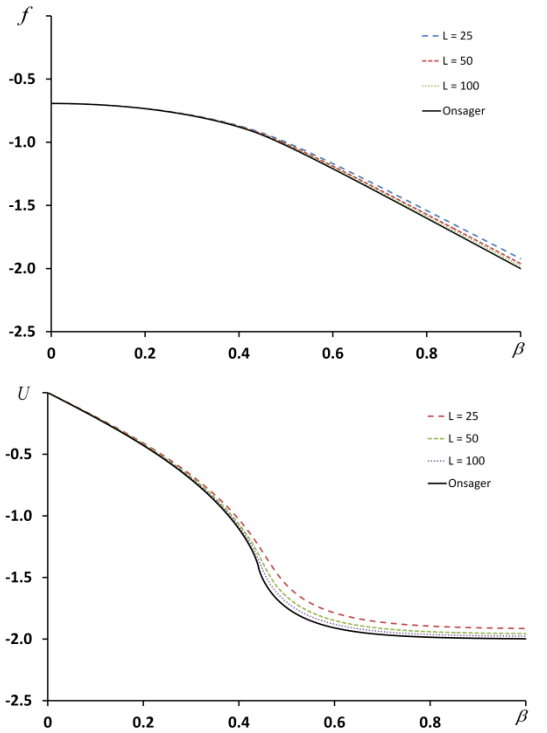


Рис. 1. Графики свободной энергии $f(\beta)$ (сверху) и внутренней энергии $U(\beta)$ (снизу) для небольших линейных размеров решётки и асимптотическое решение Онсагера ($L \to \infty$).

Отличие между решением Онсагера и поведением $f(\beta)$ для случая решётки конечного размера лучше всего наблюдать на графике второй производной (теплоёмкости C). Согласно решению Онсагера теплоёмкость имеет логарифмическую расходимость при $\beta \to \beta_{ONS}$. Естественно предположить, что для решётки конечных размеров такого не происходит, однако пик теплоёмкости $C = C(\beta)$ действительно наблюдается – тем острее, чем больше размерность решётки (см. рис. 2).

Более тщательный анализ вблизи пика теплоёмкости показал два результата. Во-первых, высота пика логарифмически зависит от размерности решётки. Во-вторых, положение пика немного сдвинуто вправо относительно β_{ONS}, но приближается к нему по мере роста размера решетки.

Анализ приведенных в таблице 1 результатов показывает, что положение пика

(критическое значение β_c), а также зависимость критических значений свободной энергии и теплоёмкости от размерности решетки хорошо аппроксимируются выражениями:

$$\beta_c = \beta_{ONS}\left(1+\frac{5}{4\sqrt{N}}\right). \qquad (9a)$$

$$U_c = -\sqrt{2}\cdot\left(1-\frac{1}{2\sqrt{N}}\right). \qquad (9b)$$

$$C_c = \frac{4\beta_c^2}{\pi}\left(\ln N - 1.7808\right). \qquad (9c)$$

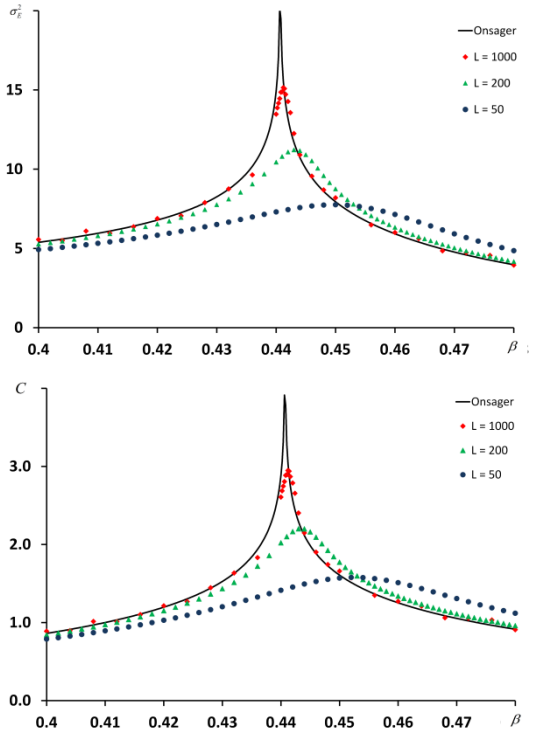


Рис. 2. Зависимость дисперсии σ_E^2 (сверху) и теплоёмкости C (снизу) от β для нескольких размеров решёток: от $N = 50\times 50$ до $N = 10^3\times 10^3$ (маркеры). Сплошной кривой показано решение Онсагера.

Относительная ошибка при аппроксимации зависимости $\beta_c = \beta_c(N)$ выражением (9a) достаточна мала: максимум ошибки ~0.3% имеет место при $L = 25$; с ростом L ошибка быстро спадает до значения 0.01% при $L = 10^3$. Качество апроксимационных формул оценивается величиной достоверности $R^2 = 1-\sum(x_{\exp}-x_{approx})^2 \Big/ \sum(x_{\exp}-\overline{x}_{\exp})^2$, где $x_{\exp}$ - экспериментальные данные, $\overline{x}_{\exp}$ - экспериментальное среднее, x_{app} - величины, полученные по апроксимационной формуле. На рис.3 показано соответствие этой формулы экспериментальным точкам. Отметим, что наблюдаемые при $L > 400$ флуктуации экспериментальных значений обусловлены тем, что точности алгоритма в использованном нами формате чисел типа double не хватало, чтобы правильно посчитать вторую производную в 4-м знаке после запятой. Относительная ошибка аппроксимации величины U_c меньше 0.4% , а величины C_c меньше 0.8%. Графики на рис.4 показывают хорошее согласие формул (9b)-(9c) с экспериментальными данными.

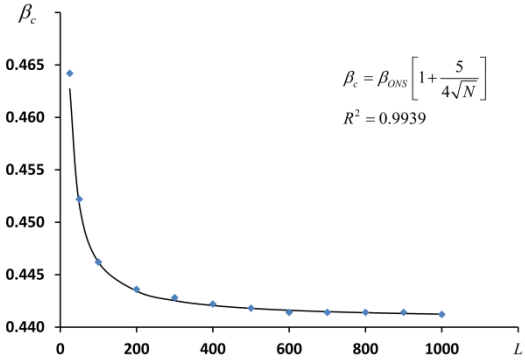


Рис. 3. Зависимость критической температуры β_C от размерности L: маркеры – данные эксперимента, сплошная кривая - аппроксимация формулой (9a).

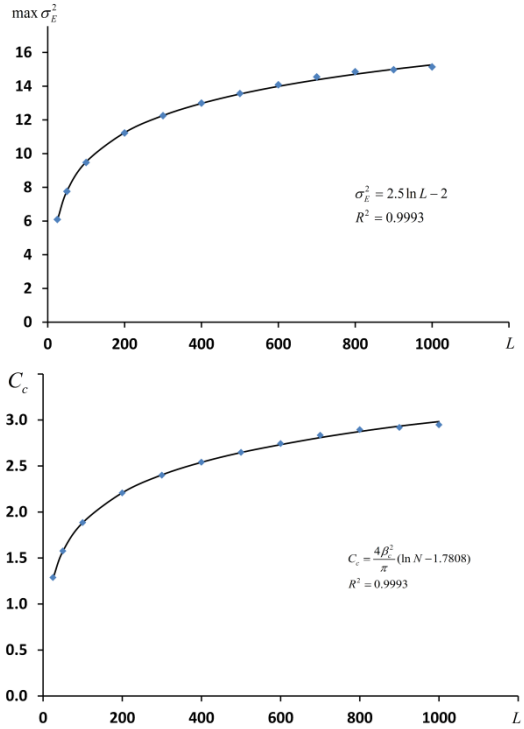


Рис. 4. Зависимость максимального значения дисперсии $\max\sigma_E^2$ (соответствует критической температуре) (сверху) и теплоёмкости C_C (снизу) от размера L: маркеры – данные эксперимента, сплошные кривые - аппроксимационные формулы.

Из приведенных в Таблице 1 данных следует, что положение пика дисперсии энергии (критическое значение β_c^*) и соответствующие этому пику значения

свободной энергии и теплоёмкости хорошо аппроксимируются выражениями:

$$\beta_c^* = \beta_{ONS}\left(1+\frac{1}{\sqrt{N}}\right). \qquad (10a)$$

$$U_c^* = -\sqrt{2}\cdot\left(1-\frac{1}{\sqrt{N}}\right). \qquad (10b)$$

$$C_c^* = 1.197\,\beta_{\mathrm{ONS}}^2\cdot\left(\ln N-1\right). \qquad (10c)$$

Эти выражения находятся в хорошем согласии с данными эксперимента: максимальное значение относительной ошибки имеет место при $L=25$ и составляет 0.6%, 2.1% и 1.2% для величин β_c^*, U_c^* и C_c^* соответственно; с ростом L величина относительной ошибки быстро спадает и при $L=10^3$ достигает значений 0.02%, 0.03% и 0.08% соответственно.

5. Обобщение решения Онсагера

Анализ показал, что можно получить аналитическое выражение, хорошо описывающие экспериментальные данные и полученные выше аппроксимационные выражения. Для этого в (6) достаточно произвести подстановку $2\beta J\to z$ и $k\to\kappa$, где

$$z=\frac{2\beta J}{1+\Delta}, \qquad \kappa=\frac{2\sinh z}{(1+\delta)\cosh^2 z}. \qquad (11)$$

L	$\beta_c\ /\ \beta_c^*$	$f_c\ /\ f_c^*$	$U_c\ /\ U_c^*$	$\sigma_c\ /\ \sigma_c^*$	$C_c\ /\ C_c^*$
25	0.4642 / 0.4556	0.9467 / 0.9351	1.3808 / 1.3288	2.4444 / 2.4678	1.2875 / 1.2642
50	0.4522 / 0.4494	0.9382 / 0.9344	1.3985 / 1.3768	2.7762 / 2.7849	1.5760 / 1.5664
100	0.4462 / 0.4454	0.9337 / 0.9326	1.4054 / 1.3978	3.0767 / 3.0782	1.8846 / 1.8797
200	0.4436 / 0.4432	0.9320 / 0.9314	1.4120 / 1.4075	3.3491 / 3.3502	2.2072 / 2.2046
300	0.4428 / 0.4422	0.9315 / 0.9306	1.4152 / 1.4078	3.4990 / 3.5001	2.4005 / 2.3955
400	0.4422 / 0.4418	0.9309 / 0.9304	1.4143 / 1.4091	3.6050 / 3.6052	2.5413 / 2.5369
500	0.4418 / 0.4418	0.9305 / 0.9305	1.4131 / 1.4131	3.6832 / 3.6832	2.6479 / 2.6479
600	0.4414 / 0.4414	0.9301 / 0.9301	1.4104 / 1.4104	3.7525 / 3.7525	2.7435 / 2.7435
700	0.4414 / 0.4414	0.9302 / 0.9302	1.4124 / 1.4124	3.8141 / 3.8141	2.8344 / 2.8344
800	0.4414 / 0.4414	0.9302 / 0.9303	1.4139 / 1.4139	3.8544 / 3.8544	2.8945 / 2.8945
900	0.4414 / 0.4414	0.9303 / 0.9303	1.4152 / 1.4152	3.8702 / 3.8702	2.9184 / 2.9184
1000	0.4412 / 0.4412	0.9301 / 0.9301	1.4132 / 1.4132	3.8914 / 3.8914	2.9477 / 2.9477

Таблица 1. Критические значения в пике теплоемкости / пике дисперсии энергии.

Сравнение получаемых при этом выражений (13) с данными эксперимента показало, что наилучшее совпадение достигается, когда подгоночные параметры Δ и δ заданы в виде:

$$\Delta=\frac{5}{4\sqrt{N}}, \qquad \delta=\frac{\pi^2}{N}. \qquad (12)$$

Совершая в (6) указанную выше подстановку (12) получим выражение для свободной энергии и вытекающие из него выражения для внутренней энергии и теплоемкости:

$$f(\beta)=-\frac{\ln 2}{2}-\ln\left(\cosh z\right)-\frac{1}{2\pi}\int_0^{\pi}\ln\left(1+\sqrt{1-\kappa^2\cos^2\theta}\right)d\theta \qquad (13a)$$

$$U=-\frac{1}{1+\Delta}\left\{2\tanh z+\frac{\sinh^2 z-1}{\sinh z\cdot\cosh z}\left[\frac{2}{\pi}\mathrm{K}_1-1\right]\right\} \qquad (13b)$$

$$C=\frac{z^2}{\pi\,\tanh^2 z}\cdot\left\{a_1\left(\mathrm{K}_1-\mathrm{K}_2\right)-\left(1-\tanh^2 z\right)\left[\frac{\pi}{2}+\left(2a_2\tanh^2 z-1\right)\mathrm{K}_1\right]\right\} \qquad (13c)$$

где K_1 и K_2 - полные эллиптические интегралы 1-го и 2-го рода соответственно:

$$\mathrm{K}_1=\int_0^{\pi/2}(1-\kappa^2\sin^2\phi)^{-1/2}\,d\phi,$$

$$\mathrm{K}_2=\int_0^{\pi/2}(1-\kappa^2\sin^2\phi)^{1/2}\,d\phi$$

и введены обозначения:

$$a_1 = p\,(1+\delta)^2,\ a_2 = 2p-1,$$
$$p = \frac{\left(1-\sinh^2 z\right)^2}{(1+\delta)^2\cosh^4 z - 4\sinh^2 z}. \qquad (14)$$

Как и следовало ожидать, в асимптотическом пределе $N\to\infty$ из (14) следует $p\to 1$, $a_{1,2}\to 1$ и выражения (13) переходят в хорошо известные [1, 2].

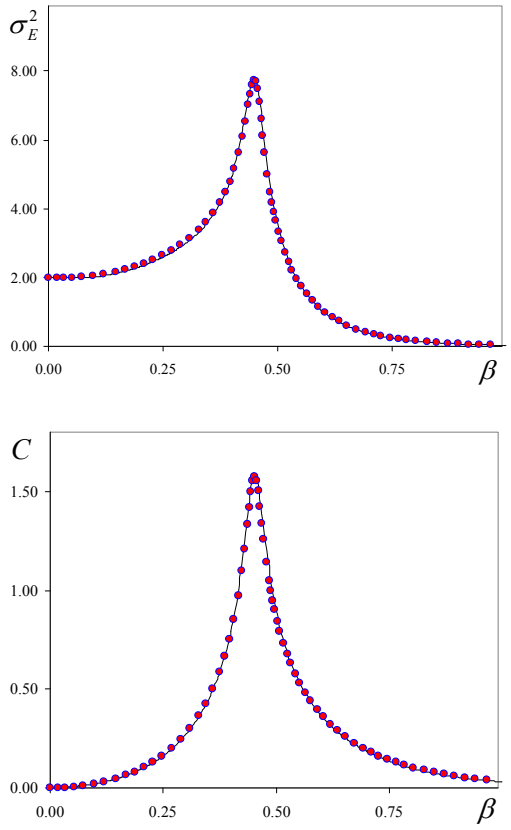


Рис. 5. Зависимость дисперсии (сверху) и теплоемкости (снизу) от β для решетки $N = 25\times 25$: сплошные кривые построены по формулам (13), маркеры – данные эксперимента.

Выражения (13) хорошо аппроксимируют результаты эксперимента даже при относительно малом размере решетки. Для иллюстрации на рисунке 5 приведены графики для дисперсии энергии и теплоемкости для решетки размера $N = 25\times 25$. Как видим, наблюдается хорошее согласие теории с экспериментом. С ростом размера решетки согласие теории с экспериментом существенно улучшается.

Анализ выражений (13) показывает, что введение поправки на конечный размер решетки приводит к незначительным изменениям в поведении свободной и внутренней энергий. А вот в формуле для теплоёмкости пропадает логарифмическая расходимость в критической точке. Действительно, анализ выражений (13) показывает, что максимум теплоемкости достигается при $\sinh z = 1$, что соответствует критической температуре

$$\beta_c = \beta_{ONS}(1+\Delta). \qquad (15)$$

Подставляя сюда Δ из (12) видим, что это выражение в точности совпадает с эмпирически установленным выражением (9a).

Разлагая функцию $C(\beta)$ в окрестности критической точки β_c и опуская полиномиальные по $\beta-\beta_c$ члены получим:

$$C(\beta) \approx \frac{4\beta_c^2}{\pi}\left\{3\ln 2 - \frac{\pi}{2} - \ln\left[4J^2\left(\beta-\beta_c\right)^2 + \frac{\pi^2}{N}\right]\right\}. \qquad (16)$$

Отсюда для критического значения теплоемкости вытекает выражение:

$$C_c = \frac{4\beta_c^2}{\pi}\left(\ln N + 3\ln 2 - 2\ln\pi - \frac{\pi}{2}\right), \qquad (17)$$

что соответствует ($2\ln\pi + \pi/2 - 3\ln 2 \approx 1.78$) эмпирически найденной формуле (9c).

Наличие выражений (13) позволяет провести анализ других характеристик, например корреляции и спонтанной намагниченности. Сначала рассмотрим зависимость от N корреляционной длины ξ. В пределе $N\to\infty$ она определяется известным выражением [1], которое мы представим в виде $\xi = -1/2\ln\eta$, где $\eta = k/(1+\sqrt{1-k^2})$. Для перехода к случаю решетки конечного размера произведем здесь указанную в (11) подстановку $k\to\kappa$. Тогда для корреляционной длины получим:

$$\xi = -\frac{1}{2\ln\bar{\eta}},\quad \bar{\eta} = \frac{\kappa}{1+\sqrt{1-\kappa^2}}. \qquad (18)$$

В критической точке $\beta=\beta_c$ величина $\kappa = \kappa(z)$ достигает своего максимума $\kappa_{\max} = 1/(1+\delta)$, а корреляционная длина своего максимального значения

$$\xi_{max} = \frac{l}{2\pi\sqrt{2}}, \qquad (19)$$

которое на порядок меньше линейного размера решетки L.

Аналогичным образом совершим подстановку (11) в полученное Янгом выражение [36]. Тогда для спонтанной намагниченности получим:

$$M_0 = \left[1-\left(\frac{\bar{\eta}}{\bar{\eta}_0}\right)^2\right]^{1/8}, \qquad (20)$$

где $\bar{\eta}_0 \approx 1/(1+\sqrt{2\delta})$ - значение функции $\bar{\eta} = \bar{\eta}(\kappa)$, достигаемое при $\beta=\beta_c$ ($M_0 = 0$

при $\beta < \beta_c$). Подчеркнем, что интерпретация выражения (20) применительно к системе конечных размеров в корне отличается от случая $N \to \infty$. Как указано в [1], среднее значение намагниченности системы, размеры которой конечны, в отсутствие внешнего поля равно нулю, поскольку для любой конфигурации с $s_i = +1$ имеется равновероятная конфигурация с $s_i = -1$. Соответственно, выражение (20) применимо для описания следующего факта: измеряемая в эксперименте намагниченность в различные моменты времени может принимать любое значение между M_0 и $-M_0$.

6. Обсуждение и заключение

На основе экспериментальных данных мы получили простые оценочные выражения (9)-(10) для критических величин, хорошо согласующиеся с экспериментом. Выбор подгоночных параметров в виде (11) дает незначительную ошибку аппроксимации в асимптотических пределах $\beta \to 0$ и $\beta \to \infty$, однако идеально описывает положение и высоты пиков. Конечно, подгоночные коэффициенты в этих выражениях можно было бы уточнить и выписать оценочные выражения с большей точностью. Однако мы не ставили перед собой такой цели – основной целью было установление тенденции изменений поведения критических параметров в зависимости от N.

Введением двух подгоночных параметров (12) получены аналитические выражения (13), обобщающие решение Онсагера на случай решетки конечного размера. Эти выражения с большой точностью описывают поведение спиновой системы даже при малых $(N = 25 \times 25)$ размерах решетки, а при $N \geq 50 \times 50$ их рассогласование с экспериментом становится меньше ошибки эксперимента.

На основании проведенных исследований можно сделать следующие выводы.

Во-первых, численный эксперимент, как правило, позволяет правильно устанавливать характерные особенности поведения спиновой системы даже при относительно малых значениях N. Увеличение N позволит лишь уточнить значение критических параметров. Однако это уточнение не столь уж существенно: как следует из (9) точность определения критических величин β_c и U_c определяется величиной $\sim L^{-1}$. Трудно говорить об уточнении величины $C_c \sim \ln N$, поскольку увеличив размер задачи от $L = 10^3$ до $L = 10^4$ порядок мы всего лишь заметим ничего не говорящее нам изменение C_c на величину ~ 30%.

Во-вторых, у систем конечных размеров исчезает предсказанная Онсагером логарифмическая расходимость теплоемкости в критической точке. То же самое справедливо и для дисперсии энергии. Этого и следовало ожидать из самых общих соображений. Вместо этого, мы наблюдаем логарифмическое нарастание критической величины теплоемкости типа $C_c \sim \ln N$. Казалось бы, при $N \to \infty$ мы переходим к пределу Онсагера ($C \to \infty$ при $\beta \to \beta_c$). Однако, такой переход на практике трудно осуществить: даже увеличив размер задачи до числа Авогадро $N \sim 10^{23}$ мы всего лишь в 4 раза изменим величину C_c по сравнению с рассмотренным нами случаем $N = 10^6$. Более того, зависимость удельной теплоемкости от N означает нарушение принципа аддитивности в классической системе, вполне заметное даже при $N \sim 10^{23}$: увеличение размера системы в два раза приведет к изменению величины C_c на 1.3%. Этот факт ничему не противоречит, поскольку нами рассмотрена модель с свободными граничными условиями, которая неаддитивна по определению. Как следует из (17) влиянием границ можно пренебречь только в пределе $\ln N >> 2$, который в условиях численного эксперимента недостижим.

В третьих, эксперимент показал, что пики кривых $\sigma_E^2 = \sigma_E^2(\beta)$ и $C = C(\beta)$ не совпадают – пик теплоемкости достигается при больших значениях β. Это вполне ожидаемый результат, поскольку теплоемкость связана с дисперсией энергии соотношением $C(\beta) = \beta^2 \sigma_E^2(\beta)$. Однако, встает вопрос – по какому из пиков определять критическое значение температуры? Действительно, первый из пиков соответствует максимуму дисперсии энергии, а второй – максимуму корреляционной длины. Оба эффекта являются характерными признаками фазового перехода. Мы, по привычке, определяли критическую точку по пику теплоемкости. При больших размерах решетки такой подход вполне оправдан: при $N \geq 400 \times 400$ расстояние между этими пиками мы уже не могли зафиксировать. Однако, при меньших размерах зазор между пиками был хорошо заметен. Скорее всего, при интерпретации эксперимента на решетках малых размеров следует говорить, что фазовый переход размазан на температурном отрезке от β_c^* до

β_c. Следовательно, численный эксперимент позволяет определить критическую температуру только с точностью до размера этого отрезка, т.е. абсолютная ошибка будет порядка величины $\pm\beta_{ONS} / 4n$.

Характер зависимости критических параметров от размера решетки проведен на примере двумерной модели Изинга. Однако, мы полагаем, что основные выводы будут справедливы и для других моделей.

Работа частично поддержана грантами РФФИ №16-31-00047 и №15-07-04861, а так же проектом ФАНО №35.14.

A generalization of the Onsager solution for a two-dimensional finite-size Ising model

M.Yu. Malsagov, I.M. Karandashev, B.V. Kryzhanovsky

Abstract. The dependence of the thermodynamic characteristics of the two-dimensional Ising model on the number of spins N was investigated by numerical methods. Experimental data obtained by the Kastelein-Fisher algorithm for the model on a square lattice $N=L\times L$ are compared with the asymptotic solution of Onsager ($N\to\infty$). Empirical expressions describing the dependence of the critical parameters on N are obtained. A generalization of the Onsager solution to the case of a lattice of finite dimensions is obtained and analytical expressions for the free energy and its derivatives (internal energy, energy dispersion, heat capacity), which describe the results of a numerical experiment, are obtained. It is shown that with increasing N the heat capacity at the critical point increases logarithmically. The limitations on the accuracy of determining the critical temperature due to the finite size of the system are indicated.

Keywords: statistical sum, normalization constant, Ising model.

Литература

1. R.J. Baxter. Exactly solved models in statistical mechanics. London: Academic Press, 1982.
2. H.Stanley. Introduction to phase transitions and critical phenomena. Clarendon Press, Oxford, 1971.
3. L. Onsager. Crystal statistics. I. A two-dimensional model with an order–disorder transition. Phys. Rev. 65 (3–4), 117–149 (1944).
4. K. Huang, Statistical Mechanics, Wiley, New York, 1987.
5. J.M. Dixon, J.A. Tuszynski, P. Clarkson. From Nonlinearity to Coherence: Universal Features of Nonlinear Behavior in Many-Body Physics. Clarendon Press, Oxford, 1997.
6. H.A. Kramers, G.H. Wannier, Statistics of the two-dimensional ferromagnet. Part I, Phys. Rev. 60(3), 252–262 (1941).
7. R. Kubo, An analytic method in statistical mechanics. Busserion Kenkyu 1 (1943) 1–13 (in Japanese).
8. J.C. Bonner, M.E. Fisher. Linear magnetic chains with anisotropic coupling. Phys. Rev. 135 (3A) A640–A658 (1964).
9. G.A. Baker Jr., G.S. Rushbrooke, H.E. Gilbert. High-temperature series expansions for the spin ½ Heisenberg model by the method of irreducible representations of the symmetric group. Phys. Rev.135 (5A), 1272–1277 (1964).
10. J. Kondo, K. Yamaji. Green's-function formalism of the one-dimensional Heisenberg spin system, Prog. Theor. Phys. 47 (3) (1972) 807–818.
11. J.J. Cullen, D.P. Landau, Monte Carlo studies of one-dimensional quantum Heisenberg and XY models, Phys. Rev. B 27 (1) (1983) 297–313.
12. J.W. Lyklema. Monte Carlo study of the one-dimensional quantum Heisenberg ferromagnet near T ¼ 0; Phys. Rev. B 27 (5) (1983) 3108–3110.
13. M. Marcu, J. Muller, F.-K. Schmatzer. Quantum Monte Carlo simulation of the one-dimensional spin-S xxz model. II. high precision calculations for S ¼ . J. Phys. A 18 (16) (1985) 3189–3203.
14. P. Schlottmann. Low-temperature behavior of the S ¼ , ½ ferromagnetic Heisenberg chain, Phys. Rev. B 33 (7) (1986) 4880–4886.
15. M. Kac and J. Ward. A combinatorial solution of the two-dimensional Ising model. Phys. Rev., 88(6), 1952.
16. M. Takahashi, M. Yamada. Critical behavior of spin-1/2 one-dimensional Heisenberg ferromagnet at low temperatures, J. Phys. Soc. Jpn 55 (6) (1986) 2024–2036.

17. C.K. Majumdar, I. Ramarao. Critical field and low-temperature critical indices of the ferromagnetic Ising model, Phys. Rev. B 22 (7) (1980) 3288–3293.
18. B.V. Kryzhanovsky, L.B. Litinskii. Generalized Bragg-Williams Equation for Systems with Arbitrary Long-Range Interaction. Doklady Mathematics, Vol. 90, No. 3, pp. 784–787 (2014).
19. B. Kryzhanovsky, L. Litinskii. Applicability of n-vicinity method for calculation of free energy of Ising model. Physica A, Vol. 468, pp. 493–507, 2017.
20. D. Amit, H. Gutfreund and H. Sompolinsky. Statistical Mechanics of Neural Networks Near Saturation. Annals of Physics, 173 (1987), 30-67.
21. J.L. van Hemmen, R. Kuhn. "Collective Phenomena in Neural Networks" in Models of Neural Networks, E. Domany, J.L van Hemmen and K. Shulten, Eds. Berlin: Springer, 1992, pp. 1-105.
22. O.C. Martin, R. Monasson, R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems. Theoretical Computer Science, 265(1-2), pp. 3-67 (2001).
23. I. Karandashev, B. Kryzhanovsky, L. Litinskii. Weighted patterns as a tool to improve the Hopfield model. Phys.Rev. E 85, 041925 (2012) .
24. G.E. Hinton, S. Osindero, Y. Teh. A fast learning algorithm for deep belief nets. Neural Computation, v. 18, pp.1527-1554 (2006).
25. M.J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. IEEE Trans. on Information Theory, 51(7):2313–2335, 2005.
26. J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Trans. on Information Theory, 51(7):2282–2312, 2005.
27. H.W.J. Blote, L.N. Shchur, A.L. Talapov. The cluster processor: new results. 1999 Int. J. Mod. Phys. C 10 1137
28. R. Häggkvist , A. Rosengren, P.H. Lundow, K. Markström, D. Andren, P. Kundrotas. On the Ising model for the simple cubic lattice. Advances in Physics, v. 56,#5,, 2007, pp.653-755.
29. Lundow P.H, Markstrom K. The critical behaviour of the Ising model on the 4-dimensional lattice. Phys. Rev. E 80, 031104 (2009) Preprint arXiv:1202.3031v1
30. Lundow P.H, Markstrom K. The discontinuity of the specific heat for the 5D Ising model. 2015, Nuclear Physics B, 895 pp.305-318.
31. J.M. Dixon, J.A. Tuszynski, E.J. Carpenter. Analytical expressions for energies, degeneracies and critical temperatures of the 2D square and 3D cubic Ising models, Physica A 349 (2005) 487–510.
32. P. Kasteleyn. Dimer statistics and phase transitions. J. Math. Phys., 4(2), 1963.
33. M. Fisher. On the dimer solution of planar Ising models. J. Math. Phys., 7(10), 1966.
34. Ya.M. Karandashev, M.Yu. Malsagov. Polynomial algorithm for exact calculation of partition function for binary spin model on planar graphs. Optical Memory & Neural Networks (Information optics), v. 26, #2, 2017. https://arxiv.org/abs/1611.00922
35. N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar Ising models. In NIPS, 2008. https://arxiv.org/abs/0810.4401
36. C.N. Yang. The Spontaneous Magnetization of a Two-Dimensional Ising Model. Phys.Rev., 65, 808 (1952).

Необходимость учёта изменения смачиваемости пород при моделировании тепловых методов добычи нефти

В.А.Юдин [1], И.В.Афанаскин [2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] yudinval@yandex.ru, [2] ivan@afanaskin.ru

Аннотация. Рассмотрена возможность изменений смачиваемости пород при применении тепловых методов добычи нефти. Приведён краткий обзор результатов экспериментальных исследований зависимости характера смачиваемости от температуры. Карбонатные породы с повышением температуры становятся более гидрофильными, результаты по песчано-глинистым породам - крайне противоречивы. На простых модельных расчётах проанализировано влияние изменения смачиваемости на динамику разработки, в сравнении с изменением вязкости нефти. Показано, что изменение смачиваемости может как значительно улучшать показатели разработки (в гидрофобных породах), так и заметно ухудшать их (в гидрофильных). Сделан вывод о необходимости детальных исследований в этом направлении, и тщательном рассмотрении возможных изменений смачиваемости при моделировании тепловых методов добычи нефти. Без этого численное моделирование добычи может быть весьма недостоверным.

Ключевые слова: тепловые методы добычи нефти, методы увеличения нефтеотдачи, смачиваемость, моделирование разработки.

Введение

По данным большинства аналитиков, производство и потребление энергии в мире в ближайшие 2–3 десятилетия будет расти, причём вклад нетрадиционных и возобновляемых источников энергии в общее производство и потребление энергии через 20–30 лет составит только около 30% [3, 4].

Вклад не возобновляемых углеводородных источников энергии (нефти, газа, угля) будет определяющим в снабжении человечества энергией, в том числе и России [1–3].

На территории России, составляющей 12,8% территории Земли, сосредоточено 12-13% прогнозных ресурсов и около 12% разведанных запасов нефти [1–4].

На сегодняшний день суммарные ресурсы нефти и битумов оцениваются в 2,066 млрд.т [1].

Однако, доказанные запасы нефти в России не велики, и при сохранении нынешних темпов её добычи их хватит менее чем на 30 лет [3].

Ввиду неблагоприятного соотношения между разведанными запасами и уровнем годовой добычи, многие специалисты прогнозируют возможное значительное падение нефтедобычи в России в недалёком будущем [1-3].

Помимо этого, наблюдается и ухудшение качества остаточных доказанных запасов [1, 2, 4].

1. Вязкие нефти как один из потенциальных источников нефтедобычи в России

По данным И.С. Гольдберга (1981), суммарные прогнозные ресурсы вязких нефтей, нефтяных песков и битумов на территории бывшего СССР составляют 30-33 млрд.т. [5], из них ресурсы тяжёлых нефтей составляют 13,4 млрд.т. [6]. По данным работы [7], от общего объёма разведанных в России запасов нефти 14% - приходится на тяжёлые нефти, 11% - на высоковязкие. Они сосредоточены в трёх основных провинциях - Волго-Уральской, Западно-Сибирской и Тимано-Печорской [7], причём 60% запасов тяжёлых нефтей сосредоточено в 15 месторождениях: Русское, Ван-Еганское, Фёдоровское, Ново-Елховское, Усинское, Торавейское, Северо-Комсомольское, Ново-портовское, Комсомольское, Вынгапуровское, Западно-Мессояхское, ТазовскоеНаульское, Ярегское, Медынское-Море, Приразломное, Сюрхаратинское и др. [7, 8].

Основная трудность в разработке месторождений вязких нефтей - низкая скорость фильтрации, обусловленная их высокой вязкостью. Применение традиционных методов – например заводнения - не позволяет достичь высокого значения коэффициента извлечения нефти ввиду вязкостной неустойчивости фронта вытеснения вязкой нефти водными или газовыми вытесняющими агентами.

При этом практически для всех таких нефтей вязкость существенно снижается с ростом температуры, обычно по одному из известных законов [10, 11]:

$$ln(\mu) = A - B \cdot ln(T),\ B > 0,$$
$$ln(\mu) = A - B/(T + C),\ B > 0$$
$$ln(\mu + A) = B \cdot T^{-C},$$

где μ - вязкость нефти, T - температура, A, B и C - экспериментальные коэффициенты.

Поэтому основными методами разработки таких месторождений являются тепловые. Они основаны на различных способах повышения температуры пласта и снижении вязкости нефти путём [4]:

- циклических тепловых обработок призабойных зон скважин,
- закачки в пласт теплоносителя (горячей воды, сухого или влажного водяного пара),
- инициирования в пласте процесса окисления и горения части нефти с закачкой в пласт воздуха или кислорода и, тем самым, создания подвижного очага окисления или горения нефти для выделения тепла.

Накопленный мировой опыт разработки залежей с высоковязкими нефтями доказал эффективность использования тепловых методов в таких случаях. Если традиционно применяемые технологии заводнения на месторождениях с нефтями повышенной и высокой вязкости могли обеспечить конечную нефтеотдачу не более 20–25%, то использование тепловых методов позволяет в ряде случаев довести нефтеотдачу до 40-45 % [9].

2. Изменение смачиваемости пород при нагреве пласта

Во многих работах отмечается возможность изменениях смачиваемости пород при нагреве, т.е. при воздействии высоких температур, которые при тепловых воздействиях разного рода могут на 100–200^{0}C превосходить начальную пластовую. Подобные изменения, в принципе, могут кардинально изменить характер фильтрации флюидов и результативность теплового воздействия.

К сожалению, приходится констатировать, что вопрос этот вообще пока слабо изучен. Более того, полученные разными авторами в различных экспериментах данные весьма противоречивы.

В обзоре [12] приводятся результаты ряда исследований изменения смачиваемости в зависимости от роста температуры при классических термических методах добычи (закачка пара, внутрипластовое горение и т.д.). В ряде экспериментов, как на насыпных песчаных моделях, так и на образцах песчаника, терригенная порода с ростом температуры становилась более гидрофильной. На насыпных моделях - остаточная нефтенасыщенность уменьшалась с ростом температуры, а на образцах песчаника об этом свидетельствовали также и изменения капиллярных кривых.

Однако в ряде экспериментов получены и прямо противоположные результаты – порода становилась более гидрофобной по мере повышения температуры, даже при закачке пара. В некоторых исследованиях, с ростом температуры сначала наблюдалось увеличение степени гидрофобности, а затем её уменьшение при последующем повышении температуры.

Более однозначны результаты, полученные в карбонатных породах, которые, согласно результатам практически всех исследователей, с повышением температуры становятся более гидрофильными.

Существует также мнение, что характер смачиваемости вообще никак не зависит от температуры, либо, при нагреве порода становится нейтральной по смачиваемости. Столь большой разброс результатов, вероятнее всего, определяется рядом объективных факторов. Во-первых, эксперименты проводились по различным методикам, с различными нефтями, и в разных условиях: на насыпных моделях, образцах керна, опытных участках. Влияние смачиваемости оценивалось также по-разному: по капиллярному впитыванию, по виду кривых относительных фазовых проницаемостей, и т.п. Во-вторых, смачиваемость зависит от многих физических факторов, по-разному реагирующих на повышение температуры. В их числе можно назвать: начальное водонасыщение и характер его изменения в процессе разработки; величина pH; состав нефти и возможность осаждения из неё асфальтенов; глинистость пород; устойчивость тонких плёнок воды на твёрдой поверхности и др. [12]. Результирующее изменение смачиваемости пород при нагреве будет зависеть от сочетания температурного поведения всех указанных факторов [12], имеющих в разных ситуациях разную значимость.

Отметим, что различие в характере изменения смачиваемости при нагреве часто отмечается и в других областях науки и техники.

Естественно, возникает вопрос, насколько сильно вероятное изменение смачиваемости пород при нагреве может повлиять на

конечный результат применения тепловых методов добыч нефти, и, соответственно, насколько необходимо его учитывать при численном моделировании таких способов разработки. Для грубой качественной оценки значимости этого фактора были выполнен ряд модельных расчётов.

3. Оценка влияния изменения смачиваемости на показатели разработки

Модельный оценочный расчёт проводился с помощью математической модели двухфазной фильтрации несмешивающихся жидкостей (нефти и воды) Dz10 [13] в трёхмерной постановке для следующих значений параметров:

1. Количество ячеек по осям X, Y, Z 10-10-1.
2. Размер ячеек по осям X, Y, Z 50-50-20 м.
3. Размеры модели 500-500-20 м.
4. Пористость 0,2 д.ед.
5. Проницаемость 30 мД.
6. Объёмный коэффициент воды: 1,01 $м^3/м^3$.
7. Вязкость воды 0,3 мПа·с.
8. Сжимаемость воды $4{,}7\cdot10^{-5}$ 1/бар.
9. Объёмный коэффициент нефти 1,1 $м^3/м^3$.
10. Вязкость нефти 1 и 10 мПа·с.
11. Сжимаемость нефти $8{,}0\cdot10^{-5}$ 1/бар.
12. Растворимость газа в нефти 108 $м^3/м^3$.
13. Давление насыщения нефти газом 70 бар.
14. Сжимаемость пласта: $4{,}7\cdot10^{-5}$ 1/бар.
15. Плотность нефти при стандартных условиях 815 $кг/м^3$.
16. Плотность воды при стандартных условиях 1056 $кг/м^3$.
17. Глубина залегания 3390 м.
18. Начальное пластовое давление 339 бар.
19. Начальная водонасыщенность (гидрофильный пласт) 0,4 д.ед.
20. Начальная водонасыщенность (гидрофобный пласт) 0,2 д.ед.
21. Добывающая скважина находится в ячейке с координатами 10-10-1.
22. Нагнетательная скважина находится в ячейке с координатами 1-1-1.
23. Добывающая скважина работает при постоянном дебите жидкости 150 $м^3$/сут.
24. Нагнетательная скважина работает при постоянном забойном давлении 419 бар.
25. Срок разработки 3600 дней (примерно 10 лет).
26. Капиллярным давлением пренебрегаем, т.к. пласт однороден по пористости и проницаемости. Влияние капиллярных сил косвенно учитывается путём введения различных ОФП для случаев гидрофобного и гидрофильного коллектора.

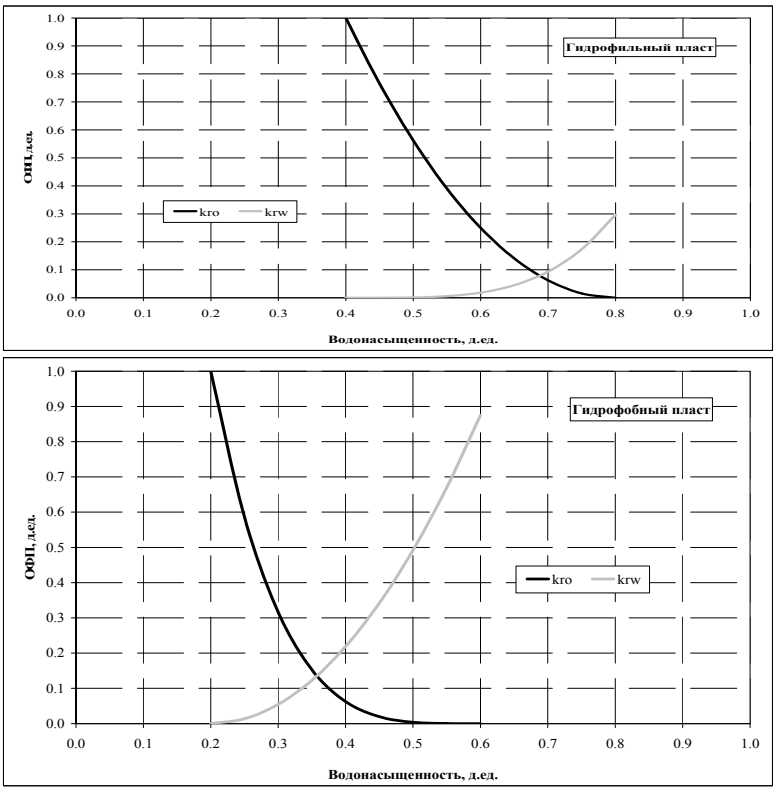


Рис.1. Принятые (по литературным данным) при расчёте кривые относительной фазовой проницаемости (ОФП) по нефти (kro) и воде (krw)

Ячейки разностной сетки прямоугольные, блочно-центрированные.

При расчётах были использованы типичные кривые относительных фазовых проницаемостей для гидрофильного и гидрофобного пласта, взятые по литературным данным и показанные на рис. 1.

Результаты расчётов приведены на рис. 2–5. Поскольку при нагреве нефтеносного пласта происходят два значимых эффекта – изменение вязкости нефти и характера смачиваемости, то на рисунках кривые, обусловленные влиянием обоих эффектов, сопоставлены со случаем сохранения начального характера смачиваемости, но уменьшения вязкости нефти.

Как видно из рис. 2 и 3, при нагреве гидрофильного пласта снижение вязкости нефти, естественно, улучшает показатель разработки, в частности, увеличивается период стабильного дебита нефти, а длительность падения дебита и роста обводнённости сокращается. В то же время, изменение поверхности породы с гидрофильной на гидрофобную сказывается на динамике разработки крайне отрицательно. Появление прорывов воды, проскальзывающей по фильтрационным каналам, поверхность которых «смазана» адсорбированными плёнками поверхностно-активных углеводородов, превышает положительный эффект от снижения вязкости нефти. Показатели разработки становятся даже хуже, чем при начальных параметрах пласта и нефти: падение дебита нефти начинается раньше и период обводнения становится более длительным. Накопленная добыча нефти значительно уменьшается.

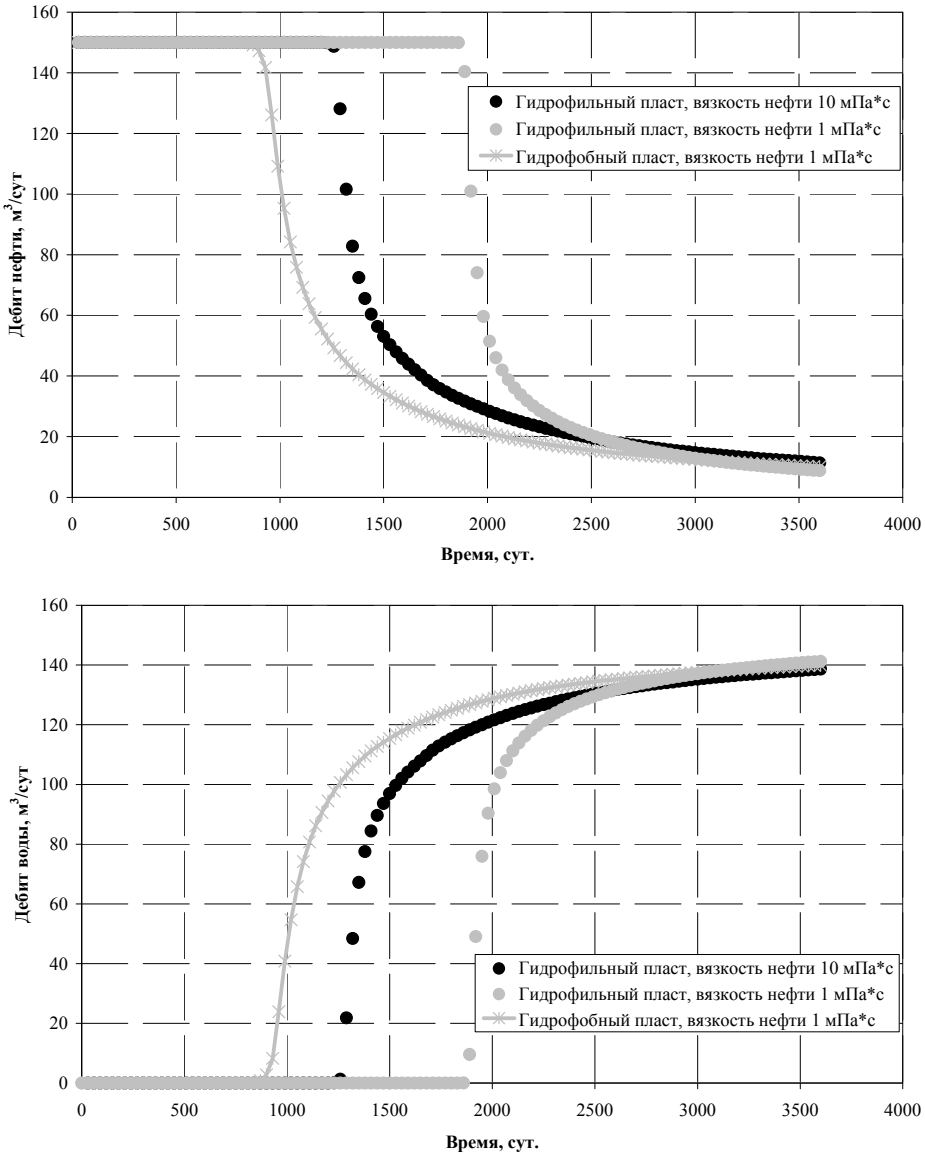


Рис. 2. Влияние изменения вязкости и характера смачиваемостина дебит воды и нефти при нагреве гидрофильного пласта

В гидрофобном пласте ситуация иная (рис. 4 и 5). В отсутствие нагрева процесс обводнения контролируется прорывами воды, которые обусловлены двумя факторами: вязкостной неустойчивостью вытеснения, поскольку вязкость нефти в 10 раз превышает вязкость воды; а также проскальзыванием воды по фильтрационным каналам, стенки которых покрыты адсорбированными углеводородами. Как показывают расчёты (рис. 4 и 5), в этом случае и уменьшение вязкости нефти, и изменение характера поверхности фильтрационных каналов – действую в одну сторону: уменьшают вероятность прорывов воды, увеличивают коэффициент вытеснения, снижают время начала падения дебита и длительность периода обводнения.

Накопленная добыча нефти значительно возрастает - за счёт обоих эффектов.

Следует подчеркнуть, что рассмотренный в статье случай во многом является иллюстративным. В действительности, при тепловых методах добычи поле температур в пласте неоднородно по пространству, поэтому характер смачиваемости поверхности также будет переменным по пространству. Это обстоятельство в статье не учитывалось.

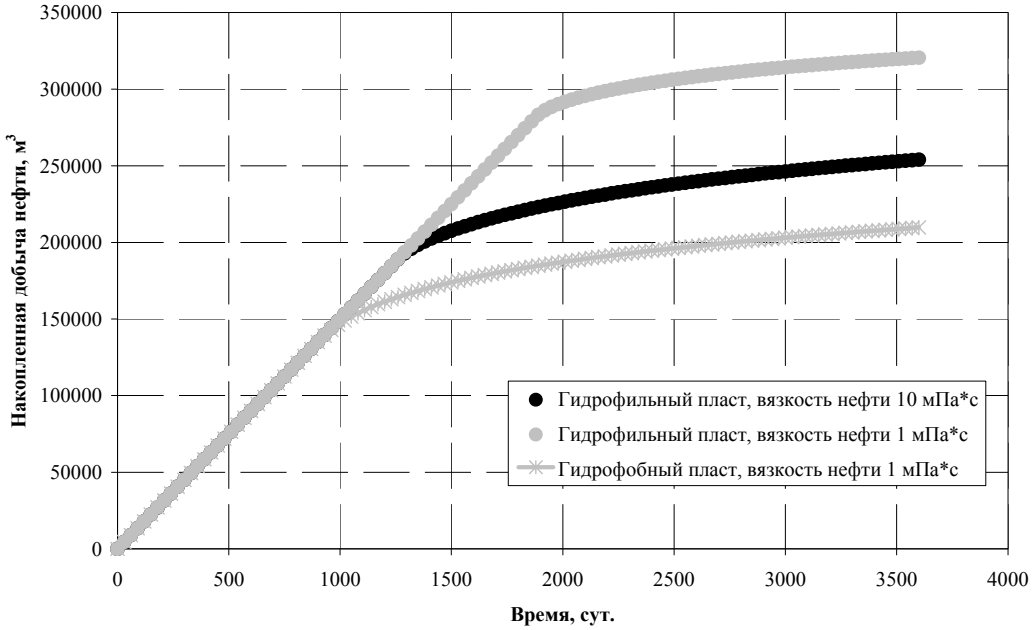


Рис. 3. Изменение накопленной добычи нефти при нагреве гидрофильного пласта

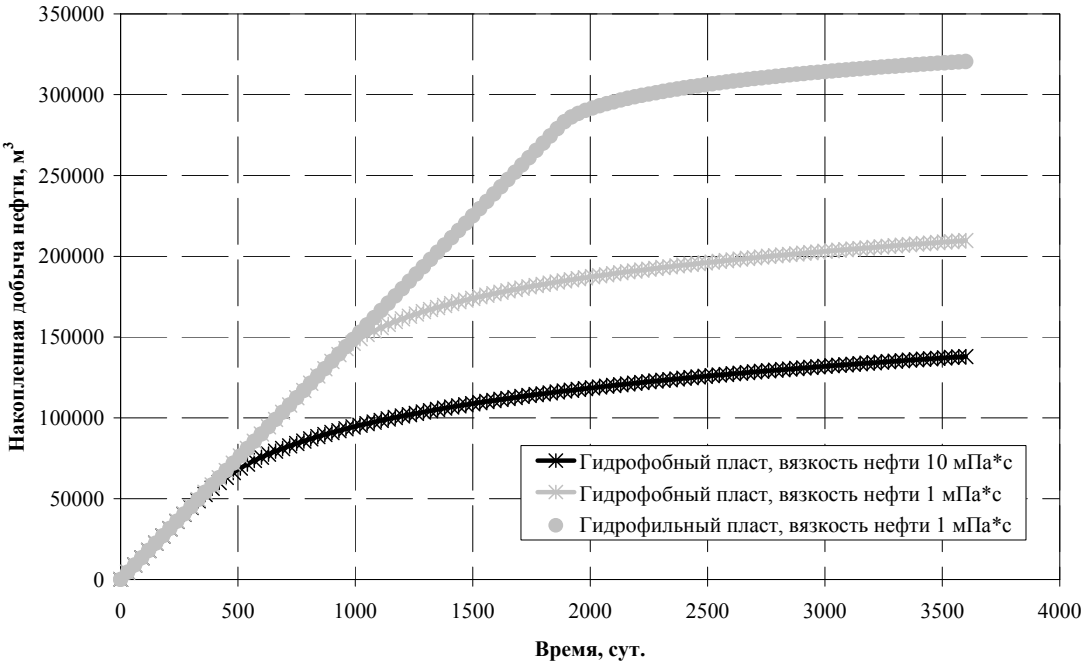


Рис. 4. Изменение накопленной добычи нефти при нагреве гидрофобного пласта

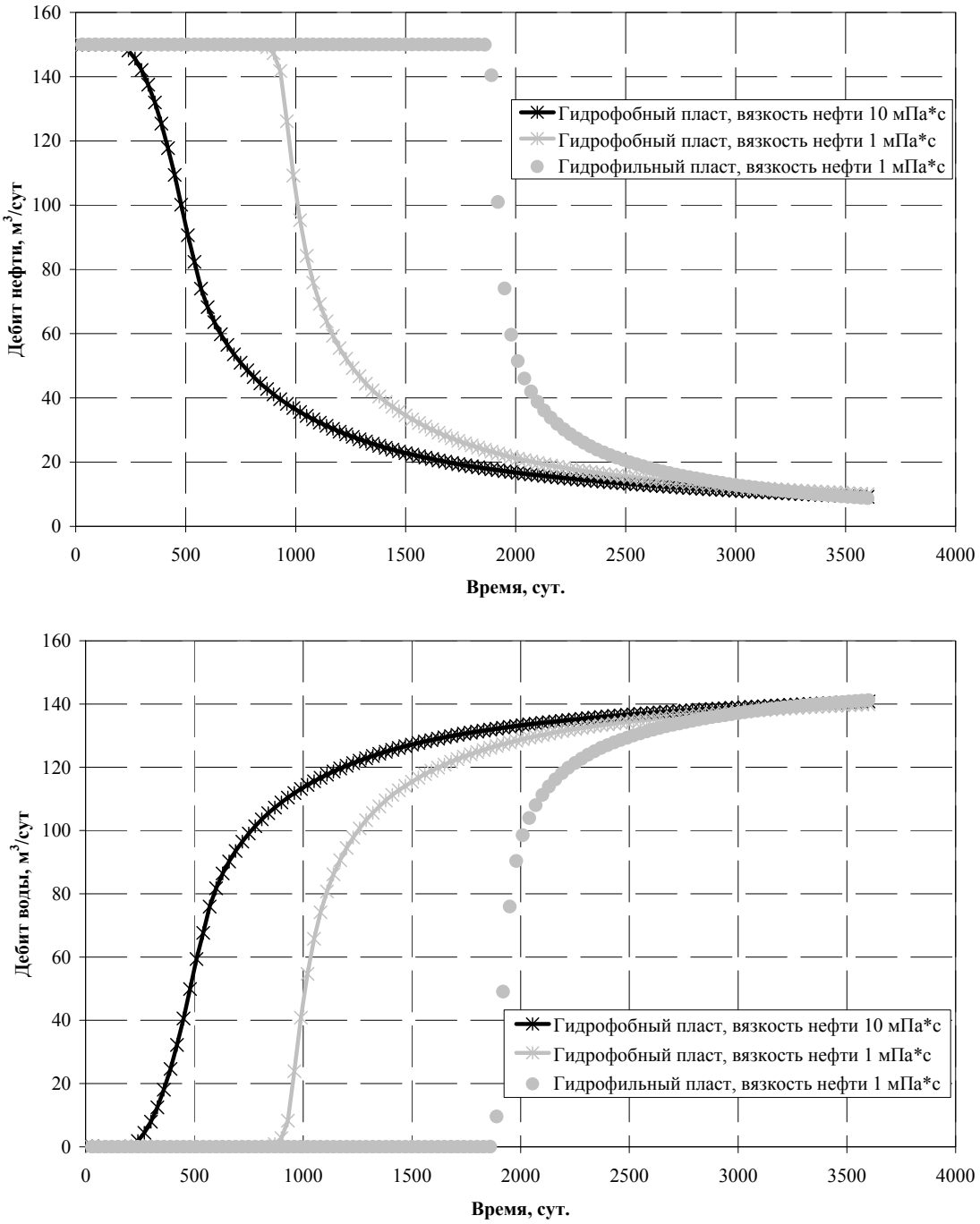


Рис.5. Влияние изменения вязкости и характера смачиваемости на дебит нефти и воды при нагреве гидрофобного пласта

Выводы

Несмотря на качественный, даже иллюстративный, характер выполненных расчётов, они, в совокупности с обзором литературных данных, свидетельствуют о том, что температурное изменение смачиваемости может оказывать значительное влияние на динамику добычи. При моделировании любых тепловых методов добычи нефти этот вопрос должен быть тщательно рассмотрен, что, к сожалению, как правило не выполняется на практике проектными организациями и НИИ.

При этом необходимо, по нашему мнению, выполнить цикл теоретических и экспериментальных работ по изучению изменения характера смачиваемости пород при нагреве, в широком интервале температур – от пластовых до, примерно, 400^0C. Такие исследования должны быть выполнены для широкого интервала параметров пласта и нефтей, охватывающего условия основных

объектов применения тепловых методов добычи нефти в нашей стране.

При моделировании тепловых методов на конкретном объекте необходимо, до начала моделирования, оценить значимость подобных эффектов в рассматриваемых условиях и необходимость их учёта. При положительном ответе – для их учёта система решаемых уравнений подземной гидродинамики должна быт соответствующим образом усложнена.

The necessity of taking into account rock wettabilty changes in thermal oil recovery methods simulation

V.A.Judin, I.V.Afanaskin

Abstract. Possibility of rock wettability changes as a result of using thermal methods of development is considered based on a brief literature review. Brief summary of experimentally obtained rocks wettability changes with temperature increase are presented. Carbonates become more water-wet with temperature increase while experimental results for terrigenous formations are quite ambiguous. On the base of simple computer modelling it is shown that the production history might be improved by wettability changes (in oil-wet rocks) or made worse (in water-wet rocks).A thorough investigation of the problem is proposed under thermobaric conditions of Russian oil fields with viscous oils on which thermal methods are planned to be applied. This is necessary to increase reservoir modeling reliability.

Keywords: thermal methods of oil development, wettability, improved oil recovery, reservoir simulation.

Литература

1. А.Орёл. Разработка трудноизвлекаемых запасов связана с инвестиционными рисками. «Нефть и жизнь», т. 108 (2016), № 8.
2. А.Н.Ищенко. О перспективах развития ТЭК России. «Недропользование - XXI век», 2017, №1, 12–13.
3. BP Energy Outlook - 2017. Режим доступа в сети интернет: http://www.imemo.ru/files/File/ru/conf/2017/07022017/07022017-PRZ-EO17-Presentation-Spencer%20short.pdf
4. В.Б.Бетелин, В.А.Юдин, И.В.Афанаскин, С.Г.Вольпин, Р.М.Кац, А.В.Королёв. Создание отечественного термогидросимулятора – необходимый этап освоения нетрадиционных залежей углеводородов России. М., ФГУ ФНЦ НИИСИ РАН, 2015.
5. Б.В.Успенский. Научно-методические основы поиска, разведки и освоения природных битумов. Дисс. на соиск. уч. степ. докт. геол.-мин. наук. Казань, 2005.
6. И.В.Николин. Методы разработки тяжёлых нефтей и природных битумов. «Наука – фундамент решения технологических проблем развития России», 2007, № 2, 58–64.
7. В.П.Якуцени, Ю.Э.Петрова, А.А.Суханов. Нетрадиционные ресурсы углеводородов - резерв для восполнения сырьевой базы нефти и газа в России. «Нефтегазовая геология. Теория и практика», т. 1 (2009), № 4.
8. Д.Ю.Чиркова, Н.А.Красноярова, И.Г.Ященко. Проблемы рационального использования вязких и тяжелых нефтей российской Арктики. «XVI Международная научно-практическая конференция имени профессора Л.П. Кулёва». Томск, Томский политехнический университет, 2015.
9. Я.Бао, Ю.Тянь, А.В.Сиднев. Глинистые низкопроницаемые коллекторы в нефтегазоносных бассейнах Китая и технологии добычи нефти. «Современные наукоемкие технологии», 2008, № 2, 57–58.
10. В.Е.Борисов. Композиционная неизотермическая модель фильтрации в пористой среде с учетом химических реакций и активной твердой фазы. «Препринты ИПМ им. М.В.Келдыша», 2013, № 91.
11. Д.Г.Антониади, А.Р.Гарушев, Б.Г.Ишханов. Настольная книга по термическим □методам □добычи □нефти. Краснодар: «Советская Кубань», 2000.
12. A.Punase, A.Zou, R.Elputranto. How Do Thermal Recovery Methods Affect Wettability Alteration? «Journal of Petroleum Engineering», V. 2014, Article ID 538021.
13. Р.М.Кац, Е.Р.Волгин, И.В.Афанаскин. Численное моделирование двухфазной фильтрации нефти и воды. «Труды НИИСИ РАН», т. 4. (2014), № 2, 141-148.

Анализ эффективности масштабирования при расчетах высокоскоростных турбулентных течений на суперкомпьютере RANS/ILES методом высокого разрешения

Л.А. Бендерский, Д.А. Любимов, А.А. Рыбаков

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail: antbar@mail.ru

Аннотация: В статье рассматривается актуальная задача расчета характеристик нестационарного турбулентного течения внутри воздухозаборника высокоскоростного летательного аппарата. Подобные расчеты ввиду своей требовательности к вычислительным ресурсам выполняются на суперкомпьютере. При использовании суперкомпьютерного вычислительного кластера, содержащего большое количество узлов, возникает проблема эффективного масштабирования выполняемых запусков. В статье описан алгоритм распределения вычислительной нагрузки между узлами суперкомпьютера с применением измельчения блочно-структурированной расчетной сетки и приведены результаты по масштабированию расчетов на суперкомпьютере МСЦ РАН.

Ключевые слова: турбулентность, высокоскоростное воздухозаборное устройство, ваихреразрешающие методы, численные методы, RANS/ILES, суперкомпьютер, масштабирование, распределение вычислительной нагрузки.

Введение

Возросший интерес к гиперзвуковым летательным аппаратам (ГПЛА) стимулирует исследование обтекания как аппаратов целиком, так и течений в отдельных их элементах. Для ГПЛА с воздушно-реактивным двигателем обязательным условием является надежная работа воздухозаборного устройства (ВЗУ), что обеспечивает надежную работу силовой установки в целом. Для этого необходимо, чтобы течение на выходе из ВЗУ было близким к стационарному, а потери полного давления на торможение в скачках уплотнения – минимальны. Кроме того, практически важным является определение границ устойчивой работы ВЗУ: границы начала срыва потока, который может наступать при изменении режима работы двигателя или условий полета.

Применение экспериментальных методов [1, 2] ограничено в силу их высокой стоимости и сложности создания реальных условий полета в аэродинамических трубах. Это стимулирует использование численного моделирования течений в гиперзвуковых ВЗУ. Наиболее распространенным является применение осредненных по Рейнольдсу уравнений Навье-Стокса (RANS) с моделями турбулентности.

В тех случаях, когда наблюдаются нестационарные явления, использование методов RANS не столь успешно, особенно в случаях, когда нестационарные явления обусловлены турбулентными эффектами, такими, как отрыв пограничного слоя при взаимодействии скачка уплотнения, отрывные течения в диффузорах, а также срыв и незапуск ВЗУ. В этом случае следует использовать методы, в которых турбулентные вихри разрешаются явным образом: DNS, LES или комбинированные RANS/LES. Последние уже успешно применяются для расчетов ВЗУ [3].

Применение вихреразрешающих методов требует применения разностных схем для конвективных членов уравнений Навье-Стокса, имеющих низкий уровень схемной вязкости. Как правило, для вихреразрешающих методов используются схемы с высоким порядком аппроксимации параметров на гранях ячеек (выше третьего), что приводит к увеличению объема передаваемых данных между вычислительными процессами.

Кроме того, вихреразрешающие методы требуют использования сеток с существенно большим количеством ячеек, чем методы RANS для явного разрешения турбулентных вихрей.

Следует также помнить, что турбулентные течения нестационарны, и для их описания также необходимо решать численно нестационарные уравнения.

При этом шаг по времени должен быть достаточно мал, чтобы описать движение всех разрешенных явным образом турбулентных вихрей, а время интегрирования достаточным для того чтобы получить с требуемой точностью осредненные параметры течения и турбулентности.

Из сказанного выше следует, что для эффективного численного решения задач такого класса требуется применение суперкомпьютеров. Использование суперкомпьютеров в расчетах подразумевает запуск задачи сразу в большом количестве параллельных процессов, которые обмениваются между собой данными [4].

При возрастании количества процессов возникают дополнительные требования к характеристикам расчетных сеток в плане равномерности распределения вычислитель-ной нагрузки между процессами. Для того, чтобы выполнить эти требования нужно перестраивать расчетную сетку в автоматическом режиме.

Механизмы равномерного распределения вычислительной нагрузки между процессами суперкомпьютера являются важной частью подготовки к вычислениям.

К ним относится измельчение блоков сетки и распределение результирующих блоков между различными процессами.

Целью настоящей работы является анализ эффективности масштабирования при расчетах турбулентных течений в высокоскоростных ВЗУ на суперкомпьютере RANS/ILES методом высокого разрешения при использовании разработанных алгоритмов подготовки расчетной сетки.

Описание RANS/ILES метода

Уравнения Навье-Стокса, описывающие течения сжимаемого газа, и уравнение переноса записаны в консервативной форме для криволинейной системы координат, сеточные линии которой совпадали с границами расчетной области и поверхностью исследуемого тела. Для их решения был использован комбинированный RANS/ILES-метод [5].

Около стенок для расчета течения решались нестационарные уравнения Навье−Стокса с моделью турбулентности Спаларта−Аллмараса. Вдали от стенок течение описывалось с помощью LES с неявной SGS-моделью − ILES.

При таком подходе отсутствует явная SGS-модель турбулентности, а ее функцию выполняет схемная вязкость разностной схемы.

Для расчета конвективных потоков на гранях расчетных ячеек была использована схема Роу, предраспадные параметры для которой вычислялись с помощью сохраняющей монотонность противопоточ-ной схемы 9-го порядка MP9.

Для реализации данной схемы необходимо знать значения параметров течения из 5 соседних ячеек с каждой стороны рассматриваемой грани. Диффузионные потоки на гранях ячеек определялись с помощью аппроксимации с центральными разностями со вторым порядком.

Интегрирование по времени выполнялось с помощью технологии «dual time stepping» – интегрирование по двойному времени. При таком подходе на каждом шаге по времени решение находится с помощью неявного метода установления по параметру, «искусственному времени».

Интегрирование по физическому времени выполнялось со вторым порядком точности.

В области около стенок, где течение описывается с помощью RANS, конвективные потоки на гранях расчетных ячеек в разностном аналоге уравнения для модели турбулентности вычислялись с помощью схемы WENO5. В области ILES модель турбулентности Спаларта–Аллмараса изменяется таким образом, чтобы турбулентная вязкость равнялась нулю [5].

Данный метод успешно применялся для расчета сверхзвуковых течений в ВЗУ ранее [3].

Постановка задачи газовой динамики

В качестве тестовой задачи было выбрано модельное ВЗУ (ВЗ) [2] высокоскоростного летательного аппарата. Его геометрия представлена на Рис. 1, ВЗУ с прямоугольным поперечным сечением шириной 54 мм с боковыми щеками, идущими от передней кромки. Торможение потока осуществлялось в двух скачках уплотнения, которые формировались изломами на нижней поверхности торможения ВЗУ.

Режимные параметры потока соответствовали числу Маха 5.9. Полное давление и температура перед ВЗУ 1.27 МПа и 810 К. Статическое давление, соответствующее режимным параметрам, ($p_{\inf}$) равно 891 Па. Число Рейнольдса, вычисленное по единице длины равно 5.2×10^6 м$^{-1}$.

В расчете число Рейнольдса, вычисленное по высоте входа в ВЗ, к которой были отнесены линейные размеры, составляло 3.12×10^5.

Эти параметры соответствовали условиям эксперимента [2]. На выходе из канала ВЗУ устанавливались разные по высоте вставки, моделирующие разную степень дросселирования. Степень дросселирования вычислялась по формуле:

$$TR = 1 - \frac{A_{t,plug}}{A_{isolator}}$$

где $A_{t,plug}$ – площадь горла образованного вставкой, $A_{isolator}$ – площадь проходной части изолятора ВЗ. Значение параметра TR, равное 0, соответствует отсутствию вставки в канале ВЗ

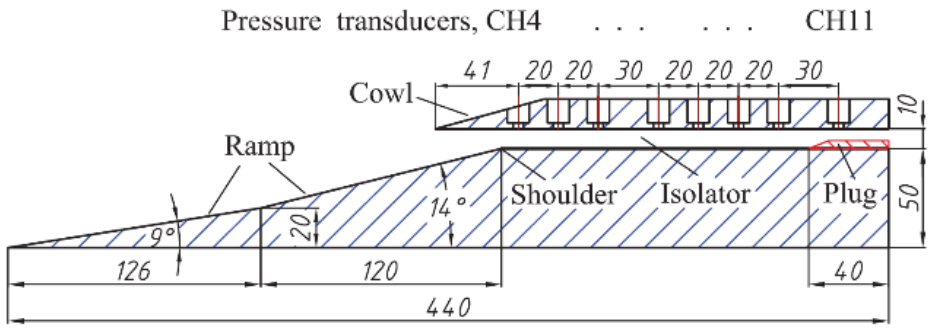


Рис. 1. Геометрия модельного ВЗУ из работы [2].

Были использованы следующие граничные условия. На входной сверхзвуковой границе, через которую поток втекает в расчетную область, фиксировались значения всех параметров течения, на выходе из ВЗУ было задано постоянное давление 50 КПа, что соответствовало условиям эксперимента [2].

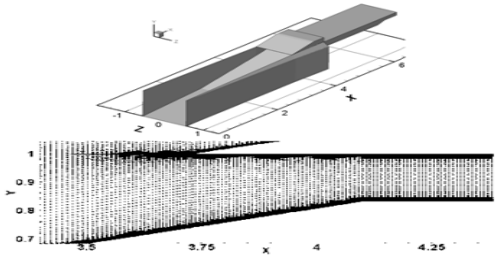


Рис. 2. Расчетная геометрия ВЗУ и фрагмент расчетной сетки.

На внешней выходной границе задавалось граничное условие сверхзвукового выхода, значения всех параметров сносились изнутри расчетной области. На твердых стенках использовалось комбинированное граничное прилипание «закон стенки». Тип условия выбирался в зависимости от значения Y^+ в центре ближайшей к стенке ячейки. Расчетная геометрия воздухозаборного устройства и фрагмент расчетной сетки показан на Рис 2.

Ниже представлены результаты расчета для расчетной сетки, состоящей из 4.13×10^6 ячеек. На Рис. 3 показано сравнение поля течения в расчете и в эксперименте для режима $TR = 0$. Видно, что течение в расчете и в эксперименте имеет схожую структуру. Количественное сравнение по

распределению давления на верхней стенке ВЗУ (Рис. 4), также показывает хорошее соответствие результатов расчета с экспериментом.

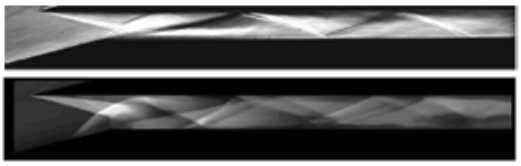


Рис. 3. Сверху поле градиента плотности (эксперимент [2]), снизу поле плотности расчет. Режим *TR*=0.

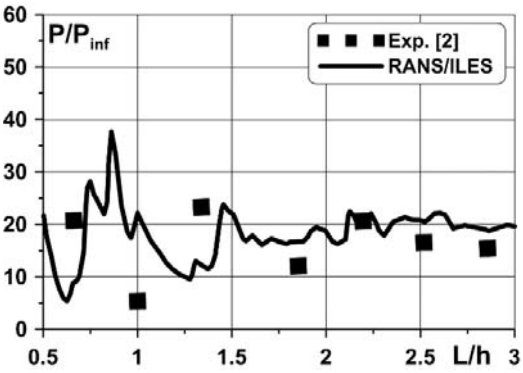


Рис. 4 Распределение давления на верхней стенке ВЗУ ($TR = 0$).

Возможность описания настоящим методом потери устойчивости течения в ВЗУ при увеличении дросселирования показана на Рис5, где представлены полученные в эксперименте [2] и при расчетах настоящим методом зависимости частоты помпажа от степени дросселирования ВЗУ.

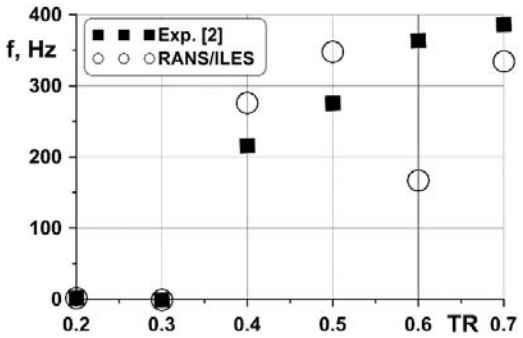


Рис. 5. Зависимость от степени дросселирования частоты помпажных колебаний.

В расчете частота помпажа определялась на основе обработки осциллограмм давления для значений TR от 0 до 0.7. Видно, что для всех исследованных случаев кроме $TR = 0.6$ частота колебаний, полученная в расчетах, хорошо совпадает с данными эксперимента [2]. Причем помпажные колебания течения в ВЗУ в эксперименте и в расчете начинаются при одинаковом TR, что очень важно для практических приложений.

Полученные результаты говорят о возможности использования данного метода для расчетов характеристик течения, определения границ устойчивой работы ВЗУ для ГПЛА.

Представление расчетной сетки

В данной работе в расчетах используется блочно-структурированная сетка [6], основным объектом которой является блок. Блок состоит из упорядоченного трехмерного массива ячеек, содержащих газодинамические параметры, ассоциированные с центрами масс ячеек. Упорядоченность размещения ячеек позволяет быстрее производить вычисления и снижает требования к объему оперативной памяти, однако строить блочно-структурированные сетки гораздо сложнее, чем неструктурированные. С каждым блоком сетки ассоциирована криволинейная система координат, которая задается тремя линиями координат в пространстве индексов (I, J, K), каждая из которых связывает пары противоположных граней блока.

Объектом, описывающим соприкосновение двух соседних блоков, является интерфейс. Интерфейс является однонаправленным, он сообщает, что у данного блока конкретная прямоугольная часть границы соприкасается с другим блоком. Чтобы определить, с какой частью другого блока граничит рассматриваемый блок, нужно рассмотреть смежный ему интерфейс. Таким образом, полная информация о касании двух соседних блоков описывается парой смежных интерфейсов.

Во время проведения расчетов различные блоки сетки обрабатываются независимо друг от друга. Однако некоторые ячейки обрабатываемого блока должны обращаться за данными к ячейкам соседних блоков, с которыми этот блок граничит через интерфейсы. Если два соседних блока обрабатываются на разных узлах

суперкомпьютера, то для обмена данными между ними можно использовать MPI-обмены. Если ячейка в процессе проведения вычислений обращается за данными на другой вычислительный узел, то будем называть ее кросс-ячейкой.

Одним из важнейших действий по управлению расчетной сеткой при расчетах на суперкомпьютере является дробление ее блоков. В связи с тем, что при запуске задач на суперкомпьютере постоянно возрастает степень параллельности (используется все больше параллельных процессов обработки блоков сетки), то для сохранения равномерности распределения блоков по вычислительным процессам требуется уметь измельчать блоки. Кроме блоков сетка содержит другие объекты, которые требуют корректировки после разделения блока, например, это интерфейсы и граничные условия. Каждый их этих объектов имеет жесткую привязку к блоку, а значит после дробления блока может возникнуть потребность его разделения. Механизм дробления блоков расчетной сетки позволяет равномерно загружать вычислительные ресурсы суперкомпьютера, что способствует эффективному масштабированию расчетной задачи при увеличении количества вычислительных узлов.

Распределение нагрузки между вычислительными узлами

Рассмотрим простой жадный алгоритм равномерного распределения блоков расчетной сетки по вычислительным узлам суперкомпьютера. Весом блока будем называть количество его ячеек, весом вычислительного узла - сумму весов сопоставленных с ним блоков. Тогда задача равномерного распределения блоков по узлам может ставиться в следующим виде: распределить блоки расчетной сетки по вычислительным узлам таким образом, чтобы вес наиболее тяжелого узла был минимален. Заметим, что точное решение данной задачи связано с большой комбинаторной сложностью, поэтому будем искать приближенное решение, не сильно отклоняющееся от оптимального.

Жадный алгоритм для достижения данной цели можно сформулировать следующим образом. В начале работы алгоритма блоки сетки не приписаны никаким вычислительным узлам. Пока есть блоки сетки, не приписанные никаким вычислительным узлам, нужно взять из них самый тяжелый блок и отнести его к самому легкому вычислительному узлу, после чего повторить данную процедуру необходимое число раз. Данный алгоритм всегда завершает работу и показывает хорошие результаты на сетках с большим количеством блоков, среди которых отсутствуют ярко выраженные крупные блоки. Однако, при наличии крупных блоков требуется использовать их дробление. В этом случае предлагается использовать следующую модификацию данного алгоритма. Сначала требуется установить предельное допустимое отклонение веса наиболее загруженного вычислительного узла от среднего значения, при достижении которого алгоритм завершает работу. Такое отклонение в дальнейшем будем обозначать dev. Применить жадный алгоритм распределения весов блоков по вычислительным узлам. Если требуемое отклонение наибольшего веса вычислительного узла от среднего значения достигнуто, то завершить работу. В противном случае разделить максимальный блок пополам по наиболее протяженному направлению, после чего произвести перераспределение.

Такой модифицированный жадный алгоритм с дроблениями пополам будем обозначать UG (от uniform greedy). Данный алгоритм также всегда завершает работу, однако в отдельных случаях может выполнять необоснованно большое количество дроблений блоков, что приводит к возрастанию количества кросс-ячеек, а значит тормозит межпроцессные обмены между блоками сетки. Для устранения крупных блоков без лишних дроблений предлагается механизм минимизации количества разрезов блоков.

Прежде всего определим, на какие части может быть разрезан конкретный блок, имеющий размеры ISize, JSize, KSize и содержащий соответственно ISize * JSize * KSize ячеек.

На возможные разрезы блока накладываются следующие ограничения. Так как блок может быть разрезан только по границам ячеек, то размеры получившихся новых блоков будут кратны одному из значений ISize * JSize, ISize * KSize, или JSize * KSize. На самом деле целесообразно выполнять разрезы только по наиболее протяженному направлению (пусть это будет ISize, для каждого блока это направление будет свое), так как это приводит к минимизации количества кросс-ячеек. Для сохранения точности расчетов запрещается выполнять разрезы блока слишком близко к границе. Для этого вводится специальный параметр margin, по которому разрешается деление блока только в сегменте [margin, ISize - margin]. Вводится еще одно ограничение, не позволяющее производить слишком мелкие блоки (задается наименьшее допустимое количество ячеек в результирующем блоке, при котором разрешено дробление).

Алгоритм распределения блоков по вычислительным узлам с минимизацией количества разрезов (в дальнейшем будем обозначать его MCC, от minimal cuts count) будем описывать в следующем виде:

1. Определить среднее ожидаемое количество ячеек, которое должно приходиться на один вычислительный узел, будем обозначать эту величину mid. Она равна общему количеству ячеек сетки, деленному на количество вычислительных узлов proc.

2. Определить максимально допустимый вес вычислительно узла на текущий момент (будем обозначать через max). Изначально max берется равным mid. Однако если в процессе распределения вес какого-либо вычислительного узла превысил mid, то величина max принимает значение веса этого узла. Таким образом max определяется как максимум из величины mid и весов всех вычислительных узлов.

3. Определить множество всех блоков, которые еще не распределены ни на один вычислительный узел. Если данное множество пусто, то алгоритм заканчивает работу.

4. Попробовать найти из рассматриваемого множества блоков такой блок, который можно распределить на один из вычислительных узлов так, чтобы вес этого узла не превысил max. Если таких блоков несколько, то нужно взять такой блок, который максимально приближает вес соответствующего вычислительного узла к отметке max. Если это удалось, то распределить найденный блок на вычислительный узел и перейти к п. 3.

5. Определить множество допустимых разрезов всех не распределенных на текущий момент блоков. Каждый потенциальный разрез делит блок на две части. Определить множество таких потенциальных результирующих блоков и из этих потенциальных блоков выбрать блок веса w и вычислительный узел веса W такие, что $W + w \le max$ и величина $max - (W + w)$ минимальна. Если такой пары блок-узел не найдено, то найти такую пару, что $W + w > max$ величина $(W + w) - max$ минимальна. Такая пара найдется всегда. После этого выполнить необходимый разрез для получения найденного блока и распределить его на соответствующий вычислительный узел, после чего перейти к п. 2.

Кроме описанных действий данный алгоритм содержит ряд эвристик, не позволяющих проявляться негативным эффектам при возникновении сложных краевых случаев (например, неконтролируемый рост величины max). Таким образом, описанный алгоритм на каждом шаге пытается выполнить такой разрез блока, чтобы максимально приблизить вес некоторого вычислительного узла к ожидаемой в среднем величине загрузки.

Для тестирования и оценки эффективности описанных выше методов распределения блоков блочно-структурированной сетки между узлами суперкомпьютерного кластера использовались три разные сетки, отличающиеся по количеству блоков: test (13 блоков, 5.8 млн. ячеек), train (30 блоков, 10.7 млн. ячеек), ref (136 блоков, 8.5 млн. ячеек).

В качестве целевого вычислительного ресурса брался гомогенный вычислительный кластер, состоящий из 64 узлов. Величина margin, характеризующая минимальное расстояние разреза от границы блока,

бралась равной 5. На Рис. 6 представлены данные применения алгоритмов UG и MCC к каждой из приведенных расчетных сеток. Описание данных на графиках: mid – средний ожидаемый вес вычислительного узла, dev – отклонение веса наиболее тяжелого узла от dev, proc – количество вычислительных узлов, cuts – количество выполненных разрезов, iface cells и cross cells – доля интерфейсных и кросс-ячеек среди всех ячеек сетки.

На каждом графике на Рис. 6, представлена гистограмма распределения блоков расчетной сетки по вычислительным узлам. Каждый столбец гистограммы соответствует одному вычислительному узлу. Высота столбца – вес соответствующего вычислительного узла. Если к вычислительному узлу отнесены несколько блоков расчетной сетки, то соответствующий столбец гистограммы разделен на несколько частей, размеры которых отражают веса блоков, также эти части раскрашены в шахматном порядке для повышения наглядности.

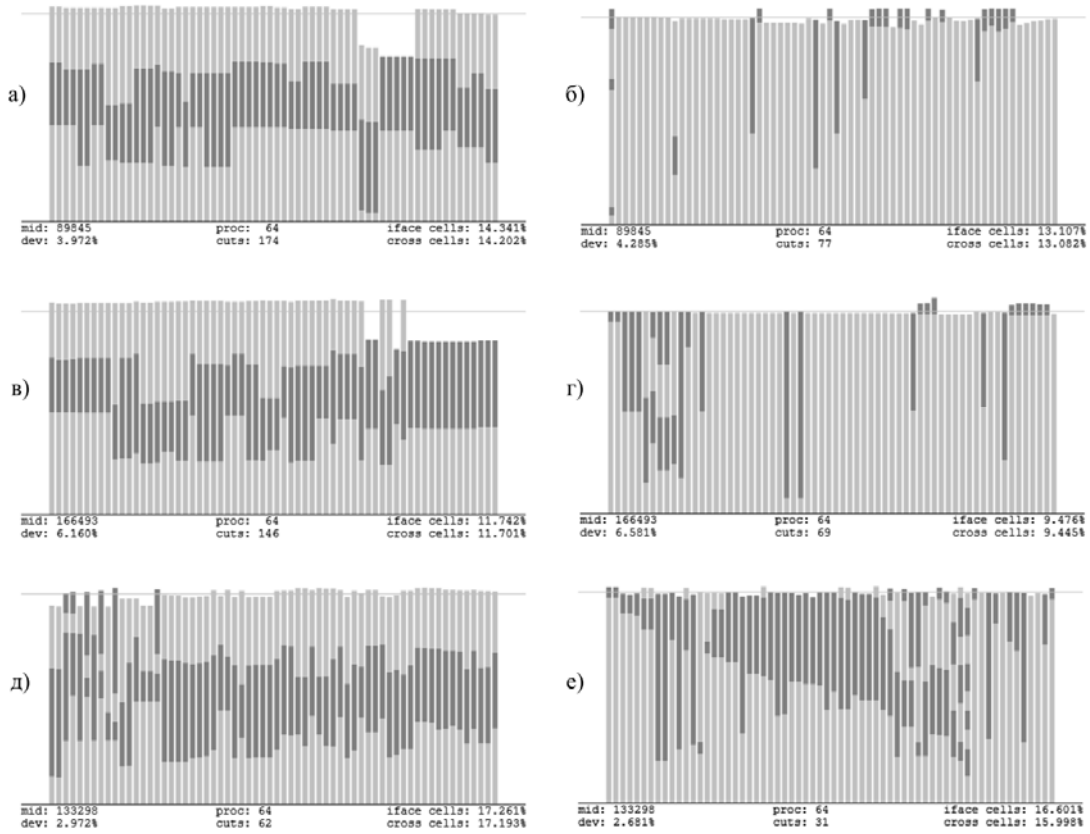


Рис. 6. Гистограммы распределения блоков расчетной сетки по вычислительным узлам суперкомпьютера для сеток test (а, б), train (в, г), ref (д, е) с помощью алгоритмов UG (а, в, д) и MCC (б, г. е).

Результаты сравнения эффективности методов UG и MCC распределения вычислительной нагрузки между узлами суперкомпьютера показывают, что использование метода MCC оправдано, так как с его помощью можно добиться распределения не худшего качества (а зачастую и лучшего), чем при использовании UG.

При этом MCC позволяет существенно сократить количество разрезов блоков сетки для достижения требуемого результата.

Также использование MCC приводит к сокращению количества кросс-ячеек в сетке, что положительно сказывается на скорости межпроцессных обменов данными.

Особенно явно достоинства метода MCC проявляются на сетках с относительно небольшим количеством блоков и наличием ярко выраженных крупных блоков.

Проведение расчетов на суперкомпьютере

Для численного исследования было выбрано модельное плоское двухскачковое ВЗУ высокоскоростного летательного аппарата, описанное в начале статьи. Для данного объекта проводились численные расчеты на суперкомпьютере с использованием RANS/ILES метода. Для проведения численных расчетов использовалась блочно-структурированная расчетная сетка, содержащая 172 блока, 848 интерфейсов, 323 граничных условия, 12.8 миллионов ячеек. Для распределения вычислительной нагрузки для данной сетки использовался алгоритм MCC с дроблением блоков.

В качестве вычислительного поля использовались узлы суперкомпьютера МВС-10П, находящегося в МСЦ РАН, каждый вычислительный узел содержит по 2 микропроцессора Intel Xeon E5-2697v3 (Haswell) [7].

Были выполнены запуски расчетов в двух конфигурациях: с запуском двух MPI процессов на каждом процессоре и с запуском четырех MPI процессов на каждом процессоре. Количество вычислительных узлов варьировалось от 1 до 18.

В качестве эталонного запуска, относительно которого считались ускорения, был выбран запуск на одном узле с двумя MPI процессами на каждом процессоре.

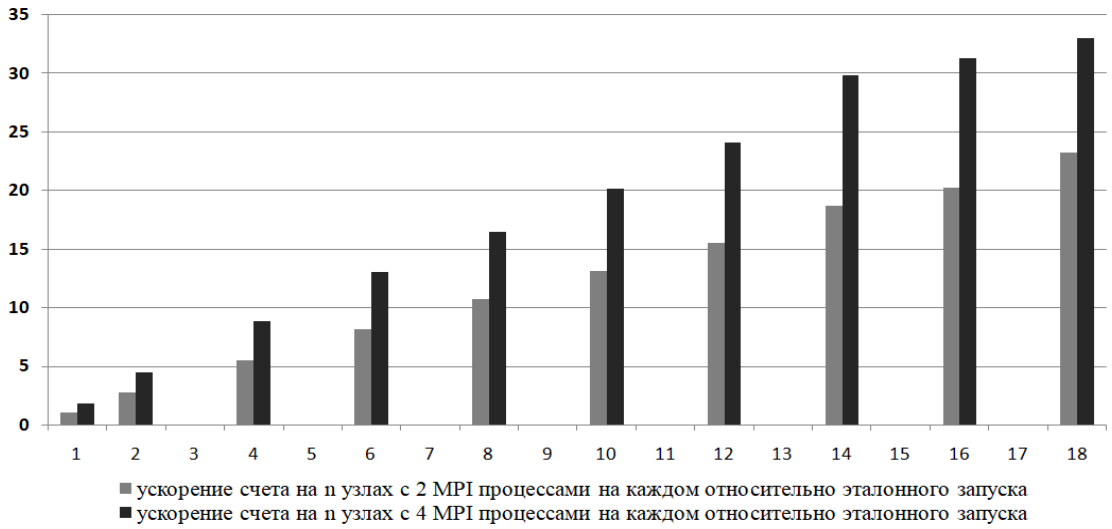


Рис. 7. Ускорение запусков на различном числе узлов относительно эталонного запуска при проведении газодинамических расчетов на модельном входном устройстве воздухозаборника на суперкомпьютере с использованием RANS/ILESметода.

Из данных, приведенных на Рис. 7, можно видеть сверхлинейное ускорение при увеличении количества расчетных узлов. Так, при использовании 18 узлов с двумя MPI процессами на каждом процессоре было достигнуто 23-хкратное ускорение относительно эталонного запуска. Такое сверхлинейное ускорение объясняется как равномерностью распределения вычислительной нагрузки между узлами суперкомпьютерного кластера, так и снижением интенсивности и повышением локальности обращений в память при увеличении количества вычислительных узлов. При использовании 4 MPI процессов на каждом процессоре эффективность счета увеличивается, при этом линейное масштабирование сохраняется.

Заключение

Показано, что применение RANS/ILES-метода высокого разрешения к расчету высокоскоростного воздухозаборного устройства позволяет определить характеристики течения и границы его устойчивой работы.

Организация параллельных вычислений в расчетных кодах RANS/ILES, а также использование специальных подходов в подготовке расчетной сетки, выраженной в дроблении ее блоков и распределении

вычислительной нагрузки между узлами суперкомпьютера, позволило достичь сверхлинейной масштабируемости расчетов задач газовой динамики на вычислительном кластере МСЦ РАН. Так при увеличении количества расчетных узлов до 18 штук было достигнуто ускорение расчетов в 23 раза.

Развитие методов повышения эффективности использования суперкомпьютеров при численном решении задач газовой динамики является актуальной задачей, так как при верной подготовке вычислений может быть достигнуто линейное и даже сверхлинейное ускорение на достаточно большом количестве узлов вычислительного кластера.

Это в свою очередь приводит к ускорению исследования и проектирования сверхзвуковых летательных аппаратов.

Работа выполнена в МСЦ РАН при поддержке программы фундаментальных исследований Президиума РАН «Фундаментальные основы технологий двойного назначения в интересах национальной безопасности».

Scaling of fluid dynamic calculations using the RANS/ILES method on supercomputer

L.A. Benderskiy, D.A. Lyubimov, A.A. Rybakov

Abstract. The actual problem of calculating the characteristics of a nonstationary turbulent flow inside a high-speed aircraft inlet is considered. Such calculations are usually performed on a supercomputer. Using a supercomputer cluster containing a large number of nodes leads to the problem of effectively scaling the runs performed. The article describes the algorithm for distributing the computational load between the nodes of a supercomputer using the block-structured grid cutting and presents the results of scaling calculations on the JSCC RAS supercomputer.

Keywords. Fluid dynamics, numerical methods, RANS/ILES, supercomputer, scaling, computational workload distribution.

Литература

1. H.-J. Tan, S. Sun, Z.-L. Yin. Oscillatory flows of rectangular hypersonic inlet unstart caused by downstream mass-flow choking. Journal of Propulsion and Power.2009, V. 25, № 1, P. 138-147.
2. Z. Li, W. Gao, H. Jiang, J. Yang. Unsteady behaviors of a hypersonic inlet caused by throttling in shock tunnel. AIAA J. 2013. V. 51, No. 10. P. 2485- 2492.
3. Д.А.Любимов, И.В.Потехина. Исследование нестационарных режимов работы сверхзвукового воздухозаборника RANS/ILES-методом // ТВТ. 2016, Т. 54, № 5, С. 784–791.
4. О.С. Аладышев, Н.И. Дикарев, А.П. Овсянников, П.Н. Телегин, Б.М. Шабанов. СуперЭВМ: области применения и требования к производительности. Известия высших учебных заведений. Электроника. 2004, № 1, С. 13-17.
5. Д. А. Любимов. Разработка и применение метода высокого разрешения для расчета струйных течений методом моделирования крупных вихрей. ТВТ. 2012, Т. 50, № 3, С. 450-466.
6. А.А. Рыбаков. Внутреннее представление и механизм межпроцессного обмена для блочно-структурированной сетки при выполнении расчетов на суперкомпьютере. Программные системы: Теория и приложения, №1 (32), 2017, С. 121-134.
7. Описание интерфейса пользователя, предназначенного для работы с интеловской гибридной архитектурой суперЭВМ (СК), где вместе с процессорами Intel Xeon используются сопроцессоры Intel Xeon Phi. URL: http://www.jscc.ru/informat/MVS-10PInter.pdf.

Численные исследования эффектов нагрева дутья в коксовой плавильной печи для минерального сырья

П.А.Войнович[1], Ю.А.Куракин [2]

[1] ФГУ ФНЦ НИИСИ РАН, СПбО Межведомственного суперкомпьютерного центра, Санкт-Петербург, Россия, [2] ФГУ ФНЦ НИИСИ РАН, СПбО Межведомственного суперкомпьютерного центра и Физико-технический институт им. А.Ф.Иоффе РАН, Санкт-Петербург, Россия, E-mail: [1,2] yurii.kurakin@mail.ioffe.ru

Аннотация: Проведены численные исследования процессов тепломассообмена в коксовой плавильной печи-вагранке непрерывного действия для минерального сырья, используемой при производстве базальтовой ваты. Показана возможность устойчивой работы печи с пониженным содержанием кокса в шихте при условии подогрева дутьевого воздуха. Дано обоснование установки на выходе печи дожигателя угарного газа, позволяющего подогревать дутьевой воздух без дополнительных затрат энергии, с целью повышения КПД плавильной печи и экономии кокса.

Ключевые слова: численное моделирование, плавильная печь для минерального сырья, горячее дутьё

Введение

В качестве источника расплава базальта при производстве минеральной ваты широко используются шахтные печи-вагранки непрерывного действия. Такая печь загружается смесью кокса и минеральных материалов (базальта, доломита и т.п.), а снизу через фурмы подаётся воздух. За счет горения кокса в печи происходит плавление и термическое разложение минерального сырья. Образующийся расплав стекает вниз и скапливается в донной области печи, откуда забирается для производства базальтового волокна.

В области фурм происходит горение кокса с образованием углекислого газа, который, поднимаясь через верхние слои шихты, частично восстанавливается до угарного газа в реакции Будара, что сопровождается поглощением тепла и уносом части кокса. Образование угарного газа является основной причиной снижения энергоотдачи кокса в печах-вагранках. Если в металлургических печах угарный газ участвует в полезных реакциях восстановления железа, в плавильных печах из-за этого падает КПД и растёт расход дорогостоящего кокса.

Одним из способов вернуть уходящее тепло является установка на выходе печи дожигателя, в котором полученная при сгорании угарного газа энергия расходуется на нагрев идущего в печь дутьевого воздуха.

Нагрев дутья давно используется для металлургических печей [1,2]. Сегодня системы *дожигатель-горячее дутьё* активно внедряются и в плавильных печах.

Изготовление и установка дожигательной системы является дорогостоящим мероприятием и требует предварительных изысканий. Большая часть опытных работ сегодня проводится в отраслевых лабораториях крупных зарубежных фирм, лидером среди которых является ROCKWOOL [3], где списанные печи используют для натурных экспериментов. В таких условиях численный эксперимент является относительно дешевым и эффективным инструментом исследований.

Объектом исследований в настоящей работе является плавильная печь установленная на предприятии "Изомин" в Московской области.

Постановка задачи

Плавильная печь представляет собой вертикальную трубу переменного сечения с относительно небольшим наклоном стенок (Рис.1). Характерный радиус ~ 1 м и высота печи ~ 4 м. Нижняя часть печи - закрытая, во время работы там содержится расплав базальта, уровень которого определяется положением сливного отверстия. Несколько выше расплава расположены фурмы.

Печь заполнена смесью кокса с сырьём (базальт + доломит). При выработке твердой фазы внутри печи она досыпается сверху коксом и сырьём в заданном соотношении.

Воздух вдувается через фурмы, в которых создается избыточное к атмосферному давление, и поднимается затем вверх через поры шихты.

Стенки печи охлаждаются водой при температуре близкой к точке кипения 100°C.

Задача считается осесимметричной.

Математическая модель и численные методы

Математическая модель основана на гибридном Лагранжево-Эйлеровском описании.

Движение газа через шихту рассматривается как течение сплошной среды через пористое тело и подчиняется закону Дарси [4,5]. Модель записана для неподвижной Эйлеровой сетки и состоит из уравнений неразрывности многокомпонентного газа и уравнения энергии:

$$\nabla C_c \rho \vec{u} = Q_c ,$$
$$\nabla \rho \vec{u} = Q_m$$
$$\nabla \left[u(\rho e + p) \right] = Q_e ,$$

$$c = \{N_2, O_2, CO, CO_2, H_2O, H_2\}$$

система замыкается уравнением движения в форме закона Дарси

$$-\nabla p = 1.75 \frac{\rho(1-\Omega)}{\Omega^3 d_{eq}} |u| \cdot \vec{u} + 150 \frac{\mu(1-\Omega)^2}{\Omega^3 d^2_{eq}} \vec{u}$$

где: ρ, u, p, C - плотность, скорость, давление и концентрация компонентов газа, Q_c, Q_m, Q_e - источниковые члены, связанные с химическими реакциями, межфазным массо- и теплообменом, радиационный теплообмен, Ω - порозность шихты. Эквивалентный диаметр d_{eq} есть отношение объема пор (порозности) к поверхности пор, которая рассчитывается исходя из локального состава шихты как сумма поверхностей частиц $d_{eq} = 4V_{pore} / S_{pore}$.

Изменение размера и температуры частиц описывается в рамках дискретной модели, когда рассматривается динамика отдельной частицы – представителя группы частиц со сходными параметрами. Характеристиками частицы являются масса m_p и температура T_p.

При расчете движения шихта рассматривается как сплошная среда. Уравнения движения решаются на Лагранжевой сетке, двигающейся вместе со средой.

Шихта движется под действием силы тяжести, согласно кинематической модели [6], горизонтальное смещение определяется скоростью вертикального смещения:

$$u_r = -2.5 \cdot d_p \cdot \frac{\partial u_z}{\partial r}$$

где d_p - средний размер частиц, r, z - радиальная и вертикальная оси.

В каждой точке пространства определяется порозность и рассчитывается вертикальное смещение. Экспериментальные значения для порозности в условиях доменной плавки лежат в диапазоне 0.3 – 0.4 [4], а среднее значение можно положить $\Omega = 0.37$.

При расчёте скорости горения кокса использована диффузионно-кинетическая модель [7], в рамках которой действительная скорость реакции горения ограничена скоростями самой химической реакции на поверхности и скоростью диффузии газовых компонентов к поверхности.

Схема химических реакций такая:

$$C(s) + O_2 \rightarrow CO_2$$
$$2C(s) + O_2 \rightarrow 2CO$$
$$C(s) + CO_2 \rightarrow 2CO$$
$$C(s) + H_2O \rightarrow CO + H_2$$
$$2CO + O_2 \rightarrow 2CO_2$$
$$2H_2 + O_2 \rightarrow 2H_2O$$
$$CaMg(CO_3)_2 \rightarrow CaCO_3 + MgO + CO_2$$
$$CaCO_3 = CaO + CO_2$$

из которых первые четыре проходят на поверхности кокса, следующие две – в объёме газа, последние две связаны с декарбонизацией доломита.

Потери энергии на излучение с единицы поверхности выражается как $q_r = \varepsilon_p \sigma T_p^4$, где $\sigma = 5.67 \cdot 10^{-8}$ Вт/м^2·К^4 - постоянная Стефана-Больцмана, а ε_p - интегральная излучательная способность поверхности частицы. Для кокса и базальта $\varepsilon_p \approx 0.8 \div 0.9$, для газа $\varepsilon_g = 0.01$, для расплава $\varepsilon_l = 0.85$.

Аппроксимации для расчёта контактного теплообмена между газом и твёрдой фазой можно найти в [8]

В тепловые и материальные балансы включены также процесс сушки шихты, поступающей в печь с некоторой начальной влажностью, декарбонизация доломита и плавление базальта. Полагается, что эти процессы идут при постоянной температуре,

а их скорость определяется количеством подводимого к частице тепла.

Расчетная область ограничена стенками печи, поверхностью расплава и верхним уровнем шихты. На боковых стенках печи и нижней границе задаётся температура и ставятся условия непроницаемости.

Теплообмен с границами рассчитывается так же как и в твердой фазе внутри печи. Температура, состав и расход дутьевого воздуха через фурмы заданы. На оси печи ставятся условия симметрии. На верхней границе задаётся давление, равное атмосферному. Шихта, поступающая через верхнюю границу имеет температуру равную температуре выходящих газов.

Для решения уравнений газовой фазы разработан оригинальный итерационный метод для поправок давления. В каждой итерации сначала по известному полю давления определяется скорость на гранях ячеек, затем, для произвольной области суммируются потоки через границу и на основе этой суммы (то есть, дисбаланса потоков массы) определяется поправка для давления Δp :

1. $u^k = f(\nabla p^k)$,
2. $\Delta p^k = F\left(Q_m + \sum \rho_i u_i^k S_i\right)$,
3. $p^{k+1} = p^k + \Delta p^k$

k - номер итерации.

Уравнение на 2-м этапе получено линеаризацией уравнения неразрывности после замены в нём скорости газа на её выражение через градиент давления этапа 1, выведенное из уравнения Дарси.

Основной рабочий режим

Рассмотрим расчет для основного рабочего режима работы печи, используемого на производстве:

Вдув: расход 6500 нм^3/час, температура 200 С, массовая доля кислорода 23%, массовая доля воды 1%

Шихта: начальная температура 23 С; влажность кокса – 3%, влажность базальта – 2%, влажность доломита – 7%; содержание золы в коксе – 13%, содержание кокса в шихте – 20%.

При запуске расчета, печь заполняется чистым коксом примерно на 2/3 объема, выше идет основная шихта с рабочим соотношением кокса и сырья (Рис.1.), и задаётся область повышенной температуры кокса вблизи фурм.

После включения дутья кокс начинает выгорать и верхние слои шихты начинают опускаться вниз образуя в итоге холостую калошу в виде буквы "М" (Рис.1).

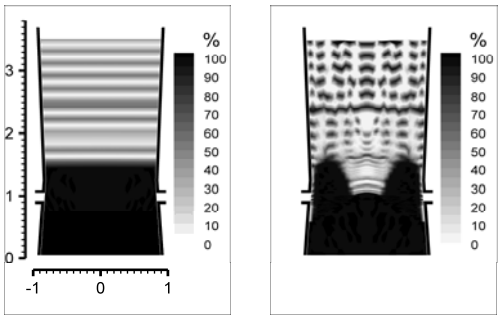


Рис.1. Объёмная доля кокса в печи при содержании кокса в загрузках 20% и температуре дутья 200С. Темные полосы соответствуют слоям кокса. Слева – начальная загрузка печи. Справа – через 4 часа после запуска.

Холостая калоша состоит из чистого кокса. Она формирует область горения вблизи фурм и является опорой для верхних слоёв шихты, не дающей твердым частицам базальта и доломита попасть в фурменный очаг или расплав в донной области печи.

После формирования холостой калоши и разогрева верхних слоёв шихты печь выходит на квазистационарный режим.

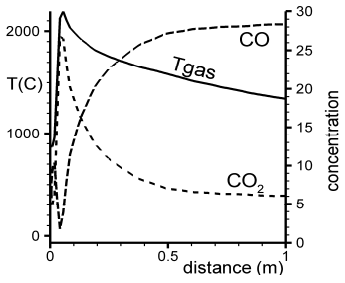


Рис. 2. Распределение температуры и состава газа вдоль линии тока от фурмы. По горизонтальной оси отложена высота от уровня фурм.

Распределение параметров газа по высоте печи показаны на Рис.2, на котором можно выделить типичные для вагранок области: 1) кислородная область, где идёт горение кокса в воздухе около фурм, где температура газа и концентрация CO_2 растут; 2) восстановительная или редукционная область, где происходит восстановление CO_2 до CO с поглощением энергии газа, там температура газа и содержание CO_2 падают, но растёт концентрация CO; и, наконец, 3) верхняя область печи, где температуры недостаточно для восстановительной реакции, концентрации CO_2 и CO почти не меняются, а температура газа падает за счет теплообмена с твердой фазой.

Зона плавления базальта находится сразу над холостой калошей (Рис.1) на расстоянии ~0.3-0.4 м от фурм. Газ попадает в зону плавления пройдя через слой чистого кокса холостой калоши и заранее потеряв часть

температуры. Восстановление CO_2 продолжается и в зоне плавления базальта.

В случае рабочего режима наблюдается некоторый избыток кокса в холостой калоше.

Соотношение компонентов шихты таково, что тепловой эффект сгорания поступающего кокса до CO_2 превышает энергию, необходимую для плавления поступающего сырья, лишняя энергия уходит на образование CO.

Высота холостой калоши подстраивается под этот избыток. Из этого можно сделать вывод, что содержание CO в отходящих газах будет расти с ростом температуры в зоне плавления и размера холостой калоши за счет интенсификации и увеличения восстановительной области.

Для верификации предложенной модели результаты расчетов сравнивались с доступными измеренными параметрами на производстве. Сравнение представлено в таблице:

	Завод	Расчет
Температура отходящих газов (C)	385	430
Производительность по шихте (т/ч)	6600	6400
Потери в стенки (МВт)	1.44 – 2.16	1.98

Также проводилось сравнение экспериментальными данными [3] (Рис.3).

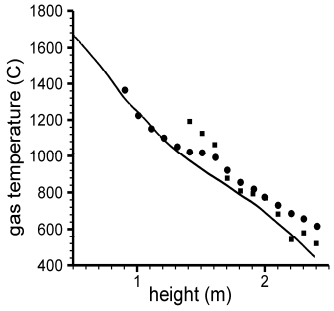


Рис. 3. Распределение температуры газа в около стенки по высоте печи. Сплошная линия – расчет, точки – данные экспериментов [3].

Влияние нагрева дутья и содержания кокса в шихте на работу печи.

Проведена серия расчётов для разных температур дутьевого воздуха и содержания кокса в загрузках.

При снижении доли кокса в загрузках (Рис.4) содержание угарного газа на выходе печи падает.

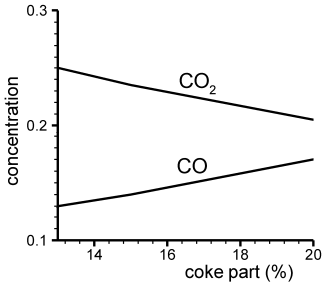


Рис.4. Зависимость концентрации CO и CO_2 в отходящих газах от содержания кокса в засыпках при температуре дутья 200 С.

С одной стороны, это означает увеличение КПД печи. С другой стороны, это может свидетельствовать о падении температуры в зоне плавления, то есть уменьшении производительности по расплаву, а также, о снижении уровня холостой калоши.

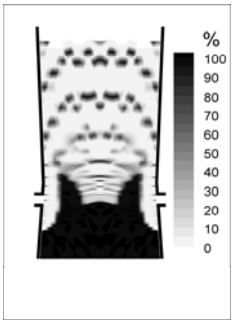


Рис.5. Объёмная доля кокса через 4 часа после запуска печи при содержании кокса в загрузках 10% и температуре дутья 200 С.

Вырождение холостой калоши хорошо иллюстрирует Рис.5, где приводятся результаты расчётов при экстремально низком содержании кокса в шихте. В этом случае в область фурм начинает поступать базальтовая фракция, что может в итоге приводит к остановке горения, и выходу из строя печи.

На производстве такая ситуация расценивается как катастрофичная, поэтому в ущерб экономии работают с избыточной долей кокса в загрузках.

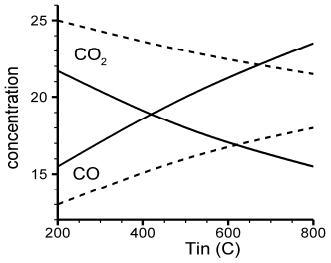


Рис.6. Зависимость концентраций CO и CO_2 в отходящих газах от температуры дутья при содержании кокса 20% (сплошные линии) и 13% (пунктирная линия)

На Рис.6 можно видеть, что при увеличении температуры дутья происходит рост содержания CO, поскольку в печь поступает дополнительная энергия. Растёт также производительность печи по расплаву, поскольку растёт температура в зоне плавления.

При температуре дутья 800 С и содержании кокса 13% концентрация угарного газа даже выше чем при холодном дутье с содержанием кокса 20%.

Состояние холостой калоши для этого режима показано на Рис 7. Если сравнивать Рис. 7 и 1 видно, что уровень холостой калоши понизился, однако области около фурм заполнены чистым коксом.

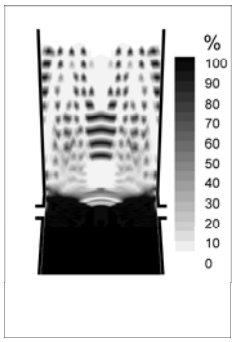


Рис.7. Объёмная доля кокса при содержании кокса в загрузках 13% и температуре дутья 800 С.

Тепловой эффект от дожигания угарного газа для этого режима оценивался исходя из концентрации CO в 16% и составил величину порядка ~5МВт, при необходимой мощности для подогрева дутьевого воздуха от 0 до 800 С порядка ~3МВт.

Заключение

Проведены численные исследования процессов в печи для плавления минерального сырья, которые позволяют говорить о следующих закономерностях.

Если рассмотреть тепловой баланс печи, то есть энергия со знаком плюс, подаваемая в печь в виде кокса, выделяющего тепло при сгорании до углекислого газа, и в виде подогрева дутьевого воздуха. С другой стороны, есть энергия со знаком минус, включающая тепло для плавления подаваемого сырья, различные потери через стенки и т.п.

Чтобы печь функционировала, их сумма должна быть положительной, то есть, в печи всегда имеется избыток энергии.

Избыток энергии "сбрасывается" во многом за счёт восстановления углекислого газа до угарного, с расходованием части кокса.

Большую роль в восстановительной реакции играет холостая колоша, в верхних слоях которой эта реакция идёт наиболее интенсивно. Чем больше холостая калоша, тем, как правило, больше CO выходит из печи.

Минимизация излишков энергии, например, за счет снижения содержания кокса в шихте, приводит к вырождению холостой калоши и риску загасания печи. То есть, значительный запас энергии в печи, и как следствие, высокое содержание CO в отходящих газах, является условием её устойчивого функционирования.

Однако, есть возможность при сохранении общего баланса перераспределить входящую энергию понижая содержание кокса, и одновременно повышая температуру дутьевого воздуха.

В расчетах два режима показывают близкие характеристики по концентрации угарного газа на выходе печи: первый - соответствует применяемому на производстве с температурой дутьевого воздуха 200 С и концентрацией кокса 20%, второй с дутьём 800 С и 13% кокса.

Более того, есть возможность вернуть в систему часть излишков энергии установив дожигатель, в котором теплота сгорания угарного газа используется для нагрева дутья. В полной системе печь+дожигатель избыток энергии значительно ниже

Таким образом, переход на более горячее дутьё с установкой дожигателя угарного газа позволяет повысить КПД печной системы и сэкономить кокс.

Кроме того, можно рекомендовать установку газоанализатора на выходе из печи, поскольку содержание CO является одним из индикаторов состояния холостой калоши.

Работа выполнена в СпБО МСЦ РАН - филиале ФГУ ФНЦ НИИСИ РАН в рамках Государственного задания.

.

Numerical investigation of blast heating in coke cupola for mineral raw melting

P.A.Voinovich, Y.A.Kurakin

Abstract: The results of numerical investigation of the processes in the cupola for mineral raw melting are presented. The cupola is used for basalt wool production. The possibility of the cupola stable functioning with low coke content is demonstrated at the air blust heating. It is shown that the installation of the carbon monoxide afterburner at cupola outlet for the heating of the air blast is reasonable for increasing of the cupola efficiency and the coke saving.

Keywords: numerical simulation, cupola for mineral raw melting, blust heating

Литература

[1] А.Д.Готлиб. Доменный процесс : Учеб. пособие для вузов / А.Д. Готлиб .— 2-е изд., испр. и доп .— Москва : Металлургия, 1966.
[2] А.Д.Готлиб. Нагрев дутья и расход кокса при выплавке чугуна / А. Д. Готлиб .— Харьков ; М. : Металлургиздат, 1947.
[3] R. Leth-Miller, A. D. Jensen, P. Glarborg, L. M. Jensen, P. B. Hansen, and S. B. Jorgensen. Investigation of a Mineral Melting Cupola Furnace. Parts I,II, Ind. Eng. Chem. Res. 2003**,** 42, 6872-6.
[4] В.А.Клименко, Л.С.Токарев. Основы физики доменного процесса. Челябинск: Металлургия. Челябинское отделение, 1991.
[5] Р.И.Нигматулин. Основы механики гетерогенных сред. – М.: Наука, 1978.
[6] R.Nedderman, U.Tuzun. Powder Technology, v.22, 1979, 234-238.
[7] Д.А.Франк-Каменецкий. Диффузия и теплопередача в химической кинетике. 3-е изд., М., Наука, 1987.
[8] Б.С.Мастрюков. Теплотехнические расчеты промышленных печей. М., Металлургия, 1972.

Исследование влияния отдельных составляющих тепловых потоков конвекции, радиации и кондукции на интервально стохастические тепловые процессы в электронных системах

А.Г.Мадера[1], П.И. Кандалов[2], М.Ж. Акжолов [3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1] alexmadera@mail.ru, [2] petrki87@gmail.com, [3] ak-1@mail.ru

Аннотация. В настоящее время при тепловом проектировании электронных систем, проведении математического и компьютерного моделирования, принимается допущение о незначительности влияния радиационной составляющей теплообмена на распределения температур. В данной работе приведены результаты численного моделирования влияния различных механизмов теплообмена (конвекции, радиации и кондукции) на интервально стохастические тепловые процессы в электронных системах. Исследования показали, что радиационная составляющая теплообмена вносит значительный вклад в значения температуры элементов электронных систем. Установлено, что пренебрежение радиационным теплообменом приводит к значительным погрешностям при проведении численного моделирования температурных полей и тепловом проектировании электронных систем.

Ключевые слова: электронный модуль, тепловой процесс, стохастический, математическое и компьютерное моделирование, мощность потребления, математическое ожидание, конвекция, радиация, кондукция

При тепловом проектировании электронных систем (ЭС) как правило пренебрегают радиационным тепловым потоком по сравнению с конвективным. Пренебрежение радиацией среди разработчиков ЭС объясняется, во-первых, сложностью учета радиационных тепловых потоков из-за их сильной нелинейной зависимости от температуры, выражаемой законом Стефана-Больцмана , что делает задачу моделирования тепловых режимов ЭС чрезвычайно сложной. Вторым доводом для пренебрежения радиацией служит ссылка на довольно низкие значения температур на кристаллах микросхем (МС), ограниченных сверху 125°С, так что, согласно оценочным расчетам, коэффициент теплоотдачи радиацией не превосходит 6 ÷ 8 В/м·К. Эти доводы не подкреплены строгим математическим анализом и общеприняты в литературе по тепловым расчетам электронных устройств [1, 2]. Между тем, как показывают детальные расчеты, влияние радиации на температуры ЭС может быть довольно значительным.

В данной статье приведены результаты численных исследований влияния конвективного, радиационного и кондуктивного теплообмена на тепловой режим конструкций ЭС, в частности для конкретной ЭС, рассмотренной в работе [3], и показано, что этим влиянием пренебрегать нельзя.

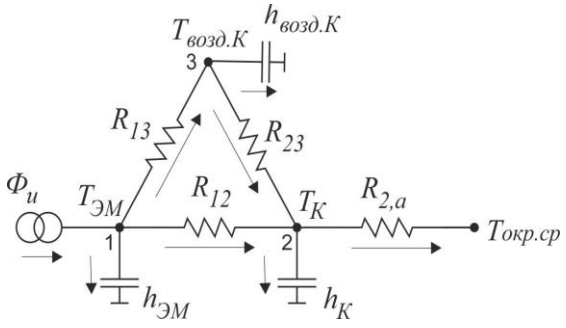


Рис. 1. Тепловая схема ЭС: ЭМ – электронный модуль, К – корпус ЭС,

Конструкция ЭС рассмотренная в работе [3] представляет собой корпус ЭС (К), с установленным внутри него электронным модулем (ЭМ), причем в [3] принято допущение, согласно которому теплообмен между ЭМ и корпусом протекает напрямую через конвективную тонкую прослойку, минуя тепловое взаимодействие с воздушной средой внутри корпуса ЭС, как от ЭМ, так и самого корпуса ЭС. Отметим, что корпус ЭС находится при этом в состоянии теплообмена с окружающей воздушной средой. Указанное допущение снижает адекватность модели теплообмена в ЭС. В отличие от работы [3], в настоящей работе данное допущение снято. А именно, принято, что между внутренней поверхностью корпуса, также как и поверхностью ЭМ, происходит теплообмен в воздушную среду внутри корпуса, что позволяет с большей степенью адекватности моделировать процессы теплообмена в ЭС. Тепловая схема (рис. 1) в этом случае, по сравнению с таковой в [3], существенно усложняется и содержит теперь вместо двух – три узла и новые схемные элементы $R_{1в}$, $R_{2в}$, (объемная теплоемкость воздуха в корпусе ЭС). Теплообмен между ЭМ и корпусом ЭС и воздушной средой внутри корпуса (*возд.К*), а также между корпусом и окружающей воздушной средой (*а – окр.ср.*), происходит при совместном действии конвекции, радиации и кондукции между ЭМ и корпусом. Математическая модель динамики развития тепловых процессов в ЭС соответствует общей тепловой модели, приведенной в [3 – 13].

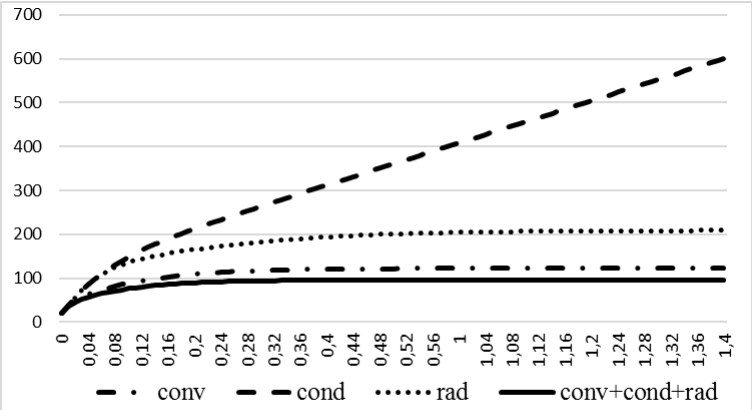


Рис. 2 . МО температуры *ЭМ*

Результаты математического моделирования математических ожиданий (МО) стохастических нестационарных температур ЭМ (узел 1, рис. 1; рис. 2), корпуса К (узел 2, рис. 1; рис. 3) и температуры воздушной среды (*возд.К*, рис. 1) внутри корпуса (узел 3, рис. 1; рис. 4), учитывающего воздействие отдельных механизмов теплообмена (конвекции, радиации и кондукции), приведены на рис. 2 (для ЭМ), рис. 3 (для К) и рис. 4 (для *возд.К*). Вычисления МО температур элементов ЭС (ЭМ, К, *возд.К*) были доведены до 100 мин. (на оси абсцисс на рис. 2, 3, 4 время отложено в час) до установления стационарного теплового процесса, протекающего в ЭС.

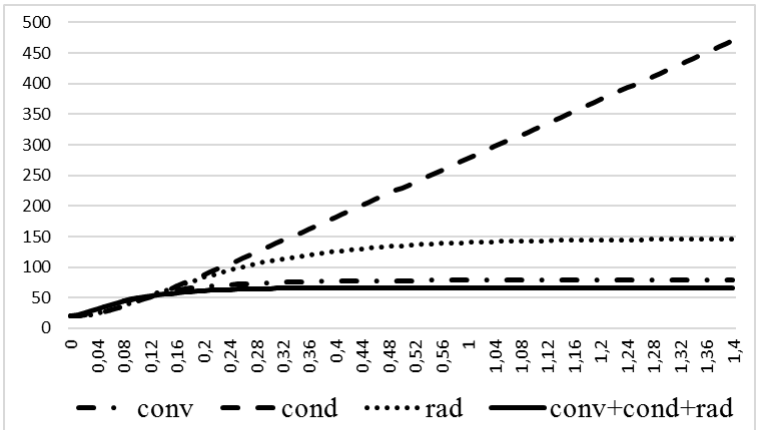


Рис. 3. МО температуры *К*

На основании расчетов МО установившихся стохастических температур элементов ЭС, а именно, ЭМ, К и *возд.К*, далее вычислялись тепловые потоки J_{conv}, J_{rad}, J_{cond}, между всеми узлами тепловой схемы

$i,j = 1,2,3$ (рис.1). Вычисления осуществлялись как для каждого механизма теплообмена (конвекции (*conv*), радиации (*rad*) и кондукции (*cond*)) в отдельности, согласно формулам:

$$J_{conv(i,j)} = A_1 \sqrt[4]{\frac{\overline{T}_i - \overline{T}_j}{L}},$$

$$J_{rad(i,j)} = A_2 \left(\overline{T}_i^4 - \overline{T}_j^4\right),$$

$$J_{cond(i,j)} = A_3 \left(\overline{T}_i - \overline{T}_j\right),$$

так и суммарного вклада конвекции, радиации и кондукции в общее значение теплового потока между любыми двумя узлами $i,j = 1,2,3$, согласно выражению

$$J_{i,j} = J_{conv(i,j)} + J_{rad(i,j)} + J_{cond(i,j)},$$

где A_J, $j = 1,2,3,4$ – коэффициенты, включающие в себя конструктивные параметры ЭМ и К, и теплофизические параметры воздушной среды внутри корпуса ЭС и в окружающей среде; $\overline{T}_i$ – МО стохастической установившейся температуры элемента i; индексы $i,j = 1,2,3$ соответствуют узлам тепловой схемы – 1 – *ЭМ* (электронный модуль), 2 – *К* (корпус ЭС), 3 – *возд.К* (воздушная среда внутри корпуса), 4 – *окр.ср.* (окружающая среда). Отметим, что кондуктивная составляющая теплообмена присутствует только для индексов 1 и 2, в остальных случаях потоков она равна нулю.

Вклад отдельных составляющих теплообмена – конвекции, радиации и кондукции – в суммарный теплообмен всей ЭС, рассчитывался в виде относительной погрешности каждого слагаемого в общем потоке (табл. 1), а именно:

$$\delta_{conv(i,j)} + \delta_{rad(i,j)} + \delta_{cond(i,j)} = 1,$$

где $\delta_{conv(i,j)} = \frac{J_{conv(i,j)}}{J_{i,j}}$,

$\delta_{rad(i,j)} = \frac{J_{rad(i,j)}}{J_{i,j}}$, $\delta_{cond(i,j)} = \frac{J_{cond(i,j)}}{J_{i,j}}$.

Таблица 1

Вклад отдельных оставляющих теплообмена в суммарный теплообмен всей ЭС

Тепловые потоки между элементами ЭС	Физическая природа потоков	Процентный вклад отдельных потоков, %
ЭМ – возд.К	Конвекция	66,21907
	Радиация	33,78093
	Кондукция	0
ЭМ – К	Конвекция	29,53467
	Радиация	43,87321
	Кондукция	26,59212
К – возд.К	Конвекция	33,90481
	Радиация	66,09519
	Кондукция	0
К – окр.ср.	Конвекция	50,14969
	Радиация	49,85031
	Кондукция	0

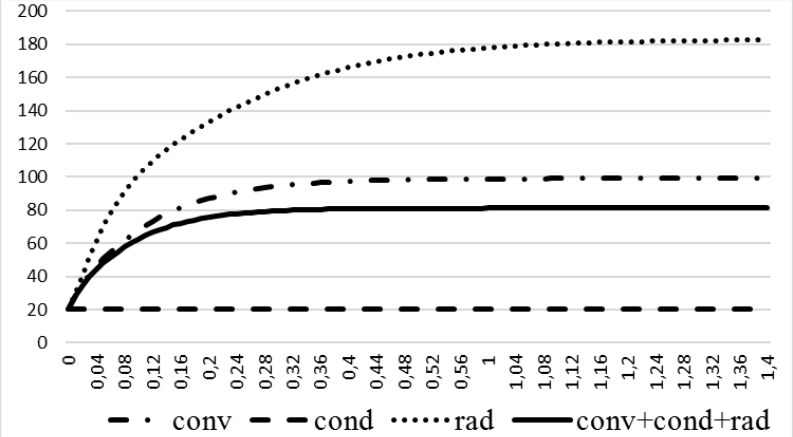


Рис. 4. МО температуры *возд.К*

Из табл. 1 следует, что радиационная составляющая теплообмена между всеми элементами ЭС (1 – *ЭМ*, 2 – *К*, 3 – *возд.К*, 4 – *окр.ср.*) изменятся от 33,8% до 49,9%. Это свидетельствует о значительном вкладе радиационной составляющей в общие тепловые потоки как между всеми элементами ЭС, так и во всей ЭС в целом. Поэтому пренебрежение радиацией приводит к неадекватности и низкой точности моделирования тепловых процессов ЭС. Что касается вклада конвективной составляющей потока в общий поток, то он является, как правило, доминирующим и изменяется в диапазоне от 29,5% до 66,2%. Однако вклад

радиации (=43,9%) в теплообмен между электронным модулем и корпусом ЭС существенно превосходит вклад конвективного теплового потока (=29,5%) между ними, что также свидетельствует о недопустимости пренебрежения радиационным теплообменом, коль скоро необходимо достичь более точного и адекватного моделирования теплообмена в ЭС.

Investigation of effect of convection, radiation and conduction on interval-stochastic thermal processes in electronic systems

A.G. Madera, P.I. Kandalov, M.J. Akzholov

Abstract. Thermal modeling and design of electronic systems is currently widely used. However, it is often assumed that thermal radiation has little influence on temperature distribution. The results of a computer simulation of the impact of various mechanisms of heat transfer was presented in this work. Research has shown that thermal radiation has contributed significantly to the system temperature. It has been found that disregard of radiation heat transfer results in significant errors in conducting thermal modeling.

Keywords. Electronic module, thermal process, stochastic, mathematical modeling and computer simulation, power consumption, expected value, convection, radiation, conduction.

Литература

1. Г.Н. Дульнев Тепло- и массообмен в радиоэлектронной аппаратуре. – М. Высш. шк., 1984
2. G. Ellison Thermal computations for electronics. – N.Y.: Taylor, 2011
3. М.Ж. Акжолов, П.И. Кандалов Влияние мощностей потребления и температуры окружающей среды на интервально-стохастические тепловые процессы в электронном модуле // Труды НИИСИРАН. 2017. Т. 7. № 4.
4. А.Г. Мадера, П.И. Кандалов. Математическое моделирование интервально стохастических тепловых процессов в технических системах при интервальной неопределенности определяющих параметров // Компьютерное исследование и моделирование. 2016. Т. 8. №3. С. 501 – 520.
5. П.И. Кандалов. Интервально стохастические тепловые процессы, протекающие в электронном модуле, заключенном в герметичном корпусе // Труды НИИСИРАН. 2016. Т. 6. №1. С. 64 – 68.
6. C. Gardiner. Stochastic methods. A Handbook for the Natural and Social Sciences – N.Y.: Springer, 2009
7. А.Г. Мадера, П.И. Кандалов. Моделирование трехмерных температурных полей в электронных модулях // Программные продукты и системы. 2010. №2. С. 36
8. А.Г. Мадера, П.И. Кандалов. Моделирование температурных полей технических систем в условиях интервальной неопределенности // Тепловые процессы в технике. 2014. № 5. С. 225 – 229
9. А.Г. Мадера Концепция математического и компьютерного моделирования тепловых процессов в электронных системах // Программные продукты и системы. 2015. № 3.С. 79 – 86
10. А.Г. Мадера, П.И. Кандалов Компьютерное моделирование температурных полей технических систем при интервально стохастической неопределенности параметров // Прикладная информатика. 2015. Т. 1(55). С. 106 – 113
11. J.C. Georgiadis. On the approximate solution on non-deterministic heat and mass transport problems // Int. J. Heat Mass Transfer. 1991. V. 33. n. 8. P.2099 – 2105
12. N.G. Kampen. van Stochastic Processes in Physics and Chemistry. – North Holland: Elsevier. 2007
13. C.J. Keller, V.W. Antonetti Statistical thermal design for computer electronics // Electronic Packaging and Production. 1979. V.19.n.3. P. 55 – 62.

Влияние мощностей потребления и температуры окружающей среды на интервально-стохастические тепловые процессы в электронном модуле

М.Ж. Акжолов[1], П.И. Кандалов[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] ak-1@mail.ru, [2] petrki87@gmail.com

Аннотация: В статье рассматривается компьютерное и математическое моделирование интервально стохастических нестационарных тепловых процессов в электронных системах. На основе разработанного метода [1], получена система матрично-дифференциальная уравнений для электронной системы, выполненной из герметичного корпуса с расположенным внутри электронным модулем. Проведено численное моделирование с применением метода Рунге-Кутта 4-го порядка. Проанализированы временные зависимости математического ожидания и дисперсии стохастической температуры электронного модуля от величин температуры окружающей среды и мощности потребления.

Ключевые слова: электронный модуль, тепловой процесс, стохастический, тепловое моделирование, мощность потребления.

Введение

В настоящее время при проектировании электронных систем (ЭС) применяются методы математического и компьютерного моделирования [1], которые основываются на допущении, что факторы, определяющие тепловые процессы, однозначно и полностью определены и детерминированы.

Однако, практика показывает, что определяющие факторы в ЭС отнюдь не являются детерминированными. Так например, значительному разбросу подвержены величины мощностей потребления у различных экземпляров микросхем одного и того же типа. При этом, данные величины заключены внутри некоторых интервалов, положение границ которых определяется технологией изготовления и последующей эксплуатацией изделия.

В современной науке большое количество работ посвящено анализу и моделированию стохастических тепловых процессов [2 – 10]. Однако, работ, посвященных моделированию нестационарных интервально-стохастических тепловых процессов недостаточно, а развитые в данных статьях подходы и методы не позволяют использовать их в компьютерном моделирование.

Реальные тепловые процессы в ЭС являются нестационарными, нелинейно зависят от значений температуры элементов в ЭС, носят интервально стохастический характер, поэтому адекватные модели и методы моделирования тепловых процессов должны учитывать указанные особенности, однако, в настоящее время в существующей литературе такие методы и модели отсутствуют.

В статье рассматривается математическое и компьютерное моделирование нестационарных нелинейных интервально-стохастических тепловых процессов в ЭС. Приводится математическая модель ЭС в виде система матрично-дифференциальных уравнений, определяющих статистические меры (математические ожидания, дисперсии, ковариации) элементов электронной системы и окружающей среды. Рассматриваются результаты компьютерного моделирования ЭС, полученные методом численного дифференцирования.

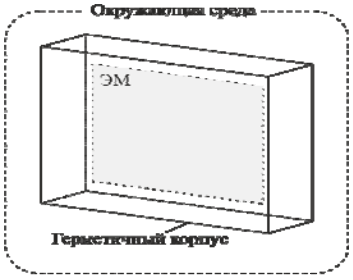


Рис. 1. Конструкция электронной системы

1. Математическая модель интервально-стохастических тепловых процессов в электронном модуле

Уравнения стохастической математической модели описывающей стохастическую математическую модель, представленную на (рис. 1), рассмотрены в работе [2]. Для проведения компьютерного моделирования преоб-

разуем систему дифференциальных уравнений в [2] к системе матрично-дифференциальных уравнений для определения статистических мер стохастических нестационарных температур электронного модуля, корпуса ЭС и окружающей среды:

- уравнение для математического ожидания:

$$H(\overline{T},t)\frac{d\overline{T}(t)}{dt}+A\overline{G}T(T,t)A^T\overline{T}(t)=\\ =\overline{\Phi}-A\overline{G}(\overline{T},t)A^T\overline{T}_a, \quad (1)$$

где $H(\overline{T},t)=diag\{h_1,h_2,...,h_n\}$ – диагональная матрица объемных теплоемкостей элементов системы; $\overline{T}(t)$ – вектор математических ожиданий (МО) температур; A – матрица инциденций; $\overline{\Phi}$ – вектор МО мощностей объемных источников теплоты; $\overline{T}_a$ – вектор МО температуры среды вблизи элементов системы; $\overline{G}$ – диагональная матрица МО тепловых проводимостей между элементами системы.

- ковариационная матрица

$$K_{TT}(t)=E\{\overset{o}{T}(t)\overset{o}{T}^T(t)\}:$$

$$\frac{dK_{TT}(t)}{dt}+H^{-1}A\overline{V}A^TK_{TT}(t)+\\ K_{TT}^T(t)A\overline{V}A^TH^{-1}=-H^{-1}AK_{GT}(t)-\\ K_{GT}^T(t)A^TH^{-1}+ \quad (2)\\ H^{-1}K_{\Phi T}(t)+K_{\Phi T}^T(t)H^{-1}-\\ -H^{-1}A\overline{V}A^TK_{T_aT}(t)-K_{T_aT}^T(t)A\overline{V}^TA^TH^{-1},$$

где $\overline{V}$ –матрица МО коэффициентов разложения в ряд Тейлора;

$K_{GT}(t)$ –ковариационная матрица между тепловыми проводимостями и температурами элементов;

$K_{\Phi T}(t)$ –ковариационная матрица между мощностями источников теплоты и температурами элементов;

$K_{T_aT}^T$ –ковариационная матрица между температурами среды и температурами элементов;

- ковариационная матрица

$$K_{GT}(t)=E\{\overset{o}{G}A^T\overline{T}(t)\overset{o}{T}^T(t)\}=\overline{R}K_{gT}(T):$$

$$\frac{dK_{gT}(t)}{dt}+K_{gT}(t)A\overline{V}^TA^TH^{-1}+\\ K_{gg}(\overline{R}+\overline{S})A^TH^{-1}=0, \quad (3)$$

где $K_{gT}(T)$ – ковариационная матрица между тепловыми проводимостями зазоров между элементами и температурами элементов;

$\overline{R}=diag\{\overline{r}_1,\overline{r}_2,...,\overline{r}_n\}$ и $\overline{S}=diag\{\overline{s}_1,\overline{s}_2,...,\overline{s}_n\}$ – диагональные матрицы, элементы которых определяются соотношениями $\overline{r}=A^T\overline{T}$ и $\overline{s}=A^T\overline{T}_a$ соответственно;

- ковариационная матрица

$$K_{\Phi T}(t)=E\{\overset{o}{\Phi}\overset{o}{T}^T(t)\}$$

$$\frac{dK_{\Phi T}(t)}{dt}+K_{\Phi T}(t)A\overline{V}^TA^TH^{-1}-K_{\Phi\Phi}H^{-1}=0, \quad (4)$$

где $K_{\Phi\Phi}$ –ковариационная матрица между мощностями источников теплоты;

- ковариационная матрица

$$K_{T_aT}(t)=E\{\overset{o}{T}_a\overset{o}{T}^T(t)\}:$$

$$\frac{dK_{T_aT}(t)}{dt}+K_{T_aT}(t)A\overline{V}^TA^TH^{-1}+\\ K_{T_aT_a}A\overline{V}^TA^TH^{-1}=0 \quad (5)$$

где $K_{T_aT_a}$ – ковариационная матрица между температурами среды.

При построении системы матрично-дифференциальных уравнений (1–5) учитывалась попарная статистическая независимость температуры среды и мощности тепловыделения источников теплоты.

2. Начальные условия и результаты моделирования

Рассмотрим результаты компьютерного моделирования (статистических мер) электронной системы представленной на рис. 1, которая описывается системой матрично-дифференциальных уравнений (1-5).

Электронная система выполнена из герметичного корпуса, внутри которого закреплен электронный модуль. Между корпусом и электронным модулем располагается воздушная среда. Электронный модуль выполнен в виде многослойной печатной платы с установленными на его нижней и верхней поверхностях различными активными и пассивными элементами, такими как микропроцессоры, микросхемы, электро-радиоэлементы, электрические разъемы и пр.

Вследствие неизбежного статистического технологического разброса изготовления, установки и монтажа используемых элементов, различные экземпляры ЭС будут иметь статистический разброс своих электрических и тепловых параметров и характеристик, обусловливая интервально стохастический характер тепловых процессов в ЭС.

Тепловая схема описанной электронной системы с соответствующими обозначениями направления тепловых потоков, номерами узлов и ветвей приведена на рис 2.

Электронный модуль и герметичный корпус представлены в схеме на рис. 2 узлами *1* и

2 с температурами $T_{ЭМ}$ и T_K, а окружающая среда узлов *3* с температурой T_a. Под воздействием теплового потока Φ_u всех активных

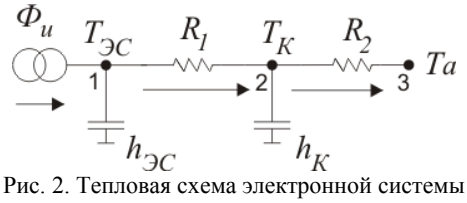


Рис. 2. Тепловая схема электронной системы

элементов происходит нагревание электронного модуля, воздушной среды внутри корпуса, собственно корпуса, и далее тепловой поток рассеивается в окружающую среду путем естественной конвекции. Причем теплообмен внутри и снаружи электронного модуля протекает одновременно с кондукцией, конвекцией и излучением.

$$h_{ЭМ} = 86,91\,;\ h_K = 287,848\,;$$

$$\vec{1} = (1,1,0)\,;\ \vec{1}_a = (0,0,1)\,;\ \vec{1}_\Phi = (1,0,0)\,.$$

Для решение системы дифференциальных уравнение (1-5) применялся численный метод решения задачи Коши для обыкновенных дифференциальных уравнений – Рунге-Кутты 4-го порядка [11]. Для величины шага $t = 10^{-2}$ время счета составляла не более 1с.

На рис. 3 и 4 показаны зависимости математических ожиданий и дисперсий стохастических температур электронного модуля при различных температурах окружающей среды T_a. В начальный момент времени вся система находится в тепловом равновесии с окружающей средой. Иначе говоря, температура электронного модуля $T_{ЭМ}$ и корпуса T_K совпадают с температурой окружающей среды $T_a = \overline{T}_a^0$ в момент времени $t = 0$. В качестве начальных

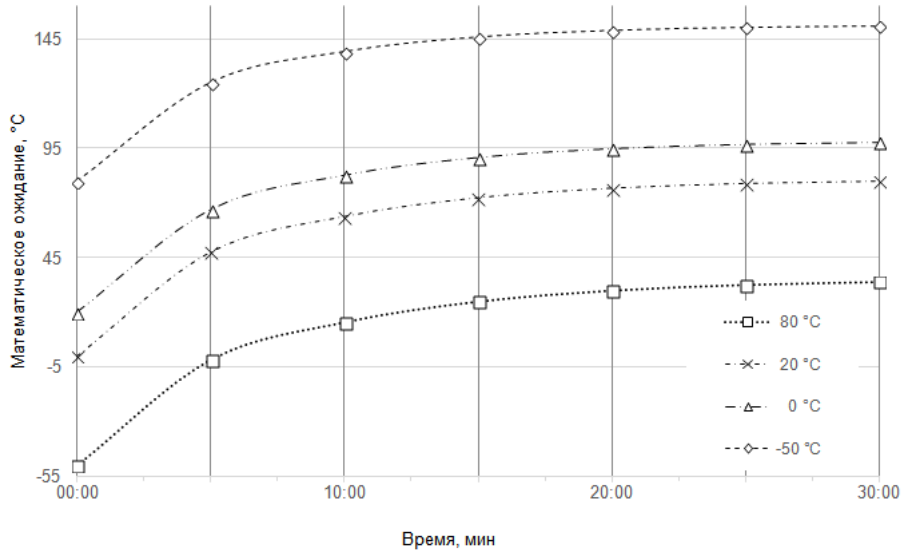


Рис. 3. Нестационарные математические ожидания (°C) стохастической температуры электронного модуля при различных температурах окружающей среды $\overline{T}_a$.

Вычисление статистических мер по полученным уравнениям математической модели (1-5) выполнялись при различных значениях температуры окружающей среды $\overline{T}_a$ и мощностей $\overline{\Phi}_u$ источника теплоты при следующих начальных условиях($t = 0$):

$$\overline{T}(0) = \overline{T}_a^0 \cdot \vec{1}\,,\ \overline{\Phi}(0) = (\overline{\Phi}_u^0, 0, 0)\,,$$

$$\overline{T}_a(0) = \overline{T}_a^0 \cdot \vec{1}_a\,,\ H(0) = diag(h_{ЭМ}; h_K; 0)\,,$$

$$K_{TT}(0) = D_{T_e} \vec{1} \cdot \vec{1}^T\,,\ K_{gT}(0) = 0\,,\ K_{\Phi T}(0) = 0\,,$$

$$K_{T_a T}(0) = D_{T_e} \vec{1}_a \cdot \vec{1}^T\,,\ K_{\Phi\Phi} = D_\Phi \vec{1}_\Phi \cdot \vec{1}_\Phi^T$$

где $\overline{T}_a^0 = 293\,K$; $D_{T_e} = 1,5K^2$; $D_\Phi = 0,7$ Вт^2;

условий рассматривалась температура окружающей $\overline{T}_a^0$ при -50, 0, 20 и 80 °C. Величина теплового потока Φ_u составляла 30 Вт.

Тепловые процессы развиваются после включения источник тепла. Установившиеся показатели значений математического ожидания стохастической температуры электронного модуля $\overline{T}_{ЭМ}$ в момент времени $t = 30$ мин составили 150,95; 97.70; 79,69 и 34,00 °C при соответствующих значениях начальной температуры окружающей среды $\overline{T}_a$. При этом дисперсии стохастической температуры ЭМ (рис. 4) составили 3,75°C^2 при $\overline{T}_{ЭМ} = 150.95$ °C; 4,25°C^2 при $\overline{T}_{ЭМ} = 97.70$ °C; 4,41°C^2 при

$\overline{T}_{ЭМ} = 79,69$ °C и 4,82 °C^2 при $\overline{T}_{ЭМ} = 34,00$ °C.

Отметим, что разности значений математического ожидания температур электронного модуля $\overline{T}_{ЭМ}$ в начальный момент времени и последующие моменты времени t равные 10 и 20 мин. при различных начальных температурах окружающей среды имеют небольшие разницы, а начиная с 30 и до 50 мин. эти разности становятся относительно одинаковые и стабильные. Динамики этих процессов развиваются пропорционально и монотонно стабильно.

Рис. 4 отображает динамику поведения нестационарной дисперсии стохастической температуры электронного модуля. Как следует из результатов численного эксперимента величина дисперсии снижается по мере роста температуры окружающей среды T_a. Такие показатели объясняются тем, что при меньшей температуре окружающей среды, градиент температуры вокруг источника больше, т.е. конвекция развивается намного интенсивнее. По мере

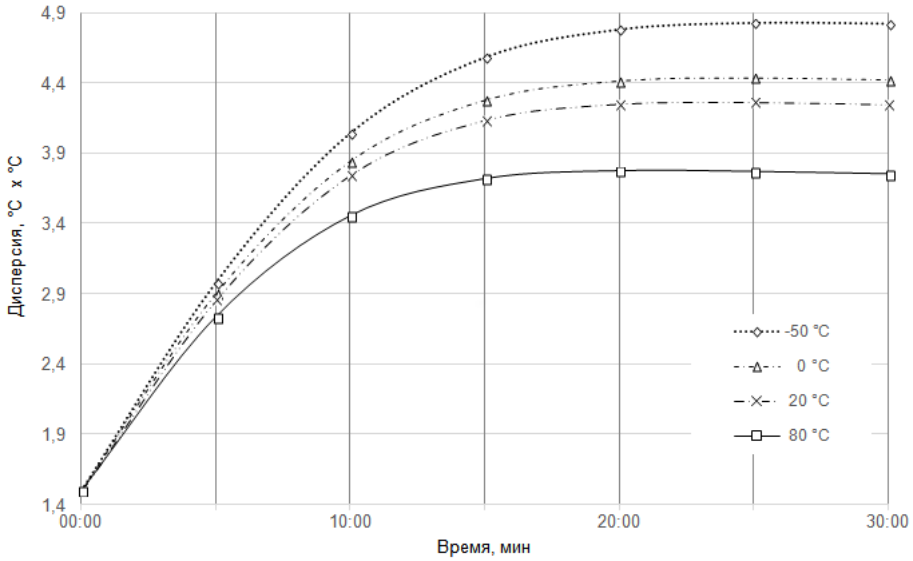


Рис. 4. Нестационарные дисперсии (°C^2) стохастической температуры электронного модуля при различных температурах окружающей среды T_a.

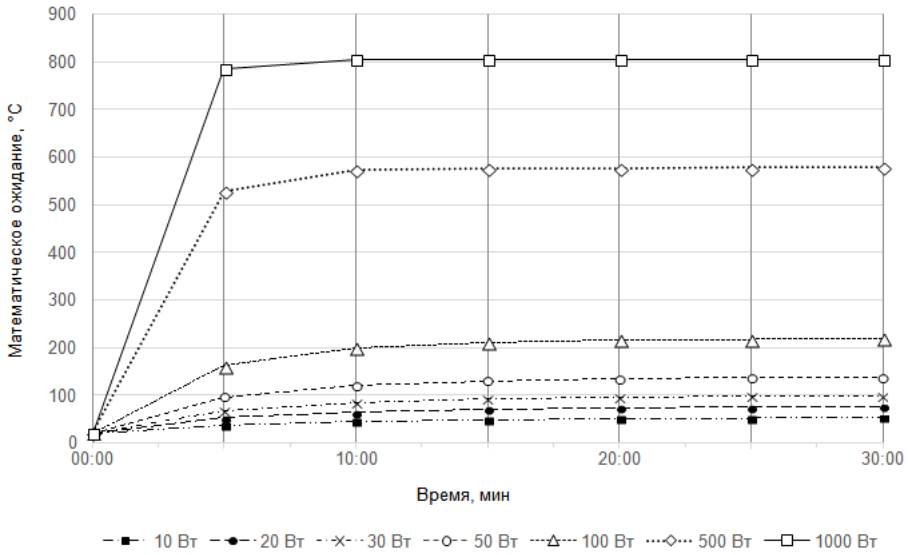


Рис. 5. Нестационарные математические ожидания (°C) стохастической температуры электронного модуля при различных мощностях $\overline{\Phi}_u$ объемного источника теплоты.

возрастания температуры окружающей среды $\overline{T}_a$ значения градиента уменьшается и влияние конвекционной составляющей на температуру электронного модуля ослабевает. Можно сказать, что повышение значения температуры выполняет роль стабилизации конвекции, что приводит к уменьшению разброса стохастической температуры электронного модуля.

На рис. 5 и 6 показаны результаты компьютерного моделирования в виде зависимости математических ожиданий и дисперсий стохастических температур электронного модуля при различных значении мощностей $\overline{\Phi}_u$ источника теплоты. В начальный момент времени $t = 0$ система находится в тепловом равновесии с окружающей средой $\overline{T}_e^0 = 20$ °С. На рис. 5 показаны зависимости математического ожидания стохастической температуры электронного при различных мощностях источника теплоты $\overline{\Phi}_u$ величиной 10, 20, 30, 50, 100, 500 и 1000 Вт.

$\overline{\Phi}_{ЭМ} = 100$ Вт ;3,39°С^2 при $\overline{\Phi}_{ЭМ} = 50$ Вт ; 4,24°С^2 при $\overline{\Phi}_{ЭМ} = 30$ Вт ;4,96°С^2 при $\overline{\Phi}_{ЭМ} = 20$ Вт ; 6,28°С^2 при $\overline{\Phi}_{ЭМ} = 10$ Вт .

Из графиков, представленных на рис.5 и 6, видно, что выход на стационарный режим значений математического ожидания стохастической температуры электронного модуля наступает при $t = 10$ мин., т.е много раньше по сравнению с дисперсии стохастической температуры электронного модуля (рис. 6), для которой стационарный режим наступает при $t = 30$ мин.

Динамика развития нестационарной дисперсии стохастической температуры электронного модуля показана на рис. 6. От начального момента времени и до момента времени 15 мин наблюдается достаточно интенсивное развитие теплового процесса. Дальнейшее развитие теплового процесса приводит к стабильному стационарному режиму, что соответствует естественному физическому

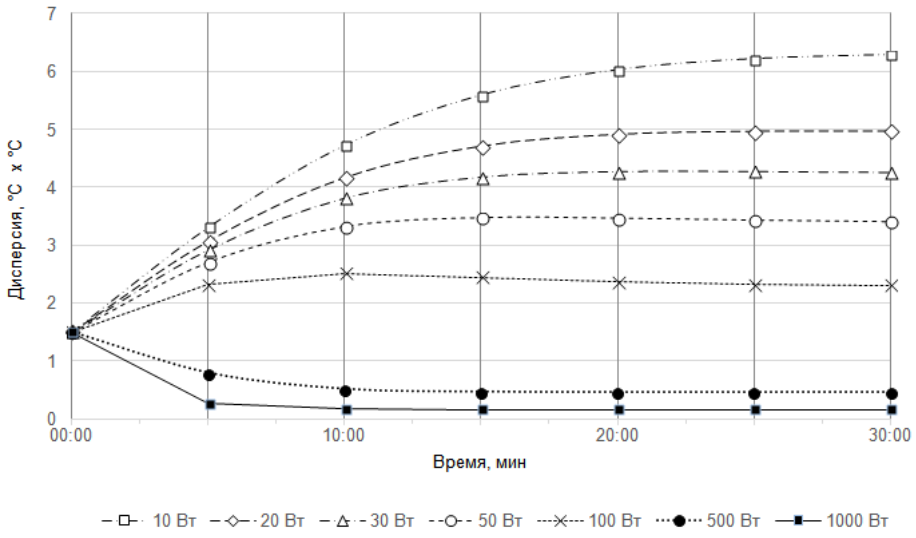


Рис. 6. Нестационарные дисперсии (°С^2) стохастической температуры электронного модуля при различных мощностях $\overline{\Phi}_u$ объемного источника теплоты.

При исследования зависимости изменения тепловых процессов от мощности источника установившиеся показатели значений математического ожидания стохастической температуры электронного модуля $\overline{T}_{ЭМ}$ в момент времени $t = 30$ мин составили 805.14, 577.37, 217.38, 136.68, 97.85, 76.02 и 51.50 °С при соответствующих значениях мощностей $\overline{\Phi}_u$ источника. При этом дисперсии стохастической температуры ЭМ (см. рис. 6) составили 0,16°С^2 при $\overline{\Phi}_{ЭМ} = 1000$ Вт ; 0,45°С^2 при $\overline{\Phi}_{ЭМ} = 500$ Вт ; 2,30°С^2 при

процессу разогрева ЭС. С ростом мощности источника тепла электронного модуля уменьшается величина дисперсии. Это обусловлено тем, что по мере увеличения мощности источника теплоты при постоянном начальном значении температуры окружающей среды влияние последней оказывается меньшее воздействие на интенсивность развития теплового процесса. Это приводит к уменьшению разброса значений дисперсии стохастической температуры электронного модуля.

Результаты решения представленные на рис. 3 – 6 показывают, что показатели МО и дисперсии стохастической температуры элек-

тронного модуля выходят на стационарный режим начиная с момента времени 30 мин. Данные результаты подтверждают, что сформулированная физическая задача является корректной, а выбранная математическая модель (системы матрично-дифференциальных уравнений) и, полученное на ее основе численное решение, является устойчивым и сходящимся.

3. Заключение

Рассмотрев в статье представлена математическая модель интервально-стохастических процессов в виде системы матрично-дифференциальных уравнений, позволяющая определять нестационарные статические меры стохастических распределений температуры элементов в ЭС: математические ожидания, дисперсии и ковариации между температурами всех элементов (ЭМ, герметичный корпус) и окружающей средой. На основе проведенного компьютерного моделирования получены и проанализированы зависимости влияния мощностей потребления и температуры окружающей среды на интервально-стохастические тепловые процессы в электронном модуле, которые показали адекватность построенной математической модели.

Метод, разработанный в статье, применяется на практике при тепловом проектировании ЭС.

Influence of power consumption and ambient temperature on the interval stochastic thermal processes in the electronic module

M.J. Akzholov, P.I. Kandalov

Abstract. The article deals with mathematical and computational modeling of interval stochastic unstady-state thermal processes in electronic systems. The authors based on [1] derived a matrix system of differential equations for electronic system which consists of hermetic enclosure and electronic module located inside. Numerical simulation are carry out using four order Runge-Kutta method. The time-dependences of the expectation value and variance of the stochastic value from ambient temperature and power consumption are analyzed.

Keywords. Electronic module, thermal process, stochastic, thermal modeling, power consumption.

Литература

1. С.Г.Бобков, А.Г.Мадера. Энергетические затраты, быстродействие и проблема теплоотвода в микропроцессорах // Программные продукты и системы. 2013. № 4. С. 104-106.
2. А.Г. Мадера, П.И. Кандалов. Математическое моделирование интервально стохастических тепловых процессов в технических системах при интервальной неопределенности определяющих параметров // Компьютерные исследования и моделирование. 2016. Т. 8. №3. С. 501 – 520.
3. П.И. Кандалов. Интервально стохастические тепловые процессы, протекающие в электронном модуле, заключенном в герметичном корпусе // Труды НИИСИРАН. 2016. Т. 6. №1. С. 64 – 68.
4. C. Gardiner. Stochastic methods. A Handbook for the Natural and Social Sciences – N.Y.: Springer, 2009
5. А.Г. Мадера, П.И. Кандалов. Моделирование трехмерных температурных полей в электронных модулях // Программные продукты и системы. 2010. №2. С. 36
6. А.Г. Мадера, П.И. Кандалов. Моделирование температурных полей технических систем в условиях интервальной неопределенности // Тепловые процессы в технике. 2014. № 5. С. 225 – 229
7. А.Г.Мадера, П.И.Кандалов. Анализ интервально стохастических температурных полей технических систем. // Программные продукты и системы. 2014. №4 (108). С. 41-45
8. J.C. Georgiadis. On the approximate solution on non-deterministic heat and mass transport problems // Int. J. Heat Mass Transfer. 1991. V. 33. n. 8. P.2099 – 2105
9. N.G. Kampen. van Stochastic Processes in Physics and Chemistry. – North Holland: Elsevier. 2007
10. C.J. Keller, V.W. Antonetti Statistical thermal design for computer electronics // Electronic Packaging and Production. 1979. V.19.n.3. P. 55 – 62.
11. Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. Численные методы: Учеб. пособие. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 600 с.

Методы распределенного моделирования воздушных потоков с помощью систем частиц

А.В.Мальцев[1], П.Ю. Тимохин[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]avmaltcev@mail.ru, [2]webpismo@yahoo.de

Аннотация: В работе рассматривается задача моделирования потоков воздуха в трехмерных виртуальных сценах. Для ее решения предлагаются методы имитации потока с помощью систем частиц. Подробно описаны распределенные методы расчета векторного поля скоростей потока в пространстве сцены. Предлагаемые подходы основаны на применении параллельных вычислений на современных многоядерных графических процессорах, что обеспечивает поддержку моделирования и визуализации в масштабе реального времени.

Ключевые слова: трехмерная сцена, виртуальная среда, система частиц, распределенные вычисления, CUDA.

Введение

В настоящее время во многих научных и технических областях деятельности человека широко используется компьютерное моделирование и визуализация виртуальной среды, имитирующей некоторую реальную обстановку. С точки зрения правильного восприятия наблюдателем такой среды, важным классом моделируемых в ней объектов являются динамические объекты, не имеющие четких геометрических границ. К ним относятся, например, струи жидкости, падающий снег, дождь и т.д.

Эти объекты состоят из огромного количества мелкоразмерных элементов и обычно моделируются в виртуальном пространстве с помощью систем частиц [1]. Также, как и другие объекты реального мира, данный класс объектов подвержен воздействию множества внешних сил. Среди них особый практический интерес представляют силы, оказываемые воздушными потоками.

Задача имитации в виртуальной среде потоков воздуха рассматривалась и продолжает изучаться современными исследователями.

Так, в работе [2] рассматривается моделирование воздействия ветра на травяные поля. В [3] и [4] авторы описывают свои модели для расчета траектории перемещения частиц снега под влиянием воздушных потоков.

Однако, большинство предлагаемых решений имеют высокую вычислительную сложность, что приводит к существенному росту времени моделирования и визуализации каждого кадра изображения виртуальной среды.

В связи с этим использование таких подходов затруднено в таких областях, как имитационно-тренажерные комплексы, где требуется формировать кадры в масштабе реального времени (частота смены кадров не менее 25 раз в секунду).

В данной статье предлагаются новые распределенные методы и алгоритмы моделирования воздушных потоков на GPU с поддержкой CUDA. Эти решения основаны на представлении таких потоков в виде систем частиц. Вычисление векторного поля скоростей воздушных потоков для виртуальной сцены в требуемый момент времени производится на основе положений и параметров активных в этот момент времени частиц. Далее рассмотрим предлагаемые подходы подробнее.

1. Моделирование воздушного потока системой частиц

Для моделирования в трехмерной виртуальной сцене потока воздуха, создаваемого каким-либо источником, будем использовать систему частиц с эмиттером, размер которого покрывает площадь испускания воздушного потока. Задаваемые параметры этой системы частиц, такие как частота генерации частиц, их скорость и время жизни, будут определять свойства воздушного потока (плотность, скорость, дальность распространения и т.д.). В данной работе рассматриваются трехмерные сцены, моделирующие открытые пространства, т.е. не содержащие крупных препятствий для прохождения воздушного потока.

Общие методы и подходы для реализации многоэлементных систем частиц на современных графических процессорах были подробно рассмотрены в исследовании [5]. Идея этих решений состоит в выполнении двух этапов обработки системы частиц в момент

формирования изображения каждого кадра виртуальной сцены. На *первом этапе* с помощью архитектуры CUDA производятся вычисление массива данных о частицах, описывающего состояние системы частиц в заданный момент времени. На данном этапе формируются новые частицы, рассчитываются текущие параметры уже существующих, а также удаляются частицы, время жизни которых истекло. *Второй этап* предполагает визуализацию полученного массива с синтезом «на лету» необходимой геометрии частиц, расчетом их освещенности и наложением текстур. Для этого задействованы вершинный, геометрический и фрагментный шейдеры.

Особенностью применения системы частиц при моделировании воздушного потока является отсутствие этапа визуализации. Применяется лишь расчет ее состояния в заданный момент времени на GPU с поддержкой CUDA.

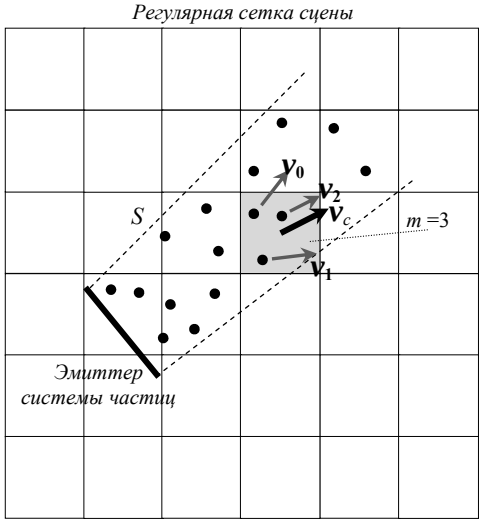


Рис. 1. Моделирование воздушного потока системой частиц

Идея предлагаемого подхода состоит в следующем. Чтобы вычислить векторное поле скоростей потока в виртуальной сцене, разделим ее пространство на небольшие трехмерные области. Для этого мы предлагаем использовать регулярную сетку – разбиение пространства сцены плоскостями, параллельными плоскостям мировой системы координат (СК) WCS, на трехмерные ячейки одинакового размера, называемые вокселами. Для решения нашей задачи мы будем использовать кубические вокселы. Детали процесса построения регулярной сетки сцены подробнее рассмотрим в п.2.

Обозначим через S систему частиц, имитирующую воздушный поток. В некоторый момент времени t каждая частица системы S может находиться не более чем в одном вокселе сетки (рис. 1). Каждый воксел в момент t содержит в себе $m \geq 0$ частиц. Поэтому можно выразить вектор $\boldsymbol{v}_c$ скорости потока в центре воксела через вектора $\boldsymbol{v}_k$ скоростей этих m частиц, где $k \in [0, m\text{-}1]$. При этом вклад $\boldsymbol{v}_k$ –го вектора будет тем больше, чем ближе k-ая частица к центру воксела. Решение задачи вычисления на GPU с поддержкой CUDA векторов скоростей для всех вокселов подробно изложено в п.3. Тогда в произвольной точке сетки сцены вектор скорости потока можно вычислить с помощью трехмерной интерполяции векторов $\boldsymbol{v}_c$ в 8-ми окружающих эту точку центрах вокселов.

2. Построение сетки сцены

Процесс создания регулярной сетки трехмерной виртуальной сцены включает в себя два этапа:

1) определение ограничивающего параллелепипеда сцены, стороны которого параллельны координатным плоскостям XY, YZ и ZX мировой СК WCS. В дальнейшем ограничивающий параллелепипед сцены (или отдельного треугольного полигона), удовлетворяющий приведенным выше условиям, будем называть просто AABB (*Axis-Aligned Bounding Box*).

2) разбиение AABB сцены плоскостями, параллельными XY, YZ и ZX, на множество вокселов.

Построение сетки для статической сцены, в которой нет динамично перемещающихся объектов, достаточно выполнить один раз при загрузке такой сцены, ограничиваясь применением только CPU. Однако, если в сцене присутствуют подвижные объекты, способные значительно изменять свое положение в пространстве, то AABB этой сцены может изменяться от кадра к кадру. В таком случае при построении сетки целесообразно использовать распределенные вычисления на современных GPU. Далее рассмотрим этот случай.

Для выполнения *1-го этапа* построения сетки загрузим в память видеоадаптера с поддержкой архитектуры CUDA координаты вершин всех n треугольных полигонов сцены, представленные в локальных СК тех объектов, которым они принадлежат и матрицы перехода из этих локальных СК в WCS. Далее найдем координаты всех загруженных вершин в WCS и определим AABB каждого i-го полигона в виде пары минимальной $B_{0,i}$ и максимальной $B_{1,i}$ вершин, окружающего его бокса, где $i \in [0, n\text{-}1]$. Для каждого треугольника эта задача решается в отдельном вычислительном потоке; эти потоки выполняются параллельно на GPU.

Исходя из рассчитанных AABB треугольников, найдем минимальную ($B_{\min}$) и максимальную ($B_{\max}$) точки для AABB всей

сцены также с помощью параллельной обработки. Эту задачу можно решить, используя функции поиска максимальных и минимальных элементов в массивах из библиотеки *Thrust* [6].

На *2-ом этапе* разобьем полученный AABB сцены плоскостями, параллельными XY, YZ и ZX, на одинаковые кубические вокселы с ребром a. Значение a вычислим по эвристической формуле

$$a = \sqrt[3]{\frac{\left(B_{\max,x} - B_{\min,x}\right)\left(B_{\max,y} - B_{\min,y}\right)\left(B_{\max,z} - B_{\min,z}\right)}{\lambda n}},$$

где λ – константа, позволяющая делать сетку более плотной или более разреженной. Размерность D сетки в вокселах по осям X, Y, Z при этом вычисляется как

$$D\left(d_x, d_y, d_z\right) = \left(\left\lceil D'_x \right\rceil, \left\lceil D'_y \right\rceil, \left\lceil D'_z \right\rceil\right),$$

$$D' = \frac{1}{a}\left(B_{\max} - B_{\min}\right),$$

где квадратные скобки обозначают округление до ближайшего целого сверху. Чтобы AABB сцены полностью вмещал полученные вокселы, пересчитаем его максимальную точку по формуле

$$B_{\max}' = B_{\min} + a \cdot D.$$

Для дальнейшей работы с построенной сеткой введем номер C воксела, вычисляемый по формуле

$$C = k_z \cdot d_x \cdot d_y + k_y \cdot d_x + k_x, \qquad (1)$$

где k_x, k_y, k_z – индексы воксела по координатным осям X, Y, Z мировой СК, начиная с нуля от точки $B_{\min}$.

3. Расчет векторного поля

Рассмотрим вычисление векторного поля скоростей воздушных потоков в виртуальной сцене на примере одного источника, моделируемого системой частиц S. Пусть общее количество частиц в системе равно p. Все расчеты будем выполнять в момент формирования текущего кадра визуализации.

Вначале определим для каждой частицы из S номер воксела регулярной сетки сцены, в котором частица находится в рассматриваемый момент времени. Для этого составим два массива M_P и M_C по p элементов каждый. Элемент массива M_P будет хранить номер частицы, а элемент с тем же индексом из массива M_C – номер соответствующего этой частице воксела сетки.

Поскольку общее количество частиц может достигать порядка 10^6 и более, то производить расчеты и заполнять массивы M_P и M_C будем на GPU с использованием архитектуры параллельных вычислений CUDA. Указанные массивы при этом будут размещаться в глобальной памяти видеоадаптера.

В предлагаемом подходе каждая частица $S_j \in S$, $j \in [0, p\text{-}1]$, обрабатываться своим потоком, имеющим индекс j. Вначале поток записывает свой индекс в элемент $M_P[j]$ массива M_P и выполняет проверку времени жизни t частицы. Если t меньше заданного для частицы максимального значения $t_{\max}$ времени жизни, то частица является активной. Тогда поток вычисляет индексы k_x, k_y, k_z воксела регулярной сетки (по осям X, Y, Z системы WCS), в котором находится частица. Вычисление производится, исходя из известных координат текущего положения $P_j(x, y, z)$ рассматриваемой частицы S_j в мировой СК:

$$k_x = \left\lfloor \frac{1}{a}\left(x - B_{\min,x}\right) \right\rfloor, \; k_y = \left\lfloor \frac{1}{a}\left(y - B_{\min,y}\right) \right\rfloor,$$

$$k_z = \left\lfloor \frac{1}{a}\left(z - B_{\min,z}\right) \right\rfloor,$$

где скобки обозначают округление до ближайшего целого снизу, a – длина ребра воксела. При этом проверяются неравенства

$$0 \le k_x < d_x, \quad 0 \le k_y < d_y, \quad 0 \le k_z < d_z,$$

где d_x, d_y, d_z – количество вокселов в сетке по каждой из осей. Выполнение всех неравенств означает, что воксел принадлежит регулярной сетке сцены, а рассматриваемая частица находится в пределах этой сетки. Тогда по найденным индексам поток вычисляет номер C_j воксела сетки с помощью формулы (1) и записывает его в элемент $M_C[j]$ массива M_C. В противном случае вместо номера C_j в $M_C[j]$ заносится значение -1. Это означает, что частица не принадлежит ни одному из вокселов сетки. Заметим, что при $t \ge t_{\max}$ частица считается неактивной и в $M_C[j]$ также заносится значение -1.

После завершения на GPU всех p потоков мы будем иметь два взаимосвязанных массива M_P и M_C, хранящих пары «номер частицы/номер воксела» (отметим, что один и тот же воксел может входить в M_C неоднократно). Эти пары отсортированы по номерам частиц. Но для расчета вектора скорости воздушного потока в каком-либо вокселе регулярной сетки нам нужно знать те частицы, которые находятся внутри него в текущий момент времени. Поэтому такая сортировка является неудобной для дальнейшего использования массивов M_P и M_C.

Действительно, имея номер воксела, необходимо делать полный перебор массива

M_C, чтобы найти номера всех расположенных в нем частиц.

Для упорядочивания пар относительно номеров вокселов регулярной сетки применим функцию *sort_by_key* из библиотеки *Thrust* [6]. Данная функция выполняет параллельную сортировку массивов методом поразрядной сортировки (radix sort), основные принципы которого описаны в [7] и [8]. На вход *sort_by_key* подаются массив ключей и массив данных, содержащие одинаковое количество элементов. Предполагается, что оба массива взаимосвязаны, а именно: *j*-ый элемент первого массива образует с *j*-ым элементом второго массива пару (ключ, информация). Функция производит сортировку ключевого массива и аналогично изменяет индексы элементов массива данных. На выходе имеем пару массивов, отсортированных по ключам. В нашем случае необходимо подать на вход *sort_by_key* массив M_C в качестве ключевого, а M_P – в качестве массива данных. В результате работы функции пары будут упорядочены по номерам вокселов.

Вектор скорости $\boldsymbol{v}_c$ воздушного потока в центре каждого воксела регулярной сетки определим с помощью усредненной суммы векторов скоростей находящихся в нем частиц с учетом весовых коэффициентов, задающих влияние каждой частицы на величину и направление $\boldsymbol{v}_c$:

$$\boldsymbol{v}_c = \frac{1}{m}\sum_{k=0}^{m-1} w_k \boldsymbol{v}_k , \qquad (2)$$

где m – количество частиц, находящихся в пределах рассматриваемого воксела, $k \in [0, m\text{-}1]$. Весовой коэффициент w_k для k-ой частицы определим в зависимости от расстояния r_k между ней и центром воксела. В нашей модели используется линейное изменение коэффициента w_k от 1 до 0 вдоль радиуса R сферы, описывающей воксел (1 соответствует центру сферы). Весовой коэффициент w_k вычисляется по формуле

$$w_k = \frac{R - r_k}{R} .$$

Подставляя его в формулу (2), получим

$$\boldsymbol{v}_c = \frac{1}{m}\sum_{k=0}^{m-1} \frac{R - r_k}{R} \boldsymbol{v}_k . \qquad (3)$$

Расчет $\boldsymbol{v}_c$ для *j*-го воксела регулярной сетки выполняется по формуле (3) в *j*-ом потоке на GPU. Массив M_C содержит отсортированные номера вокселов, но эти номера могут иметь множественные вхождения или вообще отсутствовать. Поэтому *j*-ому потоку необходимо знать смещение и количество считываемых элементов в массиве M_P. Для этого формируется таблица смещений, по которой мы для любого воксела с номером *j* сможем определить индекс первого вхождения пары «воксел *j*/номер частицы» в массивах M_C и M_P. Кроме того, нам потребуется таблица заполнения вокселов, где будет записано количество частиц, принадлежащих каждому вокселу. Данная задача решается с помощью параллельного (относительно вокселов) бинарного поиска в массиве M_C номеров вокселов.

Для вокселов, которые не содержат ни одной частицы и, следовательно, отсутствуют в массиве M_C, смещение задается равным -1, а количество частиц равным 0.

Таким образом, мы получили вектора скоростей имитируемого воздушного потока в центрах вокселов регулярной сетки. Вектор в произвольной точке внутри сетки вычисляется путем трехмерной интерполяции векторов в 8-ми окружающих эту точку центрах вокселов.

Заключение

Описанные в статье методы и алгоритмы моделирования в трехмерных виртуальных сценах воздушных потоков с помощью систем частиц основаны на использовании распределенных вычислений на современных CUDA-совместимых графических процессорах. Благодаря этому обеспечивается их работа в режиме реального времени и возможность применения в системах виртуальной реальности, имитационно-тренажерных комплексах, виртуальных лабораториях и др.

Исследования были выполнены при поддержке РФФИ, проект № 16-07-00796.

Distributed methods of air flow simulation using particle systems

A.V. Maltsev, P.Yu. Timokhin

Abstract: At this paper a task of air flow simulation in three-dimensional virtual scenes is considered. To solve it, the methods of flow imitation by means of particle system are presented. Distributed methods for calculation of flow velocity vector fields in scene space are described in detail. Proposed solutions are based on parallel computing with using of modern multicore graphics processors. This provides a support of real-time simulation and visualization.

Keywords: three-dimensional scene, virtual environment, particle system, distributed calculations, CUDA.

Литература

1. W.T.Reeves. Particle systems – a technique for modeling a class of fuzzy objects // In Computer Graphics (Proc. SIGGRAPH), 1983, p. 359-376.
2. H,Qiu., L.Chen, J.Chen. Dynamic Simulation of Grass Field Swaying in Wind // Journal of Software, 2012, no. 2, Vol.7, p. 431-439.
3. J.Tan, X.Fan. Particle system based snow simulating in real time // Procedia Environmental Sciences 10, 2011, Vol. 10, p. 1244-1249.
4. C.Martin, I,Parberry. Real Time Dynamic Wind Calculation for a Pressure Driven Wind System // Proceedings of the 2006 ACM SIGGRAPH Video Game Symposium, 2006, p. 151-154.
5. А.В.Мальцев. Реализация системы частиц в реальном времени на GPU // Программные продукты и системы, 2014, №4, с. 57-62.
6. Thrust. URL: http://docs.nvidia.com/cuda/thrust/index.html (дата обращения: 01.09.2017)
7. Radix sort. URL: https://en.wikipedia.org/wiki/Radix_sort (дата обращения: 01.09.2017)
8. Д.Кнут. Искусство программирования, том 3. Сортировка и поиск. 2-е издание. – М.: «Вильямс», 2007.

Использование семейства инструментов CMake для моделирования проектов сложных СБИС в среде Cadence Incisive

А.В. Андрианов

ЗАО НТЦ «Модуль», Москва, Россия, E-mail: andrianov@module.ru

Аннотация: По мере роста сложности современных систем на кристалле (СнК) и количества используемых аппаратных блоков, перед разработчиками встает задача построения эффективной системы моделирования проектов сложных СнК, которая бы соответствовала современным методологиям разработки и тестирования. В статье рассмотрены особенности использования инструментов из пакета Cadence Incisive совместно с CMake для управления сборкой и тестированием проекта сложной СнК, способы сокращения временных затрат на стадии компиляции и элаборации проекта, при использовании методики “Multiple Snapshot Incremental Elaboration” (MSIE) совместно с многопоточной компиляцией/элаборацией и приведено сравнение временных затрат.

Ключевые слова: современные системы на кристалле, программные компоненты, СБИС, система сборки

1. Введение

По мере роста сложности современных систем на кристалле (СнК) и количества используемых аппаратных блоков, перед разработчиками встает задача построения эффективной системы моделирования проектов сложных СнК, которая бы соответствовала современным методологиям разработки и тестирования.

На сегодняшний день, проект сложной СнК, помимо непосредственно RTL описания и стандартных библиотек, содержит большое количество программных компонентов, которые разрабатываются совместно с СнК и потому должны собираться и тестироваться вместе в ходе разработки.

Для решения поставленной задачи в дополнение к инструментам из пакета Cadence Inscisive предлагается использовать семейство инструментов CMake.

CMake – семейство инструментов с открытым исходным кодом, разработанное и поддерживаемое компанией Kitware. В состав CMake входят инструменты для сборки, тестирования и подготовки дистрибутивов программного обеспечения. Гибкий язык описания процесса сборки позволяет интегрировать в CMake поддержку языка Verilog/SystemVerilog и использовать CMake как единую платформу для сборки программных компонентов, модели СнК, а также запуска всего спектра имеющихся тестов.

В статье будут рассмотрены особенности использования инструментов из пакета Cadence Incisive совместно с CMake для управления сборкой и тестированием проекта сложной СнК, способы сокращения временных затрат на стадии компиляции и элаборации проекта, при использовании методики “Multiple Snapshot Incremental Elaboration” (MSIE) совместно с многопоточной компиляцией/элаборацией и приведено сравнение временных затрат.

2. Требования к системе сборки проекта

Возрастающая сложность СБИС накладывает набор серьезных требований на организацию процесса разработки и отладки. Система сборки должна:

- Иметь возможность параллелизовать процесс компиляции и элаборации исходного кода СБИС для сокращения времени ожидания и максимально полного использования ресурсов многопроцессорных систем.
- Предоставлять инструменты для естественного разбиения сложного проекта на под-проекты для комфортного пере-использования кода IP блоков.
- Поддерживать рекомендованную для сложных СБИС Cadence технологию Multiple Snapshot Incremental Elaboration (MSIE), существенно сокращающую время элаборации проектов сложных СБИС.
- Иметь хорошую встроенную поддержку сборки программных компонентов.

3. Почему cmake?

На сегодняшний день существует огромное количество систем сборок проектов, начиная от низкоуровневых ninja, nmake, GNU Make и до более высокоуровневых, таких как cmake, tup, maven.

Ряд возможностей cmake делает его особенно привлекательным для описания процесса сборки СБИС:

- CMake предоставляет пользователю высокоуровневый язык, на основе которого генерируется сценарий для более низкоуровневой системы сборки (GNU/Make, ninja).
- В CMake существует встроенная система запуска тестов (ctest) для которая поддерживается существующими платформами для непрерывной интеграции (напр. Jenkins)
- Наличие системы подпроектов позволяет осуществлять гибкое разбиение проекта сложной СБИС на подпроекты.
- CMake изначально поддерживает сборку вне директории исходных кодов, что позволяет использовать несколько директорий сборки (например, для различных конфигураций проекта) для одного дерева исходных кодов проекта.

4. Добавление поддержки Verilog/SystemVerilog и инструментов Cadence Incisive в CMake

На момент написания данной статьи в CMake отсутствует поддержка встроенной сборки проектов, использующих языки Verilog HDL и/или SystemVerilog. Ее можно добавить в CMake одним из двух способов.

Первый способ, это добавление нового «языка программирования» в cmake. Это позволит использовать стандартные функции add_executable() и add_library() для описания компонентов проекта.

Второй способ - добавить шаги, необходимые для сборки проекта используя add_custom_target()/add_custom_command().

Использование первого подхода сопряжено с рядом проблем. Первая и самая очевидная проблема со стороны cmake – то, что процесс сборки проекта ориентирован на сборку ПО. Схематично, процесс сборки проекта, который реализует cmake представлен на Рисунке 1.

Добавление поддержки нового языка программирования или компилятора существующего языка заключается в реализации шаблонов правил для каждого из шагов на Рисунке 1.

CMake так же ожидает от компилятора информации о зависимостях между файлами для определения того, какие цели необходимо обновлять при повторной сборке проекта.

Подобный подход плохо соотносится с инструментарием Cadence Insicive, которые используют общую базу данных для хранения результатов компиляции и элаборации, храняшщюся в директории Inca.libs.

Схематично, процесс сборки RTL инструментами Cadence Insisive представлен на Рисунке 2.

RTL описание компилируется во внутреннее представление и сохраняется в базу данных при помощи инструмента ncvlog для языка Verilog и ncvhdl для языка VHDL соответственно.

Инструмент ncelab производит элаборацию и генерирует слепок (snapshot) для моделирования при помощи ncsim.

Учитывая вышесказанное, поддержку языка Verilog HDL рациональнее

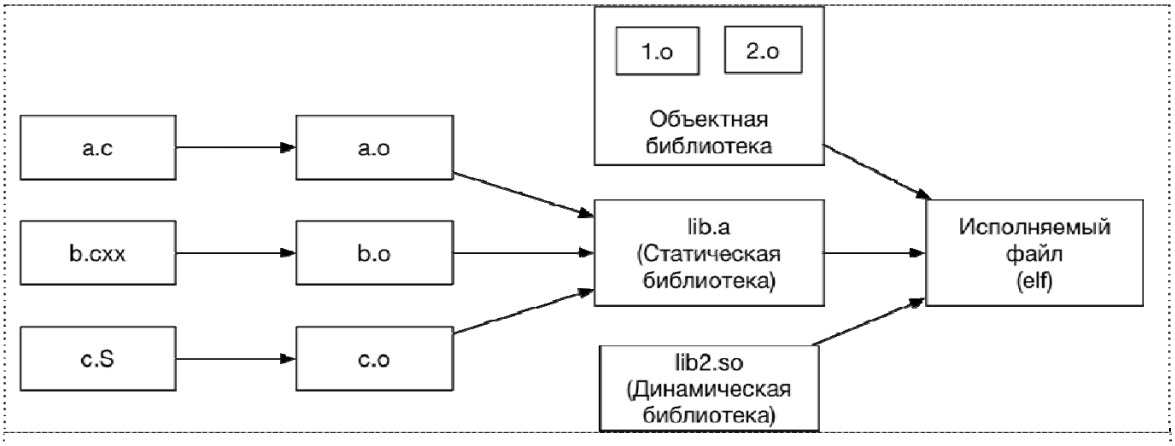


Рисунок 1. Процесс сборки C/C++ проекта при помощи CMAKE

реализовывать, добавляя дополнительные пользовательские цели в cmake.

Для упрощения работы компания Cadence предоставляет инструмент irun, автоматизирующий вызовы вышеупомянутых инструментов и позволяющий запустить компиляцию, элаборацию и симуляцию проекта СБИС одной командой.

Исходя из вышеописанных различий, для добавления поддержки Cadence Incicive был выбран второй подход. Для удобства работы, процесс сборки был разбит на два этапа, которые можно вызывать независимо. Вместо использования irun как для выполнения компиляции, так и элаборации irun используется только на этапе компиляции библиотек, а для элаборации производится вызов ncelab.

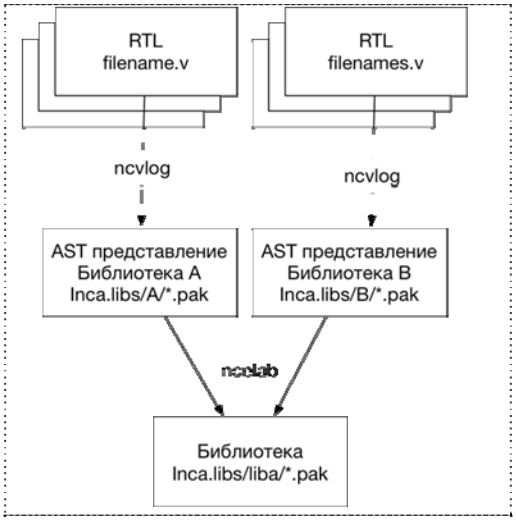


Рисунок 2. Процесс сборки HDL проекта при помощи Cadence Insicive (упрощенно)

5. Многопоточная компиляция

Инструменты Cadence используют общий файл базы данный для хранения результатов компиляции. Для организации доступа к базе данных используется система блокировок. Если процесс ncvlog или ncvhdl попытается получить доступ к библиотеке, используемой эксклюзивно другим процессом, то он будет вынужден ожидать. Ускорить процесс компиляции, используя несколько процессов компилятора в данном случае возможно только если параллельно будут компилироваться несколько библиотек, как показано на Рисунке 2. Более того, если в составе библиотеки существует инстанциирование модулей, объявленных в других библиотеках, то процесс компиляции будет блокировать соответствующую библиотеку при компиляции, что негативно скажется на времени сборки.

6. MSIE и многопоточная элаборация

Для разработки сложных СБИС, рекомендуется Cadence рекомендует разбивать проект на несколько частей и проводить элаборацию каждой из них отдельно. Такая техника носит название MSIE, или Multiple Snapshot Incremental Elaboration. При этом вводится два типа слепков для симуляции.

- Симуляционный слепок (Simulation snapshot). Может быть запущен при помощи симулятора ncsim
- Primary Snapshot, создается отдельно и используется только для подстановки при элаборации слепков.

Использование такого подхода позволяет при изменении одного из файлов проводить повторную элаборацию слепка, в состав которого входит этот файл и слепков, использующих обновленный слепок. Более того, если фрагмент проекта, из которого был создан слепок используется в нескольких экземплярах, то это позволит проводить элаборацию этого фрагмента лишь один раз. Этот процесс так же можно параллелизовать, используя несколько копий ncelab для проведения элаборации различных слепков.

При использовании данной методики необходимо учитывать несколько особенностей работы инструментов.

1. Модуль проекта, из которого создается первичный слепок не может быть параметризованным.
2. При очень большом дроблении проекта на первичные слепки, накладные расходы на компоновку первичных слепков могут превысить выигрыш по времени, полученный от использования многопоточной элаборации.
3. Если проект содержит несколько идентичных экземпляров фрагмента проекта, из которого был создан первичный слепок, то элаборация первичного слепка будет проводится один единственный раз.

7. Анализ времени компиляции и элаборации

Для сравнительного анализа времени затраченого на компиляцию и элаборацию проекта в различных вариантах сборки

использовался проект небольшой СБИС, содержащий процессорное ядро ARM, шину AXI, SRAM память, ROM память, набор стандартных блоков системного таймера, сторожевого таймера, а так же переферийные блоки Ethernet, i2c, spi. IP блоки были реализованы на языках Verilog и VHDL,SystemVerilog в данном тесте не использовался.

	Компиляция	Компиляция и элаборация	Повторная элаборация
Без MSIE	22.84	51.38	17.74
MSIE, 1 поток	22.89	61.37	7.78
MSIE, 8 потоков	20.13	28.34	5.2

Таблица 1. Время (в секундах) компиляции/элаборации проекта СБИС в различных вариантах сборки

Проект СБИС собирался в один симуляционный слепок. Входящие в состав СБИС IP блоки либо собирались в первичные слепки (MSIE) в один или восемь параллельных потоков, либо (Flat) этого не делалось, и собирался только симуляционный слепок.

Для измерения времени, затраченного на перекомпиляцию/переэлаборацию, в один из файлов IP блока вносилось несущественное изменение (переименование внутреннего сигнала).

Хост системой, где производилась компиляция и элаборация был компьютер на базе процессора Intel(R) Core(TM) i7-2630QM CPU с четырьмя физическими ядрами работающими на тактовой частоте 2.0 Ghz и 8GB оперативной памяти.

Сравнительные результаты компиляции и элаборации проекта представлены в Таблице 1.

Из полученных данных можно сделать следующие выводы:

- Использование MSIE при однопоточной сборке немного медленнее, нежели сборка непосредственно слепка для симуляции.
- Параллельная компиляция и элаборация дает наибольший выигрыш во времени.
- Несмотря на большое количество потоков, суммарная загрузка всех процессорных ядер не превышала 40% из-за ожидания блокировки базы данных процессами ncelab/ncvlog.
- Использование многопоточной сборки совместно с MSIE дает наибольший выигрыш во времени как при начальной компиляции/элаборации, так и частичной пересборке проекта.

8. Описание проекта при помощи CMakeLists.txt

Для описания проекта используется стандартный для cmake файл CMakeLists.txt, находящийся в корневой директории проекта. Стандартные функции cmake были расширены дополнительным набором функций с префиксом hdl_, чтобы предоставлять пользователю высокоуровневый язык для описания проекта.

Исходные файлы компилируются в библиотеки, добавляемые вызовами hdl_add_library().

Далее, из скомпилированных библиотек можно создавать первичные и симуляционные слепки используя вызовы hdl_add_primary_snapshot() и hdl_add_snapshot() соответственно.

Для добавления тестов в общий список тестов можно воспользоваться как стандартным вызовом hdl_add_test().

Один и тот же слепок может быть использован для запуска нескольких тестов.

Для каждой создаваемой библиотеки можно задать директории для поиска включаемых файлов через функцию hdl_include_directories(), а так же определить значение макросов через hdl_add_definitions().

Эти опции будут использованы только при компиляции указанных первым аргументом библиотек.

Пример простого проекта, описывающего сборку блока сторожевого таймера в первичный слепок, для дальнейшего использования в более крупном проекте представлен на Листинге 1.

В случае сборки вне родительского проекта, так же генерируется симуляционный слепок.

Одним из преимуществ cmake является наличие возможности включать в сборку под-проекты используя вызов add_subdirectory(). Это позволяет:

1. Размещать разрабатываемые IP блоки в отдельных репозиториях.
2. Использовать одинаковое описание файлов блока как для сборки блока в составе СБИС, так и для сборки в независимом окружении для разработки и тестирования отдельных блоков.

Для подключения проекта, пример описания которого приведен на Листинге 1, в родительском проекте достаточно вызвать функцию hdl_add_unit(), единственным аргументом которой будет служить путь к каталогу с проектом watchdog относительно файла CMakeLists.txt.

```
cmake_minimum_required(VERSION 2.8)
set(CMAKE_MODULE_PATH
${CMAKE_MODULE_PATH}
${CMAKE_SOURCE_DIR}/cmake/Modules)

project(watchdog)
include(CMakeVerilogRules)

hdl_add_library(watchdog_main
   rtl/Watchdog.v
   rtl/Wdog_Wrap.v
)

hdl_add_primary_snapshot(watchdog_main.Watchdog)

if (NOT HDL_IS_SUBUNIT)
   hdl_add_library(watchdog_tb
      test/wdt_tb.v
   )

   hdl_add_snapshot(testbench  watchdog_tb.wdt_tb)
   hdl_link_snapshots(testbench watchdog_main.Watchdog)
   hdl_add_test(watchdog_test_t1 testbench +param=v1)
   hdl_add_test(watchdog_test_t2 testbench +param=v2)
endif()

hdl_print_project_summary()
```

Листинг 1. Пример CMakeLists.txt для описания сборки блока сторожевого таймера

9. Области видимости определений макросов и путей поиска включаемых файлов

По мере роста объема проекта и активном использовании IP блоков, разработанных другими компаниями, возрастает опасность конфликта имен файлов в директориях поиска, имен модулей и/или макросов.

При использовании простого подхода и задания файлов проекта при помощи списков файлов (сгенерированными через nclaunch или вручную), и последующей передачи через опции командной строки –F/-f инструменту irun, макро-определение, объявленное в начале списка файлов будет определено до самого конца списка.

Аналогичная ситуация будет директориями поиска файлов (+INCDIR+). Использование подпроектов cmake позволяет исключить такую ситуацию, ограничивая область действия макро-определений и директорий поиска файлов конкретными библиотеками.

10. Непрерывная интеграция с использованием Jenkins CI

Современные методологии разработки и тестирования предполагают использование платформы непрерывной интеграции для периодического автоматизированного тестирования кодовой базы проекта, для выявления потенциальных проблем на самой ранней стадии. В идеальном случае, каждое изменение, перед интеграцией в кодовую базу проекта должно проходить набор автоматизированных тестов для исключения возможных проблем.

К сожалению, при разработке СБИС, полный запуск всего набора тестов на каждое внесенное изменение, зачастую, невозможен из-за значительных временных затрат, которые потребуются для тестирования в случае большого проекта.

Однако, минимальные проверки непосредственно перед внесением изменений выполнить возможно и необходимо. В зависимости от сложности проекта и доступных вычислительных ресурсов это может быть:

- Только компиляция проекта, без элаборации, для выявления очевидных синтаксических ошибок.
- Компиляция и элаборация проекта, без запуска тестов.
- Компиляция, элаборация и запуск минимального набора тестов.

Использование для выполнения этих проверок многопоточной сборки и элаборации, как было описано выше, позволяет сократить время ожидания разработчиком решения о принятии его изменений в кодовую базу проекта.

Jenkins – популярная платформа для непрерывной интеграции с открытым исходным кодом.

Существующие расширения делают ее особенно привлекательной для использования совместно с инструментами Cadence при проектировании СБИС:

Наличие поддержки ctest (расширение xUnit).

Наличие поддержки анализа предупреждений и ошибок Cadence Incisive в журнале сборки (Расширение warnings-plugin)

Using cmake family of tools for building System-On-Chip projects using Cadence Incisive

A.V. Andrianov

Abstract: With increasing size and complexity of modern 'System-On-Chip' designs developers are required to create effective project build and simulation systems that would match modern methodologies of development and testing. This article covers advanced usage of Cadence Incisive tools along with cmake for building and testing a complex 'System-On-Chip' project. It describes methods of reducing compilation and elaboration time using "Multiple Snapshot Incremental Elaboration" (MSIE) along with multi-threaded compilation/elaboration and provides benchmarks for comparison.

Keywords: cmake, cadence, incisive enterprise simulator, msie, SoC, build system

Литература

1. Scott Chacon (et al.), Pro Git, Apress, 2014.
2. Ken Martin, Mastering CMake, Kitware, 2015
3. Cadence Incisive Enterprise Simulator Refererence Manual, Cadence 2015
4. Reducing Snapshot Creation Turnaround for UVM/SV Based TB Using MSIE Approach, Cadence User Conference 2015, https://www.cadence.com/downloads/cdnlive/in/2015/VER2.pdf (Дата обращения 19.02.2017)
5. Multiple Snapshot Incremental Elaboration, DVClub 2013, http://agnify.com/wp-content/uploads/MSIE_FSL.pdf (Дата обращения 19.02.2017)
6. CMake: The Cross Platform Build System, Tanner Lovelace, Linux Magazine, Июль 2006 http://clubjuggler.livejournal.com/138364.html (Дата обращения 19.02.2017)
7. CMake Documentation, https://cmake.org/documentation/ (Дата обращения 19.02.2017)

Построение и визуализация сечения изоповерхности насыщенности вытесняемой жидкости в пористой среде

М. В. Михайлюк[1], П. Ю. Тимохин[2], А. В. Мальцев[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]mix@niisi.ras.ru , [2]webpismo@yahoo.de [3]avmaltcev@mail.ru

Аннотация: В данной работе предлагается метод построения и визуализации плоского сечения поверхности уровня (изоповерхности) насыщенности вытесняемой вязкой жидкости из образца пористой среды. Визуализация усеченной полигональной модели изоповерхности осуществляется с помощью модифицированного метода «шагающих кубиков» (Marching Cubes) в масштабе реального времени с использованием возможностей современных графических карт. Для реализации предложенных алгоритмов создан программный комплекс с эргономичным пользовательским интерфейсом.

Ключевые слова: вытеснение, фильтрация, визуализация, сечение, поверхность уровня

Введение

При добыче жидких полезных ископаемых (нефти, газоконденсата и т.д.) важными процессами являются фильтрационные течения вязких жидкостей в пористых средах. Нефтеотдача нефтеносных пластов напрямую связана со степенью вытеснения нефти с помощью воды или полимерных растворов.

Вытеснение более вязкой жидкости менее вязкой жидкостью из пористой среды является неустойчивым процессом. Эта неустойчивость приводит к нарушению поверхности раздела фаз и захвату вытесняемой жидкости внутри пористой среды, что существенно снижает качество вытеснения. Имеющиеся в пористой среде неоднородности проницаемости также могут приводить к формированию трещин и каналов в пористой матрице, что увеличивает неоднородность проницаемости [1, 2].

Для анализа результатов моделирования описанного процесса одной из важных задач является визуализация поверхности уровня (изоповерхности) насыщенности вытесняемой жидкости. В работе [3] предложена реализация метода «шагающих кубиков» (Marching Cubes) для построения и визуализации в расчетном объеме полигональной модели изоповерхности в масштабе реального времени. При этом обеспечивается возможность on-line перемещения пользователем виртуальной камеры с целью изучения 3D-модели изоповерхности с разных сторон.

В настоящей работе для повышения эффективности анализа модели изоповерхности предлагается дополнить разработанное решение методом сечений расчетного объема плоскостью, который позволяет увидеть внутреннюю структуру поверхности уровня насыщенности.

1. Построение сечения изоповерхности в расчетном объеме

Изучаемая область пористой среды имеет форму прямоугольного параллелепипеда, заполненного вытесняемой жидкостью (см. рис. 1). Через одну из граней подается жидкость меньшей вязкости. Выход осуществляется через противоположную грань области. На остальных гранях задано условие непротекания. Система уравнений, описывающая прохожде-

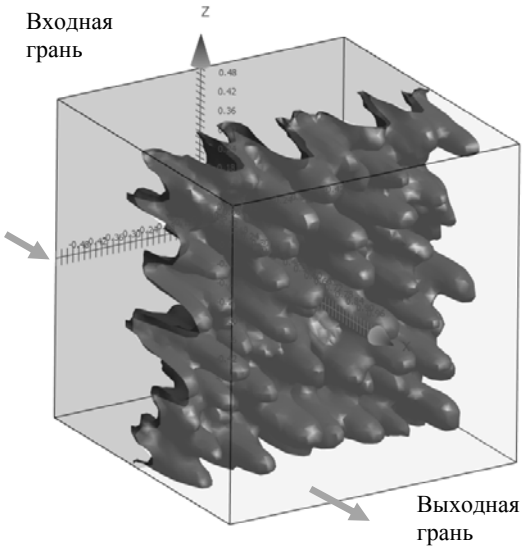


Рис. 1. Объем моделирования

ние вытесняющей жидкости (например, воды) через пористую среду, заполненную вытесняемой жидкостью (например, нефтью), а так-

же ее численное решение по расчетной сетке подробно описаны в [4, 5]. Результатами расчетов и исходными данными для визуализации являются трехмерные массивы вещественных чисел, задающих значения насыщенности s вытесняемой жидкости в ячейках объема моделирования. Каждый массив записан в бинарном файле и соответствует определенному моменту времени t моделирования. Размеры массива совпадают с размерами расчетной сетки, а значениями элементов массива являются насыщенности $s_{i,j,k}$ вытесняющей жидкости в ячейках (i, j, k) расчетной сетки.

В специально разработанной программной оболочке [3] методом Marching Cubes строится и визуализируется полигональная модель изоповерхности уровня для заданного пользователем значения $s = s^*$ насыщенности.

Плоскость сечения T определяется уравнением $ax + by + cz - d = 0$, где d - расстояние от плоскости до начала координат, а $\vec{N} = (a, b, c)$ задает вектор нормали к плоскости. Мы будем считать, что этот вектор нормализован, т.е. имеет единичную длину. Плоскость делит пространство на два полупространства. То из них, в которое направлена нормаль, будем называть положительным, а второе – отрицательным. Пересечение плоскости T с расчетным объемом образует плоскую фигуру с числом углов от 3 до 6 (см. рис. 2).

Зафиксируем некоторые значения индексов i и j, и рассмотрим множество ячеек $C_{i,j} = \{C_{i,j,k}, 0 \le k \le n_x\}$, где $n_x + 1$ - число ячеек по оси $\vec{x}$. Считаем для простоты, что каждая ячейка является кубиком со стороной Δ.

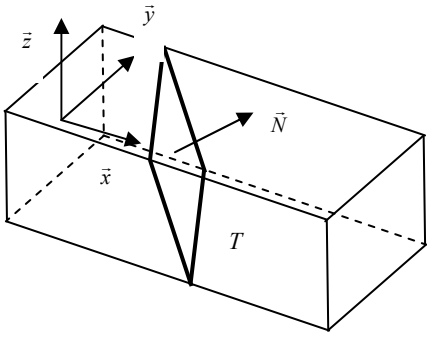


Рис. 2. Сечение расчетного объема

Нам необходимо визуализировать только ту часть, которая попадает в отрицательное полупространство и пересекает само сечение. Вычислим граничные значения k, задающие эту часть. Для более понятного описания рассмотрим двумерный случай. На рис. 3 показана сетка рендеринга в 2D, а также прямая сечения и нормаль к ней. У этой нормали $N_x \ge 0, N_y \ge 0$ (здесь ось $\vec{x}$ направлена горизонтально, а ось $\vec{y}$ вертикально). Нам необходимо визуализировать только ту часть модели, которая находится ниже линии сечения или

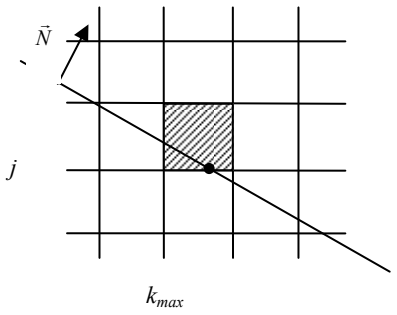


Рис. 3. Граница визуализации j - й полосы

пересекает ее. Для j -й полосы квадратов (т.е. для квадратов с вершинами $(k\Delta, j\Delta)$, $((k+1)\Delta, j\Delta)$, $((k+1)\Delta, (j+1)\Delta)$, $(k\Delta, (j+1)\Delta)$) это будет множество квадратов с k, меняющимся от $k_{\min} = 0$ до $k_{\max}$, которое мы сейчас вычислим. Как легко видеть из рисунка 3, обрабатывать надо все квадраты, расположенные левее серого и сам серый квадрат. Правее него квадраты целиком попадают в положительную полуплоскость и их обрабатывать не надо. Если точка $(0, j\Delta)$ лежит в положительной полуплоскости, то и вся j - я полоса лежит в ней, поэтому ее визуализировать не нужно и получаем $k_{\max} = 0$. Если точка $(n_x\Delta, j\Delta)$ лежит в отрицательной полуплоскости, то вся полоса лежит в ней и $k_{\max} = n_x$. В остальных случаях линия сечения пересекает j - ю полосу и $k_{\max}$ будет равно максимальному значению k, для которого точка $(k\Delta, j\Delta)$ лежит левее точки пересечения линии сечения с прямой $y = j\Delta$. Эта точка имеет координаты $((d - n_y j\Delta)/n_x, j\Delta)$. Таким образом, получаем

$$k\Delta < \frac{d - n_y j\Delta}{n_x} \text{ или } k < \frac{d - n_y j\Delta}{n_x \Delta}.$$

Наибольшее целое значение k, удовлетворяющее этому неравенству будет равно

$$k_{\max} = \left\lceil \frac{d - n_y j\Delta}{n_y \Delta} \right\rceil - 1.$$

Аналогично рассматриваются остальные три случая: когда $N_x \ge 0, N_y < 0$, $N_x < 0, N_y \ge 0$ и $N_x < 0, N_y < 0$. Для трехмерного случая таким же образом можно вычислить значения $k_{\min}$ и $k_{\max}$ для (i, j) - й полосы

и различных знаков координат N_x, N_y, N_z нормали (т.е. различных ее направлений).

2. Визуализация сечения полигональной модели изоповерхности

Построение сечения изоповерхности в расчетном объеме включает формирование полигональной модели изоповерхности в отрицательном полупространстве и на границе сечения, т.е. во всех ячейках с номерами (i, j, k) для $k_{\min,i,j} \le k \le k_{\max,i,j}$. Для этого воспользуемся методом Marching Cubes построения изоповерхности трехмерного скалярного поля (см. [3, 6, 7]). Для простоты будем считать, что сетка, в ячейках которой строится полигональная модель изоповерхности (сетка рендеринга), совпадает с расчетной сеткой насыщенности вытесняемой жидкости. Вершины каждой ячейки сетки рендеринга пронумеруем целыми числами от 0 до 7, а ребра - от 0 до 11. Вершинам, для которых значения соответствующих им ячеек расчетной сетки превышают s^*, припишем битовое значение b_V равное 1. Оставшимся вершинам – значение $b_V = 0$. Тогда любую конфигурацию из 0 и 1, приписанных вершинам каждой ячейки, можно представить в виде 8 битов, т. е. одного байта, который мы будем обозначать $K_{i,j,k}$. Каждой такой конфигурации ячейки сетки рендеринга однозначно соответствует набор треугольников (полигонов), который будет являться частью строящейся изоповерхности. Этот набор запишем в массив T как элемент с номером $K_{i,j,k}$. Вершины полигонов набора находятся только на тех ребрах ячеек сетки, на концах V_1 и V_2 которых $b_{V_1} \ne b_{V_2}$, при этом положение этих вершин определяется с помощью линейной интерполяции на основе значений s^*, $s(V_1)$ и $s(V_2)$. Алгоритм построения полигонов таков, что на каждом ребре ячейки будет не более одной вершины полигона, и полигоны соседних ячеек будут иметь общие вершины или ребра. Тем самым, полученная в результате полигональная модель будет замкнутой и не содержащей дыр. Для реализации алгоритма необходимы следующие структуры данных:

- Массив M_E, элементы которого соответствуют ребрам сетки рендеринга. Первоначально он пустой, а затем в те элементы, которые соответствуют ребрам, содержащим вершину полигона, запишутся координаты этой вершины и нормаль $\vec{N}$ в ней.
- Массив U из 256 элементов, соответствующих всевозможным битовым конфигурациям ячеек рендер-сетки. В каждом элементе массива U записаны локальные номера ребер треугольников (от 0 до 11) участка изоповерхности для такой конфигурации.
- Массив C_K размером равный числу ячеек сетки рендеринга, элементами которого являются значения конфигураций $K_{i,j,k}$ этих ячеек для $k_{\min,i,j} \le k \le k_{\max,i,j}$.
- Массив Π всех полигонов модели усеченной изоповерхности, в который будут записываться полигоны участков изоповерхности внутри ячеек сетки рендеринга с индексами $k_{\min,i,j} \le k \le k_{\max,i,j}$. Каждый полигон представлен индексами трех вершин – элементов массива M_E, причем вершины перечислены в таком порядке, который однозначно определяет направление нормали к полигону, соответствующее его лицевой стороне.

Заметим, что координаты конкретных вершин или ребер сетки рендеринга хранить не надо, т.к. их легко вычислить, зная размер ячейки. Таким образом, мы получаем следующий

Алгоритм построения полигональной модели усеченной изоповерхности

Начало.

1. Вычисляем значения $k_{\min,i,j}$ и $k_{\max,i,j}$ для всех $i \in [0, n_y], j \in [0, n_z]$.
2. Создаем пустой массив M_E по числу всех ребер сетки (в порядке перебора индексов (i, j, k)).
3. Для каждой вершины V_1 сетки вычисляем значение b_V.
4. Для каждой ячейки $C_{i,j,k}$, где $k_{\min,i,j} \le k \le k_{\max,i,j}$ сетки рендеринга вычисляем конфигурацию $K_{i,j,k}$ и записываем в массив C_K.
5. Цикл по всем ребрам $E_{i,j,k}$ сетки рендеринга для $k_{\min,i,j} \le k \le k_{\max,i,j}$

   Если для вершин V_1 и V_2 этого ребра $b_{V_1} \ne b_{V_2}$, то вычисляем на ребре точку

   $$P_{E_{i,j,k}} = V_1 + \frac{s^* - s(V_1)}{s(V_2) - s(V_1)}(V_2 - V_1).$$

   Записываем координаты $P_{E_{i,j,k}}$ в элемент массива M_E, соответствующий

$E_{i,j,k}$, и ставим флаг, что это ребро содержит точку.

Конец цикла.

6. Цикл по всем элементам $C_{K,i,j,k}$ массива C_K

   Получаем значение $K_{i,j,k}$ элемента $C_{K,i,j,k}$.

   В массиве U выбираем элемент с индексом $K_{i,j,k}$, в котором записаны номера ребер треугольников (полигонов) участка изоповерхности, попадающего в эту ячейку.

   По этим номерам вычисляем соответствующие элементы массива M_E.

   Тройки номеров элементов массива M_E записываем в массив Π полигонов модели усеченной изоповерхности.

   Для каждого записанного полигона вычисляем нормаль и добавляем ее к нормалям $\vec{N}_{V_i}$ вершин этого полигона в массиве M_E.

Конец цикла.

Конец.

После конца работы алгоритма в заполненных элементах массива M_E поле $\vec{N}$ нормали будет содержать сумму нормалей ко всем полигонам, содержащим эту вершину. Так как получившийся вектор может иметь произвольную длину, производим его нормирование.

Визуализация полигональной модели усеченной изоповерхности производится с помощью графической библиотеки OpenGL. С помощью функций этой библиотеки устанавливается и направляется в нужную сторону источник освещения, задаются свойства этого источника, а также свойства материала, с помощью которого моделируются визуальные свойства изоповерхности. Кроме того, задаются некоторые параметры расчета освещенности, например, интенсивности испускаемого фонового, диффузного и зеркального излучения, а также сфокусированность источника. Для материала модели изоповерхности задаются коэффициенты отражения фоновой, диффузной и зеркальной составляющей света, попадающего на лицевую сторону поверхности уровня и степень его зеркального отражения. На рис. 4 показан пример визуализации модели усеченной изоповерхности насыщенности вытесняющей жидкости для $s^* = 0.1$.

Интерфейс включает как изменение ориентации расчетного объема (вместе с сечением), так и перемещение, и изменение ориентации самой плоскости сечения. Повороты расчетного объема вокруг его осей координат осуществляются с помощью мыши: перемещение мыши влево/вправо или вперед/назад (с нажатой левой кнопкой) поворачивает расчет-

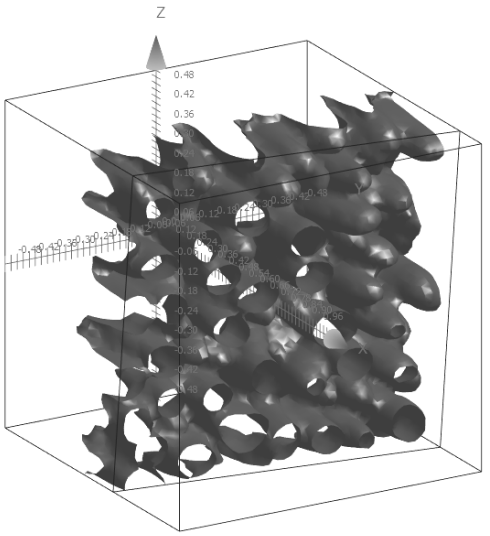


Рис. 4

ный объем вокруг осей Z и Y соответственно. Вращение колесика мыши приводит к приближению или удалению камеры. Параллельное перемещение плоскости сечения и ее повороты производятся с помощью клавиатуры. Повороты изменяют значения соответствующих координат нормали $\vec{N}$ к плоскости сечения. Таким образом, можно строить и визуализировать различные сечения изоповерхности насыщенности и изучать ее внутреннюю структуру в масштабе реального времени. Для ускорения работы алгоритма его целесообразно выполнять на графической карте, используя технологию шейдеров.

Заключение

В данной работе предложен эффективный метод построения и визуализации плоского сечения изоповерхности вытесняемой вязкой жидкости из образца пористой среды.

Исходной информацией для визуализации являлся 3D-массив насыщенностей вытесняющей жидкости, на основе которого строится полигональная модель изоповерхности с помощью модифицированного метода «шагающих кубиков».

Визуализация осуществлялась с применением современных технологий обработки данных на графических ускорителях.

Плоскость сечения можно перемещать и поворачивать в масштабе реального времени, что дает возможность подробнее изучить строение поверхности уровня.

Работа выполнена при поддержке гранта РФФИ № 16-29-15099 офи_м.

Construction and visualization of saturation isosurface cross section for displaced fluid in a porous medium

M. V. Mikhaylyuk, P. Yu. Timokhin, A. V. Maltsev

Abstract: The paper proposes a method of construction and visualization of plane cross section of saturation level surface (isosurface) for viscous fluid displaced from a sample of a porous medium. Visualization of clipped polygonal isosurface model is carried out in real-time using a modified "Marching Cubes" method and modern graphics hardware capabilities. To implement the proposed algorithms, a software with an ergonomic user interface was developed.

Keywords: displacement, filtration, visualization, cross section, isosurface

Литература

1. N.N.Smirnov, J.C.Legros, V.F.Nikitin, E.Istasse, L.Schramm, F.Wassmuth, D.A.Hart. Filtration in artificial porous media and natural sands under microgravity conditions (2003) . Microgravity Science and Technology, 14 (2), pp. 3-28.
2. N.N.Smirnov, V.F.Nikitin, O.E.Ivashnyov, A.Maximenko, M.Thiercelin, A.Vedernikov, B.Scheid, J.C.Legros. Microgravity investigations of instability and mixing flux in frontal displacement of fluids (2004) Microgravity Science and Technology, 15 (2), pp. 35-51.
3. М. В. Михайлюк, П. Ю. Тимохин, А. В. Мальцев, В. Ф. Никитин, Е. И. Скрылева, В. В. Тюренкова. Моделирование и визуализация процесса вытеснения нефти из пористой среды // Вестник кибернетики. – 2016. – № 3. – С. 35-41.
4. N.N.Smirnov, V.F.Nikitin, A,V.Norkin, A.B.Kiselev, J.C.Legros, E.Istasse. Microgravity investigation of capillary forces in porous media // Space Forum. 2000. № 6 (1-4). pp. 1-10.
5. И.В.Козлов, Е.И.Скрылева. Математическое моделирование и обработка эксперимента по вытеснению нефти водой из неокомских песчаников // Вестник кибернетки. 2016. №2. С. 138-145.
6. Marching cubes. URL: https://ru.wikipedia.org/wiki/Marching_cubes (дата обращения: 02.08.2016).
7. Polygonising a scalar field. URL: http://paulbourke.net/geometry/polygonise/ (дата обращения: 02.08.2016).

Имитационное моделирование виртуального робота-кентавра в космических тренажерах

Е.В. Страшнов[1], **М.А. Торгашев**[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]*strashnov_evg@mail.ru,* [2]*mtorg@mail.ru*

Аннотация: Рассматривается задача имитационного моделирования виртуального робота-кентавра в режиме реального времени применительно к космическим тренажерам. Для этого спроектированы и созданы виртуальная сцена участка поверхности Луны и модель робота-кентавра. С помощью разработанного виртуального пульта осуществляется управление движением робота и звеньями его манипуляторов в командном режиме. Для апробации применяемых методов и алгоритмов моделирования робота-кентавра рассмотрены базовые операции, которые робот в состоянии выполнить на поверхности Луны.

Ключевые слова: антропоморфный робот-кентавр, виртуальная сцена, поверхность Луны, имитационно-тренажерный комплекс, реальное время

Введение

Одним из перспективных направлений развития космонавтики является дальнейшее исследование и освоение Луны. В настоящее время во многих странах ведется разработка космических программ, в которых планируется создание стационарной обитаемой лунной базы [1]. Предполагается, что лунная база будет предназначена для проведения научных исследований в области планетологии, астрономии и других дисциплин. Кроме того, особый интерес представляет добыча на Луне полезных ископаемых [2], в том числе редкого для Земли изотопа гелия-3, который в дальнейшем может быть использован в термоядерных реакторах. Однако проблема освоения Луны состоит в том, что длительное пребывание человека на поверхности Луны опасно для его здоровья, так как он становится подвержен воздействию ионизирующего излучения и попаданию метеоритных осколков. Поэтому альтернативным решением является использование роботов [3] вместо человека для выполнения различных работ. Такие роботы могут обладать свойством антропоморфности, то есть быть подобными по строению человеку, что позволит им выполнять ту же самую работу, что и человек. Кроме того, эти роботы должны быть оснащены средствами передвижения для свободного перемещения по поверхности Луны. Примером таких роботов является мобильный робот-кентавр, представляющий собой торсовый антропоморфный робот, установленный на колесную или гусеничную тележку. Предполагается, что при выполнении большинства технологических задач управление таким роботом будет осуществляться человеком-оператором дистанционно. В этом случае оператор должен пройти процесс обучения, чтобы у него сформировались необходимые навыки управления роботами. Однако использование реальных роботов в процессе обучения с одной стороны является нежелательным в силу риска поломки робота при ошибке оператора, а с другой стороны существуют трудности воссоздания на Земле условий и местности, которые характерны для Луны. Предлагаемым решением является обучение операторов в системах виртуального окружения, включающих виртуальные модели роботов в виртуальной среде. Это позволит расширить границы применимости роботов, отработать выполнение конкретных операций, смоделировать нештатные ситуации и т.д.

В данной работе в рамках имитационно-тренажерного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН, рассматривается моделирование виртуального робота-кентавра в виртуальной сцене участка поверхности Луны. Управление роботом осуществляется в командном режиме с помощью виртуального пульта и включает управление движением тележки и всеми звеньями антропоморфного робота. Моделирование динамики включает моделирование отдельных тел, шарнирно-связанных тел, колес, а также определение и разрешение коллизий при пересечении тел. Применяемые методы и алгоритмы управления и моделирования динамики отработаны на технологических операциях перемещения робота и расчистки поверхности Луны.

1. Виртуальная модель участка поверхности Луны

В данной статье рассматривается стадия проектирования виртуальной сцены поверхности Луны без наличия элементов и конструкций, необходимых для строительства лунной базы. Это необходимо для проведения исследований, какие задачи может выполнять робот

на начальной стадии освоения Луны. Поэтому в системе моделирования 3Ds Max была создана виртуальная сцена (см. рис. 1) поверхности Луны размером 465 на 400 метров, представляющая собой равнину (перепады высот не превышают 50 метров) с небольшим количеством кратеров. Кроме того, в сцене содержатся камни различной формы и размера. Исходя из того, что робот в дальнейшем может взаимодействовать с данными камнями, было рассмотрено несколько типов камней: большие камни весом больше 1 тонны, средние камни весом от 100 до 500 кг и маленькие камни весом до 100 кг. В исследовании предполагается, что робот-кентавр в состоянии поднять маленький камень, а средний камень только сдвинуть. Также в сцене содержатся еще мелкие камни (щебень), на которые робот может только наехать.

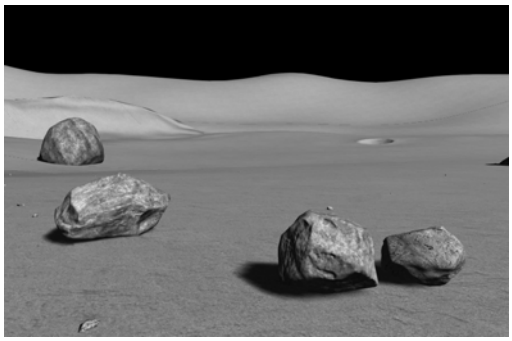


Рис.1. Виртуальная сцена участка Луны

Рельеф поверхности Луны задается с помощью полигональной модели, представляющей собой множество треугольников. Эти треугольники имеют различную величину и форму. Для данной сцены был выбран такой уровень детализации рельефа, что получилось порядка 150000 треугольников. В задаче определения пересечения объектов с поверхностью Луны используется регулярная прямоугольная сетка высот в плоскости XY с шагом 4 см, построенная на основе полигональной модели. Для каждого узла сетки задается материал поверхности и вычисляется высота на основе интерполяции вершин треугольника, в зависимости от того, куда попала точка узла сетки. С алгоритмом построения сетки высот с помощью полигональной модели можно ознакомиться в [4].

Для моделирования динамики роботов и тел необходимо правильно задать материалы, используемые в сцене, так как по ним определяются их свойства: плотность материала, коэффициенты трения скольжения, качения и восстановления для пары материалов и т.д. В виртуальной сцене поверхности Луны используются два типа материалов: лунный грунт (реголит) для поверхности Луны и материал булыжника для камней. Лунный грунт представляет собой рыхлый слой пыли, поэтому характеризуется большим коэффициентом сцепления для различного типа шин. Так для шин с металлическим протектором коэффициент трения скольжения равен 0.9. Кроме того, для данного типа материала характерен малый коэффициент отскока (восстановления) при ударе, и, например, для пары материалов булыжника и грунта с песком данный параметр равен 0.01. При вычислении массы камней используется плотность лунного грунта, которая равна 2600 кг/м^3.

Отдельной задачей при моделировании динамики твердых тел является определение и разрешение коллизий твердых тел. Существующие алгоритмы определения коллизий тел работают для объектов определенной формы (параллелепипеды, сферы и т.д.), в то время как камни в сцене Луны имеют достаточно сложную форму. Поэтому, чтобы определять коллизии камней с другими объектами, их окружают наборами аппроксимирующих боксов (параллелепипедов). При этом существенным условием является использование наименьшего числа таких боксов для заданной точности приближения. Для больших камней достаточно окружить поверхность камня по периметру, не позволяя роботу во время движения проникать внутрь камня, в то время как средние и маленькие камни окружены более детально, что требует в среднем 50 аппроксимирующих боксов.

2. Модель робота-кентавра

Виртуальная модель робота-кентавра (см. рис. 2) создана в системе моделирования 3Ds Max.

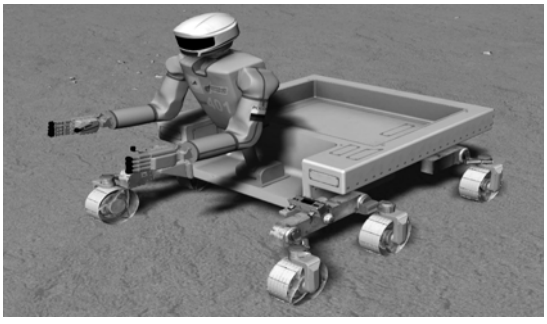


Рис. 2. Виртуальная модель робота-кентавра

Конструкция робота состоит из человекоподобного торса, прикрепленного к колесной платформе. Модель торсового робота по своим визуальным характеристикам полностью соответствует реальному антропоморфному роботу SAR-401. Она состоит из торса, рук и головы. К голове прикреплена виртуальная камера, позволяющая оператору осуществлять контроль над роботом через шлем виртуальной реальности.

Конструкция робота-кентавра предполагает, что его торсовая часть может свободно поворачиваться вокруг оси, позволяя ему класть небольшие предметы в специальное углубление на платформе. Колеса крепятся к платформе с помощью осевых поперечных подвесок.

Динамические характеристики робота задаются набором виртуальных объектов, которые создаются с помощью специального конструктора в системе моделирования 3Ds Max. В модели робота-кентавра задействованы следующие виртуальные объекты: центры масс, двигатели, аппроксимирующие боксы и динамические объекты колес.

Объект «центр масс» служит для задания массы и главных моментов инерции звена. На каждом шаге моделирования тензор инерции тела вычисляется по формуле $I = R^T I' R$, где I' – диагональная матрица главных моментов инерции звена, R – матрица перехода из локальной системы координат звена в мировую систему координат.

Объект «двигатель» предназначен для задания параметров управляемого шарнира робота. К таким параметрам относятся пусковой момент двигателя $M_П$, его скорость холостого хода $\omega_{хх}$, постоянная времени $t_П$, минимальный и максимальный угол поворота в шарнире, максимальный момент трения в шарнире $M_{тр}$.

Параметры двигателя задействованы для вычисления момента τ, развиваемого двигателем [5]:

$$\tau = M_П \left(U - \frac{\omega_{дв}}{\omega_{хх}} \right),$$

где $\omega_{дв}$ – угловая скорость двигателя, U – напряжение, подаваемое на двигатель.

Аппроксимирующие боксы создаются так, чтобы наиболее точно окружить геометрию деталей робота, и в дальнейшем используются для определения и разрешения коллизий робота с объектами виртуального окружения [6, 7].

В динамическом объекте «колесо» задаются характеристики колеса и его подвески. Такими параметрами являются материал протектора колеса, ход подвески колеса, параметры двигателя колеса при наличии привода и параметры рессоры колеса. К параметрам рессоры относятся номинальная масса $m_{ном}$ (масса робота, приходящаяся на данное колесо), коэффициент жесткости рессоры k_s и коэффициент затухания колебаний рессоры k_d. Эти параметры участвуют в вычислении силы контакта колеса F робота с поверхностью [8]:

$$F = m_{ном} g + k_s \Delta z + k_d \Delta \dot{z},$$

где Δz – текущее изменение хода подвески, g – ускорение свободного падения.

В данной статье рассматривается управление роботом-кентавром в командном режиме. Для этого с помощью специальных редакторов созданы виртуальный пульт и функциональная схема управления. Виртуальный пульт управления (см. рис. 3) представляет собой двумерное схематическое изображение торсовой части робота, на котором задано расположение двигателей и камер. Кроме того, пульт содержит джойстик и кнопки стрелок. Джойстиком

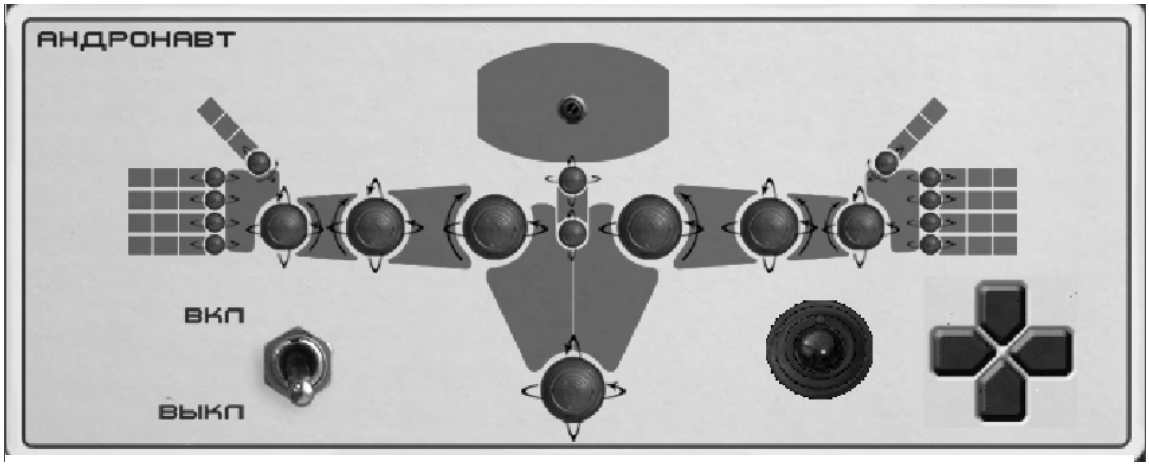


Рис. 3. Виртуальный пульт управления роботом-кентавром

осуществляется управление двигателями в шарнирах, а кнопками стрелок – управление перемещением робота.

Функциональная схема представляет собой набор вычислительных блоков, соединенных между собой. На входы функциональной схемы поступают управляющие сигналы блоков пульта управления, а на выходах формируются сигналы на блоки исполнительных органов (двигателей или колес). На рис. 4 представлен фрагмент функциональной схемы управления роботом-кентавром. В данной схеме блоки кнопок перемещения задают управляющие сигналы, которые передаются в функциональную схему. Функциональная схе-

ма вычисляет управляющие сигналы (напряжения, подаваемые на двигатели колес, и сигналы тормозных муфт) для динамического объекта «колесо».

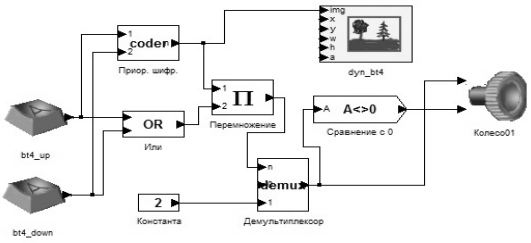


Рис. 4. Функциональная схема управления

Командный режим управления роботом характеризуется тем, что оператор выбирает один или несколько двигателей и, воздействуя на джойстик, управляет движением в выбранных сочленениях. Схема управления сконструирована таким образом, чтобы робот мог схватить объект. Поэтому если выбрать на пульте двигатели локтевых суставов и воздействовать на джойстик, руки робота будут двигаться, либо навстречу друг другу, либо в противоположном направлении. Кроме того, для фиксации захвата в некоторых двигателях срабатывает торможение, обеспечивающее захват объекта роботом. Для управления перемещением робота созданы специальные кнопки на пульте управления, с помощью которых осуществляется движение тележки (вперед и назад) и поворот робота.

3. Выполнение роботом-кентавром технологических задач

Созданные сцены участка поверхности Луны и модели робота-кентавра используются в имитационно-тренажерном комплексе, разработанном в ФГУ ФНЦ НИИСИ РАН и включающем системы управления, моделирования динамики и стерео визуализации. Система управления обеспечивает дистанционное управление роботом в командном режиме с помощью виртуального пульта. Система динамики рассчитывает новые положения и ориентации объектов, а система визуализации обеспечивает синтез этой сцены и вывод ее на экран. Время работы всех систем для формирования одного кадра не превышает 40 мс.

Процесс моделирования робота-кентавра в имитационно-тренажерном комплексе состоит из нескольких этапов:

1. Оператор тренажера воздействует на элемент пульта управления, формируя исполнительные команды. Если это джойстик, то в схему управления передается величина отклонения джойстика.

2. Переданные исполнительные команды обрабатываются в функциональной схеме, которая формирует управляющие команды на исполнительные органы (например, напряжения, подаваемые на двигатели робота).

3. Осуществляется передача данных по информационному протоколу из системы управления в систему динамики.

4. Система динамики определяет моменты двигателей и другие внешние силы.

5. Для каждого тела вычисляется его линейная и угловая скорость с учетом разрешения коллизий и механических связей в шарнирах.

6. Определяются новые координаты и ориентации всех объектов.

7. На основе аппроксимирующих контейнеров определяются коллизии тел.

8. По информационному протоколу передаются мировые матрицы объектов из системы динамики в систему визуализации.

9. Осуществляется визуализация с учетом новых положений и ориентаций объектов.

В рамках данного программного комплекса рассмотрены перечни задач, которые робот может выполнять на поверхности Луны. Среди них были отработаны сценарии, которые включают в себя перемещение робота по поверхности Луны с учетом различных неровностей поверхности и коллизий с камнями, а также расчистку участка Луны от камней.

В силу того, что сила тяжести на Луне примерно в 6 раз меньше, чем сила тяжести на Земле, для обеспечения перемещения робота-кентавра по Луне был выбран металлический протектор колес с дополнительным усилением сцепления. В данной задаче отрабатывается как движение робота, так и возможное столкновение с камнями. Результаты моделирования показали, что в процессе перемещения робот может либо наехать на камень, либо его сдвинуть.

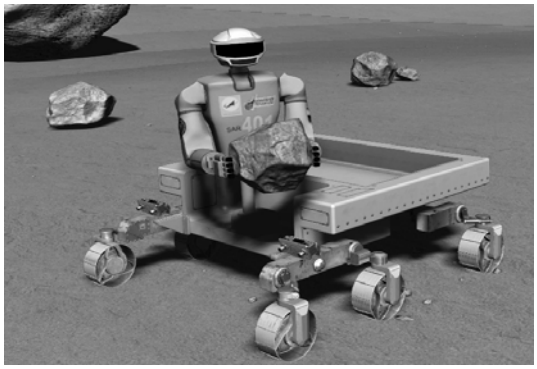


Рис. 5. Захват роботом камня

Задача расчистки участка Луны от камней состоит из несколько этапов. Сначала робот должен подъехать и развернуться так, чтобы его передняя часть оказалась перед камнем.

Робот может захватить только небольшой камень весом порядка 50 кг, и для улучшения захвата на ладонях рук работа используется материал резины. Затем робот берет и кладет камень в тележку и перевозит его в другое место, где вытаскивает камень из тележки и кладет в нужное место. На рис. 5 показана загрузка камня в тележку робота.

Проведенные испытания показали, что рассмотренные задачи выполняются роботом в режиме реального времени с необходимой точностью вычислений.

Заключение

В дальнейшем на предложенной модели робота-кентавра можно отработать более сложные задачи, связанные со строительством и обслуживанием лунной базы. Кроме того, на данной модели можно отработать другие типы управления, например, копирующий и супервизорный.

Исследование было выполнено при поддержке РФФИ в рамках научного проекта № 17-07-00137.

Simulation of a virtual centaur robot in the space training complexes

E.V. Strashnov, M.A. Torgashev

Abstract: This paper considers the problem of real-time simulation of virtual centaur robot with regard to space training complexes. For this, a virtual scene of the lunar surface area, and a centaur robot model were designed and created. Using the developed virtual remote controller, the command mode for motion control of robot and links of its manipulators was implemented. For approbation of the applied centaur robot simulation methods and algorithms, the basic operations that robot is able to perform on the lunar surface are considered.

Keywords: anthropomorphic centaur robot, virtual scene, Moon surface, training complex, real time

Литература

1. А.Н. Перминов, Н.Ф. Моисеев, Н.Н. Севастьянов, Н.А. Брюханов, Г.А. Сизинцев, В.В. Синявский, Б.И. Сотников, С.Ф. Стойко. Перспективы освоения Луны // Известия РАН. Энергетика, 2006, № 1, стр. 3-14.
2. Н.А. Брюханов, В.П. Легостаев, А.А. Лобыкин, В.А. Лопота, Г.А. Сизенцев, В.В. Синявский, Б.И. Сотников, И.М. Филиппов, В.В. Шевченко. Использование ресурсов Луны для исследования и освоения Солнечной системы в XXI веке // Космическая техника и технологии, 2014, том 4, № 1, стр. 3-14.
3. А.Н. Тимофеев, И.В. Шардыко. Проблемы применения в космосе антропоморфных роботов // Космическая робототехника, 2013, № 1, стр. 37-41.
4. М.В. Михайлюк, П.Ю. Тимохин. Моделирование поверхности виртуального полигона // Труды НИИСИ РАН, 2016, том 6, № 2, стр. 38-41.
5. Е.В. Страшнов, М.А. Торгашев. Моделирование динамики электроприводов виртуальных роботов в имитационно-тренажерных комплексах // Издательство “Новые технологии”, Мехатроника, автоматизация, управление, 2016, том 17, № 11, стр. 762-768.
6. М.В. Михайлюк, А.М. Трушин. Расчет коллизий прямоугольных параллелепипедов в задачах динамики // Труды НИИСИ РАН, 2012, том 2, № 2, стр. 48-55.
7. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов // Труды НИИСИ РАН, 2014, том 4, № 2, стр. 95-105.
8. М.В. Михайлюк, А.М. Трушин. Моделирование динамики колес в виртуальных сценах // Труды НИИСИ РАН, 2012, том 2, № 1, стр. 47-53.

Моделирование и визуализация порового пространства керна

М.В. Михайлюк[1], А.В. Мальцев[2], М.А. Торгашев[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия

E-mail's: [1]mix@niisi.ras.ru, [2]avmaltcev@mail.ru, [3]mtorg@mail.ru

Аннотация: В работе предлагается метод визуального представления порового пространства произвольного параллелепипеда из объема моделирования исследуемого керна в виде виртуальной трехмерной модели графа. Построение графа выполняется на основе данных из NC файлов, которые содержат информацию о коэффициентах поглощения рентгеновского излучения в образцах керна. Визуализация полигональной модели графа выполняется в масштабе реального времени с использованием технологии VBO. Также рассматривается задача построения графика пористости выделенной части керна.

Ключевые слова: керн, гидроразрыв, граф, визуализация, виртуальная модель, поровое пространство.

Введение

В геологоразведочных работах важным этапом является получение керна. Керн представляет собой столбик извлеченной породы диаметром около 20 см. и может иметь различную длину. Изучение керна позволяет оценить наличие и качество нефтяного месторождения, оценить запасы и условия залегания нефтяных пластов, определить показатели нефте- и водонасыщенности, пористости, глинистости и т.д. Для последующего моделирования исследователи часто строят цифровую модель керна, для чего можно использовать методы рентгеновской микротомографии [1-3]. В этой модели керн разбивается на слои толщиной в несколько микрон и каждый слой представляется в виде двумерного цифрового массива. Числа в этом массиве кодируют значения какого-либо параметра (материала, плотности, нефтенасыщенности и т.д.) в элементарном объеме керна. Таким образом, весь керн можно задать в виде трехмерного массива чисел. Особый интерес представляет анализ порового пространства керна – пустот (возможно, заполненных газом) через которые при добыче будет вытесняться нефть. Одним из удобных для анализа методов является представление порового пространства в виде графа, в котором полости задаются вершинами, а каналы – ребрами. В данной работе предлагается метод визуализации порового графа с помощью графической библиотеки OpenGL и технологии вершинных буферов (VBO – vertex buffer object). Кроме того, рассматривается вопрос построения графика пористости выделенной области цифровой модели керна.

1. Формат представления данных керна

Для проекта «Цифровое месторождение [4, 5] очень важно эффективное числовое представление исходных данных. В связи с этим серьезные требования предъявляются к форматам представления научных данных. Чаще всего они ориентированы на огромные объемы данных различных типов, обеспечивают быстрый доступ к этим данным, допускают использование на различных программных платформах, позволяют добавлять новые типы данных, а также включают описание типов, размеров, наименований данных и т.д. В данной работе мы используем формат NC, разработанный объединением университетов в области исследований атмосферы (University Corporation for Atmospheric Research). С файлами NC можно работать при помощи бесплатных инструментов netCDF от Unidata - набором программных модулей интерфейса на языках программирования Java, C, C++ и др.

Основные компоненты в NC файле включают размерности (dimensions), переменные (variables) и атрибуты (attributes), каждая из которых включает имя и идентификационный номер. Размерности указывают интервалы изменения индексов в многомерных массивах, переменные – имена данных, а атрибуты – их типы. В данной работе мы использовали данные о керне, предоставленные ООО «Системы для микроскопии и анализа». Они состоят из 184 файлов формата NC, каждый из которых содержит значения масштабированного коэффициента поглощения рентгеновского излучения в ячейках сетки размером

2600x2600x40. Эти значения записываются в виде беззнаковых целых 16-рязрядных чисел. Размер квадратной ячейки равен 0,0029895601 миллиметра.

Для доступа к информации, записанной в файлах формата NC в работе использовалась свободная библиотека утилит от Unidata. С ее помощью данные записываются во внутренние структуры системы визуализации.

2. Представление порового пространства в виде графа

При предсказании эффективности работы нефтяных скважин важное значение имеет моделирование гидроразрыва пласта [6] и течения жидкости в керне [7, 8], а также связанная с ними задача изучения структуры порового пространства породы. Для проведения такого изучения предлагается использовать 3D визуализацию порового пространства произвольно выбранного параллелепипеда из объема моделирования керна. Выбор параллелепипеда осуществляется заданием отрезков по каждому измерению. Отрезок задается указанием минимального и максимального номеров ячеек по каждой оси. Поровое пространство определяется как совокупность ячеек выбранного объема, в которых коэффициент поглощения рентгеновского излучения находится в области $D = [0, P]$. Обычно P выбирается равным 11000.

Одним из способов визуального представления порового пространства является отображение его в виде виртуальной трехмерной модели графа, в котором узлы показывают поры, а ребра – каналы между этими порами. В нашей модели каждый узел представляется в виде сферы с диаметром d, зависящим от размера соответствующей ему поры. Ребра-каналы будем визуализировать прямоугольными параллелепипедами с длиной l_c, равной расстоянию между связываемыми узлами, и сечением a x b, позволяющим задавать геометрию каналов.

Перед визуализацией графа необходимо подготовить по данным из NC файла списки для узлов и ребер, которые также будут записаны в файлы. Для файла вершин предлагается следующая структура каждой строки:

$$n, x, y, z, d,$$

где n – уникальный номер узла (в нашей реализации совпадающий с номером строки файла), $n \in [0, S\text{-}1]$, S – общее число узлов; x, y, z – вещественные числа, задающие координаты узла внутри области выбранного параллелепипеда из объема моделирования керна (в глобальной системе координат). Файл ребер будет содержать строки формата:

$$n_1, n_2, a, b,$$

где n_1, n_2 – номера связываемых ребром узлов из списка узлов, $n_1, n_2 \in [0, S\text{-}1]$, $n_1 \neq n_2$.

Процесс отрисовки графа включает этап предобработки и непосредственно визуализацию. В процессе предобработки выполняется чтение информации из файлов узлов и ребер в массивы структур, расположенные в оперативной памяти. Далее для каждого узла генерируется полигональная модель сферы. С целью оптимизации хранения данных и последующего рендеринга, сферы представляются в виде четырехугольных стрипов (*quad strip*). Координаты вершин стрипа для сферы вычисляются по алгоритму, описанному в [9], в зависимости от трех параметров: точки (x, y, z) расположения узла, диаметра d узла и коэффициента h гладкости поверхности. Первые два параметра считываются из соответствующей узлу строки списка узлов. Коэффициент h позволяет регулировать количество вершин и полигонов в модели сферы. Чем он больше, тем больше вершин и полигонов будет содержать модель, а значит, будет более гладкой, и наоборот. В используемом алгоритме число вершин модели равно $2h[h/2]$, где квадратные скобки означают взятие целой части числа. Задавая h, необходимо понимать, что существенное его увеличение приведет к падению скорости визуализации. Среднее значение h, обеспечивающее приемлемый результат, равно 20. Координаты вершин стрипов записываются последовательно для каждой сферы в вершиный массив V_S.

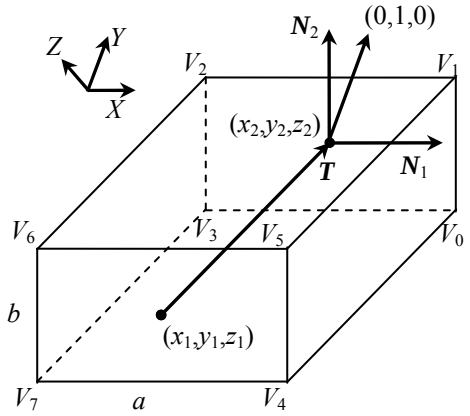


Рис. 1. Модель ребра-канала в виде параллелепипеда

Далее для каждого ребра из списка ребер генерируется полигональная модель параллелепипеда без торцов (рис. 1), имеющего длину

$$l_c = | \boldsymbol{T} | = | (x_2, y_2, z_2) - (x_1, y_1, z_1) |,$$

где (x_1, y_1, z_1), (x_2, y_2, z_2) – точки расположения узлов n_1 и n_2 соответственно, $\boldsymbol{T}$ – вектор от

точки с номером n_1 к точке с номером n_2. Чтобы осуществить построение такой модели, вычислим нормализованные вектора $\boldsymbol{N}_1$ и $\boldsymbol{N}_2$ нормалей к двум ее смежным граням:

$$\boldsymbol{N}_1 = [\boldsymbol{T}_{\text{norm}} \times \boldsymbol{T}_{\text{axis}}] / |[\boldsymbol{T}_{\text{norm}} \times \boldsymbol{T}_{\text{axis}}]|,$$
$$\boldsymbol{N}_2 = [\boldsymbol{N}_1 \times \boldsymbol{T}_{\text{norm}}] / |[\boldsymbol{N}_1 \times \boldsymbol{T}_{\text{norm}}]|,$$

где $\boldsymbol{T}_{\text{norm}} = \boldsymbol{T}/l_c$, $\boldsymbol{T}_{\text{axis}}$ – «опорный» вектор построения. В качестве $\boldsymbol{T}_{\text{axis}}$ в нашем решении выбирается базовый вектор (0, 1, 0), если он не коллинеарен с $\boldsymbol{T}_{\text{norm}}$, и (1, 0, 0) в противном случае. Тогда вершины модели ребра определим по формулам:

$$V_0 = (x_2, y_2, z_2) + 0.5a\boldsymbol{N}_1 - 0.5b\boldsymbol{N}_2;$$
$$V_1 = V_0 + b\boldsymbol{N}_2;\ V_2 = V_1 - a\boldsymbol{N}_1;\ V_3 = V_0 - a\boldsymbol{N}_1;$$
$$V_4 = V_0 - \boldsymbol{T};\ V_5 = V_1 - \boldsymbol{T};$$
$$V_6 = V_2 - \boldsymbol{T};\ V_7 = V_3 - \boldsymbol{T}.$$

Получившийся для ребра параллелепипед так же, как и сферу для узла, можно представить в виде четырехугольного стрипа, который будет состоять из следующей последовательности вершин: V_1, V_5, V_4, V_0, V_3, V_7, V_6, V_2, V_1, V_5. Координаты вершин стрипов для всех ребер графа записываются последовательно для каждого ребра в вершиный массив V_P.

Этап визуализации сгенерированных моделей узлов и ребер предлагается выполнять с использованием технологии вершинных буферов VBO [10]. Она позволяет один раз загрузить в видеопамять все неизменяющиеся данные для рендеринга, а в каждом кадре лишь отсылать команду на визуализацию нужных примитивов, что существенно уменьшает время визуализации. В нашем случае необходимо загрузить в VBO-буферы (заранее созданные с помощью функции *glGenBuffers*) вершинные массивы V_S и V_P, используя функцию *glBufferData*.

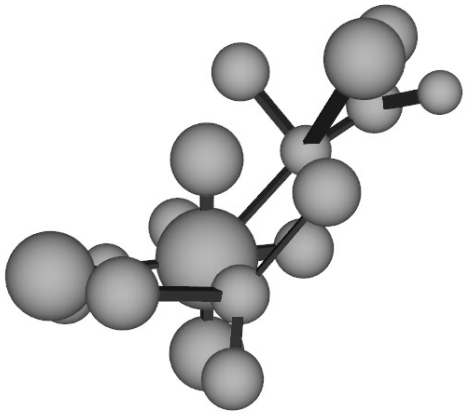


Рис. 2. Пример модели графа порового пространства

Визуализацию виртуальных моделей в виде четырехугольных стрипов, вершины которых будут считываться из VBO-буферов, выполним в цикле по этим моделям с помощью метода *glDrawArrays*. В качестве его параметров будем указывать тип визуализируемого элемента (*GL_QUAD_STRIP*), индекс первой вершины стрипа текущей модели, и количество вершин в одном стрипе. На рисунке 2 представлен пример визуализированного трехмерного графа порового пространства. Узлы и ребра раскрашены в разные цвета.

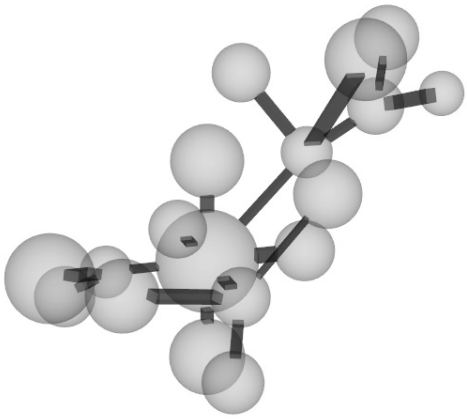


Рис. 3. Модель графа с применением полупрозрачности

Повышение наглядности общей структуры графа при большом числе узлов и ребер можно обеспечить с помощью придания его элементам свойств полупрозрачности. Для этого перед рендерингом необходимо с помощью функции *glEnable* включить возможность использования альфа-канала (параметр *GL_ALPHA*), отвечающего за прозрачность граней, и смешивания цветов перекрывающих друг друга объектов (параметр *GL_BLEND*), а также задать необходимый коэффициент полупрозрачности для материалов моделей. Результат визуализации графа с использованием полупрозрачности показан на рисунке 3.

3. Построение графика пористости

Пористость $\sigma(P)$ выделенного объема для заданного отрезка $[0, P]$ определяется как отношение числа $N_{пор}$ ячеек порового пространства в этом объеме к полному числу N ячеек этого объема, т.е.

$$\sigma = N_{пор} / N .$$

Функция пористости $f_\sigma(\ell)$ определяется как множество значений пористости σ для всех $\ell \le P$. Заметим, что функция $f_\sigma(\ell)$ является монотонно возрастающей, так как поровое пространство $D' = [0, P']$ включает в себя поровое пространство $D'' = [0, P'']$ для всех $P' \le P''$.

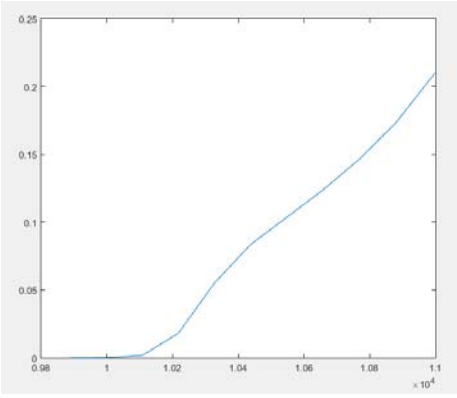


Рис. 4. График пористости
в выделенном объеме

В общем случае для построения графика функции $f_\sigma(\ell)$ на отрезке [0,11000] нужно вычислить ее значение для всех точек этого отрезка. Заметим, однако, что горизонтальное разрешение экрана монитора равно 1024 пиксела, поэтому, в силу монотонности функции $f_\sigma(\ell)$, достаточно вычислить и визуализировать ее значения для аргумента с шагом 10. Разделим отрезок [0,11000] на 1100 частей (каждый длиной 10), посчитаем, сколько ячеек порового пространства попадает в каждую часть, и запишем эти числа в массив $T[i]$ $(i = 0,...,1099)$. Для определения номера части, в которую попадает ячейка, достаточно вычислить целую часть i от деления ее значения на 10 и добавить 1 к элементу $T[i]$. Теперь надо добавить к каждому элементу $T[i]$ значения все предыдущих элементов. Это можно сделать в цикле. Затем визуализируются оси координат (абсцисса – от 0 до 1000, ордината – от 0 до 1) и сам график функции $f_\sigma(\ell)$ с помощью отрезков, соединяющих точки $(i-1, T[i-1])$ и $(i, T[i])$. Таким образом, получаем следующий

Алгоритм.

1. Считываем в массив $S[L, M, K]$ данные выделенного объема из файлов формата NC.
2. Цикл по всем $0 \le \ell < L$, $0 \le m < M$, $0 \le k < K$ (по всем элементам E массива $S[L, M, K]$)
   Если $S[\ell, m, k] \le 11000$ (т.е. $E \in D$), то
   $$i = \left[\frac{S[\ell, m, k]}{10}\right]; \quad T[i] = T[i] + 1;$$
   Конец если.
   Конец цикла.
3. Цикл по i от 0 до 1099
   $$T[i] = T[i] + T[i-1];$$
   Конец цикла.
4. Визуализация осей координат и графика функции $f_\sigma(\ell)$.

На рисунке 4 показан пример визуализации функции пористости для выделенного объема размером 200 ячеек по каждому измерению.

Заключение

В результате данного исследования предложен метод визуального представления пористого пространства изучаемой породы в виде трехмерной виртуальной модели графа. Для рендеринга такой модели были разработаны алгоритмы, основанные на применении современных методов и подходов к обработке данных на видеоадаптерах, в том числе, технологии вершинных буферов. Также был предложен алгоритм построения графика пористости произвольной области из керна.

Предложенные решения могут найти применение в задачах изучения образцов керна, а также при исследовании поровых пространств, полученных в результате гидроразрыва породы.

Simulation and visualization of core's pore space

M.V. Mikhaylyuk, A.V. Maltsev, M.A. Torgashev

Abstract: The paper proposes the method for pore space visual representation of an arbitrary parallelepiped from investigated core's simulation volume in the form of 3D virtual graph model. Graph creation is performed on the basis of data from NC files which contain information about X-ray radiation absorption coefficients in core samples. Visualization of polygonal graph model is real-time performed with using of VBO technology. It's also considered the plotting task of porosity graphic for core's selected part.

Keywords: core, hydraulic fracturing, graph, filtration, visualization, virtual model, pore space.

Литература

1. Я.В.Савицкий. Современные возможности метода рентгеновской томографии при исследовании керна нефтяных и газовых месторождений // Вестник ПНИПУ. Геология. Нефтегазовое и горное дело, 2015, № 15, с. 28-37.
2. В.Б.Бетелин, В.Ф.Никитин, Н.Н.Смирнов, Е.В.Михальченко, Е.И.Скрылева, Л.И.Стамов, В.В.Тюренкова. Компьютерный керносимулятор – подходы и методы // Вестник кибернетики, 2015, №4, с. 33-44.
3. В.В.Мизгулин, Н.А.Штуркин, Е.Ю.Нурканов, Р.М.Кадушников, С.С. Сафонов. Метод трехмерного статистического анализа микроструктуры и порового пространства керна по теневым изображениям с рентгеновского томографа // Сборник тезисов III научно-практической конференции «Математическое моделирование и компьютерные технологии в разработке месторождений», Уфа, 2010.
4. В.Б.Бетелин. Проблемы создания отечественной технологии «цифровое месторождение» // Международная конференция «Математика и информационные технологии в нефтегазовом комплексе». Сургут, 2014, с. 15-17.
5. В.Б.Бетелин. О проблеме импортонезависимости в нефтегазовой отрасли и машиностроении России // Математика и информационные технологии в нефтегазовом комплексе, 2016, с. 18.
6. Д.А.Пестов, Н.Н.Смирнов, А.В.Акулич, В.В.Тюренкова. Математическое моделирование задачи распространения трещины гидроразрыва // Вестник кибернетики, 2017, т. 25, № 1, с. 80-92.
7. В.Ф.Никитин, Л.И.Стамов, Е.В.Михальченко. Трехмерное математическое моделирование течения вязких жидкостей в многосвязной системе каналов и пор // Международная конференция «Математика и информационные технологии в нефтегазовом комплексе». Сургут, 2016, с. 76-78.
8. V.Nikitin, V.Tyurenkova, E.Skryleva, M.Mikhailyuk, N.Smirnov. Computer visualization of fluid displacement instability in porous medium // Proceedings of International Astronautical Congress 2017 (IAC 2017).
9. Как нарисовать сферу вручную.
URL: http://www.gamedev.ru/code/forum/?id=41764 (дата обращения: 27.09.2017).
10. Основы VBO в OpenGL.
URL: http://www.vbomesh.blogspot.ru/2012/02/vbo-opengl.html (дата обращения: 27.09.2017).

Оптимизация алгоритмов быстрого преобразования Фурье для специализированного векторного сопроцессора с учётом иерархической структуры памяти

А.А. Бурцев

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: burtsev@niisi.msk.ru

Аннотация: Классические алгоритмы быстрого преобразования Фурье (БПФ), основанные на многократном исполнении над различными парами элементов комплексного вектора унифицированной операции – так называемой «бабочки Фурье», можно существенно ускорить, если такую операцию реализовать аппаратно. В составе микропроцессора ВМ8 семейства КОМДИВ предусмотрен специализированный векторный сопроцессор, способный, в частности, выполнять такую операцию комплексной арифметики как одну команду. Алгоритмы БПФ, реализованные для микропроцессора ВМ8 с применением этой команды, позволили значительно ускорить выполнение операций преобразования Фурье над комплексными векторами (длины $n=2^p$) одинарной и двойной точности. Но при этом было замечено, что получаемый коэффициент ускорения сильно варьируется в зависимости от того, помещаются ли элементы обрабатываемого вектора целиком в кэш-память соответствующего уровня (или целиком в регистры сопроцессора). С учётом этого, принимая во внимание особенности иерархического строения памяти микропроцессора, первоначально реализованные алгоритмы БПФ удалось впоследствии усовершенствовать и обеспечить тем самым стабильное ускорение операций БПФ даже для больших векторов, целиком не вмещающихся в кэш-память. Результаты такого усовершенствования алгоритмов БПФ и представлены в данной статье.

Ключевые слова: микропроцессоры семейства КОМДИВ, векторный сопроцессор, кэш-память, цифровая обработка сигналов, операция БПФ.

Введение

Для поддержки высокопроизводительных вычислений в ФГУ ФНЦ НИИСИ РАН разрабатывается семейство отечественных микропроцессоров КОМДИВ. В их составе наряду с универсальными процессорами предусматриваются также и разнообразные арифметические сопроцессоры, призванные способствовать ускорению научно-технических расчётов для определённых специфических областей применения. Так, микропроцессор 1890ВМ8Я (ВМ8) [1], оснащённый специализированным векторным сопроцессором (CPV) комплексной арифметики, позволил в несколько раз ускорить решение типовых задач линейной алгебры и цифровой обработки сигналов, что было подтверждено практической разработкой ряда тестовых программ [2,3].

Однако, следует признать, что ускорение производительности, получаемое от применения высокоскоростных команд CPV, в некоторых случаях оказывается ниже ожидаемого из-за низкоскоростной работы команд, призванных обеспечивать своевременную подкачку данных из памяти в регистры CPV и обратно.

Память, используемая в современных микропроцессорах для хранения обрабатываемых данных, как правило, устроена иерархически: подвергаемые обработке объекты данных могут располагаться в регистрах процессора (сопроцессора), в кэш-памяти 1-го или 2-го уровня, в оперативной или даже во внешней памяти (на жёстком диске). Соответственно и времена доступа к этим объектам могут сильно различаться в зависимости от того, на каком уровне такой иерархии они размещены в данный конкретный момент времени.

Чтобы при такой организации памяти достигать максимальной производительности разрабатываемых программ, следует применять при их разработке более совершенные алгоритмы. Учитывая существующую неоднородность памяти и её иерархическое строение, подобные алгоритмы должны задавать такой оптимальный порядок обработки данных, при котором требуемые объекты всякий раз оказывались бы по возможности в самой быстродействующей части памяти.

Примером подобного усовершенствованного алгоритма может служить широко известный блочный алгоритм перемножения матриц

[4], который при исполнении в среде с иерархически устроенной памятью обеспечивает значительно лучшую производительность по сравнению с обычным классическим алгоритмом перемножения матриц. Применение подобного алгоритма на микропроцессоре ВМ8 с использованием CPV позволило добиться стабильного выигрыша [2] в производительности для задач перемножения матриц сколь угодно больших размеров.

В данной статье рассматриваются возможные варианты усовершенствования классического алгоритма быстрого преобразования Фурье (БПФ) с прореживанием по времени для векторов длины $n=2^p$, которые на микропроцессоре ВМ8 с применением CPV обеспечивают лучшую производительность в условиях имеющейся иерархически устроенной памяти.

1. Векторный сопроцессор CPV

Микропроцессор ВМ8 помимо обычного сопроцессора (CP1) плавающей арифметики снабжён дополнительным сопроцессором, способным выполнять не только базовые операции комплексной арифметики, но также и групповые усложнённые операции вещественной арифметики. Этот сопроцессор может за одну команду выполнить одну и ту же операцию сразу над парами, четвёрками и даже восьмёрками чисел, что позволяет характеризовать его как векторный сопроцессор, т.е. сопроцессор, приспособленный для обработки векторов вещественных и комплексных чисел.

Векторный сопроцессор (CPV), входящий в состав микропроцессора ВМ8, содержит 64 128-битных регистра, в каждом из которых может содержаться: или 1 комплексное число двойной точности (DC), или 2 комплексных числа одинарной точности (SC), или 2 вещественных числа двойной точности (DR), или 4 вещественных числа одинарной точности (SR). Для каждого из этих 4-х форматов значений, помещённых в его регистры, CPV обеспечивает соответствующие ему вычислительные команды. Например, каждая из команд умножения с накоплением (см. таблицу 1) выполняет свойственную ей операцию для одной тройки (z,y,x) величин типа DC, либо двух троек типа SC или DR, либо сразу для четырёх троек типа SR.

Таблица 1. Команды умножения с накоплением

cmd z,y,x	**z=y·x**	**z=z+y·x**	**z=z-y·x**	**z=z+y·x y=z-y·x**
1 DC (9)*	**cmul.d**	**cmadd.d**	**cmsub.d**	**cmaddsub.d**
2 SC (7)*	**cmul.s**	**cmadd.s**	**cmsub.s**	**cmaddsub.s**
2 DR (5)*	**vmul.d**	**vmadd.d**	**vmsub.d**	**vmaddsub.d**
4 SR (4)*	**vmul.s**	**vmadd.s**	**vmsub.s**	**vmaddsub.s**
(t)* - *длительность команды в тактах (если команда уже находится в кэше)*				

Длительность вычислительных команд зависит от типа обрабатываемых величин (см. в скобках: 4 такта для SR, 5 для DR, 7 для SR и 9 для DC). Но поскольку CPV разрешает на каждом такте начинать исполнение новой команды вычислительного потока, то в итоге можно добиться такой максимальной производи-тельности CPV, при которой n его вычислительных команд смогут исполниться всего за n+8 тактов.

Команды CPV для работы с памятью (см. таблицу 2) позволяют загрузить 128-битное значение целиком в регистр (vldm) или заполнить его старшую и младшую половину 64-битными значениями по отдельности (vld, vldh). Одной командой (vldq) можно загрузить два соседних регистра, прочитав из двух смежных 128-разрядных слов памяти от 2-х до 8-ми чисел различных форматов (2 DC или 4 SC или 4 DR или 8 SR). Аналогичные команды (vsdm, vsd, vsdh, vsdq) предусмотрены и для сохранения значений регистров в памяти.

Таблица 2. Команды работы с памятью

	команды загрузки	команды сохранения
256 /32 L2(5)*	**vldq**, **vldqx**	**vsdq**, **vsdqx**
64 /8 L1(3)*	**vld**, **vldh**, **vldx**, **vldhx**, **vldlx**	**vsd**,**vsdh**, **vsdx**, **vsdhx**
128 /16 L1(3)*	**vldm**, **vldd**, **vlddh**, **vldmx**, **vlddx**, **vlddhx**, **vlddlx**	**vsdm**, **vsdd**, **vsddh**, **vsdmx**, **vsddx**, **vsddhx**
(t)* - *длительность команды в тактах (если данные уже в L1(L2)-кэше)*		

Для указания виртуального адреса в этих командах можно использовать базовую адресацию с относительным смещением (vld) или базовую индексную (vldx). Получаемый адрес должен быть выровнен на границу обрабатываемого слова, т.е. кратен 8, 16 или 32 (см./d).

Архитектурой КОМДИВ предусмотрена возможность в каждом такте взять на исполнение две очередные команды, если эти команды разных потоков. Это позволяет совместить во времени вычислительные команды CPV над одной группой данных с командами CPV для загрузки/сохранения в/из памяти другой группы данных.

Самая продуктивная команда (**cmaddsub**), предусмотренная в CPV над комплексными числами, эквивалента исполнению 10 арифметических операций (4 умножений и 6 сложений) над вещественными числами двойной точности или 20 арифметических операций над вещественными числами одинарной точности. Такой командой реализуется (одна для DC и две для SC) базовая операция цифровой обработки сигналов (ЦОС), называемая бабочкой Фурье (БФ). Именно при регулярном исполнении этой команды микропроцессор ВМ8 и мо-

жет достигать своей пиковой производительности (16 ГФлопс на частоте 800 Мгц).

2. Иерархическая структура памяти микропроцессора

Применить высокопроизводительные команды CPV можно лишь над теми данными (числами), которые уже помещены в регистры CPV. В микропроцессоре BM8 данные, первоначально хранящиеся в оперативной памяти, доставляются в регистры сопроцессора CPV (как и в регистры процессора), перемещаясь через два уровня кэш-памяти (см. рис.1).

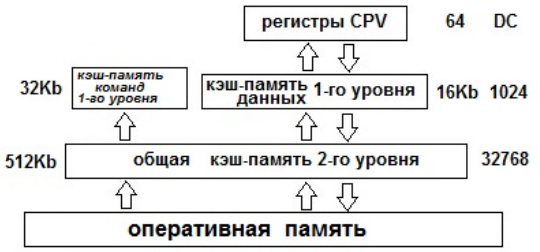


Рис. 1. Иерархическая структура памяти

В составе микропроцессора BM8 предусмотрена кэш-память данных 1-го уровня объёмом 16 Кбайт, кэш-память команд 1-го уровня объёмом 32 Кбайт, а также общая кэш-память команд и данных 2-го уровня объёмом 512 Кбайт. Кэш-память данных 1-го уровня содержит 4 секции по 128 строк в каждой. Одна строка кэша содержит 32 байта данных, которые как единое целое помещаются в кэш или выгружаются из него. В строке кэша хранятся также тэги и 24 старших разряда адреса того участка памяти, которому эти данные принадлежат. Кэш-память 2-го уровня содержит 4096 строк в каждой из 4-х секций, а в каждой строке хранятся 18 старших разряда адреса.

В какие моменты и какие данные надлежит перемещать из оперативной памяти в кэш-память и из кэш-памяти одного уровня в другой, зависит, вообще говоря, от принятой политики кэширования, т.е. стратегии управления кэш-памятью. Хотя микропроцессор BM8 поддерживает 8 разнообразных политик кэширования, команды CPV рекомендуется применять при такой политике, которая условно характеризуется фразой «сквозная запись с локализацией с обратной записью в кэш 2-го уровня» (её код равен 3).

Кратко опишем правила, соблюдаемые при такой политике при исполнении команд (vldm,vsdm), затрагивающих все уровни кэш-памяти. Допустим, требуется прочитать данные в регистры из кэша 1-го уровня. Если при этом возникает промах, т.е. нужной строки данных там нет, то происходит обращение в кэш 2-го уровня. Если строка таких данных присутствует в кэше 2-го уровня, то они передаются в кэш 1-го уровня (и в регистры). При промахе, т.е. если строки требуемых данных нет и в кэше 2-го уровня, данные из оперативной памяти загружаются в кэш обоих уровней.

При исполнении команды записи данных из регистров в память следует различать 4 разных случая. Если строка обновляемых данных присутствует в кэшах обоих уровней, то осуществляется запись данных в оба кэша. Если строка представлена только в кэше 1-го уровня, то осуществляется запись в этот кэш, а также инициируется запись в основную память через буфер, при этом данные ставятся лишь в очередь на запись в память, а поток команд не блокируется, а продолжает исполняться дальше. Если в кэше 1-го уровня такой строки нет, но она есть в кэше 2-го уровня, то запись осуществляется в кэш 2-го уровня, а затем уже обновлённая строка данных загружается из него в кэш 1-го уровня. Если же такой строки нет ещё ни в одном из кэшей, то соответствующая строка считывается из основной памяти и обновлённая новыми значениями данных записывается в кэши обоих уровней, но содержимое памяти при этом пока не обновляется. В этом случае строка помечается специальным флагом, означающим, что хранимые в ней данные должны записываться в память, когда эту строку потребуется вытолкнуть из кэша.

Если в регистрах CPV можно хранить лишь 64 комплексных числа двойной точности (DC), то в кэш-памяти 1-го уровня может поместиться уже 1024 таких числа, а в кэш-памяти 2-го уровня – 32768 чисел. Но это возможно только в тех случаях, когда между адресами помещённых в кэш данных не возникает соперничества за одни и те же места кэш-памяти, в результате которого они вынуждены выталкивать друг друга. Так, например, 256 элементов одной строки матрицы Y[8][256] комплексных чисел двойной точности, которые следуют в памяти друг за другом, могут целиком размещаться в кэш-памяти 1-го уровня, а 8 элементов одного её столбца – нет. Чтобы понять, почему, поясним специфику внутреннего устройства кэш-памяти.

При обращении в кэш-память 1-го уровня младшие разряды адреса с 5-го по 11-ый определяют индекс – номер строки в одной из 4-х секций, где могут быть размещены данные с таким адресом. Для данных с одним и тем же индексом в кэш-памяти 1-го уровня предусмотрены лишь 4 места для их размещения, по одной строке в каждой секции. Так что новый претендент с тем же индексом вынужден выталкивать из кэша одну из тех строк, что была помещена туда ранее.

Элементы одного столбца матрицы Y[8][256] следуют в памяти с шагом (страйдом) в 256 элементов DC, а значит, их адреса в па-

мяти различаются смещением 256×16= 4096 =2^{12} =0x1000, так что все 8 элементов столбца будут претендовать на места с одним и тем же индексом в кэш-памяти 1-го уровня, а поскольку в ней таких мест всего 4, то они будут всё время выталкивать друг друга из кэш-памяти.

При обращении в кэш-память 2-го уровня индекс строки вычисляется по разрядам адреса с 5-го по 16-й. Поэтому конфликтовать за 4 места в ней будут 32-х байтные отрезки данных, расположенные друг от друга в памяти со смещением 2^{17}= 0x20000= 131072= 8192×16. Таковыми могли бы, например, стать элементы одного столбца матрицы Z[8][8192] из комплексных чисел двойной точности (DC).

Таким образом, имеющаяся кэш-память хорошо приспособлена для обработки векторов, элементы которых занимают непрерывный участок памяти, но не совсем удобна для работы с данными, элементы которых размещаются в памяти с некоторым постоянным адресным смещением. Учитывая такую специфику кэш-памяти, в данной статье ограничимся анализом производительности алгоритмов, выполняющих операцию преобразования Фурье только над непрерывными векторами комплексных чисел двойной точности и использующих при этом команды CPV (vldm,vsdm), затрагивающие все уровни кэш-памяти.

3. Классический алгоритм БПФ

В результате дискретного преобразования Фурье (ДПФ) из вектора комплексных чисел X_N (длиной N) получается комплексный вектор Y_N, элементы которого $Y_{m\ (m=0,1,..,N-1)}$ вычисляются по формуле:

$$Y_m = \sum\{ X_k \cdot W(m \cdot k,N) \}_{k=0,1,..,N-1} \text{ , где}$$

$$W(q,N) = \exp(-J \cdot 2\pi \cdot q/N) \text{ (J – мнимая единица).}$$

Вычисление ДПФ непосредственно по этой формуле неэффективно, ибо требует порядка $O(N^2)$ арифметических операций. Известны более совершенные алгоритмы, объединённые общим названием «быстрое преобразование Фурье» (БПФ), позволяющие вычислить ДПФ за $O(N \cdot \log_2 N)$ арифметических операций.

В качестве основы для дальнейшего анализа рассмотрим известный классический алгоритм Кули-Тьюки с прореживанием по времени для длины $N=2^P$.

Тот алгоритм можно эффективно реализовать на CPV, поскольку в нём вектор Y_N получается на месте исходно вектора X_N путём многократного поэтапного выполнения над различными его парами элементов одной и той же базовой операции, так называемой «бабочки Фурье», а такая операция, как уже было замечено, реализуется всего одной командой CPV (cmaddsub). Напомним кратко суть этого алгоритма (подробнее он описан в [3]).Будем обозначать формулой BF(A,B,V) и называть «бабочкой Фурье» такую единую операцию над комплексными величинами A и B с коэффициентом V, в результате которой новые значения A′ и B′ для этих величин вычисляются на основе их предыдущих значений по формулам:

$$A' = A + B \times V,\ B' = A - B \times V$$

Описываемый алгоритм БПФ можно схематично представить состоящим из двух частей. В первой части выполняется перестановка (сортировка) элементов вектора X_N в так называемом бит-реверсном порядке. При такой бит-реверсной перестановке (БРП) каждый элемент вектора с индексом k, двоичное значение которого задаётся набором битов $k=(b_p,b_{p-1},..,b_2,b_1)$ длиной p ($p=\log_2 N$), переставляется с элементом вектора с индексом m, двоичная запись которого представляется набором тех же битов, но записанных в обратном порядке: $m=(b_1,b_2,\ldots,b_{p-1},b_p)$.

Вторая часть алгоритма складывается из p этапов (см. рис. 2 (а,б,в) для N=32, 512, 8192). На каждом t-ом этапе (t=1,..,p) вектор X длиной N разбивается по определённому правилу на N/2 пар элементов. Пары образуются из элементов, расположенных в векторе на расстоянии s ($s=2^{t-1}$, s=1,2,4,..,N/2) элементов друг от друга. На t-ом этапе над каждой парой вида (X_m,X_{m+s}) выполняется операция бабочки Фурье $BF(X_m,X_{m+s},V_m)$ с $V_m=W(m,2^t)=W(m \cdot 2^{P-t},N)$. Операции BF с одинаковым значением коэффициента V_m выгодно исполнять группой, а сам коэффициент не вычислять, а брать из заранее подготовленной таблицы (Cf) по индексу $u=m \cdot d$ ($d=2^{P-t}$). С учётом этого вторую часть описываемого алгоритма БПФ можно представить (на языке Си) в виде:

```
s=1; h=2; d=N/2;
 for(t=1, t<=P; t++ ) {
  for(k=0,u=0; k<s; k++, u=u+d ) {
   V= Cf[u];//V=exp(-I*(2*Pi*k/2^t));
   for(m=k; m<N; m=m+h )
    BF(X[m],X[m+s],V);
  }//for k
   h=h*2; s=s*2; d=d/2;
 }//for t
```

Такой алгоритм на языке Си для микропроцессора BM8, реализованный на обычном сопроцессоре CP1 плавающей вещественной арифметики , при чём в качестве базового. Будем называть его **вариантом A** и сравнивать с ним все последующие варианты его усовершенствования, рассчитанные на применение векторного сопроцессора CPV.

Для понимания дальнейших модификаций этого алгоритма отметим одну важную его особенность. Обозначим БПФ-h(X_f) операцию БПФ, которая выполняется без БРП над отрезком вектора X длиной h ($h=2^t$), начиная с элемента X_f. И заметим, что в результате исполне-

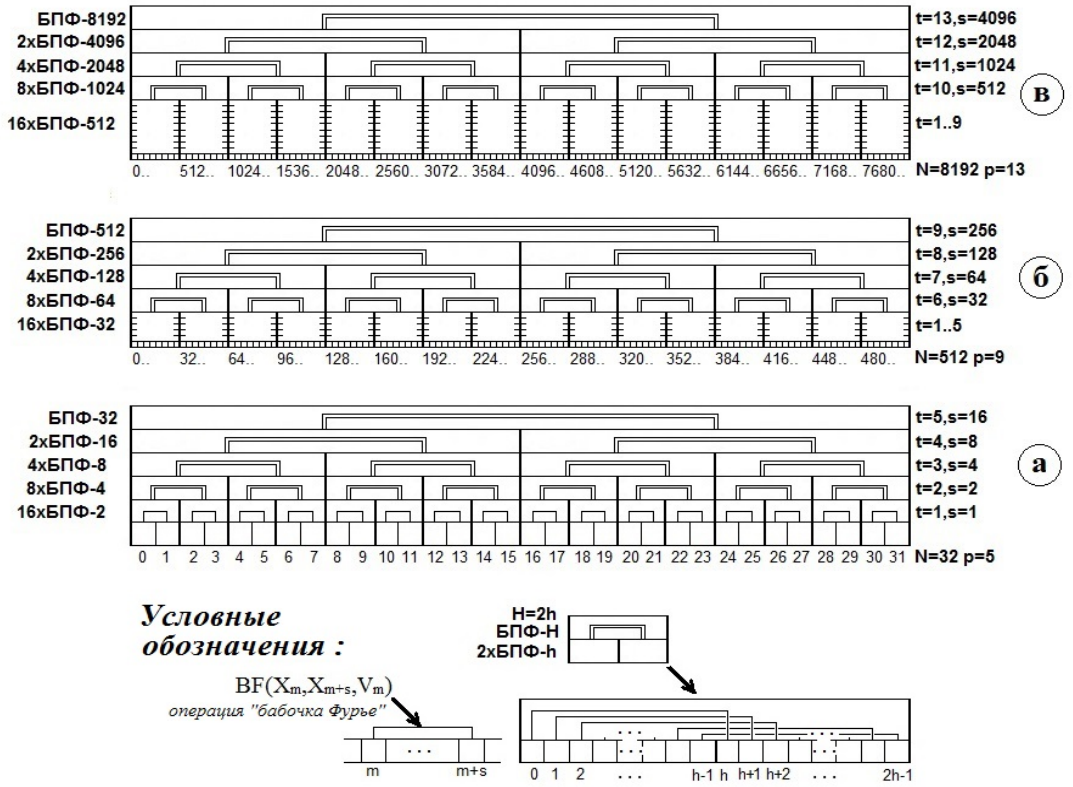


Рис. 2. Схема 2-ой части алгоритма БПФ для N=32 (а), N=512 (б), N=8192 (в)

ния очередного шага цикла по t совершается несколько операций БПФ-h над серией векторов из подряд стоящих элементов длины $h=2^t$. Сначала при t=1 операция БПФ-2 совершается над каждой парой соседних элементов, затем при t=2 операция БПФ-4 совершается над соседними четвёрками элементов, потом БПФ-8 над восьмёрками и т.д. На очередном (t+1)-ом шаге цикла каждая пара соседних векторов длины h, над которыми уже была исполнена операция БПФ-h (на t-ом шаге цикла), объединяется в вектор длины H=2·h и подвергается совместной обработке, так что на их месте получается вектор, являющийся результатом операции БПФ-H.

Далее в статье будем сравнивать различные варианты алгоритмов БПФ, применяя их к обработке вектора комплексных чисел двойной точности одной и той же длины в так называемом режиме inplace («на месте»). Это означает, что результат применения БПФ должен оставаться на месте того же самого вектора, а не записываться в какой-то другой вектор.

Нам потребуется сравнивать различные варианты алгоритмов БПФ по производительности (по быстродействию). В качестве количественной характеристики длительности алгоритма будем замерять число тактов, затраченных на его второе исполнение подряд. Для этого составляется тестовая программа, которая вызывает каждый испытуемый алгоритм два раза, но замеряет длительность исполнения (в тактах) его второго вызова.

Все приводимые в данной статье показатели числа тактов получены при запуске тестовых программ на плате микропроцессора BM8 с частотой процессора 400 МГц и частотой DDR-памяти 400 МГц.

4. Алгоритм БПФ для CPV

Для усовершенствования описанного выше алгоритма БПФ воспользуемся теми возможностями, которые предоставляет CPV.

Заметим, что применяя CPV, можно не только вычислять операцию BF за одну команду, но ещё и использовать 64 регистра CPV как своеобразную кэш-память 0-го уровня.

Операция BF(A,B,V) может вычисляться одной командой cmaddsub.d rA,rB,rV, только если требуемые ей величины A,B,V уже размещены в регистрах CPV rA, rB, rV и должны оставаться там как результаты её выполнения.

Если же эти величины расположены в памяти (по адресам adrA, adrB, adrV), и туда же следует поместить результаты, тогда для полного исполнения операции BF(A,B,V) потребуется 6 команд:

```
vLdm rA,adrA; vLdm rB,adrB; vLdm rV,adrV;
 cMaddsub.d rA,rB,rV
 vSdm rA,adrA; vSdm rB,adrB;
```

Значит, на одну команду, вычисляющую «бабочку Фурье», потребуется 3 команды загрузки и 2 команды сохранения: **BF= 3L+1M+2S**. На каждом шаге (этапе) цикла по t вычисляется N/2 операций BF, а всего $p=\log_2 N$ этапов. Следовательно, такой способ вычисления БПФ для вектора длиной N потребует выполнения $(3L+1M+2S)\cdot\log_2 N\cdot N/2$ команд. Так, например, для БПФ длиной N=32 получится $(3L+1M+2S)\cdot\log_2 32\cdot 32/2$ = 240L+80M+160S = 480 команд.

В приведённой выше формуле на каждую вычислительную команду приходится аж 5 команд работы с памятью. Количество команд, вычисляющих «бабочку Фурье», в нашем алгоритме уменьшить не получится. А вот количество команд обращения к памяти сократить как раз таки можно. И на это будут направлены все наши дальнейшие усилия.

Это в теоретической математике от перестановки слагаемых сумма, как известно, не меняется. В вычислительной же математике (когда сумма вычисляется на калькуляторе или программой на компьютере) из-за возникающих ошибок округления чисел итоговый результат суммирования может существенно зависеть от порядка сложения слагаемых.

А в компьютерных системах, где обрабатываемые данные хранятся в неоднородной (иерархически устроенной) памяти, от порядка выполнения элементарных операций может существенно зависеть не только сам получаемый результат, но и время его вычисления, т.е. производительность вычислительных программ. Чтобы поднять производительность программ, выполняющих БПФ с применением CPV, будем изменять порядок вычисления операций «бабочек Фурье», пытаясь получить оптимальный вариант алгоритма БПФ с учётом иерархического строения памяти.

4.1. БПФ длины 32 для CPV

Сначала рассмотрим самый благоприятный случай, когда количество команд обращения в память можно сократить до минимума.

Для исполнения операции БПФ над вектором длиной N элементов необходимо вычислить $\log_2 N\cdot N/2$ бабочек Фурье и при этом хотя бы по одному разу прочитать из памяти в регистр CPV каждый элемент вектора, а затем записать в него обновлённое значение.

Значит, в лучшем случае для вычисления $\log_2 N\cdot N/2$ бабочек потребуется загрузить N элементов вектора (да ещё N/2 коэффициентов) в регистры, а после выгрузить их обратно.

Но такой благоприятный вариант возможен только, если у CPV хватит регистров для размещения всех N+N/2 величин (для элементов и коэффициентов). Поскольку таких регистров всего 64, значит, содержать все элементы вектора и все коэффициенты в регистрах CPV на протяжении вычислений всех бабочек Фурье возможно лишь при выполнении БПФ длины не более 32 (если рассматривать в качестве длины только степени двойки $N=2^P$).

Чтобы выполнить БПФ для вектора длиной N=32 (БПФ-32), потребуется вычислить 80 бабочек: по 16 бабочек на каждом из 5 этапов ($p=\log_2 32=5$). Элементы вектора можно прочитать из памяти в 32 регистра лишь один раз перед 1-ым этапом, а далее оставлять их в регистрах, и только при завершении последнего 5-ого этапа отправлять полученные значения в память. Ещё 16 регистров потребуется загрузить коэффициентами, необходимыми для вычисления этих 80 бабочек. Итого будет занято 48 регистров CPV.

Заметим, что первоначальную загрузку значений элементов вектора в регистры можно совместить с их бит-реверсной перестановкой, т.е. в регистр rXk (k=0,1..,31) загружать из памяти такой элемент X[m] вектора, индекс m которого соответствует бит-реверсному двоичному значению индекса k.

В результате для выполнения БПФ длины 32 потребуется: 1) 16 команд v**L**dm для загрузки 16 коэффициентов; 2) 32 команды v**L**dm для чтения в регистры 32-х элементов вектора в бит-реверсном порядке; 3) 80 команд c**M**addsub.d для вычислений 80 бабочек Фурье (по 16 на каждом этапе); 4) 32 команд v**S**dm для записи из регистров в память новых значений для 32-х элементов вектора.

Итого всего: 16**L**+32**L**+80**M**+32**S**=160 команд вместо 480, т.е. общее количество команд станет в 3 раза меньше, при этом количество команд обращения к памяти сократится в 5 раз (400/80). Такой способ выполнения БПФ с применением CPV, применимый для БПФ малого размера, охарактеризуем как алгоритм БПФ **варианта Б** для N<=32. Для N=32 он реализуется функцией, код которой составлен на ассемблере и подробно изложен в [3] (см. функцию **cp3_VFFT32**).

Аналогичные функции были реализованы (на ассемблере) и для выполнения на CPV операций БПФ длиной N=16,8,4,2. Результаты ускорения БПФ, полученные этими функциями на CPV, приведены в таблице 3.

Заметим, что такого же значительного ускорения, как для случая N=32 (в 19 раз!), для N<32 обеспечить не удалось. Почему же? Казалось бы, при исполнении БПФ длиной N<32 количество команд обращения к памяти можно также минимизировать, храня все необходимые величины в регистрах CPV. Но, оказывается, этого недостаточно для обеспечения столь же высокого быстродействия алгоритма БПФ.

Таблица 3. Ускорение БПФ (Б) на CPV для N<=32

P	$N=2^P$	Вариант А на CP1	Вариант Б на CPV	КБА
1	**2**	272	80	3.40
2	**4**	398	81	4.91
3	**8**	906	105	8.63
4	**16**	1769	137	12.91
5	**32**	3809	199	19.14

КБА – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает вариант Б на CPV по отношению к варианту А на CP1 (= такты варианта А / такты варианта Б)

Всё дело в том, что одна команда cmaddsub.d исполняется 9 тактов, но на каждом такте можно начать исполнение новой такой команды с другими регистрами. При N=32 на каждом этапе (шаге цикла по t) выполняется группа из 16 команд cmaddsub.d.

И почти всегда удаётся расставить эти команды так, чтобы используемые в них регистры успевали бы получать свои результаты до того, как на следующем этапе начнётся исполнение команд, в которых они могут потребоваться. Так что никаких задержек в исполнении команд при БПФ для N=32 не возникает.

При N=16 на каждом этапе выполняется 8 команд cmaddsub.d, при N=8 ещё меньше – 4, при N=4 – 2, при N=2 – всего 1 команда.

И для этих случаев избежать задержек, возникающих из-за зависимостей по регистрам между командами cmaddsub.d разных этапов, уже не удаётся. Эти вынужденные задержки как раз и снижают производительность вычисления БПФ для векторов меньших размеров (N<32).

4.2. БПФ длины $N=2^p$ >32 для CPV

Вторая часть алгоритма БПФ для вектора произвольной длины $N=2^p$>32, которая начинает выполняться после бит-реверсной перестановки (БРП) элементов, на первых 5-ти этапах над каждым отрезком вектора из 32-х подряд стоящих элементов совершает по сути операцию БПФ длиной 32, но только уже без БРП. Чтобы сократить число команд обращения к памяти на первых 5 этапах, изменим порядок вычислений. И будем для каждого отрезка вектора из 32-х элементов выполнять вычисления «бабочек Фурье» всех 5-ти этапов сразу: БПФ-32(X_j) для j=0,32,64,…A на последующих этапах (t=6,..), чтобы не допускать задержек между вычислительными командами и командами обращений к памяти, будем вычислять бабочки Фурье группами по 16 штук. В итоге получим алгоритм БПФ **варианта Б**, 2-ую часть которого можно выразить схематично (на языке Си) в виде:

```
VWLoad32(Cf32);
for(k=0;k<N>>5;k++)DoVFFT32(&X[k*32]);
s=32; h=2; d=N/64;
for(t=6, t<=P; t++ ) {
 for(k=0,u=0;k<s;k=k+16,u=u+16*d) {
  VFFT_Load16W(&Cf[u],d);
  for(m=k; m<N; m=m+h )
    VFFT_Do16BF(&X[m],&X[m+s]);
 }//for k
 h=h*2; s=s*2; d=d/2;
}//for t
```

Характеристика ассемблерных функций:
VWLoad32 – загружает 16 коэфф-тов для БПФ-32
DoVFFT32 – выполняет БПФ-32, но без БРП
VFFT_Load16W – загружает 16 коэф-тов для группы
VFFT_Do16BF – вычисляет сразу 16 бабочек Фурье

Этот алгоритм **варианта Б**, более подробно описанный в [3], был первоначально реализован на микропроцессоре BM8 с применением CPV. Проведённые тестовые испытания подтвердили, что он позволяет в несколько раз ускорить вычисления БПФ (см. табл. 4).

Таблица 4. Ускорение БПФ (Б) на CPV для N>32

P	$N=2^P$	Вариант А на CP1	Вариант Б на CPV	КБА
6	**64**	8408	1273	**6.60**
7	**128**	18762	2545	**7.37**
8	**256**	41789	5504	**7.59**
9	**512**	92517	13042	**7.09**
10	**1024**	208301	37781	**5.51**
11	**2048**	503124	99571	**5.05**
12	**4096**	88778	220080	**4.95**
13	**8192**	2340812	480100	**4.88**
14	**16384**	5123577	1182503	**4.33**
15	**32768**	14424729	6097307	**2.37**

КБА – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает **вариант Б** на CPV по отношению к **варианту А** на CP1 (= такты варианта **А** / такты варианта **Б**)

Однако, было замечено, что с ростом длины векторов обеспечиваемый этим алгоритмом коэффициент ускорения существенно снижается. Особенно заметны его резкие падения, когда длина обрабатываемых векторов превышает пороговые значения N=32, 512, 16384, которые как раз характеризуют, какое количество элементов вектора (комплексных чисел двойной точности) при исполнении БПФ может гарантировано поместиться в регистрах CPV (32), в кэш-памяти 1-го уровня (512) и в кэш-памяти 2-го уровня (16384).

Поэтому возникла потребность в дальнейшем усовершенствовании применяемого алгоритма БПФ с учётом иерархического строения памяти и неоднородного характера доступа к хранимым в ней на различных уровнях объектам данных.

5. Оптимизация БПФ для CPV

Сначала рассмотрим, какие ещё приёмы усовершенствования алгоритма БПФ можно применить для обработки векторов, помещающихся целиком в кэш-память 1-го уровня. А потом попробуем применить аналогичные приёмы и в отношении векторов больших размеров, превышающих объём не только кэш-памяти 1-го уровня, но и 2-го уровня тоже.

5.1. Оптимизация БПФ длины 32<N<=512

Алгоритм БПФ варианта Б, начиная с 6-го этапа, предполагает вычисление операций BF («бабочек Фурье») последовательно над всеми парами элементов вектора, которые образуются на очередном этапе. И поскольку результаты вычислений всех BF текущего этапа уже не удаётся сохранять в регистрах CPV до следующего этапа, данный алгоритм при вычислении каждой BF принуждает исполнять 4 команды обращения к памяти: 2 команды загрузки v**L**dm до и 2 команды сохранения v**S**dm после. Так, например, при обработке вектора длиной N=512 на этапах 6-9 требуется выполнить 4·512/2=1024 операций BF, значит, при этом потребуется 1024·4=4096 команд обращения к памяти. (Команды загрузки коэффициентов мы здесь учитывать не будем, т.к. они загружаются один раз перед циклом обработки целой группы BF).

Чтобы сократить число команд обращения в память, попробуем изменить порядок вычислений BF таким же приёмом, который применили ранее (в п.4.2) для обработки блоков из 32-х элементов. Для этого представим вектор X длины N как матрицу Y[H][32] (N=Hx32), которая расположена в памяти по строкам, так что отрезки из 32-х подряд стоящих элементов вектора X образуют строки этой воображаемой матрицы.

Допустим, над каждой такой строкой матрицы уже выполнена операция БПФ-32. Далее действия, совершаемые над элементами этой матрицы на последующих этапах (при t=6,7,8,9), можно образно представить как выполнение нескольких специфических операций БПФ длиной 2, 4, 8 и 16 над столбцами этой матрицы, как над векторами, элементы которых следуют в памяти с шагом (страйдом) 32. Такое представление для N=512 наглядно продемонстрировано на рис. 3.

Специфика этих операций БПФ над столбцами заключается не только в том, что в них не применяется БРП, но ещё и в том, что в этих БПФ «бабочки Фурье» вычисляются с другими коэффициентами. Формулы вычисления этих коэффициентов или правила формирования их индексов для доступа в единую приготовленную заранее таблицу коэффициентов Cf уточним позднее. А пока договоримся обозначать БПФ-H*(X_k) операцию БПФ над вектором длины H, который образуют элементы k-го столбца воображаемой матрицы Y, размещённой на месте вектора X.

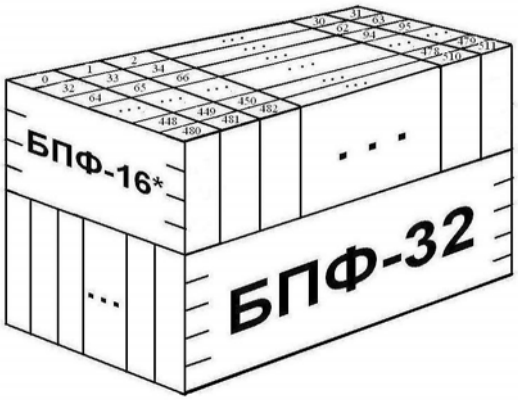


Рис. 3. Схема алгоритма БПФ-512 на основе БПФ-32

Как показано на рис. 3 для длины N=512, работа, выполняемая алгоритмом БПФ на этапах 1-5, заключается в исполнении 16-ти операций БПФ-32 над строками воображаемой матрицы, а работа, выполняемая на этапах 6-9, заключается в исполнении 32-х операций БПФ−16* над столбцами этой же матрицы.

Изменим порядок вычислений BF на этапах 6-9. Будем сразу выполнять все вычисления для очередных двух столбцов, необходимые для совершения над ними полных операций БПФ-16*. При этом будем держать результаты всех промежуточных вычислений BF для этих столбцов в регистрах CPV.

Это позволит существенно сократить количество команд обращений в память. Для обработки каждого столбца потребуется лишь 16 команд загрузки в начале и 16 команд сохранения в конце исполнения полной операции БПФ-16* над столбцом. Итого при обработке всех столбцов понадобится (16**L** +16**S**)·32=1024 команды обращения к памяти (не считая команд загрузки коэффициентов), т.е. почти в 4 раза меньше (4096/1024), чем на этапах 6-9 в алгоритме БПФ варианта Б для N=512.

Такую же оптимизацию алгоритма БПФ можно попытаться осуществить и для других значений N. При N=256 этапы 6-8 можно превратить в цикл исполнения 32-х операций БПФ-8* над столбцами матрицы Y[8][32]. При этом на каждом шаге цикла целесообразно обрабатывать уже по 4 столбца сразу, чтобы избежать вынужденных задержек из-за зависимостей, возникающих между командами cmaddsub, исполняемых над элементами одного столбца. Но количество команд обращений в память удастся сократить лишь в 3 раза: (8**L**+8**S**)·32=512 команд вместо 3·128·4=1536.

Для N=128 этапы 6-7 превратятся в цикл исполнения операций БПФ-4* над 32-мя столбцами, сразу 8 из которых надо обрабатывать на одном шаге. А сократить число команд с памятью удастся лишь в 2 раза.

А для N=64 этот приём оптимизации выигрыша вообще не даёт. Можно, конечно, для единообразия алгоритма БПФ превратить 6-ой этап в цикл исполнения операций БПФ-2* над 32-мя столбцами, обрабатывая их по 16 за раз, но всё равно для каждой пары элементов одного столбца придётся исполнить 2 команды загрузки и 2 команды сохранения.

Отметим, что описанным приёмом оптимизации можно было бы воспользоваться и для N=1024, превратив этапы 6-10 алгоритма БПФ в цикл исполнения 32-х операций БПФ-32* над столбцами воображаемой матрицы Y[32][32]. При этом количество команд обращения к памяти (при выполнении этапов 6-10) можно было бы сократить с 5·512·4=10240 команд до (32**L**+32**S**)·32=2048, т.е. в 5 раз!

Однако, вектор из 1024-х элементов (комплексных чисел двойной точности) вместе с другими величинами (коэффициентами, локальными переменными, располагаемыми в стеке), необходимыми для выполнения БПФ такого размера (1024), уже не помещается в кэш-памяти 1-го уровня. Поэтому алгоритм БПФ для N=1024 выгоднее перестроить по-другому, так, чтобы обрабатываемая в текущий момент часть вектора всегда находилась в кэш-памяти 1-го уровня.

5.2. Оптимизация БПФ длины N>512

Допустим длина вектора N>512, так что он не помещается целиком в кэш-памяти 1-го уровня. В таком случае для вычисления БПФ вектор выгодно разбить на блоки по 512 элементов, над каждый из которых сначала осуществить операцию БПФ-512 (как в п. 5.1 и на рис. 3). А последующие действия (на этапах t=10,11,..) снова представим, как выполнение специфических операций БПФ-F** над 512-ью столбцами воображаемой матрицы Z[F][512] (N=F×512), но с особыми коэффициентами. Такое представление алгоритма БПФ для N=8192 иллюстрируется на рис. 4.

И выполнять операцию БПФ-F** будем сразу над несколькими столбцами: по 2 БПФ-16** для N=8192, по 4 БПФ-8** для N=4096, по 8 БПФ-4** для N=2048, по 16 БПФ-2** для N=1024, сохраняя элементы этих столбцов в регистрах CPV, чтобы сократить количество команд обращения к памяти и избежать нежелательных задержек из-за зависимостей, которые могли бы возникать между командами cmaddsub одного столбца.

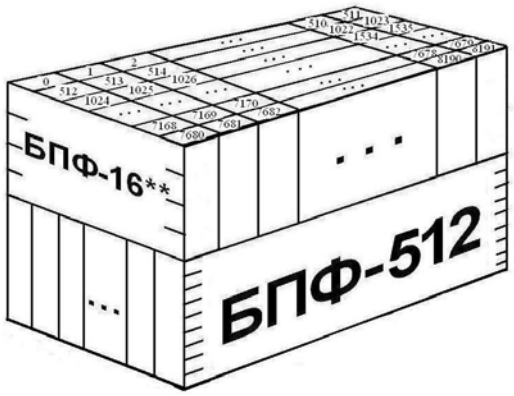


Рис. 4. Схема БПФ-8192 на основе БПФ-512

Заметим, что подобное разложение вычисления БПФ большого размера на стадии выполнения множественных операций БПФ малого размера, можно осуществить разными способами в зависимости от того, на какие блоки меньшего размера разбить весь обрабатываемый вектор. Например, представив вектор длиной N=8192 как 3-х мерный массив Z[32][8][32], 2-ую часть алгоритма БПФ (после БРП) можно было бы разделить на такие стадии. Сначала выполнить БПФ-256 для каждой плоскости − матрицы из 8 строк по 32 элемента, для чего после 8 операций БПФ-32 над каждой строкой исполнить 32 операции БПФ-8* над столбцами. А затем уже исполнить 256 операций БПФ-32** над колонками, образуемыми 32-мя элементами, лежащими на одинаковых позициях в разных плоскостях.

Чтобы определить, какой способ разложения окажется оптимальным для заданного $N=2^P$, составим обобщённый алгоритм БПФ. В нём в качестве параметров, описывающих конкретный вариант разложения, будем задавать разнообразные размеры блоков, над которыми будут выполняться множественные операции БПФ различных стадий. И выполнив такой алгоритм многократно для всевозможных вариантов разложения, выясним, какой же из этих вариантов обеспечивает наилучшую производительность.

5.3. Обобщённый алгоритм БПФ для N>32

Возьмём за основу алгоритм БПФ для N>32, описанный в п.4.2, и перестроим в нём тот фрагмент, который задаётся циклом по этапам t=6..P. Распределим действия этих этапов на три стадии. Пусть первая из них (назовём её R-стадией) охватывает R этапов, начиная с 6-го, вторая (Q-стадия) – Q следующих этапов после первой, а завершающая (V-стадия) – все остальные этапы вплоть до P-го. Для полноты картины назовём базовой (B-стадией) стадию исполнения операций БПФ-32 на этапах 1-5.

Параметры R и Q (0<=R,Q<=5) задают конкретный способ (вариант) разложения алгоритма БПФ. При заданном разложении обрабатываемый вектор представляется состоящим из блоков, каждый из которых является трёхмерным массивом Z[F][H][32] ($H=2^R$, $F=2^Q$). Блок складывается из F плоскостей. Каждую из плоскостей можно рассматривать как матрицу размером H×32, состоящую из H строк по 32 элемента, или как единый вектор из RL элементов (RL=H·32). Весь блок можно тоже обрабатывать как вектор длиной QL элементов (QL=F·RL=F·H·32).

Схематично обобщённый алгоритм БПФ можно представить в следующем виде:

```
R=..; Q=..; H=1<<R;F=1<<Q; // H=2^R,F=2^Q
RL=H*32;QL=F*RL; SR=32/H;SQ=32/F;
VWLoad32(Cf32);
for(pQ=0;pQ<N;pQ=pQ+QL)
{// do БПФ-QL(X_pQ)
 for(pR=0;pR<QL;pR=pR+RL){//БПФ-RL(X_pQ+pR)
  for(k=0;k<RL;k=k+32) //== B- стадия ==
   DoVFFT32(&X[pQ+pR+k]);//БПФ-32(X_pQ+pR+k)
  if(R>0)
  for(k=0;k<32;k=k+SR) //== R-стадия ==
   DoVFFTsHxSR
   (&X[pQ+pR+k],&Cf,k,N>>6,32);
 }//for pR
 if(Q>0)
 for(k=0;k<RL;k=k+SQ)   //== Q- стадия ==
  DoVFFTsFxSQ
   (&X[pQ+k],&Cf,k,N>>(6+R),RL);
}//for pQ
if(N>QL)
{s=QL;h=2*QL;d=N>>(6+R+Q);
 for(t=6+R+Q,t<=P;t++){ //== V-стадия ==
  for(k=0,u=0;k<s;k=k+16,u=u+16*d){
   VFFT_Load16W(&Cf[u],d);
   for(m=k;m<N;m=m+h )
     VFFT_Do16BF(&X[m],&X[m+s]);
  }//for k
  h=h*2; s=s*2; d=d/2;
 }//for t
}
```

Характеристика ассемблерных функций:
DoVFFTsHxSR =TblFun[R-1];
– выполняет БПФ-H* над SR столбцами
DoVFFTsFxSQ =TblFun[Q-1];
– выполняет БПФ-F** над SQ колонками
Вызываемая функции выбираются из таблицы:
TblFun [5]= { DoVFFTs2x16, DoVFFTs4x8,
DoVFFTs8x4, DoVFFTs16x2, DoVFFTs32x1 };

Основная идея перестройки алгоритма БПФ с учётом особенностей функционирования иерархической памяти состоит как раз в том, чтобы разложить этот алгоритм на такие части, при выполнении которых обрабатываемые порции данных целиком умещались бы в памяти одного уровня: строки матриц − в регистрах CPV, плоскости матриц− в кэш-памяти 1-го уровня, а блоки плоскостей − в кэш-памяти 2-го уровня.

Для совершения операции БПФ длины QL с очередным блоком над его элементами выполняются три стадии обработки (B-, R-, Q-). В каждой его плоскости, представленной в виде матрицы Y[H][32], сначала над её строками выполняются H операций БПФ-32, относящиеся к B-стадии, а затем исполняются операции БПФ-H^* над 32 столбцами матрицы, в обработке которых и заключается сущность R-стадии (подробно это объяснялось в п. 5.1 и иллюстрировалось на рис. 3).

Наконец, после того, как над всеми плоскостями (матрицами) блока выполнены стадии обработки B и R, и по сути совершены операции БПФ-RL над их элементами как над векторами длины RL, наступает черёд стадии Q. На этой стадии исполняются операции вида БПФ−F^{**} над всеми RL колонками плоскостей блока, как это объяснялось в п.5.2 и иллюстрировалось на рис. 4 (для N=8192). После завершения Q-стадии обработки над блоком оказывается исполненной операция БПФ-QL как над вектором длины QL.

Если N>QL, то на завершающей V-стадии выполняются оставшиеся этапы обработки, действия которых задаются циклом по t от t=6+R+Q до t=P. И на каждом этапе между парами, образуемыми элементами соседних блоков, выполняются операции «бабочки Фурье» так же, как ранее уже было описано циклом по t в алгоритме БПФ п.4.2.

В приведённой выше схеме алгоритма показано, что основная работа по выполнению R-стадии осуществляется путём многократных вызовов (в цикле по k) ассемблерной функции **DoVFFTsHxSR**, которая исполняет операцию БПФ-H^* сразу над несколькими (SR) столбцами матрицы. На самом же деле организуется вызов по таблице одной из пяти функций, предусмотренных для исполнения либо операции БПФ-32^* над одним столбцом (при R=5), либо операций БПФ-16^* над 2-мя (R=4), либо БПФ-8^* над 4-мя (R=3), либо БПФ-4^* над 8-мью (R=2), либо БПФ-2^* над 16-ю столбцами сразу (при R=1).

Каждая из таких функций реализована на ассемблере, что позволяет использовать высокопроизводительную команду cmaddsub для вычисления «бабочек Фурье» и сократить число команд обращения к памяти (элементы всех обрабатываемых столбцов загружаются в регистры CPV в начале и записываются обратно в память уже в конце функции).

Вызов любой из этих функций сопровождается списком из 5 параметров: (X,V,k, M,L). Параметры X и k задают начальный адрес и номер первого обрабатываемого столбца, параметр L задаёт шаг (страйд) для перемещения к следующему элементу. Параметр V указывает на таблицу, из которой выбираются коэффициенты для операций «бабочек Фурье», а пара-

метр M задаёт множитель, необходимый для вычисления индексов доступа в эту таблицу.

Индексы элементов k-го столбца определяются формулой: k+j×L (j=0,1,..,H-1). При вычислении операции $BF(X_m,X_{m+s},V_m)$ над парой элементов (X_m,X_{m+s}) этого столбца требуется использовать коэффициент V_m (m=k+j×L), который на t-ом этапе определяется формулой: $V_m=W(m,2^t)=W(m\cdot 2^{P-t},N)$. Первоначальное значение множителя 2^{P-t} задаётся параметром функции M, который на последующих этапах уменьшается вдвое. Так что требуемый для BF коэффициент можно не вычислять, а просто выбирать из приготовленной заранее таблицы V по индексу u=m×M =(k+j×L)×M.

Аналогичным выбором по той же таблице реализуется и вызов ассемблерной функции **DoVFFTsFxSQ** для исполнения операций БПФ−F^{**} над SQ колонками плоскости, который предусмотрен в цикле по k на Q-стадии приведённой выше схемы алгоритма. Заметим, что при вызове выбранной функции на R-стадии для параметров L и M задавались значения L=32, M=N/64, а на Q-стадии при таком вызове задаются совсем другие значения: L=RL, $M=2^{P-6-R}=N/(2\times RL)$. Представленный выше обобщённый алгоритм БПФ был реализован на микропроцессоре BM8 и опробован для каждого значения $N=2^p$ от 64 до 32768 при различных сочетаниях установочных параметров R и Q. Опытным путём с помощью многократных тестовых прогонов для каждого N было выяснено, при каких значения R и Q достигается оптимальная производительность выполнения операций БПФ длины N. Выявленные таким образом оптимальные значения параметров R и Q приведены в таблице 5.

Таблица 5. Оптимальные значения R и Q для $N=2^p$

P	$N=2^P$	**R**	H	RL	**Q**	F	QL
6	**64**	**1**	2	64	**0**		
7	**128**	**2**	4	128	**0**		
8	**256**	**3**	8	256	**0**		
9	**512**	**4**	16	512	**0**		
10	**1024**	**5**	32	1024	**0**		
11	**2048**	**5**	32	1024	**0**		
12	**4096**	**5**	32	1024	**2**	4	4096
13	**8192**	**3**	8	256	**5**	32	8192
14	**16384**	**5**	32	1024	**4**	16	16384
15	**32768**	**4**	16	512	**5**	32	16384

Добавим в схему обобщённого алгоритма выбор оптимальных значений параметров R и Q по таблице:

```
int TableR[10]={1,2,3,4,5,5,5,3,5,4};
int TableQ[10]={0,0,0,0,0,0,2,5,4,5};
R=TableR[P-6]; Q=TableQ[P-6]; //P=log2N
```

И назовём **вариантом B** получившийся в результате такой оптимизации алгоритм БПФ.

5.4. Результаты оптимизации БПФ для векторного сопроцессора CPV

Оптимизированный по варианту B алгоритм БПФ для N>32 был реализован с применением CPV и опробован на микропроцессоре BM8 для обработки векторов комплексных чисел двойной точности. Полученные показатели его производительности представлены в таблице 6. В её правых столбцах КВА и КВБ приводятся коэффициенты выигрыша, которые обеспечивает этот оптимизированный вариант (B) алгоритма по отношению к алгоритмам предыдущих вариантов A и Б.

Таблица 6. Ускорение БПФ (B) на CPV для N>32

P	$N=2^P$	**Вариант B на CPV**	**КВА**	**КВБ**
6	**64**	1520	**5.53**	**0.84**
7	**128**	2444	**7.68**	**1.04**
8	**256**	4639	**9.01**	**1.19**
9	**512**	9663	**9.57**	**1.35**
10	**1024**	25801	**8.07**	**1.46**
11	**2048**	67140	**7.49**	**1.48**
12	**4096**	150162	**7.25**	**1.47**
13	**8192**	316194	**7.40**	**1.52**
14	**16384**	838478	**6.11**	**1.41**
15	**32768**	3899233	**3.70**	**1.56**

КВА – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает **вариант B** на CPV по отношению к **варианту A** на CP1 (= такты **A** / такты **B**)

КВБ – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает **вариант B** на CPV по отношению к **варианту Б** на CPV (= такты **Б** / такты **B**)

Как видно из этой таблицы, проведенная оптимизация алгоритма позволила существенно (почти в 1,5 раза) поднять производительность исполнения на CPV операции БПФ для векторов больших размеров (>512). Не очень значительный выигрыш получается для N=256 (19%) и N=512 (35%). Почти никакого выигрыша такой способ оптимизации не даёт для N=128. А для N=64 он вообще не подходит, так как ухудшает производительность алгоритма.

6. Особый алгоритм БПФ для N=64

Вариант B не позволяет ускорить алгоритм БПФ для N=64, т.к. требует такого же количества команд обращения к памяти, как и вариант Б: 56**L**+56**S** (БРП), 16**L**+2×(32**L**+32**S**) на B-стадии, 32**L**+64**L**+64**S** на 6-ом этапе, итого 232**L**+184**S**=416 команд. Можно ли исполнить БПФ-64, сократив число таких команд?

Воспользуемся тем, что при исполнении БПФ-32 16 регистров CPV остаются свободными (32 регистра хранят 32 обрабатываемых

элемента вектора и 16 − коэффициенты BF). И тем, что 16 коэффициентов БПФ-32 можно будет использовать при БПФ-64 в качестве коэффициентов с чётными индексами.

Будем называть чётными те элементы вектора, которые в итоге исполнения операции БПФ-64 окажутся на местах с чётными индексами (0,2,..,62), а нечётными − те, которые окажутся на местах с нечётными индексами (1,3,..,63). Для обработки 16-ти чётных элементов младшей половины вектора и 16-ти нечётных элементов старшей половины вектора будем использовать одни и те же 16 регистров CPV. И предложим такой порядок исполнения БПФ длины 64.

После выполнения БПФ-32 над старшей половиной вектора выгрузим в память только её нечётные элементы и загрузим в освобождённые ими регистры чётные элементы младшей половины вектора. Нечётные элементы младшей половины загрузим в оставшиеся свободными 16 регистров. Тогда после выполнения БПФ-32 над младшей половиной можно будет сразу приступать к исполнению операций BF над парами, которые образуются из чётных элементов младшей и старшей половины вектора.

Далее чётные элементы младшей половины выгрузим в память, а на их место загрузим временно отгруженные в память нечётные элементы старшей половины. Загрузим также 16 коэффициентов с нечётными индексами в регистры, в которых ранее были коэффициенты с чётными индексами. Теперь исполним BF над нечётными парами, образуемыми нечётными элементами младшей и старшей половины вектора. В результате после сохранения в памяти всех нечётных элементов будет завершена операция БПФ длины 64.

Загрузку элементов вектора в регистры будем выполнять сразу с учётом их бит-реверсного порядка. Заметим, что временно выгружаемые из регистров нечётные элементы старшей половины не могут испортить элементы, которые будут с учётом бит-реверсного порядка загружаться потом в регистры для обработки младшей половины, т.к. для элемента младшей половины с индексом $k=0b_4b_3b_2b_1b_0$ может быть загружен только чётный элемент из вектора с индексом $m=b_0b_1b_2b_3b_40$.

Посчитаем, сколько команд обращения к памяти потребуется при описанном способе исполнения БПФ длины 64. Загрузка коэффициентов и элементов для старшей БПФ-32: 16**L**+32**L**, +16**S** для временной выгрузки нечётных элементов старшей половины, загрузка элементов для младшей БПФ-32: 32**L**, +16**S** для выгрузки чётных элементов младшей половины, +16**L** для загрузки нечётных элементов старшей половины, +16**L** для загрузки нечётных коэффициентов и потом +48**S** для записи в память оставшихся элементов вектора. Итого получается: 112**L**+80**S** = всего 192 команды обращения к памяти, что почти в 2 раза меньше (416/192≈2,17), чем в вариантах Б и В.

Такой способ выполнения БПФ с применением CPV охарактеризуем как алгоритм БПФ **варианта Д** для N=64.

Он был реализован ассемблерной функцией, в которой команды обращения к памяти ещё и совмещались по времени (т.е. исполнялись параллельно) с командами вычислений бабочек Фурье как и при N=32. В результате для операции БПФ длины 64 удалось добиться такого же ускорения как и для БПФ длины 32.

Таблица 7. Ускорение БПФ (Д) на CPV для N=64

P	**$N=2^P$**	**Вариант Д на CPV**	**КДА**	**КДБ**
6	**64**	431	**19.51**	**2.95**

КДА – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает **вариант Д** на CPV по отношению к **варианту А** на CP1 (= такты варианта **А** / такты варианта Д)

КДБ – коэффициент выигрыша или ускорения, который для операции БПФ обеспечивает **вариант Д** на CPV по отношению к **варианту Б** на CPV (= такты варианта **Б** / такты варианта Д)

Это подтверждают полученные показатели производительности описанного алгоритма БПФ **варианта Д** для N=64, которые представлены в таблице 7.

Заключение

Алгоритм исполнения операции быстрого преобразования Фурье, первоначально реализованный для микропроцессора 1890ВМ8 семейства КОМДИВ с применением команд векторного сопроцессора, был усовершенствован с учётом особенностей функционирования иерархически организованной памяти.

В результате проведённой оптимизации алгоритма удалось существенно (почти в 1,5 раза) поднять производительность операции БПФ для векторов комплексных чисел большого размера (длины>512). Это позволило обеспечить на CPV стабильное ускорение для операции БПФ в том числе и при обработке массивов данных, целиком не вмещающихся в кэш-память 1-го или 2-го уровня микропроцессора 1890ВМ8Я (ВМ8) семейства КОМДИВ.

Optimization of algorithms of fast Fourier transform for the specialized vector coprocessor taking into account hierarchical structure of memory

A.A.Burtsev

Abstract. The classical algorithms of the fast Fourier transform (FFT) based on repeated execution over different couples of elements of a complex vector of the unified operation – so-called "Fourier butterfly", it is possible to accelerate significantly if to realize such operation by means of hardware. As a part of the microprocessor VM8 of the KOMDIV family the specialized vector coprocessor capable, in particular, to execute such operation of complex arithmetics as one command is provided. The FFT algorithms realized for the BM8 microprocessor using this command allowed to accelerate considerably execution of operations of Fourier transform over complex vectors (of length $n=2^p$) of a single and double precision. But at the same time it was noted that the received acceleration coefficient strongly varies depending on whether elements of a processed vector entirely in the cache memory of appropriate level are located (or entirely in registers of the coprocessor). Taking into account it, in view of features of a hierarchical structure of memory of the microprocessor, the algorithms of a FFT originally realized it was succeeded to improve and provide afterwards thereby a stable acceleration of operations FFTs even for the big vectors which entirely are not holding in the cache memory. Results of such enhancement of algorithms of a FFT are also provided in this article.

Keywords: microprocessors of the KOMDIV family, vector coprocessor, cache memory, digital signal processing, operation FFT.

Литература

1. Микросхема 1890ВМ8Я. URL: http:// https://www.niisi.ru/1890ВМ8Я.pdf (дата обращения: 26.07.2017).

2. А.А. Бурцев. О возможности оптимизации некоторых функций библиотеки линейной алгебры с помощью векторного сопроцессора // Труды НИИСИ РАН, т.4 №2. М.: Изд-во НИИСИ РАН, 2014. с.5-15.

3. А.А. Бурцев. О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье // Труды НИИСИ РАН, т.5 №2. М.: Изд-во НИИСИ РАН, 2015. с.138-147.

4. Kazushige Goto, Robert A. van de Geijn. Anatomy of high-performance matrix multiplication. // ACM Trans. on Mathematical Software, vol. 34, No. 3, 2008, pp.1-25.

Обработка исключительных ситуаций с использованием библиотеки мониторинга

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]grntl@niisi.ras.ru, [2]kostas@niisi.ras.ru, [3]ashch@niisi.ras.ru

Аннотация: Статья посвящена вопросам реализации средств управления обработкой исключительных ситуаций в многопоточной прикладной программе в среде операционной системы реального времени, функционирующей на многопроцессорной вычислительной системе. Рассматривается архитектура библиотеки мониторинга, а также технологические аспекты ее применения.

Ключевые слова: контролируемое выполнение, исключительная ситуация, исключение, библиотека мониторинга, многопоточная программа, поток управления, сигнал, операционная система реального времени, многопроцессорная система.

Введение

Исключительная ситуация (exception) – состояние внешних данных, устройств ввода-вывода или вычислительной системы в целом, в котором дальнейшее выполнение прикладной программы становится невозможными или бессмысленными. Классические примеры подобных ситуаций: деление на нуль, ошибка чтения данных с внешнего устройства, исчерпание ресурсов (память, дисковое пространство), прием сигнала выключения питания [1].

Программист может парировать подобные ситуации, используя структуры типа **`if-else`**, в которых реализована проверка критических условий выполнения, однако в этом случае алгоритм прикладной программы теряет прозрачность, обрастает многочисленными проверками и блоками обработки ошибок. Также следует принять во внимание, что требуемые для выполнения проверок библиотечные функции (или соответствующие системные вызовы ОС РВ) могут отсутствовать, и программист оказывается перед необходимостью разработки дополнительного функционала.

В состав ОС РВ Багет 2.6 входит гибкий механизм регистрации и обработки исключительных ситуаций на уровне ОС. Использование этого механизма позволяет программисту реализовать принцип контролируемого выполнения [2] в прикладной программе. Однако, чтобы избежать «ложного чувства безопасности» [5], программисту следует максимально точно определить множество возможных исключительных ситуаций и разработать соответствующие достоверные обработчики таких ситуаций (с полным тестовым покрытием, обеспечивающим высокую надежность кода). С точки зрения трудоемкости эта задача сравнима (или более трудоемка) с парированием исключительных ситуаций посредством конструкций **`if-else`**, о котором говорилось выше.

Библиотека мониторинга (БМ) [2, 6] реализует функции, которыми должна обладать система, спроектированная в соответствии с принципами контролируемого выполнения [2, 3, 4]. В соответствии с [2], программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы в критических условиях (например, при поступлении в программу некорректных данных, ошибок в среде исполнения или в самой прикладной программе). Средства контролируемого выполнения прикладной программы содержат механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, а также генерацию управляющих воздействий, в случае отклонения поведения программы от эталона.

Библиотека мониторинга выполняет контроль работоспособности, корректности и целостности прикладной программы на основании данных, получаемых от агентов мониторинга. При этом регистрация и обработка ошибок являются задачами, решаемыми прикладным программистом, и

он может использовать библиотечные функции БМ, в том числе и инструменты протоколирования и восстановления, предоставляемые БМ.

В данной статье рассматриваются приемы и решения использования механизма управления исключительными ситуациями в прикладной программе средствами БМ.

1. Архитектурные особенности обработки исключений с использованием БМ

Архитектура БМ и структура прикладной программы, работающей совместно с библиотекой мониторинга на одном процессоре, рассмотрена в [2].

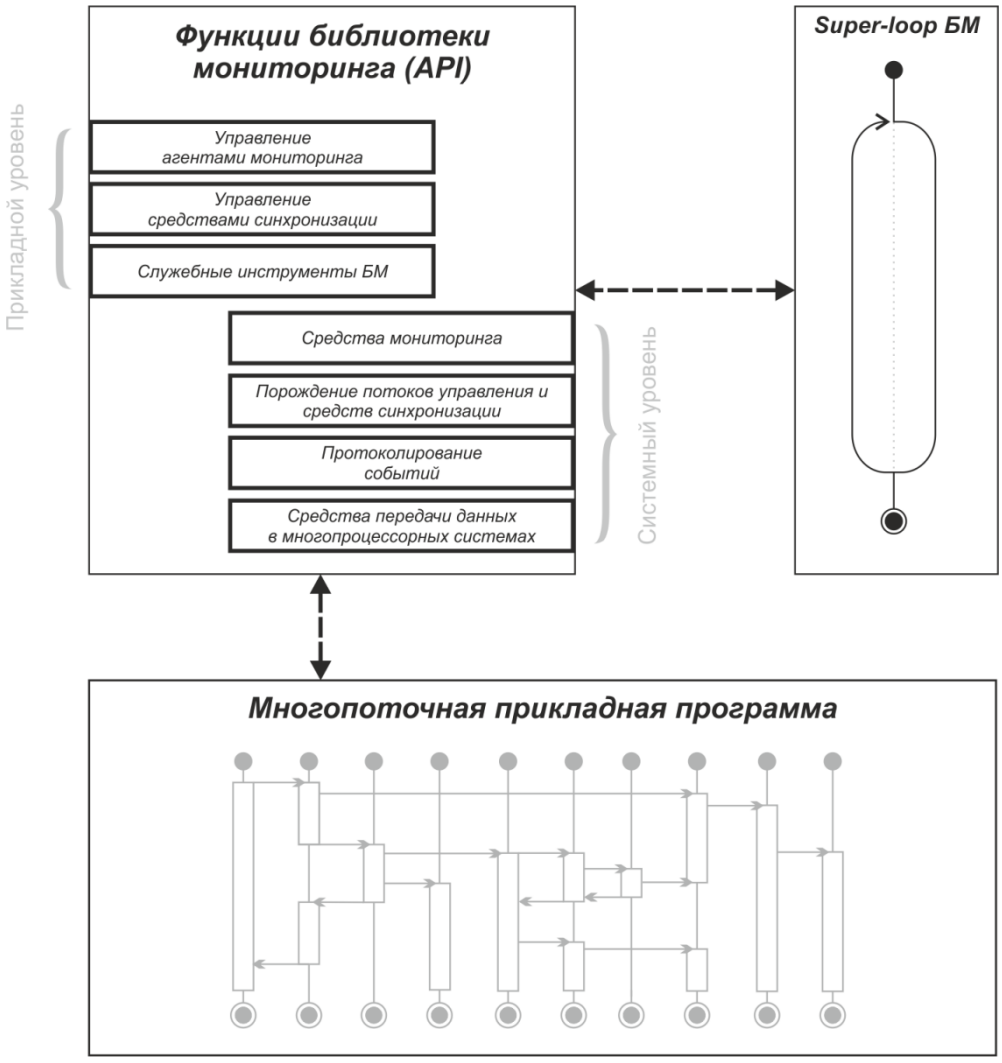


Рисунок 1. Схема взаимодействия БМ и прикладной программы

Структура прикладной программы, работающей совместно с библиотекой мониторинга на многопроцессорной системе, рассмотрена в [6].

На рисунке 1 представлена схема взаимодействия БМ с прикладной программой на уровне функций (API).

При старте ОС РВ управление передается головной функции БМ, реализующей **`super-loop`** [7], в котором выполняется последовательное обращение к функциям БМ системного уровня (см. рисунок 1). Эти функции в соответствии с заданной конфигурацией прикладной программы выполняют порождение потоков управления, средств синхронизации, мониторинга и протоколирования [6].

Возникновение исключительной ситуаций в потоке управления прикладной программы (т. е. на прикладном уровне) приведет к приостановке этого потока; БМ зарегистрирует отказ потока по таймауту (агент мониторинга в заданном потоке управления перестанет информировать локальный монитор о своей работе), при

этом причину отказа, способ восстановления и время отказа установить невозможно.

Для обработки исключительных ситуаций в БМ реализован дополнительный набор функций (так называемый «слой исключений», представленный на рисунке 2).

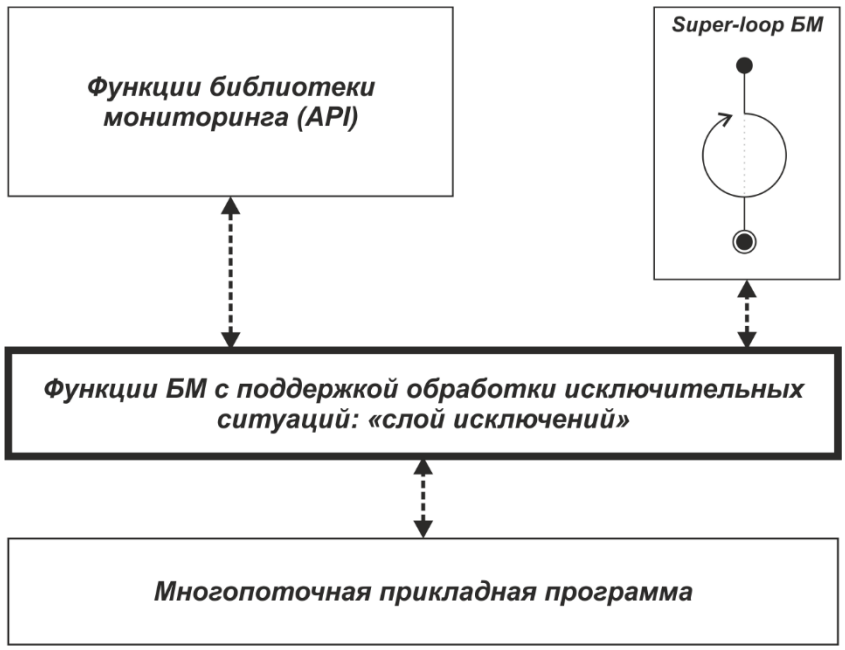


Рисунок 2. Схема взаимодействия БМ и прикладной программы с дополнительным набором функций

Из рисунка 2 следует, что прикладная программа, а также головная функция БМ, обращаются к API БМ через «слой исключений», функции которого обеспечивают управление исключительными ситуациями (регистрация и обработка).

Таким образом, исключительная ситуация, возникшая при выполнении прикладной программы, будет перехвачена БМ, которая запротоколирует общую информацию о возникшей исключительной ситуации и попытается восстановить работоспособность прикладной программы (сформирует реакцию на исключительную ситуацию).

2. Структура функции с обработкой исключительных ситуаций

В соответствии с документацией ОС РВ Багет 2.6 [1] в прикладной программе можно использовать гибкий способ обработки исключительных ситуаций аналогичный конструкции **`TRY-CATCH`** в C++, применимый как в потоках управления, так и в функциях обработки прерываний. Для этой цели предназначены макрооперации **`tryBegin()`**, **`tryCatch()`**, **`tryEnd()`**.

Ууниверсальный «скелет» функций из «слоя исключений» БМ соответствует исходному тексту, представленному в Листинге 1.

```
#include <try.h>
int _func1_exept( void )
{
    tryBegin();
        func1();
    tryCatch();
      struct tryINFO *p;
      tryInfo(p);
        checkpoint_agent(
                    p->cause);
        return -1;
    tryEnd();
    return 0;
}
```

Листинг 1. Исходный текст функции БМ с обработкой исключительных ситуаций

В примере из Листинга 1 при возникновении исключительной ситуации в функции **`func1()`** управление передается оператору, непосредственно следующему за **`tryCatch()`**. Если при выполнении функции **`func1()`** исключительная ситуация не возникнет, то участок

программы между **tryCatch()** и **tryEnd()** не будет исполняться и передается оператору, непосредственно следующему за **tryEnd()**.

3. Агенты мониторинга

Использование библиотеки мониторинга предполагает, что программист выполнил проектирование системы в терминах потоков управления, а именно – определил количество групп функциональных потоков управления, количество потоков управления, входящих в каждую группу, и разработал для каждого потока управления итеративную функцию, выполняющуюся в теле цикла, на языке Си [2]. При этом в итеративных функциях должны быть определены точки останова, и в этих точках должны быть размещены вызовы агентов мониторинга – функции прикладного уровня БМ **ckeckpoint_agent()**.

Реализация «слоя исключений» позволяет упростить интеграцию БМ в прикладную программу для «быстрого старта»: не требуется размещение агентов мониторинга в прикладных функциях. Мониторинг прикладной программы в этом случае заключается в регистрации исключительных ситуаций и самовосстановлении системы.

Следует отметить, что в блоке **tryCatch() - tryEnd()** может быть реализован запуск пользовательского обработчика исключительной ситуации, который должен быть определен для заданной головной функции на этапе конфигурирования БМ [6]. Кроме того, в **tryCatch() - tryEnd()** может быть выполнен стандартный сценарий: протоколирование системных структур БМ, связанных с заданным потоком управления, или повторный запуск потока управления.

4. Отслеживание значений переменных

Механизм обработки исключительных ситуаций позволяет реализовать проверку, например, значений целочисленных переменных непосредственно в процессе выполнения прикладной программы. Эта проверка реализована по аналогии с функцией **assert()**.

Метод проверки состоит в том, что в «слой исключений» вводится функция **is()**, в которую в качестве аргумента передается контролируемая переменная (см. Листинг 2)

```
int is(int retcode,
const int etalon, int causecode);
```

Листинг 2. Прототип функции отслеживания значений целочисленных переменных

В функции выполняется соответствующая проверка, и в случае отказа (результат проверки – ложь) возбуждается исключительная ситуация посредством системной функции **throwException()**. Пример реализации функции **is()** представлен в листинге 3.

```
#include <try.h>
void is(int retcode,
        const int etalon,
        int causecode ) {
   if (retcode != etalon) {
     throwException(
           causecode,
           (void*)NULL );
   }
}
```

Листинг 3. Пример реализации функции **is()**

5. Выводы

Представленная в статье технология управления исключительными ситуациями позволяет повысить надёжность приложений, использующих библиотеку мониторинга, упростить их отладку и увеличить точность регистрации ошибок и отказов. Применение данной технологии приводит к созданию отказоустойчивых программ, гарантирующих обработку исключительных ситуаций. Технология освобождает разработчика от рутинных действий по использованию типовых конструкций операционной системы.

Изложенный в статье подход применим как к однопроцессорным, так и к многопроцессорным системам. Создание библиотеки мониторинга выполнялось в рамках работ ФГУ ФНЦ НИИСИ РАН по технологии разработки систем реального времени.

Handling Exceptions Using the Monitor Library

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: The Article is devoted to the implementation of management tools handling exceptional situations in a multi-threaded application program in the real-time operating system environment, functioning on a multiprocessor computer system. Special attention is paid to the monitor library architecture, as well as the technological aspects of its application.

Keywords: controlled execution, exception, monitor library, multithread program, thread, signal, real-time operating system, multiprocessing system.

Литература

1. В.Л.Безруков, А.Н.Годунов, П.Е.Назаров., В.А.Солдатов, И.И.Хоменков. Введение в ос2000. Вопросы кибернетики. Информационная безопасность, Операционные системы реального времени, Базы данных /Под ред. чл.-корр. РАН В.Б.Бетелина. - Москва; НИИСИ РАН, 1999, – с. 76-106.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков. Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН. – Том 5, № 2. Построение программ реального времени. Москва, 2015, с. 113-121.
3. В.А.Галатенко. Контролируемое выполнение / В.А. Галатенко, К.А. Костюхин, Н.В. Шмырев – М: НИИСИ РАН, 2012. – 157 с.
4. В.Б. Бетелин. Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, N- 6, с. 45-55.
5. T. Cargill. Exception handling: A false sense of security. C++ Report, Nov-Dec 1994. Доступно по адресу: http://ptgmedia.pearsoncmg.com/images/020163371x/supplements/Exception_Handling_Article.html
6. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков. Библиотека мониторинга для многопоточных программ. // Труды НИИСИ РАН. – Том 7 № 1. Построение программ реального времени. Москва, 2017, с. 70-74.
7. M. Nahas and A. M. Nahhas. Ways for implementing highly-predictable embedded systems using time-triggered co-operative (TTC) architectures. INTECH Open Access Publisher, 2012.

Разработка программного комплекса моделирования тепловых процессов для теплового проектирования электронных систем

П.И. Кандалов[1], С.А. Кошкин[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]petrki87@gmail.com, [2] don.kell@mail.ru

Аннотация. В работе рассмотрена задача разработки высоконадежной системы автоматизированного теплового проектирования. Основное внимание уделено требованиям, предъявляемым к данным системам. Рассмотрена структура многофункционального программного комплекса теплового проектирования электронных систем МФПК-ТП-ЭС. Приведены недостатки существующих программных комплексов теплового проектирования электронных систем.

Ключевые слова. Программный комплекс, электронная система, тепловые процессы, тепловое проектирование, компьютерное моделирование, стохастический, математическое ожидание.

Введение

Разработка современных электронных систем (ЭС) требует использование высоконадежных систем автоматизированного проектирования. Практика создания ЭС различного назначения, предназначенных для эксплуатации в условиях воздействия тепловых, механических, вибрационных, химических, радиационных и космических факторов и экстремальных параметров окружающих сред, предъявляет жесткие требования к данным системам по всем видам дестабилизирующих воздействий. Тенденция к ужесточению и повышению требований к системам автоматизированного проектирования ЭС диктует, в свою очередь, повышение как уровня адекватности математического и компьютерного моделирования ЭС, так и многофункциональности проектирования.

В настоящее время методы математического и компьютерного моделирования, применяемые в системах автоматизированного теплового проектирования, строятся на допущение, что факторы, определяющие тепловые процессы, полностью известны и однозначно определены, то есть являются детерминированными, что приводит к результатам несоответствующим реальности. Практика проектирования и создания ЭС показывает, что реальные температурные поля технических систем являются недетерминированными и носят неопределенный интервальный характер. Неопределенность температурных полей обусловливается неопределенным характером параметров и факторов, определяющих тепловой режим ЭС [1], а пренебрежение их характером может приводить к существенным ошибкам при проектировании ЭС и нежелательным последствиям, как – выход характеристик ЭМ за пределы допустимых значений, потеря работоспособности, снижение быстродействия, надежности, помехозащищенности, невыполнению поставленной задачи [2; 3].

Разработка многофункционального программного комплекса (МФПК) автоматизированного теплового проектирования ЭС является критически важной для создания новых конкурентоспособных ЭС и охватывает все проблемы, связанные как с моделированием значений температуры в различных точках ЭС при различных условиях эксплуатации и испытаний, проектируемых ЭС, так и с конструированием и проектированием эффективных систем теплоотвода и охлаждения ЭИ.

Создание отечественных МФПК для проектирования ЭС является особенно актуальным на современном этапе, поскольку обеспечивает решение важнейшей задачи по импортозамещению и импортобезопасности программного обеспечения (ПО).

В статье рассматривается архитектура создаваемого программного комплекса, разбирается актуальность создания, выдвигаются основные требования к функциональным возможностям, необходимым для превосходства над конкурентами.

Назначение и структура программного комплекса

МФПК ТП-ЭС предназначен для обеспечения разработчиков элементов и ЭС в целом, высокоэффективным, адекватным реальности, с развитой пользовательской оболочкой инструментом, предназначенным для проведения всестороннего, многовариантного математиче-

ского и компьютерного моделирования тепловых процессов и температурных распределений в ЭС.

Разрабатываемый МФПК ТП-ЭС должен удовлетворять определенным требованиям и обладать следующими функциональными возможностями:

1) Моделировать тепловые режимы и температурные распределения в ЭС на различных иерархических уровнях [4], содержащих:

1 уровень – элементы ЭС (микросхемы (МС), электро- радиоэлементы (ЭРЭ));

2 уровень – электронные модули (ЭМ), включающие многослойную печатную плату, установленные на ней МС и ЭРЭ, теплоотводы, систему охлаждения ЭМ, конструктивные элементы крепления и монтажа ЭМ;

3 уровень – панель (блок, субблок и т.п. по различной терминологии), объединяющая несколько ЭМ вместе с конструктивными элементами крепления и монтажа, систему интерфейса, теплоотводы и систему охлаждения панели;

4 уровень – стойку, в состав которой входят несколько панелей, элементы конструкции крепления и монтажа панелей в стойке, система интерфейса и охлаждения.

2) Учитывать нелинейный, динамический и трехмерный характер тепловых процессов [4 – 10], развивающихся в ЭС, сложный характер теплообмена между элементами ЭС, а также разнородность конструктивных теплофизических параметров элементов на различных иерархических уровнях ЭС.

3) Обладать возможностью моделирования тепловых режимов и процессов в ЭС с учетом тепловой обратной связи, заключающейся во взаимовлиянии и взаимодействии электрических и тепловых процессов на различных иерархических уровнях ЭС между собой, которая возникает в силу значительной зависимости электрических параметров и потребляемой элементами мощности [1] от температуры, а температуры элементов – от изменения мощностей потребления элементов [11 – 14]. Причем тепловая обратная связь в ЭС может быть как положительной, так и отрицательной. В первом случае, если не принять специальных мер по охлаждению ЭС, будет наблюдаться лавинообразный рост мощности потребления и температуры, ведущий к перегоранию элемента, во втором − снижение мощности потребления с ростом температуры при одновременном падении быстродействия.

4) Моделировать тепловой режим ЭС при наличии взаимно-обусловливающих эффектов, таких как тепловая обратная связь, интервально статистический разброс тепловых и электрических параметров ЭС, воздействие дестабилизирующих механических, климатических и радиационных факторов.

5) Учитывать влияние внешних тепловых воздействий на тепловые режимы ЭС со стороны окружающих технических систем [15, 16].

6) Архитектура МФПК-ТП-ЭС должна содержать высокоэффективное математическое вычислительное ядро, а также развитую сервисную пользовательскую оболочку, отвечающую современным требованиям к сложным программным комплексам и системам [17]. Сервисная оболочка должна обеспечивать визуальную, наглядную и удобную для восприятия форму задания исходных данных, представления полученных результатов и визуализацию результатов в виде цветных изображений изотерм температурных распределений и других тепловых характеристик ЭС на различных иерархических уровнях.

Основу МФПК-ТП-ЭС (рис. 1) составляет графический интерфейс, который формирует иерархическое представление ЭС, обеспечивает задание исходных данных объекта моделирования и выполнят передачу данных между различными компонентами системы. Препроцессор создает требуемую математическую модель, основываясь на модулях информационного обеспечения, включающих на каждом уровне иерархии:

- тепловые параметры и характеристики ЭС и элементов;
- электрические параметры и характери-

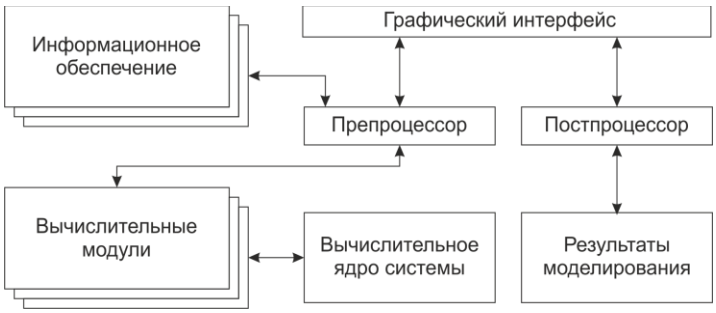


Рис. 1. Структурная схема МФПК-ТП-ЭС

стики ЭС и элементов;

- статистические меры электрических и тепловых параметров и характеристик элементов ЭС;

- параметры конструкций ЭС;

- параметры окружающей среды и теплоносителей внутри ЭС.

В состав МФПК-ТП-ЭС включены вычислительные модули следующего назначения:

- моделирование параметров детерминированных тепловых режимов ЭС;

- моделирование интервально стохастических параметров тепловых режимов ЭС;

- моделирование интервально стохастических температурных распределений ЭС и параметров теплового режима ЭС”

- моделирование тепловой обратной связи ЭС;

- моделирование интервально стохастических температурных распределений ЭС и параметров теплового режима ЭС с учетом влияния тепловой обратной связи;

Математическая модель, сформированная постпроцессором, передается в вычислительное ядро системы и после завершения расчета постпроцессор формирует результаты моделирования в виде удобном для пользовательского представления (таблицы, графики и т.д.). Расчетные значения температурных распределений и параметров ЭС отображаются в графическом интерфейсе пользователя.

Разработка МФПК-ТП-ЭС позволит исключить целый ряд недостатков присущих современным зарубежным ПК для тепловых расчетов, не позволяющими использовать их в практике теплового проектирования ЭС, в том числе отечественной:

- методы и модели, как и математические и компьютерные вычислительные алгоритмы, заложенные в зарубежных ПК, неизвестны пользователю и не раскрываются, что не позволяет судить об их адекватности, валидности и релевантности. Отсутствие информации об области и диапазоне применимости моделей, заложенных в конкретном ПК, сведений о математических моделях, методах решения уравнений математических моделей, используемых в данном ПК, не позволяет разработчикам новой техники доверять получаемым в ПК результатам и выводам и ориентироваться на них при создании новых ЭС;

- содержащиеся по умолчанию в зарубежных ПК физические и конструкционные параметры и характеристики материалов, элементной базы, полностью привязаны к зарубежным технологиям, конкретным компаниям, конструкциям и принятым стандартам, которые не применимы к отечественным материалам, технологиям, конструкциям и стандартам;

- библиотеки элементов (микросхем и электро- радиоэлементов), «вшитые» в структуру зарубежных ПК, не доступны для анализа специалистам по теплообмену, а их существенное отличие от отечественной номенклатуры, делает их неприменимыми к отечественной практике теплового проектирования;

- многофункциональность зарубежных ПК во многом ограничена вопреки декларируемым в них возможностям и зачастую не соответствует реальной практике при попытках их применения к расчетам тепловых процессов в конкретных конструкциях ЭМ. Так, в зарубежных ПК отсутствует такая важная для адекватного теплового анализа ЭМ возможность, как моделирование температурных распределений в многослойной печатной плате ЭМ, в которой на сегодняшний день может насчитываться более нескольких десятков разнородных слоев. Например, в ПК BetaSoft, количество слоев в печатной плате не превосходит 3-х, а в других зарубежных ПК имеется возможность задания лишь одного единственного усредненного слоя, причем метод приведения множества слоев к одному не приводится.

В настоящее время разработка программного комплекса МФПК ТП-ЭС проводится с использованием интегрированной среды разработки Microsoft Visual Studio на языке C# [17]. Применение объектно-ориентированного подхода обеспечивает МФПК высокую степень модульности благодаря таким свойствам, как инкапсуляция, полиморфизм и позднее связывание. Модульность архитектуры МФПК позволяет эффективно расширять возможности и модернизировать программный комплекс и обеспечивает создание надежного и эффективного комплекса моделирования.

Заключение

В работе представлена структура программного комплекса, предназначенного для моделирования тепловых процессов электронных систем на различных уровнях иерархии. Программный комплекс предлагает широкий спектр возможностей для проведения теплового моделирования, обработки и визуализации результатов моделирования.

Приведены системные недостатки зарубежных ПК, которые не позволяют рекомендовать их к безусловному применению в практике теплового проектирования ЭС, что наряду с решением проблемы импортозамещения программного обеспечения, еще больше повышает актуальность разработки и создания отечественных МФПК для теплового проектирования ЭС.

Software development for modeling of thermal processes for the thermal design of electronic systems

P.I. Kandalov, S.A. Koshkin

Abstract. The problem of development of highly reliable computer-aided design system of thermal design is considered. The article focused on the requirements required of the system. The structure of a multifunctional software package for the thermal design of electronic systems is considered. The disadvantages of the existing software systems for the thermal design of electronic systems are given.

Keywords. software, electronic system, thermal process, thermal design, computer simulation, stochastic, expected value.

Литература

1. С.Г.Бобков, А.Г.Мадера. Энергетические затраты, быстродействие и проблема теплоотвода в микропроцессорах // Программные продукты и системы. 2013. № 4. С. 104-106.
2. Конструкторско-технологическое проектирование электронной аппаратуры: Учебник для вузов / К.И. Билибин, А.И. Власов, Л.В. Журавлева и др.; под общ. ред. В.А. Шахнова. – 2-е издание, перераб. И доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. – 568с.
3. G.N. Ellison. Thermalcomputationsforelectronics. Conductive, radiative, andconvectiveaircooling. NY.: CRCPress, 2011. 416 p.
4. А.Г. Мадера Моделирование теплообмена в технических системах. М.: Науч. Фонд «Первая исслед. лаб. им. акад. В.А. Мельникова», 2005. 208 с.
5. А.Г. Мадера, П.И. Кандалов. Математическое моделирование интервально стохастических тепловых процессов в технических системах при интервальной неопределенности определяющих параметров // Компьютерное исследование и моделирование. 2016. Т. 8. №3. С. 501 – 520.
6. А.Г. Мадера, П.И. Кандалов. Моделирование трехмерных температурных полей в электронных модулях // Программные продукты и системы. 2010. №2. С. 36.
7. А.Г. Мадера, П.И. Кандалов. Моделирование температурных полей технических систем в условиях интервальной неопределенности // Тепловые процессы в технике. 2014. № 5. С. 225 – 229.
8. А.Г. Мадера Концепция математического и компьютерного моделирования тепловых процессов в электронных системах // Программные продукты и системы. 2015. № 3.С. 79 – 86
9. А.Г. Мадера, П.И. Кандалов Компьютерное моделирование температурных полей технических систем при интервально стохастической неопределенности параметров // Прикладная информатика. 2015. Т. 1(55). С. 106 – 113
10. А.Г.Мадера, П.И.Кандалов. Анализ интервально стохастических температурных полей технических систем. // Программные продукты и системы. 2014. №4 (108). С. 41-45
11. V. Camarchia, F. Capuleti, M. Pirola, S.D.G. Guettieeri, G. Ghione. Self-Consistent Electrothermal Modeling of Class A, AB, and B Power GaN HEMTs Under Modulated RF Excitation // IEEE Transactions
12. C.J. Keller, V.W. Antonetti. Statistical thermal design for computer electronics // Electronic Packaging and Production. – 1979. – V.19, No. 3. – P. 55 – 62.
13. O. Mueller. Internal thermal feedback in four-poles especially in transistors // Proceeding of the IEEE. 1964. V. 52 (8). P. 924 – 930.
14. N. Rinaldi, V. D'Alessandro. Theory of electrothermal behavior of bipolar transistors: part II-two-finger devices // IEEE Transactions Electron Devices, vol. 52, no. 9, pp. 2022 -2033, 2005.
15. А.Н. Чеканов. Расчеты и обеспечение надежности электронной аппаратуры: учебное пособие М.: КНОРУС, 2012. 440 с.
16. И.П. Норенков. Основы автоматизированного проектирования: Учебник для вузов / 4-е изд., перераб. и доп. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2009.
17. Дж. Рихтер. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. 4-е изд. – СПб.: Питер, 2017. – 896 с.: ил. – (Серия “Мастер-класс”)

Транспонирование трехмерной матрицы с использованием специализированного сопроцессора СР2 микропроцессора КОМДИВ128-РИО

М.С. Аристов[1], А.М. Щегольков[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]maristov@niisi.ras.ru, [2]ashch@niisi.ras.ru

Аннотация: В статье рассматривается реализация транспонирования трехмерной матрицы программированием специализированного сопроцессора обработки сигналов СР2 в составе процессора КОМДИВ128-РИО, разработанного в НИИСИ РАН. Приводятся основные особенности архитектуры КОМДИВ128-РИО и СР2 и возникающие в связи с этим сложности программирования данного сопроцессора. Приводится метод повышения производительности разрабатываемых для СР2 программ.

Ключевые слова: транспонирование трехмерной матрицы, микропроцессоры семейства КОМДИВ, К128-РИО, специализированный сопроцессор СР2.

Введение

Задача обработки больших массивов данных в режиме реального времени является одной из актуальных проблем во многих областях науки и техники, в том числе, при обработке сигналов, трехмерном моделировании, создании реалистичных изображений.

Для решения подобных задач в НИИСИ РАН разработан и используется процессор 1890ВМ7Я (КОМДИВ128-РИО), включающий в свой состав 128-разрядный специализированный сопроцессор СР2 с архитектурой SIMD [1], предназначенный для применения в многопроцессорных вычислительных комплексах, ориентированных на большой объем вычислений и рекомендуемый для применения в системах цифровой обработки сигналов.

Настоящая статья посвящена эффективной реализации программы транспонирования трехмерной матрицы с использованием сопроцессора СР2.

1. Архитектурные особенности процессора 1890ВМ7Я

Процессор 1890ВМ7Я состоит из ядра – основного универсального процессора, DMA-контроллера и специализированного сопроцессора СР2 (далее – сопроцессора СР2, или просто СР2), предназначенного для цифровой обработки сигналов и других объемных вычислений на основе 32-разрядных вещественных чисел. Архитектура сопроцессора СР2 позволяет достичь максимальной производительности 8-GFlops. Особенность архитектуры сопроцессора СР2 состоит в том, что он последовательно выполняет вычислительные операции над несколькими потоками данных. Подобная архитектура накладывает определенные ограничения на возможности сопроцессора СР2, однако подобное решение позволяет достичь высокой производительности в случае цифровой обработки сигналов.

Сопроцессор СР2 содержит четыре вычислительные секции, каждая из которых имеет собственную (локальную) память данных, поэтому перед запуском программы на СР2 и после ее завершения необходимо выполнить пересылки данных между памятью основного процессора и памятью вычислительных секций СР2. Задачу передачи данных между памятью основного процессора и памятью вычислительных секций СР2 решает DMA-контроллер. Сопроцессор СР2, основной процессор и DMA-контроллер работают параллельно и независимо друг от друга, что можно эффективно использовать при обработке сигнальных данных.

Используя особенности параллельной работы DMA-контроллера и СР2 можно, с использованием DMA-контроллера, запустить, например, чтение/запись данных, размещаемых (размещенных) в первых половинах памяти вычислительных секций СР2, параллельно запустить программу СР2

для обработки данных, размещенных во вторых половинах памяти вычислительных секций СР2, а по окончании обработки и пересылки данных из первых половин памяти вычислительных секций СР2 в память основного процессора поменять местами указанные действия – запустить программу СР2 для обработки данных, размещенных в первых половинах памяти вычислительных секций СР2, и параллельно запустить чтение/запись данных, размещаемых (размещенных) во вторых половинах памяти вычислительных секций СР2. Подобная техника параллельной работы независимых устройств позволяет увеличить общую производительность в сравнении с последовательным запуском каждой из стадий работы программы.

В состав сопроцессора СР2 входят одна управляющая и четыре вычислительных секции. Каждая 64-разрядная команда СР2 исполняется параллельно на всех вычислительных секциях. В одной команде СР2 сочетаются одна арифметическая команда и одна управляющая. Арифметическая команда выполняет арифметические действия над данными, хранящимися в регистрах вычислительных секций.

Управляющая команда осуществляет чтение данных в регистры или запись данных из регистров. В одной команде СР2 одновременно задается арифметическая и управляющая команды. Исполнение арифметической и управляющей команд, размещенных в одной команде СР2, происходит параллельно.

Регистры сопроцессора 64-разрядные, а архитектура команд устроена таким образом, что позволяет размещать в одном 64-разрядном регистре два 32-разрядных числа. Вычислительные операции выполняются над 64-разрядными данными, что позволяет выполнять операции над двумя 32-разрядными числами за один такт в одной вычислительной секции.

В сопроцессоре СР2 выполнение всех команд конвейеризовано, то есть, команда может быть передана на исполнение до того, как будет готов результат предыдущей команды. Количество тактов, за которое исполняется команда, зависит от команды. Для арифметических операций конвейер имеет длину 8 тактов, для управляющих – 3 такта. Таким образом, после запуска, например, команды сложения вещественных чисел, результат будет записан только на 8-й такт, а параллельная управляющая команда чтения из памяти секции – на 3-й такт. Различные команды сопроцессора СР2 отрабатывают за различное количество тактов.

Изложенные выше архитектурные особенности позволяют реализовать два возможных метода оптимизации вычислений.

Первый метод подразумевает развертку циклов, используя свободные регистры, и объединение схожих команд в блоки длиной 8 или 16 тактов.

Второй метод – это конвейеризация. Конвейеризацию необходимо использовать, если количество регистров для развертки циклов недостаточно. Однако, так как результат работы команды записывается в регистр не сразу, а, например, на 8-й такт, то до записи результата данные в этом регистре можно использовать в других вычислительных командах.

Методы оптимизации программ для процессора КОМДИВ128-РИО подробно изложены в работе [3].

2. Алгоритм работы СР2

Алгоритм работы сопроцессора СР2 состоит в следующем. Код, выполняемый на универсальном процессоре под управлением операционной системы, загружает в память инструкций сопроцессора СР2 программу (исполняемый код; исходный код пишется на языке Ассемблера для СР2 [2]), а в память данных сопроцессора СР2 – данные, используемые программой СР2, после чего инициирует выполнение загруженной программы на СР2. По окончании выполнения программы на СР2 устанавливается флаг, доступный для проверки коду, выполняемому на универсальном процессоре. Проверив данный флаг, можно выгрузить результат работы из памяти данных СР2 для дальнейшей обработки на универсальном процессоре в среде операционной системы.

Однако особенности реализации механизмов пересылки данных между памятью основного процессора и памятью СР2 накладывают определенные требования к реализации программ для СР2.

DMA-контроллер оперирует непрерывными областями памяти по 16 байт, при этом адрес начала такой области памяти должен быть выровнен к определенной границе. Таким образом, в случае работы с данными типа **`float`** нет возможности записать первый элемент данных в одну область памяти СР2, второй – в другую, и так далее. Последовательные четыре элемента типа **`float`** копируются из памяти

основного процессора в память вычислительных секций СР2 и обратно последовательно.

3. Задача транспонирования трехмерной матрицы

В общем виде задачу транспонирования трехмерной матрицы комплексных чисел на языке программирования Си можно представить так:

```
//цикл по размерности x
for (x=0; x<xSize; x++)
   //цикл по измерению y
   for (y=0; y<ySize; y++)
      //цикл по измерению z
      for (z=0; z<zSize; z++)
{
         out[z][y][x].re =
             in[x][y][z].re;
         out[z][y][x].im =
             in[x][y][z].im;
       }
```

В этой программе **xSize**, **ySize** и **zSize** – размеры матрицы по соответствующим измерениям **x**, **y**, **z**.

3.1. Программная реализация задачи транспонирования матрицы для СР2

При программной реализации задачи для СР2 нужно учитывать особенности работы сопроцессора СР2. Как и любую другую программу для сопроцессора СР2, программную реализацию этой задачи можно разделить на три этапа:

– загрузка данных из памяти основного процессора в память вычислительных секций СР2;

– обработка данных программой для сопроцессора СР2;

– выгрузка данных из памяти вычислительных секций СР2 в память основного процессора.

В зависимости от размера исходной матрицы, можно выполнить программу как за один проход, так и за несколько.

В случае нескольких проходов разбивать массив можно по размерности **y** таким образом, что каждая секция СР2 обрабатывает только один подмассив по размерности **y**.

3.1.1. Загрузка данных из памяти основного процессора в память вычислительных секций СР2

Так как загрузка данных из памяти основного процессора в память вычислительных секций СР2 возможна только блоками по 16 байт, что соответствует двум комплексным числам типа **float**, то загрузка будет осуществляться следующим образом:

```
in[0][0][0] → память 1 секции
in[0][0][1] → память 1 секции
in[0][0][2] → память 2 секции
in[0][0][3] → память 2 секции
in[0][0][4] → память 3 секции
in[0][0][5] → память 3 секции
in[0][0][6] → память 4 секции
in[0][0][7] → память 4 секции
in[1][0][0] → память 1 секции
in[1][0][1] → память 1 секции
in[1][0][2] → память 2 секции
in[1][0][3] → память 2 секции
in[1][0][4] → память 3 секции
in[1][0][5] → память 3 секции
in[1][0][6] → память 4 секции
in[1][0][7] → память 4 секции
in[2][0][0] → память 1 секции
in[2][0][1] → память 1 секции
```

…

Для передачи данных между памятью основного процессора и памятью вычислительных секций СР2 используется DMA-контроллер. Управление работой DMA-контролера выполняет программа в виде дескриптора или в виде связного списка дескрипторов требуемых действий, которая располагается в памяти основного процессора. DMA-контроллер последовательно считывает дескрипторы и выполняет действия по передаче данных, предписанные дескрипторами [4].

Указанный выше алгоритм загрузки данных из памяти основного процессора в память вычислительных секций СР2 с использованием DMA-контроллера реализуется следующей программой формирования связного списка дескрипторов и передачи данных из памяти основного процессора в память вычислительных секций СР2:

```
void load_to_cp2(float * INaddr,
       int zSize, int ySize,
       int xSize)
{
   struct lmem_img lmem_lo;
   struct mem_img gmem_lo;
```

```
   unsigned long len =
            2*sizeof(float)*
            zSize*ySize*xSize
   lmem_lo.base = 0x0000;
   lmem_lo.half =
         DRV_CP2_MEM_HALF_ONE;
   lmem_lo.isize_b = len;
   gmem_lo.isize_b =
             len/(countrows);
   gmem_lo.esize_i =
             4*countrows;
   gmem_lo.ld =
    zSize*ySize*2*sizeof(float);
   gmem_lo.vbase = INaddr;
   drv_queue_enq(
     DRV_QUEUE_2Dx1D_4XSPLIT_IN,
     &gmem_lo, &lmem_lo);
}
```

3.1.2. Обработка данных сопроцессором CP2

После загрузки данных в память вычислительных секций CP2 необходимо запустить программу сопроцессора CP2 для перекладывания элементов внутри памяти вычислительных секций CP2.

В первой секции памяти CP2 данные расположены следующим образом:

```
[in[0][0][0], in[0][0][1],
in[1][0][0], in[1][0][1],
in[2][0][0], in[2][0][1], …]
```

Однако, так как пересылка и запись данных возможна только кусками по 16 байт, необходимо переставить элементы таким образом, чтобы в блоках по 16 байт были последовательные данные для результирующего массива:

```
[in[0][0][0], in [1][0][0],
in[2][0][0], in [3][0][0],
in[0][0][1], in [1][0][1],
in[2][0][1], in [3][0][1],
in[0][0][2], …]
```

Аналогичные перестановки необходимо сделать и в памяти остальных вычислительных секций CP2.

Указанный выше алгоритм реализует представленная ниже программа на языке Ассеблера для CP2 [2]:

```
cp2m_repack_kern_start:
   clr r0
   clr r1
   clr r4
   move a0,g3  ## Адрес входного
               ## массива IN
   move a4,g4  ## Адрес входного
               ## массива OUT
   move a1, a0
   move n0, g1
   do g1, Looprepack1
      do g0, Looprepack0
         lw f0, (r0) + n0
         nop;;nop
         sw f0, (r4) + 1
      Looprepack0: nop
      upaddr (r1) + 1
      move a0,a1
   Looprepack1: nop
   nop;;nop
   stopi 0
cp2m_repack_kern_end:
```

В этой программе в глобальные переменные помещаются следующие данные:

– в переменную **g3** – адрес входного набора данных;

– в переменную **g4** – адрес записи результирующего набора данных;

– в переменную **g0** – размерность исходного массива по оси **x**;

– в переменную **g1** – размерность исходного массива по оси **z**.

Так как перемещение данных происходит не последовательно, нельзя располагать входные и результирующие данные в одной и той же области памяти вычислительных секций CP2.

3.1.3. Выгрузка данных из памяти вычислительных секций CP2 в память основного процессора

В соответствии с расположением результирующих данных в памяти вычислительных секций CP2 считывать их из памяти вычислительных секций CP2 необходимо в следующем порядке:

```
память 1 секции → out[0][0][0]
память 1 секции → out[0][0][1]
память 1 секции → out[0][0][2]
память 1 секции → out[0][0][3]
память 2 секции → out[1][0][0]
память 2 секции → out[1][0][1]
память 2 секции → out[1][0][2]
память 2 секции → out[1][0][3]
память 3 секции → out[3][0][0]
память 3 секции → out[3][0][1]
…
```

Указанный выше алгоритм выгрузки данных из памяти вычислительных секций

CP2 в память основного процессора с использованием DMA-контроллера реализуется следующей программой формирования связного списка дескрипторов и передачи данных из памяти вычислительных секций CP2 в память основного процессора:

```
void load_from_cp2 (
   float * OUTaddr, int zSize,
   int ySize, int xSize)
{
      struct lmem_img lmem_lo;
      struct mem_img gmem_lo;
      unsigned long len =
         2*sizeof(float)*
         zSize*ySize*xSize;
      lmem_lo.base = 8*0x0800;
      lmem_lo.half =
         DRV_CP2_MEM_HALF_ONE;
      lmem_lo.isize_b = len;
      gmem_lo.isize_b =
         xSize *2*sizeof(float);
      gmem_lo.ld =
         zSize*xSize*
         2*sizeof(float);
      gmem_lo.esize_i = 4;
      gmem_lo.msize_i = ySize;
      gmem_lo.mld =
         zSize*ySize*xSize*
         maxcells*2*
         sizeof(float);
      gmem_lo.vbase = OUTaddr;
      drv_queue_enq(
   DRV_QUEUE_3Dx1D_4XSPLIT_OUT,
   &gmem_lo,&lmem_lo);
}
```

4. Результаты измерения времени вычисления и выводы

Транспонирование матрицы комплексных чисел размером 512, 16 и 6 по осям **x**, **y** и **z** соответственно на основном процессоре заняло 30 мс. При этом при использовании сопроцессора CP2 время составило 1.3 мс. Таким образом, использование сопроцессора CP2 увеличило производительность в 23.07 раза.

Однако программная реализация алгоритма транспонирования трехмерной матрицы комплексных чисел для сопроцессора CP2 накладывает определенные ограничения на размерности матрицы, а именно:

– размерность по оси **x** должна быть кратна 4;

– размерность по оси **z** должна быть кратна 4.

Производительность сопроцессора CP2 при решении подобных задач значительно выше производительности основного процессора, поэтому даже если исходная матрица не удовлетворяет перечисленным выше ограничениям, ее приведение к подходящему виду и решение задачи транспонирования трехмерной матрицы комплексных чисел с использованием сопроцессора CP2 будут выполнены значительно быстрее, чем решение такой задачи с использованием только основного процессора.

3D-Matrix Transposition Using Specialized Co-processor CP2 of KOMDIV128-RIO CPU

M.S. Aristov, A.M. Shchegolkov

Abstract: The paper describes the implementation of 3D-Matrix transposition by programming specialized digital signal co-processor CP2 of KOMDIV128-RIO CPU developed in SRISA RAS. The key specs of KOMDIV128-RIO and CP2 are provided and main problems of programming this co-processor are considered. The method of improving performance of software developed for CP2 is given.

Keywords: 3D-Matrix transposing, the KOMDIV CPU family, K128-RIO, specialized co-processor CP2

Литература

1. Микросхема интегральная 1890ВМ7Я (КОМДИВ128-РИО), указания по применению. М.: НИИСИ РАН, 2009.
2. Программное изделие «Ассемблер для специализированного сопроцессора (СР2) в составе микропроцессора Комдив128-РИО (АССК128)». Руководство программиста. М.: НИИСИ РАН, 2014.
3. М.С. Аристов. Методы оптимизации программ для процессора КОМДИВ128-РИО. «Труды НИИСИ РАН», 2014, Т. 4, № 2, с. 33-39.
4. Программа «Библиотека цифровой обработки сигналов (БЦОС)». Справочник по функциям с интерфейсом ср2m программы БЦОС. М.: НИИСИ РАН, 2014.

Организация однонаправленной передачи данных в защищённый сегмент вычислительной сети

А.В.Баранов, А.А.Завальский

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail: antbar@mail.ru

Аннотация. Работа посвящена вопросам применения программно-аппаратного комплекса «ТОК-11» для организации канала однонаправленной передачи данных в защищённый сегмент локальной вычислительной или территориально распределённой сети. В статье приведена схема применения комплекса «ТОК-11», рассмотрены политика безопасности и регламент доступа к каналу однонаправленной передачи данных. Для возможности обеспечения контроля и предоставления доступа к каналу однонаправленной передачи рассмотрена предложенная авторами архитектура соответствующей программной системы.

Ключевые слова: канал однонаправленной передачи данных, защищённый сегмент сети, политика безопасности, регламент доступа

1. Введение

В Межведомственном суперкомпьютерном центре РАН не первое десятилетие ведутся научно-исследовательские работы в области методов построения сетей передачи данных для высокопроизводительных вычислений. Важное внимание при этом уделяется вопросам защиты данных как в локальных [1], так и глобальных [2] вычислительных сетях. Одним из актуальных современных направлений исследований является поиск и разработка решений для создания защищённого сегмента сети суперкомпьютерного центра [3], при этом возникает необходимость организации однонаправленной передачи данных из открытого сегмента сети в защищённый.

В настоящее время для передачи данных из одного сегмента локальной вычислительной или территориально распределённой сети в другой сегмент, имеющий более высокий уровень конфиденциальности, часто используются различные комплексы однонаправленной передачи. Примером сертифицированного устройства, осуществляющего гарантировано однонаправленную передачу данных между сегментами сети, является программно-аппаратный комплекс «ТОК-11». Функциональность комплекса «ТОК-11» обеспечивает передачу данных между сегментами сети, но не управление доступом пользователей к каналу однонаправленной передачи данных. Отсутствие функции управления доступом создаёт угрозы информационной безопасности для канала однонаправленной передачи (далее – КОП) как со стороны потенциального злоумышленника, так и вследствие случайных ошибочных действий пользователей.

Целью настоящей работы является разработка организационных и методологических принципов предоставления и контроля доступа к каналу однонаправленной доставки информации в закрытый сегмент локальной вычислительной сети, а также разработка архитектуры программной системы, реализующей эти принципы.

2. Программно-аппаратный комплекс «ТОК-11»

Разработанный ООО «Праймтек» программно-аппаратный комплекс (далее – ПАК) «ТОК-11» – комплекс однонаправленной передачи данных – устройство, обеспечивающее передачу информации из одной ЛВС в другую и не позволяющее передачу в обратном направлении. ПАК «ТОК-11» предназначен для осуществления гарантированной автоматической однонаправленной передачи данных из открытых информационно-телекоммуникационных сетей (в том числе сети Интернет) в изолированные информационные системы.

В соответствии с [4], основными функциями ПАК «ТОК-11» являются:

- автоматическое подключение к информационной системе актуальных баз данных компьютерных вирусов по защищённому каналу передачи данных для получения актуальных баз данных компьютерных вирусов;
- автоматическая доставка в изолированные информационные системы актуаль-

ных баз данных компьютерных вирусов для сертифицированных антивирусных средств;

- гарантированная автоматическая однонаправленная передача в изолированные информационные системы файлов/каталогов любой организации и объёма, веб-страниц.

Конструктивно ПАК «ТОК-11» (рис. 1) выполнен в виде двух компонентов размером 2U для монтажа в стойку, условно обозначаемых как «Передатчик» и «Приёмник».

Устройство «Передатчик» предназначено для подготовки и передачи данных в «Приёмник», подключенный к закрытому сегменту ЛВС.

Устройство «Приёмник» принимает данные, проверяет их целостность и предоставляет доступ к ним.

Устройства «Приёмник» и «Передатчик» соединены между собой одножильным оптическим кабелем.

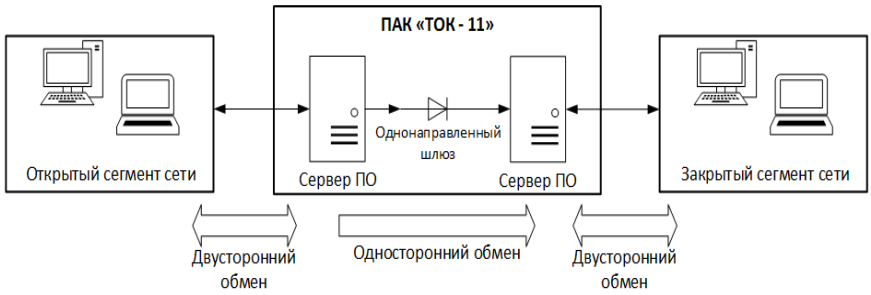


Рис. 1. Схема применения программно-аппаратного комплекса «ТОК-11»

В состав устройства «Передатчик» входят следующие компоненты:

- сертифицированное устройство однонаправленной передачи данных (шлюз «З-Кросс»);
- сервер со специализированным программным обеспечением.

Шлюз «З-Кросс» является средством гарантировано однонаправленной передачи информации из открытых сетей в сети, в которых циркулирует информация ограниченного доступа. Однонаправленность передачи реализована с использованием программируемой логической интегральной схемы фирмы Xilinx. Контроль питания, хранение конфигурации устройства, загрузка конфигурации в ПЛИС, работа дисплея обеспечиваются микроконтроллером на базе ядра ARM7.

При передаче данных из открытого сегмента в закрытый шлюзом осуществляется фильтрация IP-пакетов в соответствии с таблицей разрешённых к передаче адресов и протоколов.

Сервер, входящий в состав устройства «Передатчик», имеет следующие характеристики:

- операционная система Debian Linux;
- процессор Intel Core i3;
- 8 гигабайт оперативной памяти.

При работе «Передатчика» используется следующее ПО:

- веб-сервер Apache Tomcat;
- программный комплекс «Альфикс-АП», разработанный ООО «Р-Альфа Лаб»;
- антивирусное средство М-42, разработанное ООО «Праймтек».

Программный комплекс «Альфикс-АП» является сертифицированным средством криптографической защиты передаваемой информации и используется для установления защищенного соединения с информационной системой актуальных баз данных компьютерных вирусов через сеть Интернет.

Антивирусное средство (далее – АВС) М-42 предназначено для проверки файлов, передаваемых в закрытый сегмент сети через ПАК «ТОК-11».

В состав устройства «Приёмник» входят следующие компоненты:

- устройство приема данных от шлюза однонаправленной передачи;
- сервер со специализированным программным обеспечением.

Устройство приёма данных от «Передатчика» представляет собой медиаконвертер.

На его вход по подключаемому через разъем типа «SC» оптоволоконному кабелю подаётся сигнал, после чего данные, пришедшие из открытого сегмента сети, с помощью кабеля UTP передаются на вход сервера, входящего в состав устройства «Приёмник».

Сервер, входящий в состав устройства «Передатчик», имеет следующие характеристики:

- ОС Windows Server 2008 R2;
- процессор Intel Xeon;
- 16 гигабайт оперативной памяти;
- SSD объемом 128 гигабайт для хранения пользовательских данных.

Диск разбит на три раздела:

- системный;

- раздел хранения обновлений антивирусных баз;

- раздел хранения данных, полученных от устройства «Передатчик».

При работе «Приёмника» используется ПО:

- веб-сервер Apache Tomcat;

- антивирусное средство М-42.

Получение обновлений антивирусных баз данных происходит следующим образом.

1. Используя веб-интерфейс настройки устройства «Передатчик», администратор сети выбирает, базы данных каких антивирусов необходимо получать, а также устанавливает таймер, по которому будет происходить получение.

2. Используя программный комплекс «Альфикс-АП», сервер, входящий в состав устройства «Передатчик», устанавливает защищённое соединение с информационной системой актуальных баз данных компьютерных вирусов.

3. Из информационной системы загружаются актуальные версии баз данных компьютерных вирусов для выбранных антивирусных средств и сохраняются во внутреннем хранилище устройства «Передатчик».

4. С помощью однонаправленного шлюза «З-Кросс» полученные обновления в виде архива передаются в устройство «Приёмник», там разархивируются и помещаются в раздел жёсткого диска, предназначенный для хранения баз данных.

После выполнения указанной последовательности администратор сети может обращаться к устройству для получения актуальных баз.

Передача файлов из открытой сети в закрытую происходит следующим образом (рис. 2).

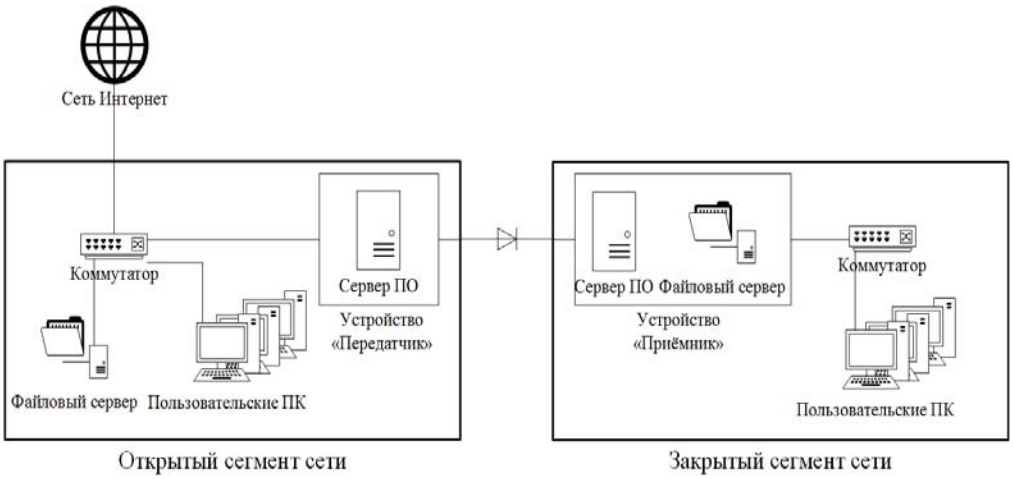


Рис. 2. Схема однонаправленной передачи информации в закрытый сегмент

1. Используя веб-интерфейс настройки устройства «Передатчик», администратор сети выбирает источник, из которого будут передаваться файлы в закрытый сегмент.

Помимо этого, настраивается таймер, по истечении которого файлы забираются из выбранного источника, а также таймер хранения, по истечении которого файлы удаляются из выбранного источника.

2. С помощью АВС М-42 файлы, предназначенные для передачи, проверяются на наличие заражений. Заражённые файлы удаляются, информация о них записывается в журнал работы устройства «Передатчик».

3. Файлы, прошедшие проверку, загружаются непосредственно в устройство «Передатчик», для каждого рассчитывается хэш-сумма, записываемая в текстовый файл.

После этого передаваемые файлы и файл с хэш-суммами архивируются и с помощью однонаправленного шлюза «З-Кросс» передаются в устройство «Приёмник».

4. Как только от устройства «Передатчик» приходит архив с файлами, его содержимое разархивируется на смонтированный в «Приёмнике» RAM-диск.

5. Для разархивированных файлов снова высчитывается хэш-сумма и сверяется со значением в полученном текстовом файле.

6. Полученные разархивированные файлы снова проходят проверку АВС М-42, после чего помещаются в раздел жёсткого диска файлового сервера, предназначенный для хранения файлов.

После выполнения указанной последовательности пользователи закрытого сегмента сети могут обращаться за файлами на файловый сервер.

3. Организационное обеспечение безопасности комплекса однонаправленной передачи

Для поддержания необходимого уровня информационной безопасности требуется определение целей и задач информационной безопасности, выявление актуальных угроз и определение потенциальных нарушителей. Применительно к каналу однонаправленной передачи, основываясь на [5], указанные вопросы должны быть освещены в Политике безопасности сетевых ресурсов канала однонаправленной передачи.

Кроме этого, важным является определение категорий лиц, имеющих доступ к КОП, порядка предоставления и изъятия доступа, описание действий, подлежащих регистрации.

Для освещения данных вопросов требуется разработка Регламента доступа к каталогам канала однонаправленной передачи.

В соответствии с рекомендациями [5], политика безопасности разрабатывается с целью обеспечения управления информационной безопасностью согласно требованиям организации и включает в себя разделы, посвящённые описанию различным аспектам информационной безопасности.

При рассмотрении особенностей применения КОП был выявлен ряд опасных с точки зрения информационной безопасности факторов, а именно существующее подключение открытого сегмента к сети Интернет, сопровождение ПАК «ТОК-11» сотрудниками организации-разработчика комплекса.

В тексте документа данные факторы нашли свое отражение при описании каналов возможных атак, в модели угроз безопасности и модели нарушителей.

Разработка политики безопасности, как политики нижнего уровня, происходила с учётом действующих нормативных актов, таких как политика безопасности верхнего уровня и регламент получения учётных записей пользователей.

Политика безопасности содержит следующие разделы:

- задачи обеспечения информационной безопасности;
- объекты, защищаемые политикой;
- модель угроз безопасности, включающую описание возможных каналов для атак на защищаемые объекты и классификацию актуальных угроз безопасности;
- модель нарушителей безопасности, определяющую классификацию нарушителей, для каждого класса нарушителей описаны возможности по реализации атак, наиболее вероятные каналы атак и группы людей, которых можно отнести к классу нарушителей;
- методы и меры по противодействию угрозам безопасности, описанным в модели угроз;
- порядок реагирования ответственных за информационную безопасность сотрудников на возникающие инциденты;
- порядок пересмотра политики.

При разработке разделов, посвящённых порядку реагирования на инциденты безопасности и порядку пересмотра политики, авторы отталкивались от существующих организационных процессов.

При разработке регламента доступа к входным и выходным каталогам канала однонаправленной передачи авторами преследовалась цель создания документа, освещающего следующие аспекты:

- категории должностных лиц, имеющих доступ к каналу, а также их права при обращении к каналу;
- порядок предоставления и изъятия прав доступа;
- порядок протоколирования доступа пользователей.

В частности, были определены способы фиксации факта предоставления доступа, определены причины, на основании которых он может быть предоставлен или ограничен, проработаны вопросы изменения прав при переводе сотрудников из одного подразделения в другое. Регламент доступа содержит следующие разделы:

- структура сетевых ресурсов КОП;
- группы пользователей, имеющие доступ к каталогам КОП;
- права групп пользователей, допущенных к КОП;
- порядок предоставления и ограничения доступа пользователей к КОП;
- порядок регистрации событий доступа пользователей при обращении каталогам КОП.

4. Система предоставления и контроля доступа к каналу однонаправленной передачи

Авторами была разработана архитектура системы предоставления и контроля доступа (далее – СПКД) к каналу однонаправленной передачи данных (рис. 3).

Система решает следующие задачи:

- предоставление доступа к каналу однонаправленной передачи;

- контроль изменения прав доступа к каналу однонаправленной передачи;

- настраиваемый контроль доступа пользователей к каналу однонаправленной передачи, включающий информирование о произошедших попытках несанкционированного доступа.

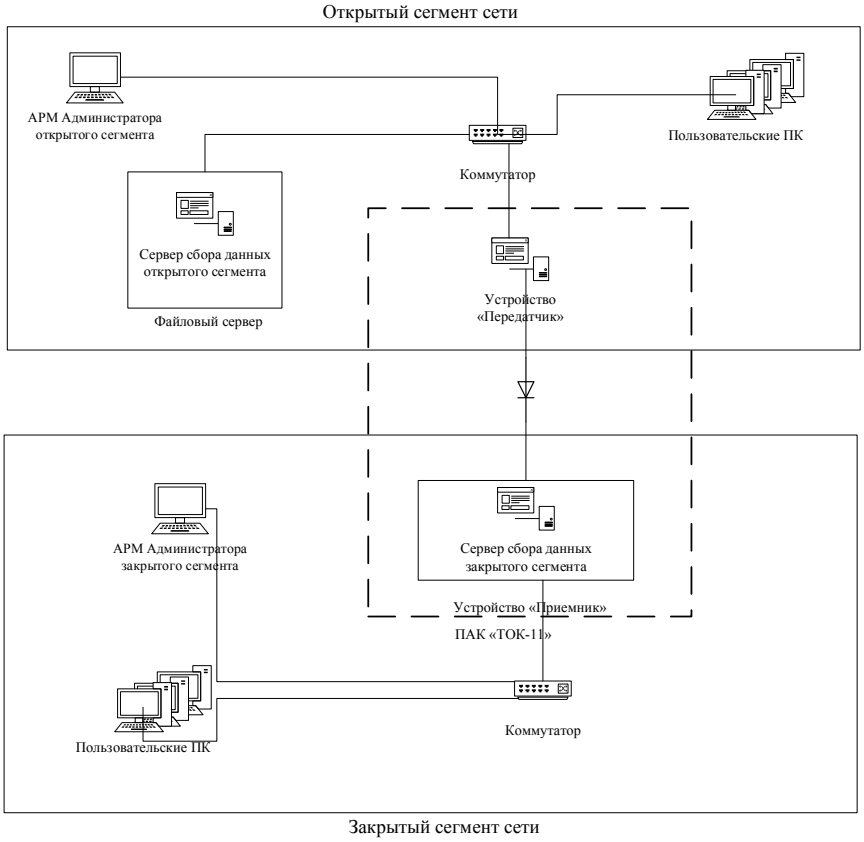


Рис. 3. Архитектура системы предоставления и контроля доступа к каналу однонаправленной передачи данных

Всю систему можно условно разделить на две подсистемы: открытого и закрытого сегментов. Каждая подсистема построена в соответствии с клиент-серверной архитектурой и состоит из автоматизированного рабочего места (далее – АРМ) администратора и сервера сбора данных.

АРМ администратора открытого сегмента предназначено для управления доступом к входным каталогам КОП, настройки контролируемых событий для входных каталогов.

Сервер сбора информации открытого сегмента предназначен для сбора данных о доступе к входным каталогам КОП и их передачи на сервер сбора информации закрытого сегмента.

Сервер сбора информации закрытого сегмента предназначен для сбора данных о доступе к входным и выходным каталогам комплекса, обеспечивает хранение данных об изменении прав доступа к входным и выходным каталогам, событиях доступа к входным и выходным каталогам.

АРМ администратора закрытого сегмента предназначено для управления доступом к выходным каталогам, отображения информации о событиях доступа к входным и выходным каталогам КОП, настройки контролируемых событий для выходных каталогов, отображения прав доступа к входным и выходным каталогам.

На рисунке 4 приведена схема функциональной структуры системы предоставления и контроля доступа.

Подсистемы обработки запросов обеспечивают взаимодействие между АРМ администратора и серверами сбора данных открытого и закрытого сегментов соответственно.

Подсистема генерации лог-файлов сервера сбора данных открытого сегмента выполняет функцию преобразования данных об изменении прав доступа пользователей и событиях доступа или попыток доступа к входным каталогам канала однонаправленной передачи в xml-файлы специального формата и последующей передачи в закрытый сегмент.

Подсистема приёма и обработки лог-файлов сервера сбора данных закрытого сегмента предназначена загрузки информации из лог-файлов в подсистему хранения данных.

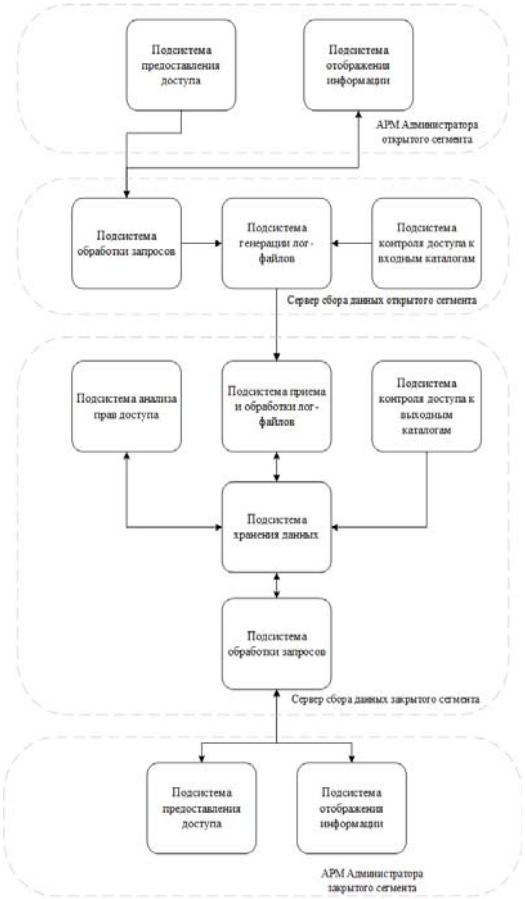


Рис. 4. Функциональная структура системы контроля и предоставления доступа к КОП

Подсистемы генерации лог-файлов и их приёма и обработки являются составными частями подсистемы передачи данных от сервера сбора данных открытого сегмента на сервер сбора данных закрытого сегмента.

Подсистема анализа прав доступа предназначена для отслеживания изменений прав доступа к входным и выходным каталогам, совершённых не через СПКД.

Подсистемы контроля доступа выполняют функцию сбора и анализа информации о событиях доступа к каталогам канала однонаправленной передачи, сохранение этой информации в подсистеме хранения данных.

Подсистема хранения данных представляет собой базу данных, содержащую информацию:

- о событиях доступа к входным и выходным каталогам;
- журналы изменения прав доступа к входным и выходным каталогам.

Подсистемы предоставления доступа осуществляют управление правами доступа пользователей к входным (на АРМ администратора открытого сегмента) и выходным (на АРМ администратора закрытого сегмента) каталогам КОП. С помощью этих подсистем возможна настройка подлежащих регистрации событий доступа к каталогам.

Работоспособность рассмотренной архитектуры была подтверждена по результатам опытной эксплуатации макета СКПД.

5. Заключение

По итогам настоящей работы можно сделать следующие выводы. Существующие сертифицированные средства могут быть с успехом применены для создания канала однонаправленной передачи данных в защищённый сегмент сети, однако их функциональных возможностей недостаточно для комплексного обеспечения информационной безопасности. Последнее может достигнуто за счёт предложенной авторами системы предоставления и контроля доступа на основе политики безопасности и регламента доступа к каналу однонаправленной передачи.

Работа выполнена в МСЦ РАН при поддержке программы Президиума РАН I.5П «Проблемы создания высокопроизводительных, распределенных и облачных систем и технологий. Интеллектуальные информационные технологии и системы».

Organization of unidirectional data transmission into a network's protected segment

A.V.Baranov, A.A.Zavalskiy

Abstract. The paper is devoted to the application of the TOK-11 software and hardware complex for organizing a channel of unidirectional data transmission into a local or global network's protected segment. The application scheme of the complex TOK-11, security policies, and access procedure to the unidirectional data transmission channel are presented. The software system architecture designed by the authors in order to provide control and access to the unidirectional transmission channel is considered.

Keywords: unidirectional data transmission channel, network's protected segment, security policies, access procedure

Литература

1. О.С. Аладышев, М.Р. Биктимиров, М.А. Жижченко, А.П. Овсянников, В.М. Опилек, Б.М. Шабанов, Н.Ю. Шульга. Особенности построения объединенной сети суперкомпьютерного центра. «Программные продукты и системы», (2008), № 2. С. 9-12.

2. Г.И. Савин, Б.М. Шабанов, П.Н. Телегин, О.И. Вдовикин, А.П. Овсянников, И.А. Козырев, В.В. Корнеев, Д.В. Семёнов, А.В. Киселёв, А.В. Кузнецов. Инфраструктура грид для суперкомпьютерных приложений. «Известия высших учебных заведений. Электроника», (2011), № 1 (87). С. 51-56.

3. Б.М. Шабанов, А.П. Овсянников, А.В. Баранов, Е.А. Киселёв, С.А. Лещев, Б.В. Долгов, Д.Г. Гуменный, Д.Л. Шуров. Решение задач фотореалистичной компьютерной графики на базе защищённой инфраструктуры суперкомпьютерного центра коллективного пользования. «Суперкомпьютерные дни в России: Труды международной конференции (25-26 сентября 2017 г., г. Москва)», М.: Изд-во МГУ, 2017. С. 733-741.

4. Комплекс однонаправленной доставки информации в изолированные системы «ТОК-11/ТОК-QP». [В Интернете] https://primetech.ru/products/tok (дата обращения: 12.10.2017)

5. ГОСТ Р ИСО/МЭК 27002-2012. Информационная технология (ИТ). Методы и средства обеспечения безопасности. Свод норм и правил менеджмента информационной безопасности.

Оптимизация ведения складского учета

Г.Л. Левченкова

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: galka_lev@mail.ru

Аннотация: Рассматривается способ организации данных и методы занесения дополнительной информации в систему складского хранения/перемещения материальных ценностей с целью получения актуального фактического состояния дел.

Ключевые слова: складская система, хранение, перемещение, учет

1. Введение

При рассмотрении известных автору складских систем, внедренных на предприятиях, была замечена ограниченность возможности отражения фактического состояния дел, касающихся месторасположения материальной ценности (далее по тексту МЦ). Например, согласно учету "по бухгалтерии" предмет находится на складе, но фактически выдан на руки работнику для, например, проверки работоспособности или для пересчета, то есть не "навсегда", а на несколько часов. Материальная ценность может быть передана на участок сборки для проверки совместимости, и только по результатам исследований будет требоваться сделать перемещение "по бухгалтерии". Бывает ситуация, когда согласно документам предмет учтен на складе, но на полке не лежит: документы на получение проведены бухгалтерией, но материальная ценность на склад еще не передавалась. Обратный случай: МЦ принесли, положили на стеллаж, но бухгалтерские документы еще не приняты к учету, то есть "по бухгалтерии" материальной ценности еще нет. И в каждом из этих случаев информацией "что, где когда и зачем" обладает только работник склада, который выдал или принял товар лично. Причем этот сотрудник рассчитывает на свою память либо записывает информацию на бумажках, которые складывает, копит и иногда просматривает.

Таким образом, действительное положение дел оказывается недоступным через установленную систему учета. И по прошествии некоторого количества времени может возникнуть ситуация, когда что-то потребуется найти, а на полке этого не окажется. Да и вспомнит ли кто-нибудь из складских работников, кто именно обладает хоть какой-никакой информацией по данному вопросу?

В настоящей статье предлагается способ организации данных, который позволит хранить в актуальном виде всю информацию, касающуюся нахождения материальной ценности: как фактически, так и "по документам".

2. Информация

Определим, какие основные данные необходимо иметь в распоряжении, чтобы корректно оперировать процессами учета, хранения и перемещения материальных ценностей:

- наименование МЦ;
- единица измерения;
- количество;
- документ оприходования (принятие к учету "по бухгалтерии" на предприятии);
- документ поступления (при приеме на склад "по бухгалтерии");
- фактическое принятие на склад (отметка о поступлении);
- документ выдачи со склада (при передаче со склада "по бухгалтерии");
- фактическая передача со склада (отметка о выдаче);
- отметка о фактическом наличии/отсутствии МЦ на складе (временное состояние, длящееся до получения документов либо до принятия/отпуска МЦ);
- место расположения/хранения МЦ;
- примечания.

Помимо вышеперечисленных данных, которые на первый взгляд однозначно определяют МЦ, необходимо иметь возможность указать дополнительные сведения, которые будут влиять на процессы дальнейшего хранения и перемещения товаров. Причем указать не в поле "Примечание", а в отдельно расположенных колонках, по которым можно будет впоследствии проводить отбор и поиск нужной информации:

- код товара (МЦ);
- упаковка;
- инициация закупки (документ);
- назначение;
- партия (гарантия);
- серийный/инвентарный номер;
- уточняющее описание МЦ;
- признак: разукомплектован или "вставлен" в изделие;
- признак "незавершенности" записи;
- признак "архивности" записи.

3. Приведение информации к "стандартному" виду

Слово "стандартному" взято в кавычки не случайно, поскольку СТАНДАРТ на каждом предприятии может отличаться от принятых в других учреждениях. Кроме того, необходимо учитывать, что "правила учета" на складе зачастую видоизменяются, дорабатываются самими складскими работниками, чтобы МЦ учитывались однозначно, независимо от того, как они проведены "по бухгалтерии".

Таким образом, в системе складского учета следует "стандартизировать" рассматриваемые ниже признаки МЦ.

Наименование МЦ, единица измерения, количество: основополагающая информация и, заметим, далеко не однозначно определяемая. К сожалению, не все базы данных способны предоставить достаточное количество учетных полей, которые необходимо (с точки зрения складского работника) заполнить информацией о "наименовании" поступившей МЦ. Кроме того, зачастую наименование вводится в базы данных работниками бухгалтерии, которые переносят его дословно из документа поступления (накладной на поставку), и, поскольку у различных поставщиков одинаковый товар может называться по-разному, то и в учетных данных купившего МЦ предприятия плодится множественное количество записей, среди которых выбрать одинаковые наименования весьма затруднительно. В итоге на склад вместе с МЦ поступает документ, в котором название товара весьма приблизительно соответствует однозначному пониманию "предмета", который должен учитываться.

Следовательно, для создания корректной записи, содержащей информацию о наименовании МЦ, в складской системе потребуется не одно поле базы данных, а несколько, именно:

- наименование "по приходным документам" (это именно те "слова", которые указаны в накладной поступления на предприятие);
- наименование "по бухгалтерии" (занесенное в бухгалтерскую систему название, в основной массе совпадает с наименованием "по приходным документам");
- наименование "по складу" (истинное, приведенное к максимально однозначному виду название МЦ, не дающее разночтений в толковании) – одно из основных полей, по которым производится учет и поиск товара на складе. Особое внимание следует уделить вводу русских/английских и больших/маленьких букв наименования, и если работник склада не может этого определить однозначно, то следует проконсультироваться у специалиста;
- единица измерения "по приходу" (указанная в накладной поступления на предприятие единица измерения);
- единица измерения " по бухгалтерии " (указанная в накладной поступления на склад единица измерения);
- единица измерения "по складу" (принятая к учету на склад единица измерения). Здесь напрашивается комментарий: в документах (накладных поступления) указывается единица измерения того товара, который был продан, но еще не факт, что организация, которая купила товар, к учету будет принимать именно такие элементы. Например, купили 10 бидонов с краской. Но к учету на складе принимаются 200 литров, поскольку выдавать со склада будут именно в литрах;
- код товара по складу (идентификационный номер МЦ, не связанный с упаковкой, единицами измерения и прочими несущественными отличиями товаров друг от друга). Многие производители включают в наименование своих товаров метод упаковывания либо кодируют в названии технологию изготовления, цвет и т.д., таким образом, товары разных фирм-производителей имеют различное наименование, но по функциональной сути являются полными аналогами. В этих случаях отбор строк базы данных по признаку "код товара по складу" приводит к выбору всех нужных товаров, хранящихся на складе, несмотря на то, что они могут носить разные учетные названия;
- код товара по бухгалтерии (идентификационный учетный номер МЦ, присвоенный в бухгалтерской системе учета при поступлении товара в организацию);
- "направление деятельности" – бухгалтерская учетная информация, необходимая для ведения учета по заказам

предприятия (присутствует в документах передачи на склад "по бухгалтерии);

- количество "по приходу" (указанное в накладной поступления на предприятие либо первично принятое к учету бухгалтерией);

- количество "для учета на складе" (принятое на хранение количество). В связи с изменением единиц измерения при приеме на склад, количество также корректируется. Кроме того, оно может измениться в следующих случаях: допустим, на складе хранятся некие комплекты, состоящие из 1-го "пакетика", содержащего 1 винт, 1 шайбу и 1 гайку. А в новом поступлении эти комплекты скомпонованы по два, то есть в одном комплекте "по накладной" лежит 2 винта, 2 гайки и 2 шайбы. Тогда к учету на склад следует принять не 10 комплектов, как указано в накладной, а 20-ть. Другой пример: согласно конструкторской документации изготовлена "Печатная плата" – 1 штука. Но по сути на ней расположены 10 одинаковых сегментов – заготовок для десяти одинаковых модулей. Эти сегменты впоследствии будут отделяться друг от друга, на них будут впаиваться компоненты и, в итоге, будут собраны 10-ть изделий. Поэтому на складе следует учесть не одну "Печатную плату", а десять;

- количество "в партии" и признак ("кодовое слово") партии. На некоторые виды товаров вместе с материальными ценностями поступают этикетки, в которых указаны гарантийные обязательства, или срок годности, или вид приемки товара. И информация, относящаяся к партии, зачастую распространяется на упакованные вместе МЦ, поступившие по нескольким накладным, то есть на складе будут заведены разные записи на каждую накладную. Для того, чтобы впоследствии можно было определить, на какие именно компоненты распространяются сведения отдельно взятой этикетки, необходимо иметь поле, которое будет связывать записи о движении МЦ с "идентификатором партии";

- упаковка. Эта категория подразумевает наличие трех полей:

1) признак упаковывания вместе с МЦ из другой партии (накладной);

2) количество дыр в ленте/паллете или признак того, что все компоненты хранятся без специальной упаковки (россыпью). Некоторые компоненты поступают упакованными в ленту (или паллету и т.д.), причем для последующей передачи на производство важно, чтобы сама лента была длиннее, чем место в ней, которое занимают компоненты. В связи с этим информация о длине ленты (о количестве пустых + непустых мест) оказывается важной при выборе партии для выдачи;

3) количество "лишних" компонент в ленте (паллете/ячейке). При поступлении мелких компонент несколько штук в ней могут оказаться "сверх положенного", то есть по накладной закуплено 10 штук, а в ленте находится 15. Это не случайно, поскольку при пайке есть вероятность выпадения (потери) мелких компонент, и на производство следует передавать количество чуть больше положенного. Процент "лишних" (запас для потери) обычно невелик, но при выдаче следует учитывать и это количество. Иногда, при покупке большой партии товара, продавец "дарит крупные подарки", и вместо 300 оптических мышек, к примеру, в партии оказывается 301 штука, то есть на склад принимается санкционированный бонус в виде 1-го бесплатного комплекта.

Документы перемещения МЦ. Каждая запись из этой категории более или менее стандартная и состоит из трех полей:

1) Фамилия получившего / или название фирмы-поставщика (для документов извне) / или название подразделения (для перемещения внутри организации);

2) вид и № документа (акт, накладная, …);

3) дата документа.

К этой категории относятся: документ оприходования (принятие к учету "по бухгалтерии" на предприятии); документ поступления (при приеме на склад "по бухгалтерии"); фактическое принятие на склад (отметка о поступлении); документ выдачи со склада (при передаче со склада "по бухгалтерии"); фактическая передача со склада (отметка о выдаче).

Отметка о наличии/отсутствии МЦ/документов на складе. Признак временного отсутствия какой-то из основных записей или самого товара, так называемый "признак незавершенности записи". Например:

- когда МЦ выдана, но еще не все документы выдачи оформлены;

- когда МЦ еще не принесли на хранение, но документы поступления уже есть;

- МЦ фактически лежит на полке, а документов поступления еще нет либо документы выдачи со склада уже сделаны.

Практически это означает, что запись, относящаяся к движению данной МЦ в указанном количестве, не оформлена до конца и требует изменения.

Место расположения/хранения МЦ. Казалось бы, должно быть однозначное толкование месторасположения, однако, несмотря на очевидность, эти данные должны располагаться минимум в двух полях:

1) месторасположение:
помещение+стеллаж+полка+коробка;
2) признак "совместного хранения".

Под "признаком совместного хранения" подразумевается, что данные МЦ УПАКОВАНЫ вместе с точно такими же МЦ, но записи о поступлении на склад у этих двух (или более) партий имеют свои отличительные черты. Например, МЦ предназначены для сборки разных изделий, или у них разные гарантийные талоны, или МЦ числятся "по бухгалтерии" на разных счетах, и т.д. То есть, согласно документам и другим опознавательным признакам записи об этих МЦ различаются, но фактически товары находятся/содержатся в одной упаковке. Наличие признака "совместное хранение", соответственно, подразумевает, что количество МЦ, указанное на упаковке (на коробке, находящейся на стеллаже), не будет совпадать с количеством товара в просматриваемой записи, но должно являться суммой количеств из каждой записи, которые ссылаются на рассматриваемое "месторасположение". Заметим, также, что поле "совместное хранение" может быть совмещено с полем "1) признак упаковывания с …" категории "упаковка", рассмотренным выше, но лучше оставить его независимым дополнительным признаком.

Признак "совместного хранения" не взводится, если "сверх означенного в записи" в упаковке содержатся только "лишние" (не учтенные "по бухгалтерии") компоненты. То есть если в ленте запакованы 5 шт. компонент из данного получения + 1 "лишняя". Другими словами, признак "совместного хранения" устанавливается только в отношении записей об учтенных МЦ. Отметка о наличии "лишних" компонент доступна в электронной записи (количество "запасных компонент" будет ненулевым), и рекомендуется наклеить на такую упаковку дополнительный снимающийся уточняющий стикер с указанием: здесь упакованы
учтенных - 10 штук.
и запасных товаров (неучтенных) – 5 штук.

Инициация закупки (документ) и назначение. Не лишней при учете на складе будет информация о том, кому потребовалось, собственно, купить ту или иную МЦ (если, конечно, такие сведения доступны). Здесь могут быть указаны название подразделения, физическое лицо, приказ, записка. Нередко бывают ситуации, когда нужно вызвать специалиста, который может проанализировать - "а то ли купили, что нужно?" либо оценить качество приобретенного товара, и работник склада должен знать, к кому обращаться. Заполнение же поля "назначение" может помочь в отборе товаров, предназначенных для выполнения определенных задач. Зачастую именно по полю "назначение" проводится принудительное разбиение записей о хранящихся МЦ: вместо одной записи о поступлении 100 штук вентиляторов вводится две записи – о 40-ка штуках для "Сборки изделия А" и 60-ти штуках для "Сборки изделия С".

Подбор/зарезервировано. Признак (вводится, как название_потребности) того, что данные МЦ предназначены для выдачи со склада с определенной целью. То есть, они специально помечены, как "зарезервированные для ПОТРЕБНОСТИ", чтобы не учитывать их "наличие" при подсчете имеющихся на складе резервов. Рекомендуется размещать такие записи в отельном перечне/сводке, чтобы они не были видны, как доступные к новому подбору для выдачи, или подсвечивать их (выделять строку, например, салатовым цветом). Помеченные этим признаком записи могут относиться как к уже сложенным в один "ящик" товарам, готовым к передаче, так и к МЦ, которые еще расположены на своих местах на стеллажах.

Серийный и инвентарный номер обязательно должны располагаться в отдельных полях, чтобы можно было эффективно вести поиск. Рекомендуется вводить эти номера однотипно, то есть не допускать того, чтобы у одних изделий они выглядели как "№ 134", а у других как "N234" или как "s/n 456".

Уточняющее описание МЦ является по сути уже "почти примечанием", но, тем не менее, поскольку для отбора необходимых товаров бывают нужны уточнения их характеристик (красный, открытый/закрытый, с винтами, комплект), то целесообразно вести такое поле отдельно от общих примечаний.

Признаки: разукомплектовано (доукомплектовано) и "вхождение в изделие". В случае, когда на учете в организации по бухгалтерии стоит ЭВМ, она, к сожалению, часто имеет одну ЦЕНУ на весь "комплект": системный блок, монитор, клавиатура, мышь, колонки. Однако, эти составные части ЭВМ иногда выходят из строя или передаются на склад для хранения, и в последующем выдаются со склада не как ЭВМ, а именно как монитор или клавиатура…, то есть – "по частям". Соответственно, на складе такие составные изделия (ЭВМ в нашем примере)

следует учитывать по частям, а для определения принадлежности к "комплекту ЭВМ" нужен признак того, что данная МЦ появилась в результате разукомплектования, и ссылка на объект, который был таким образом частично разукомплектован (частично, потому что по бухгалтерии на учете по-прежнему осталась одна ЭВМ). Бывает и обратная ситуация, когда закупленный жесткий диск, к примеру, устанавливается в системный блок - в другую МЦ. В этом случае необходим признак того, что изделие (системный блок ЭВМ) доукомплектовано, а у жесткого диска должна присутствовать ссылка на именно ту ЭВМ (системный блок), куда его вставили, и признак "вставлено" в изделие.

Примечания. Данное поле имеет назначение, аналогичное комментариям программировании. Если некоторые фразы, вносимые сюда, часто повторяются, либо по ним чаще ведется поиск, то целесообразно вынести такие данные в отдельную колонку, а в поле "примечания" оставить менее важную информацию.

Признак "архивности" записи. Такой признак проставляется у записей, относящихся к МЦ, которые выданы со склада.

Если впоследствии эта МЦ будет передана на хранение еще раз, то для неё будет заведена новая запись, а эта, "старая", будет сохранена в "истории".

Признак позволяет не замусоривать общее состояние (вид списка) склада сведениями, относящимися к уже не хранящимся на складе товарам.

Рекомендуется размещать архивные записи в отельном перечне/сводке, чтобы они не были видны, как доступные к новому подбору для выдачи и подсвечивать их (выделять строку, например, красным цветом).

Особое внимание дате заведения учетной записи на склад не придается, однако нелишним будет иметь представление о дате размещения МЦ на полке.

Эта информация содержится в поле "дата" документа "принятие на склад (отметка о поступлении)".

При разработке складской системы, основанной на учете рассмотренных выше признаков МЦ, следует расположить каждый из них в отдельной графе (колонке), организовать возможность выбора строк по любому из признаков или их совокупности, а также предусмотреть итоговое сложение "количества" после применения установленных фильтров.

4. Порядок занесения информации

Для начала следует определиться со степенью "обязательности" заносимой информации. То есть: какая информация у записи является основополагающей, какая уточняющей, и какая – переменной. Очевидно, что для хранения и учета как минимум необходимы сведения о наименовании, количестве (с единицами измерения) и месторасположении МЦ. Но каждая из этих записей делится на множество частей, которые, в свою очередь, имеют те или иные приоритеты в заполнении данными. Рассмотрим все перечисленные в разделе 3 элементы данных и отдельные записи, разделив их на три группы по принципу порядка ввода в складскую систему.

4.1. Первоначальное заполнение при поступлении

Рекомендуется при начальном вводе новой записи все поля заполнять каким-нибудь незначащим символом, например @, чтобы отличать "поле, не содержащее значения", от "пустого поля, которое еще не редактировалось".

В зависимости от того, что мы держим в руках, может быть пять вариантов действий, совершаемых при занесении первоначальных сведений в поля базы данных.

Первый и самый предпочтительный вариант: если в нашем распоряжении МЦ и документы зачисления на склад. Порядок действий в этом случае:

Вводятся данные из документа перемещения "по бухгалтерии" на склад:

- № и дата документа поступления на склад, ФИО получившего;
- наименование по бухгалтерии;
- код товара по бухгалтерии;
- единица измерения по бухгалтерии;
- количество "по приходу";
- направление деятельности по бухгалтерии.

Проставляется отметка о поступлении (ФИО получившего, дата, может быть, есть акт передачи или какая-то другая информация: "Принес Иванов И.И.", например).

Анализируются данные, часть которых может быть указана на самих изделиях или на упаковке:

- определяется и указывается "наименование по складу";
- определяется и указывается "единица измерения по складу";
- в соответствующее поле вводится код товара по складу;
- перечисляются серийные номера (при их наличии);
- при необходимости заполняется поле "уточняющее описание";
- при наличии гарантийных этикеток или иных документов, относящихся к упаковке, эта информация заносится в поле "примечание", и заполняются поля "количество в партии" и "кодовое слово (идентификатор) партии";
- при обнаружении упаковывания товара вместе с таким же наименованием из другого получения в графе УПАКОВКА проставляется "признак упаковывания вместе с МЦ из другой партии", взводится флажок "признак незавершенности записи", который будет удален после занесения в базу записей обо всех упакованных совместно изделиях, и указывается, если этого не было сделано ранее при заполнении предыдущих данных, "кодовое слово (идентификатор) партии".

Если на момент заведения записи в вашем распоряжении есть еще какая-то дополнительная информация, которая подлежит занесению в остальные поля и распространяется на всё количество товара, которое принимается к учету на складе именно этой записью (например, инициация закупки (документ) или назначение), то эти данные также вносятся. Последующие действия с МЦ и вводимой записью будут заключаться в определении месторасположения компонент на стеллажах и разбиении записи согласно предназначению.

Анализируются упаковки поступив-шего товара

- запись о поступившем количестве "по накладной" в размере 100 штук разбивается на две: 10 штук, упакованных в коробку и 90 штук, упакованных в ящик. Несмотря на то, что у принятых на хранение изделий может быть один гарантийный талон, признак "совместное хранение" не устанавливается, поскольку изделия упакованы раздельно;

ВАЖНО: все поля, которые были заполнены до "размножения" строки, останутся без изменений, за исключением, может быть, серийных номеров изделий, которые попадут в разные упаковки.

- при необходимости у каждой записи/упаковки указывается "количество дыр в ленте/ячеек в паллете" (группа полей УПАКОВКА);
- количество "запасных элементов". Если в упаковке находятся какие-то "лишние" количества компонент, то устанавливается "признак незавершенности записи" и оставляется взведенным до тех пор, пока в базу не заведутся все записи, покрывающие документами все количество упакованных совместно товаров. Если на настоящий момент уже известно, что это "запасные элементы", то в случае, когда их можно отделить от партии ("лишняя мышка") это лучше осуществить, а если отделять "лишние" нецелесообразно, то следует заполнить поле "количество запасных компонент";
- упаковки размещаются на складе (на полке) и месторасположение заносится в соответствующее поле;
- при дальнейшем разбиении записей на количества, согласно распределению, или назначению, или еще каким-то данным, распространяющимся на часть товара из упаковки, следует установить в соответствующем поле "признак совместного хранения" (упаковка на полке одна, а записей в базе может оказаться несколько).

Первоначальную регистрацию МЦ на складе можно считать завершенной. Если в ближайшее время ожидается получить дополнительные данные о данной партии товара, которые нужно обязательно учесть, то устанавливается флажок "признак незавершенности записи", а в поле, требующее уточнения, вводятся символы НЕТ, чтобы понимать причину "незавершенности записи" и иметь возможность быстро выбрать требующие изменения строки из множества строк базы данных.

Второй, менее предпочтительный вариант поступления товара. Если в нашем распоряжении МЦ и хоть какие-то документы, отличные от бухгалтерских документов зачисления на склад:

1) обязательно устанавливается "признак незавершенности записи";
2) делается отметка о поступлении (ФИО получившего, дата);
3) в систему заносится вся информация, которая имеется в нашем распоряжении на настоящий момент.

Особо отметим, что обязательны к заполнению поля:

- "наименование по складу";
- "единица измерения по складу";
- код товара по складу;
- серийные номера (при их наличии).

4) упаковки размещаются на складе (на полке) и месторасположение заносится в соответствующее поле;

5) в поля, требующие обязательного ввода информации при занесении новой учетной записи о товаре, вводится слово НЕТ;

6) после поступления документов ввод сведений производится, как описано в ***первом*** варианте.

Третий случай: в нашем распоряжении только материальная ценность. Заполняются данными поля:

1) взводится "признак незавершенности записи";

2) в систему заносится вся информацию, которая может быть извлечена из описания упаковки или"со слов сотрудника, который принес МЦ".

Обязательно заполняются:

- "наименование по складу";
- "единица измерения по складу";
- код товара по складу;
- серийные номера (при их наличии);

3) упаковки размещаются на складе (на полке) и их месторасположение заносится в соответствующее поле;

4) в полях, требующих последующего обязательного заполнения, указывается "НЕТ";

5) после поступления документов ввод сведений производится, как описано в ***первом*** варианте.

Четвертый вариант поступления. В нашем распоряжении только документы о зачислении товара на склад. Заносится следующая информация:

1) устанавливается "признак незавершенности записи";

2) заполняются соответствующие документам поступления поля записи;

3) проставляется "НЕТ" в местах, требующих обязательного заполнения;

4) запись выделяется красным цветом;

5) после поступления МЦ снимается цветовая пометка, и ввод сведений производится, как описано в ***первом*** варианте.

Ситуация пятая: имеется отдельно упакованная МЦ, про которую достоверно известно, что она "лишняя"/запасная, и принесена на склад, чтобы учесть в единой базе предприятия, что она есть в наличии.

Неучтенная компонента, которой, при утере или утрате, можно заменить необходимую для сборки изделия деталь. Повторимся – "отдельно упакованная "лишняя" МЦ", потому что упакованные совместно с учтенными компоненты указываются в соответствующем поле записи учтенного товара.

1) в систему заносится вся информация, которая может быть почерпнута из описания упаковки или "со слов сотрудника, который принес МЦ";

2) особое внимание уделяется учетной группе полей, уточняющих наименование (код по складу, наименование по складу….), чтобы данная МЦ могла быть отобрана при поиске соответствующего учтенного товара;

3) упаковки размещаются на складе (на полке) и их месторасположение заносится в соответствующее поле.

ВАЖНО: Для хранения "лишних" (неучтенных) МЦ следует отвести отдельный стеллаж (помещение, полку). Во-первых, в этом случае запасные компоненты не будут перемешиваться с основными/учтенными, а во-вторых, список всех "лишних" компонент можно будет получить выбором месторасположения (и не нужно будет заводить дополнительный признак "неучтенности").

4) если известно, что данная МЦ получена вместе с какой-либо учтенной партией товара, то соответствующие поля о документах поступления допускается заполнять теми же данными, что содержатся в основной записи поступившей партии (учтенной).

Особый случай: МЦ были выданы со склада ранее (для проведения тестирования, например), и теперь возвращаются (полностью или частично) на хранение.

В этой ситуации вместо ввода новой записи допустимо копирование строки (записи), которая на настоящий момент является архивной, в основной перечень изделий, находящихся на складе, и приведение в должный вид соответствующих "новому приходу" полей.

Если все обязательные поля новой записи заполнены, следует снять "признак незавершенности записи": данные внесены в достаточном количестве, чтобы найти (на полке) и идентифицировать МЦ (по документам).

4.2. Уточняющая информация

В процессе хранения возникают моменты, когда некоторые сведения о находящихся на складе товарах требуют уточнения или корректировки. В первую очередь это относится к строкам, у которых установлен признак "незавершенности записи". Но могут изменяться (или появляться вновь), например, следующие данные, которые подлежат занесению в соответствующие поля:

- фирма-производитель. Можно занести эту информацию в поле "Примечание", если она не часто востребована. В случае постоянного использования этих данных следует завести отдельное поле записи, чтобы учитывать эти сведения;
- назначение. Корректируется при поступлении соответствующей информации на склад. В случаях, когда поле "назначение" предполагает разделение товара на две или более частей, строка множится, чтобы в каждой получившейся записи было указано только одно "назначение";
- направление деятельности по бухгалтерии. Может измениться в процессе изменения "назначения". Следует заменить старое значение поля новым;
- подбор/зарезервировано при получении работниками склада указания о выдаче или подборе нескольких наименований МЦ с целью выдачи в будущем целого набора компонент, у запрошенных товаров проставляется признак "название потребности". Это может быть ссылка на "запрос о выдаче", или особое "новое" место расположения на стеллаже. Строка/запись с нужным количеством МЦ выделяется цветом, чтобы отметить факт резервирования, или подлежит перемещению в отдельный список. При этом сам товар может до момента выдачи оставаться на своем прежнем месте на полке. В некоторых случаях удобно сразу отделить необходимое количество каждой МЦ и сложить их в вместе, тогда подобранную группу компонент размещают на стеллаже в отдельном ящике, и соответствующую пометку вводят в каждую запись для этих МЦ в поле "месторасположение";
- дополнительная информация. Не будем вдаваться в подробности, но иногда о хранящемся товаре на склад поступает довольно непредсказуемая информация, которую нелишне будет записать, раз уж она кому-то показалась значимой;
- разукомплектование. Процесс разделения одной учетной записи на две и более необходим при запросах на выдачу "части" МЦ. Например, под наименованием "Соединитель АВС" на складе хранится разъем, в который вкручены винты, но выдать просят только сам разъем в количестве 2-х штук, а винты от него надо оставить учтенными на складе.

В этом случае, сначала нужно отделить от партии соединителей две штуки, разбив запись о наличии 20-ти соединителей на 2шт. и 18-ть. Затем запись о двух штуках надо разбить на две. В одной из них следует указать "отделяемое" наименование - "винт DAS" 4 штуки (в каждый разъем вкручено по 2 винта). Если такие винты уже учитывались на складе как самостоятельный элемент, то ввести ранее принятое к учету наименование.

В поле "разукомплектовано" для винтов следует указать ссылку на запись с 2-мя соединителями, которые остались без крепежа, а в поле "примечание" причину разукомплектования и оставшиеся после разукомплектования элементы.

В другой записи будут два "Соединителя АВС", у которых в поле "разукомплектовано" будет стоять ссылка на вторую запись – о "винтах", а в поле "примечание" будет указана причина разукомплектования и пометка "без винтов DAS".

Поскольку "доукомплектование" не является уточняющей информацией, а скорее относится к "выдаче со склада" (компонент, который будет вставлен в изделие, по сути будет "выдан в изделие", а доукомплектованное изделие может изменить наименование, быть "выдано со старым наименованием" и "принято вновь с новым наименованием"), то эту операцию рассмотрим в следующем подразделе.

- разбиение упаковок на части. Самая кропотливая и требующая аккуратности работа по учету МЦ. Относится к "уточняющим сведениям", но по сути является следствием действий по работе с товарами, подлежащими "подбору/резервированию":

1) находим товар, запись о котором подлежит редактированию, в базе данных и на полке (берем в руки упаковку);
2) следует убедиться в том, что взятая упаковка содержит именно столько МЦ, сколько получается при сложении количеств в выбранных строках базы данных;
3) от упаковки отделяется (или делается пометка на упаковке) то количество товара, которое следует выдать/зарезервировать;
4) записи разбиваются (строки копируются и редактируются "количества по складу") таким образом, чтобы разбиение МЦ на новые упаковки (или назначения) соответствовало этим записям;

5) корректируется количество "запасных элементов", которые указаны в случае совместного хранения в поле "примечание" учтенного изделия. Если к выдаче предназначен "кусок", включающий запасные компоненты, то указываем их, а в соответствующей записи, откуда эти компоненты были взяты, уменьшаем количество;

6) у каждой записи/упаковки фиксируется изменившееся "количество дыр в ленте/ячеек в паллете";

7) если в отобранных строках был установлен "признак упаковывания вместе с МЦ из другой партии (накладной)", и к выдаче предназначен весь "кусок из другой партии" целиком, и теперь он отдельно будет размещаться на полке, то данный признак нужно снять у всех строк. После разделения в каждой из упаковок остались товары, которые получены по одному приходному документу;

8) если после обработки записей было произведено разделение упаковок, то у каждой "теперь отдельно упакованной партии" следует снять "признак совместного хранения". Напомним, что этот признак устанавливается при совместном упаковывании МЦ, записи о которых имеют какие-то отличительные черты, например, назначение, гарантия, названия документов, даты, примечания и т.п.;

9) заметим, что остальные поля (за исключением, конечно, содержащих серийные номера изделий, новое распределение и новое "месторасположение") у записей, которые были затронуты при операциях подбора и резервирования, остаются без изменений.

4.3. Данные о выдаче со склада

"Выдачей со склада" будем называть фактическое исчезновение МЦ из помещения склада. Все остальные состояния товара делятся на "хранение" и "подбор". То есть, выдача считается осуществившейся, если МЦ вставлена в какую-то другую МЦ и не лежит в коробке, либо кто-то пришел и вынес товар за дверь складского помещения, либо у МЦ изменилось наименование, и товар снят с учета и "исчез" со склада по этой причине. Соответственно, в записи "исчезнувшего" товара:

1) сразу должна быть заполнена группа полей "отметка о выдаче":

- *Фамилия* получившего / или название фирмы (для представителей другой организации)/ или название изделия, в состав которого вошел этот товар;
- *вид и № документа* (акт, расписка, пометка "в_руки"…, данные о доукомплектовании какого-то другого изделия данной компонентой). Работники склада могут вести журнал выдачи, или сами готовить/печатать неофициальные документы передачи (акты, расписки), которые будут храниться на складе, подписанные получившим изделие лицом;
- *дата* передачи/дата доукомплектования.

2) при наличии официальных ("по бухгалтерии") документов передачи, или акта о доукомплектовании, или списании материальных запасов для формирования комплекта, информация должна быть занесена в группу полей "документ выдачи со склада". У доукомплектованного "выдаваемой компонентой" изделия следует отразить этот факт в графе "Примечание", а если наименование "нового" изделия изменилось, то отметить выдачу со склада "старого изделия" и оформить прием на учет "нового наименования" этого товара. Если официальных ("по бухгалтерии") сведений на момент выдачи еще нет, то нужно проставить "признак незавершенности записи" и в графы, подлежащие обязательному заполнению, проставить слово НЕТ.

3) строка с записью об изделии, которое больше не находится на полке в помещении склада должна быть окрашена (помечена красным цветом). В случае, если все обязательные поля заполнены, следует снять "признак незавершенности записи" и взвести признак "архивности записи", а затем перенести строку в "архив" (при ведении соответствующего архивного перечня).

Следует отметить особый случай, который состоит в том, что документы "по бухгалтерии" на выдачу уже оформлены, но МЦ всё еще находится на складе. При таком положении дел запись должна быть помечена, как "подбор/зарезервировано":

- установлен "признак незавершенности записи",
- в поле "название потребности" вводится "ВЫДАЧА";
- запись выделяется салатовым цветом или подлежит перемещению в отдельный "список зарезервированных для выдачи изделий".

При выдаче неучтенных (лишних) компонент заполнению подлежат только графы "отметка о выдаче", и запись можно переводить в раздел "архивное состояние" - проставлять "признак архивности записи".

Напомним также, что в момент выдачи с "запасом неучтенных компонент" следует

проверить корректность заполнения группы полей "упаковка" у записей, которые связаны признаком "совместное хранение" и отражают общее количество "лишних" МЦ в упаковке. В запись, предназначенную к выдаче, заносится фактическое количество передаваемых (отрезаемых от мотка) неучтенных компонент, то есть выдается 10 учтенных, но отрезается кусок ленты с 15-ю компонентами, 5-ть из которых лежат как "лишние". А в запись, где учтен оставшийся от партии товар, заносится "новое количество лишних".

Еще один особый случай: пришел Петров и хочет "только посмотреть" на МЦ, "только посмотреть и проверить…, и сразу вернет". Производятся следующие действия:

1) Петрову выдается МЦ;
2) работником склада оформляются документы о выдаче (это может быть расписка или акт передачи, или подпись получившего на ксерокопии документа поступления на склад..);
3) эта информация заносится в группу полей "отметка о выдаче" базы данных;
4) устанавливается признак незавершенности записи;
5) запись выделяется красным цветом;
6) в полях "официальные ("по бухгалтерии") документы на выдачу" проставляется слово НЕТ;
7) после того, как Петров (или другой сотрудник) вернет МЦ, рекомендуется:
    7.1) в группу полей "официальные ("по бухгалтерии") документы на выдачу" внести данные типа: "возвращено на склад Ивановым такого-то числа";
    7.2) данную запись перевести в архивное состояние;
    7.3) продублировать её копированием в "актуальный список МЦ, хранящихся на складе";
    7.4) снять признак архивности у "новой" записи, снять выделение цветом, убрать упоминание о выдаче, очистить группу полей "официальные документы выдачи";
    7.5) занести в группу полей "фактическое принятие на склад (отметка о поступлении)" новую информацию о приеме на склад.

Таким образом, во-первых, в последующем можно будет увидеть "историю" перемещения данной МЦ, и, во-вторых, в каждый момент времени в базе данных будет находиться актуальная информация о наличии товара.

Раз уж мы упомянули о "расписке" при выдаче со склада, следует отметить, что все документы, которыми оперируют складские работники, можно разделить на две крупные категории (имеются в виду документы, которые относятся к перемещению материальных ценностей): ПРИЕМ на склад и ВЫДАЧА со склада. Внутри каждой из категорий два раздела: официальные документы и внутренние передаточные (неофициальные). Чтобы быстро найти "официальные" документы, удобно раскладывать их в папки "по номеру документа" (этот номер заведен в соответствующую группу полей каждой записи). Чтобы организовать хранение неофициальных отметок о приеме/выдаче, рекомендуется присваивать этим "заметкам" складские порядковые номера, которые и заносить в запись о товаре. Например, расписка о выдаче с номером out_037_17 будет находиться в папке "отметки о выдаче со склада в 2017 году" под номером 37. А акт приема-передачи (на склад) с номером in_302_16 будет подколот в папку "отметки о поступлении на склад 2016 года" под номером 302.

Еще раз отметим, что заполнение полей, содержащих информацию о передаче со склада, влечет за собой последующий перенос записи "в архив". Но перенос в эту категорию зависит, от состояния полноты сведений как о выдаче, так и о предшествующем поступлении товара. При отсутствии данных в любых "обязательных полях", в том числе незавершенного состояния ввода данных при принятии на склад, следует оставить у записи "признак незавершенности записи". Окончательное перемещение записи "в архивное состояние" будет осуществлено только после замены данных обязательных полей необходимыми сведениями. Архив записей рекомендуется размещать в отдельном списке/перечне, чтобы не засорять "актуальное состояние наличия товаров" на складе.

5. Заключение

Описанная система учета МЦ была разработана для предприятия, на котором количество учитываемых наименований не превышает 600 000 единиц. Она зарекомендовала себя, как очень эффективная, удобная и легкая в использовании программа, позволяющая оперативно получить любую исчерпывающую информацию о МЦ, находящихся в организации.

Основной особенностью этой базы данных является одноуровневая организация информации, облегчающая восприятие.

Так называемая ЗАПИСЬ, в которой содержатся сведения о МЦ, является одной строкой (все поля записи расположены в одной строке), и пользователю доступна расстановка в нужной последовательности колонок таблицы, состоящей из этих записей-строк. Кроме того, наложение фильтров на строки позволяет быстро отображать необходимые для анализа сведения.

Несмотря на множество полей, которые требуют заполнения, ввод данных занимает совсем немного времени, но содержащиеся в базе сведения оптимально быстро могут быть проанализированы. Доступ к информации о хранящихся на складе МЦ возможен в любой момент в полном объеме, и не зависит от памяти работника склада.

Optimization of warehouse management system

G.L. Levchenkova

Abstract. Here is a method of organization of data and entry additional information in the warehouse registry list of material valuables. The goal is to get a conspectus of the general current of affairs.

Keywords: stocking system, automated storage and retrieval warehouse

Литература

1. Стюарт Эмметт. Искусство управления складом. Минск, "Гревцов Паблишер", 2007.

2. 1С:Предприятие. Версия 7.7. Конфигурация "Торговля+Склад" 9.2. Описание. М., Фирма «1С», 2002.

3. 1С:Бухгалтерия бюджетного учреждения 8. Руководство по ведению учета (описание типовой конфигурации). М., Фирма «1С», 2009.

4. Г.Л. Левченкова. Система "ЗАКУПКА И РАСПРЕДЕЛЕНИЕ". Обеспечение потребностей предприятия в комплектующих, материалах и готовых изделиях. "Организация документооборота и учета. Организация закупки и распределения", Москва, НИИСИ РАН, 2006, 76-234.

5. В.Н. Давыдов, Г.Л. Левченкова. Особенности проведения ревизии склада при смене информационной системы учета. "Технологии программирования. Учетные системы. Ортонормированные системы. Сборник статей под редакцией академика РАН В.Б.Бетелина", Москва, НИИСИ РАН, 2008, 93-114.

6. Г.Л.Левченкова, А.Г. Прилипко. Перенос информации между базами данных, существующими в средах с разными операционными системами. "Труды НИИСИ РАН", т. 1 (2011), №1, 77-85.

Метод итерационного проектирования встроенных средств тестирования с компрессией

М.С. Ладнушкин

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: maximsl@gmail.com

Аннотация: Проведен анализ встроенных систем сжатия тестовых данных скан-схем с учетом падения контролируемости и наблюдаемости внутренних узлов с ростом числа внутренних скан-цепей. Предложен итерационный метод проектирования скан-схем с компрессией, позволяющий оптимизировать параметры скан-схемы для достижения максимального коэффициента компрессии при минимальных аппаратурных затратах.

Ключевые слова — средства тестирования, отбраковка, скан-технология, компрессия тестовых сигналов, генерация тестовых векторов, моделирование.

1. Введение

Тестирование с использованием встроенных скан-цепей позволяет обнаруживать неисправности типа «залипание» и переходные неисправности в цифровых СБИС с высоким коэффициентом покрытия (95% и выше). С ростом числа элементов на кристалле растёт число потенциальных неисправностей, а значит и объём тестовых данных, что увеличивает требования к объёму памяти тестирующего оборудования (тестера). Также, как следствие, растёт время тестирования каждой микросхемы.

Технологии сжатия тестовых данных позволяют в десятки раз сократить время отбраковки микросхем с помощью встроенных на кристалл СБИС блоков компрессии и декомпрессии [1] (см. рис. 1). Согласно данным технологиям, тестер в сжатом виде передаёт данные на вход микросхемы по ограниченному числу тестовых каналов (I), после чего встроенный в СБИС Декомпрессор распространяет сигналы на внутренние N скан-цепи (N>I). На выходе скан-цепей данные сжимаются в Компрессоре до ограниченного числа выходных портов микросхемы Q (Q<N) и в сжатом виде передаются обратно в тестирующее устройство. За счёт сокращения длин скан-цепей сокращается время загрузки/выгрузки данных (фазы сдвига скан-тестирования).

Методы сжатия данных используют основную особенность тестовых последовательностей для скан-тестирования: эти последовательности содержат лишь f = 0,1…5% значимых данных, остальные данные заполняются случайным образом, и на выходе не анализируются. Это позволяет их эффективно сжимать, теоретически в пределе в f^{-1} раз [2].

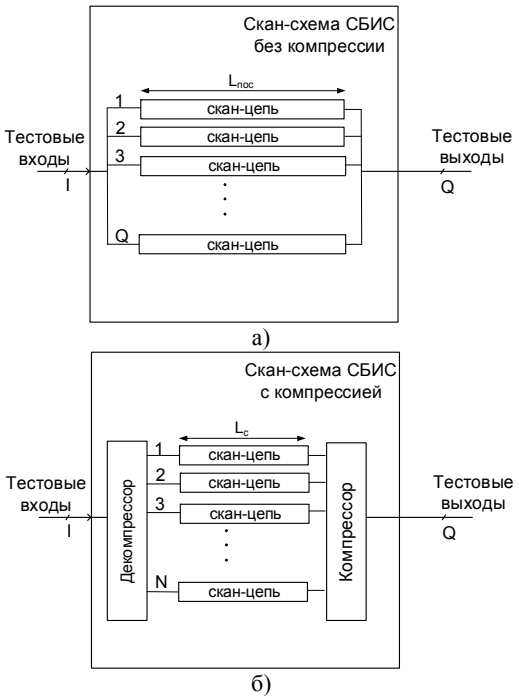


Рис. 1. Скан-схемы СБИС без компрессии (а) и с компрессией (б)

Однако меньшее число тактов фазы сдвига не всегда ведет к сокращению времени тестирования из-за возможного увеличения (инфляции) тестовых векторов с ростом числа скан-цепей, которое вызвано двумя факторами [3]:

1) Недостаточная контролируемость сигналов через блок Декомпрессора в виду неспособности Декомпрессора декодировать необходимое число значащих бит при загрузке тестовых данных в скан-цепи;

2) Недостаточная наблюдаемость выходных сигналов скан-цепей через Компрессор, вызванная маскированием тестовых откликов неизвестными X-значениями.

Данные факторы связаны с особенностями каждой отдельной СБИС, которые оцениваются такими характеристиками как коэффициент заполнения f значащими битами загружаемых тестовых векторов и плотность X-значений (D_x), возникающих после фазы захвата данных. Данные характеристики скан-схемы могут быть получены путём генерации тестовых векторов на всём объёме неисправностей схемы, но эти оценки дадут информацию только о верхнем теоретическом пределе коэффициента компрессии для данной СБИС.

Некоторыми авторами проделаны исследования оценки стоимости каждой отдельной конфигурации скан-схемы с компрессией на ранних стадиях разработки, с целью определения оптимального коэффициента компрессии для выбранной СБИС, из которых однако следует, что теоретические оценки на ранних стадиях являются крайне неточными [4], [5]. Техника предсказания оптимальных характеристик скан-схемы на основе конкретного проекта путём виртуального создания различных блоков компрессии и ускоренной генерации тестов для полученной скан-схемы дают неплохие результаты по нахождению конфигурации с максимальным коэффициентом компрессии [6]. Но такой подход может оказаться довольно длительным (большое число итераций), а также в некоторых случаях привести к выбору неоптимальной конфигурации из-за большого числа параметров скан-схемы и отсутствия алгоритма выбора значений (в основе техник лежит метод поиска алгоритмом Гаусса, согласно которому оптимизация осуществляется по каждому отдельному параметру перебором на всём интервале значений). В частности, результаты экспериментов показали, что при увеличении числа тестовых каналов (входов/выходов) скан-схемы максимум коэффициента компрессии достигается при большем количестве скан-цепей [7].

В данной работе на основании диапазонов изменения значащих бит и роста плотности X-значений с увеличением числа скан-цепей предложен итерационный метод проектирования скан-схемы с компрессией, позволяющий добиться максимального коэффициента компрессии при минимальных аппаратурных затратах. В качестве критерия оценки эффективности компрессии предложена функция стоимости. Экспериментальные результаты подтверждают эффективность метода.

2. Анализ элементов скан-схемы СБИС с компрессией

Коэффициент компрессии – основная характеристика скан-схемы с сжатием данных, показывающая во сколько раз сократилось время тестирования и объём тестовых данных при неизменном тестовом покрытии неисправностей благодаря применению встроенных блоков сжатия:

$$C = \frac{t_{пос}}{t_c}, \quad (1)$$

где $t_{пос}$ – время тестирования без компрессии, t_c – время тестирования с компрессией. Без компрессии в I скан-цепей загружаются данные напрямую с тестирующего устройства, а данные снимаются с Q выходов, I=Q (см. рис. 1а). В общем случае время тестирования равно

$$t = P \cdot L \cdot F, \quad (2)$$

где P – число тестовых векторов в данном режиме, L – длина скан-цепей в данном режиме, F – частота тестового синхросигнала. Если T – число скан-триггеров микросхемы, то длина скан-цепей в режиме компрессии и без компрессии равна:

$$L_c = T / N, \quad (3)$$

$$L_{пос} = T / I, \quad (4)$$

где N – число внутренних скан-цепей, I – число каналов тестера. Для простоты будем считать I=Q. Отсюда коэффициент компрессии равен:

$$C = \frac{P_{пос}}{P_c} \cdot \frac{N}{I}, \quad (5)$$

где $P_{пос}$ – число векторов без компрессии, P_c – число векторов с компрессией, Q –число выходов.

Из формулы видно, что коэффициент компрессии прямопропорционален числу скан-цепей и обратно пропорционален числу входов. Увеличение числа скан-цепей ведет к росту числа тестовых векторов с компрессией из-за недостаточной контролируемости декодированных сигналов, а также наблюдаемости кодируемых на выходе сигналов. Первый множитель – это величина обратная инфляции тестовых векторов $Infl = \frac{P_c}{P_{пос}}$. Она

характеризует проявление этих факторов и показывает во сколько раз число векторов с компрессией больше числа векторов без компрессии. Рассчитать заранее значение инфляции для скан-схемы с достаточной точностью невозможно, но можно определить границы параметров, при которых она будет минимальна.

Контролируемость определяется способностью декодировать необходимое число значащих бит при загрузке данных в скан-цепи. Данные из тестера поступают на вход скан-схемы в сжатом виде, каждый такт передаётся I сигналов. Декомпрессор осуществляет декодирование I сигналов в N выходных сигналов (N>I). Декодирование может осуществляться простым разветвлением каждого входного сигнала на несколько скан-цепей (Иллинойс скан), либо с использованием комбинационной или последовательной схем [8]. С точки зрения декодирования эти схемы отличаются различной способностью декодировать n значащих бит на выходе. Так, например, для I = 8 и N = 80 вероятность декодировать n = 6 значащих бит для комбинационных схем ниже 10%, а для последовательных схем на основе 64-разрядного сдвигового регистра с обратной связью – более 90%. Тем не менее ни одна из этих схем не сможет декодировать n = 12 значащих бит [9].

Многочисленные производственные тесты показывают, что коэффициент заполнения тестовых векторов значимыми битами f убывает в ходе генерации тестовых векторов с 5% до 0,1%, составляя в среднем по всем векторам 0,2% [10]. То есть для любого набора тестовых векторов P_c

$$f = A_{sign}(P_c) / A_{dec}(P_c), \quad (6)$$

$$f \subset [f_{\min}; f_{\max}], \quad (7)$$

где A_{dec} - объём декодированных данных, A_{sign} - объём значащих бит в этом объёме, f_{max} – максимальный коэффициент заполнения в первых векторах скан-тестирования, f_{min} – минимальный коэффициент заполнения в последних тестовых векторах, нацеленных на единичные неисправности.

Если максимальная длина внутренних скан-цепей равна L_c, то объём декодированных данных одного тестового вектора и объём значащих бит в этом векторе будет равняться

$$V_{dec} = N \cdot L_c, \quad (8)$$

$$V_{sign} = n \cdot L_c, \quad (9)$$

где N – число скан цепей, n - среднее число значащих бит в одном такте сдвига.

Для числа векторов P_c, необходимых для тестирования всех неисправностей объём всех декодированных данных A_{dec} и объём значащих бит в этом объёме A_{sign} будет выражается формулами:

$$A_{dec} = V_{dec} \cdot P_c, \quad (10)$$

$$A_{sign} = V_{sign} \cdot P_c. \quad (11)$$

Отсюда по формуле (6) среднее значение коэффициента заполнения на протяжении всего теста значащими битами составит:

$$f = n / N, \quad (12)$$

где n – среднее число значащих бит в одном такте сдвига, N – число скан-цепей. Формула показывает, что увеличение числа скан-цепей N приведёт к увеличению числа значащих бит n, что потребует увеличения числа входных переменных (входов I).

Наблюдаемость выходных сигналов скан-цепей определяется сжатием N сигналов в Q. Компрессия осуществляется, как правило, комбинационной схемой методом сложения по модулю 2 выходных сигналов (реже – с использованием комбинационной схемы на основе MISR (multiple input signature register)) скан-цепей, однако в виду присутствия неизвестных X-значений в выгружаемых данных, сигналы предварительно маскируются [8], [11]. X-значения в скан-цепях возникают в связи с наличием в СБИС логических блоков, модели к которым отсутствуют или которые не могут быть должным образом промоделированы во время скан-тестирования, например, аналоговые и заказные блоки (приёмопередатчики, регистровые файлы и т.д.). Потеря наблюдаемости происходит именно по причине маскирования, и растёт с ростом плотности X-значений на выходах скан-цепей.

Если предположить, что распределение X-значений внутри каждой скан-цепи линейное и вероятность захвата скан-триггером X-значения равна p, то вероятность возникновения x числа неизвестных значений на выходах скан-цепей в любой такт сдвига будет равна:

$$D_x = C_N^x \cdot p^x (1-p)^{N-x}, \quad (13)$$

где D_x – плотность “X” на входе, C_N^k – сочетание из N цепей по k, N – число скан-цепей, p – вероятность захвата «X» скан-триггером, x – число X-значений на выходах скан-цепей в одиночном сдвиге. Если, например, положить, что p=0,03, то распределение вероятности величины D_x в зависимости от числа скан-цепей N при различных параметрах x показана на рис. 2. Видно, что с ростом числа скан-цепей стремительно растёт число «X» на выходах

скан-цепей, и в данном примере при N>600 вероятность возникновения 8 и более «X» стремится к 100%.

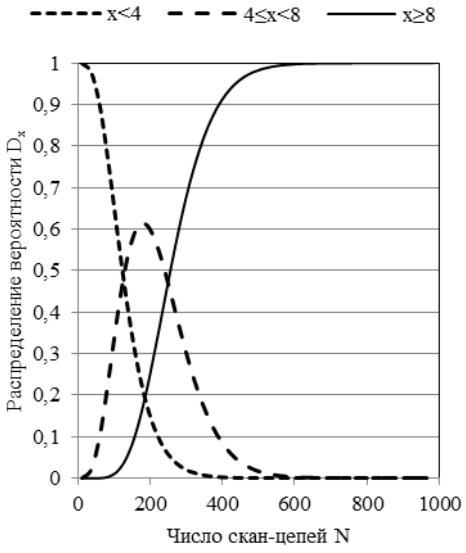


Рис. 2. Распределение вероятности D_x в зависимости от числа скан-цепей N

Декомпрессор из N в Q сигналов может быть представлен в виде матрицы с N строками и Q столбцами, в которой каждый элемент i-ой строки и j-ого столбца a_{ij} равен 1 тогда и только тогда, когда j-ый выход компрессора зависит от i-ого входа. В противном случае элемент a_{ij} = 0. Если считать, что матрица заполнена случайным образом, причём вероятность $P(a_{ij}=1) = p_1$ ($P(a_{ij}=0) = 1-p_1$), то можно сказать, что никакой входной сигнал из N не будет маскирован x числом X-значений, то есть будет сохранена 100% наблюдаемость скан-цепей и вероятность этого события P равна [12]:

$$P=\cdot(1-p_1(1-p_1)^x)^Q. \quad (14)$$

Минимум функции $P(p_1)$ достигается при $p_1 = 1/(x+1)$, а вес каждой строки будет равен:

$$W_i = \frac{Q}{x+1}, \quad (15)$$

где Q – число выходов, x – число X-значений. Отсюда учитывая (14), получаем вес каждой строки матрицы

$$W_i = \frac{Q}{N \cdot D_x + 1}, \quad (16)$$

где D_x – плотность X-значений на входе в компрессор.

Формула (16) показывает, что с ростом N, а также с ростом числа “X” на входе компрессора вес строк матрицы будет сокращаться, в противном случае X-значения будут маскировать выгружаемые данные сокращая наблюдаемость остальных выходных сигналов.

3. Итерационный метод проектирования

В качестве критерия оценки стоимости компрессии рассмотрим коэффициент эффективности компрессии с точки зрения аппаратурных затрат на дополнительную логику сжатия:

$$E_c = \frac{C^2}{S_c}, \quad (17)$$

где C – коэффициент компрессии, S_c – площадь компрессора и декомпрессора.

В общем случае площадь S_c представляет собой сумму площади занимаемой комбинационной частью схемы и последовательной частью:

$$S_c = S_{seq} + S_{comb}, \quad (18)$$

причём

$$S_{comb} \approx S_{comb0} \cdot N, \quad (19)$$

где S_{comb} – площадь комбинационная части, S_{seq} – площадь последовательной части, S_{comb0} – площадь, приходящаяся на одну скан-цепь, N – число скан-цепей.

Если положить, что I = Q, то используя (5) и (17) получаем:

$$E_c \approx \frac{N}{Infl^2 \cdot I^2 \cdot S_{comb0}}, \quad (20)$$

где E_c – эффективность компрессии, Infl – функция инфляции, I – число входов данных скан-схемы, N – число каналов.

Таким образом, для создания скан-схемы с максимальным коэффициентом компрессии и минимальными аппаратурными затратами на дополнительную логику сжатия на кристалле, необходимо найти такое число скан-цепей N_e, и число входов I_{min} (Q_{min} выходов), при которых достигается максимум функции эффективности компрессии E_c.

Так как инфляция тестовых векторов Infl должна быть рассчитана экспериментально, для определения максимума E_c был предложен итерационный метод, предусматривающий последовательное создание скан-схем с различными параметрами, генерацию тестовых векторов и расчёт значения функции стоимости в каждой итерации. (см. рис. 3).

Рис. 3. Предложенный метод проектирования

Сначала определяются границы числа скан-цепей N исходя из выбранной архитектуры декомпрессора по формуле (12). Далее на основании диапазона N выбирается наименьшее число входов I и выходов Q скан-схемы. После чего выполняется разработка скан-схемы с выбранными значениями I и Q и начальным значением N_0, которое соответствует минимуму выбранного диапазона. Затем осуществляется генерация тестовых векторов для полученной скан-схемы и фиксируется тестовое покрытие T_{cov}, которое было достигнуто на данном наборе тестов. Далее вычисляется E_c по формуле (17). После чего вычисляется следующее значение числа скан цепей по формуле

$$N_{i+1} = N_i + \Delta N, \qquad (21)$$

где N_i – число скан-цепей в предыдущей итерации i, ΔN – шаг итерации и выполняется разработка новой скан-схемы с числом скан-цепей N_{i+1}.

Процесс останавливается в том случае, если E_c с каждой новой итерацией начинает убывать – это означает, что максимальное значение было пройдено. Также, стоит заметить, что сравнивать E_c при разных параметрах имеет смысл лишь в том случае, если тестовое покрытие неисправностей T_{cov} для сравниваемых архитектур одинаково. То есть, если $T_{cov}(N_{i+1}) < T_{cov}(N_i)$, то процесс дальнейшей оптимизации прекращается.

В результате работы метода будут получены параметры скан-схемы, при которых будет достигаться максимальный коэффициент компрессии при минимальных аппаратурных затратах для данной СБИС и данных методов компрессии/декомпрессии.

4. Скан-схема с комбинационной логикой сжатия

Рассмотрим в качестве примера архитектуру скан-схемы с X-компрессией и X-маскированием [13]. Эта комбинационная архитектура состоит из широковещательного декомпрессора и компрессора на основе деревьев ИСКЛ-ИЛИ (XOR) с дополнительным блоком маскирования X-значений (см. рис. 4).

Рис. 4. Структура скан-схемы СБИС с X-компрессией и X-маскированием

Декомпрессор представляет собой сеть мультиплексоров (MUX), которая декодирует входные I сигналов в N сигналов скан-цепей и сигналы управления блоком маскирования. Число мультиплексоров подбирается таким образом, чтобы любые 3 выходных сигнала могли быть однозначно заданы с помощью входных I сигналов в каждый такт сдвига [14]. Тогда согласно (7) и (12) имеем $f = 3/N$ и $f \subset [0{,}001; 0{,}05]$,откуда $N \subset [60; 3000]$.

Выходной блок сжатия представляет собой блок маскирования элементами И части входных сигналов и блок компрессора на основе деревьев ИСКЛ-ИЛИ. Согласно данной архитектре верно неравенство

$$N \le Q \cdot 2^{Q-1}, \qquad (22)$$

где N – число скан-цепей, Q – число выходов [13]. Отсюда получаем, что минимальное значение Q для N из диапазона выше равно 10.

Таким образом мы получили границы, в которых будет достигаться максимальный коэффициент эффективности E_c.

5. Результаты моделирования

Предложенный итерационный метод был использован для поиска оптимальной конфигурации скан-схемы для четырёх различных СБИС. Характеристики СБИС после логического синтеза представлены в

табл. 1. Это СБИС разного типа с разным на порядок числом триггеров и заказных блоков в своем составе.

Таблица 1. Характеристики СБИС

Характеристика	№1	№2	№3	№4
Технология	250 нм	65 нм		
Число триггеров, тыс.	76,1	345,1	514,4	588,2
Напряжение питания, В	3,3	1,0		
Число портов ввода/вывода	140	311	639	513
Число блоков ОЗУ	18	358	536	478
Число заказных блоков	6	13	68	136
Площадь логики, мм^2	47,5	60,4	59,0	47,8

Создание скан-схем и генерация тестовых векторов осуществлялись с помощью САПР Synopsys. Для каждой из четырех СБИС были созданы 14 скан-схем с различным числом внутренних скан-цепей N из диапазона, определённого выше с шагом $\Delta N = 200$. Для каждой схемы была выполнена генерация тестовых векторов и рассчитан коэффициент эффективности E_c по формуле (17). Для унификации значений E_c для СБИС с разными проектными нормами в качестве площади использовалось число логических вентилей.

Для сравнения для каждой СБИС были созданы скан-схемы с числом скан-цепей равным 5120 - максимальным для данной архитектуры при количестве входов/выходов равным 10/10.

На рис. 5 изображена диаграмма значений эффективности компрессии в зависимости от числа скан-цепей N. Из диаграммы видно, что максимальное значение E_c достигается при $N \leq 2000$ для всех четырёх проектов.

Ниже представлена диаграмма изменения коэффициента компрессии C также в зависимости от числа скан-цепей (см. рис. 6).

На основе найденных максимумов функции $E_c(N_e)$, были получены значения N_e. Характеристики итоговых скан-схем - время тестирования t (частота синхросигнала – 10 МГц), коэффициент компрессии C, тестовое покрытие T_{cov} и аппаратурные затраты S_c для каждой из СБИС - отражены в табл. 2. Для сравнения в таблицу также помещены результаты генерации векторов для скан-схем с максимальным числом скан-цепей $N_{max} = 5120$.

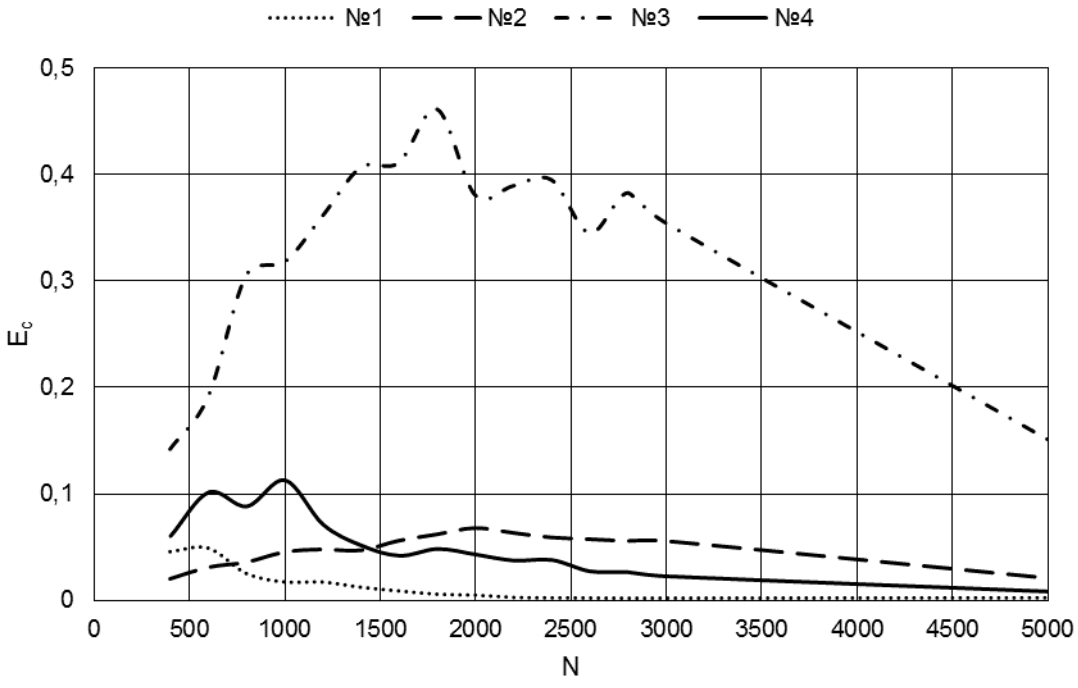


Рис. 5. Эффективность компрессии E_c в зависимости от числа скан-цепей N для четырёх СБИС

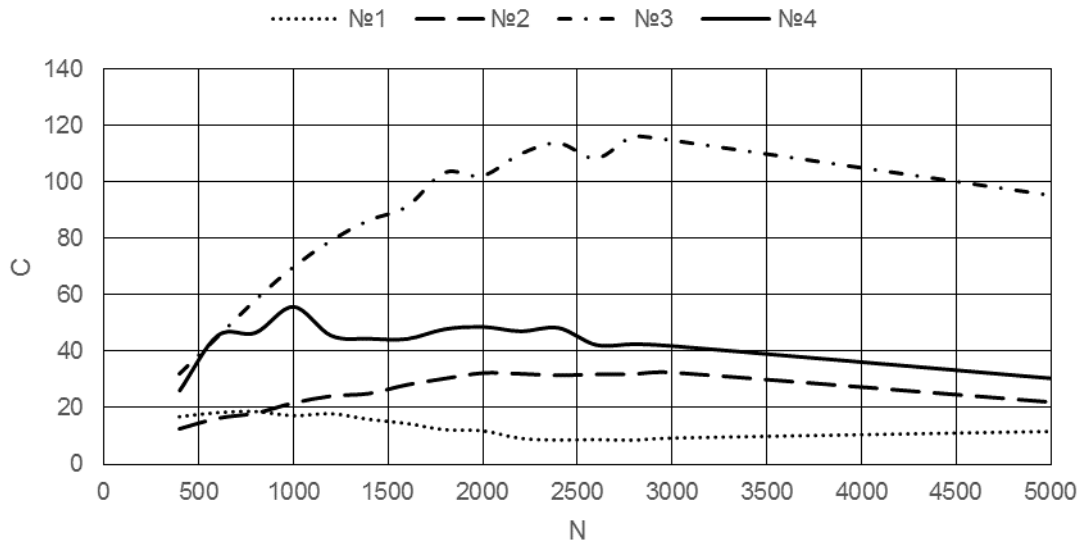


Рис. 6. Коэффициент компрессии C в зависимости от числа скан-цепей N для четырёх СБИС

Таблица 2. Характеристики полученных скан-схем СБИС

СБИС	I/Q	N_e	$E_c(N_e)$	$t(N_e)$, с	$C(N_e)$	S, 10^3 вент.	T_{cov}, %	N_{max} = 5120				
								$E_c(N_{max})$	$t(N_{max})$, с	$C(N_{max})$	S, 10^3 вент.	T_{cov}, %
№1	10/10	600	0,05	0,06	18,4	6,7	92,0	0,002	0,14	11,8	66,4	91,98
№2	10/10	2000	0,07	0,88	32,4	15,3	89,9	0,02	1,32	21,4	24,0	89,6
№3	10/10	1800	0,46	2,22	103,3	23,1	92,7	0,14	2,43	94,1	63,6	92,6
№4	10/10	1000	0,11	1,22	55,8	20,5	94,5	0,008	2,29	29,7	110,8	93,9

Если рассмотреть участок N $\subset$ [2000; 3000] на рис. 6, то можно заметить, что коэффициент компрессии C для СБИС №2 на нём практически не меняется, причём C(2000) = 32,4, а C(3000) = 32,6. Использование существующих методов поиска привело бы к выбору архитектуры с числом скан-цепей 3000 для СБИС №2, что является неоптимальным, если учесть, что разница в коэффициенте компрессии составляет 0,6%, а аппаратурные затраты при этом больше на 23%.

Обращаясь к табл. 2, важно отметить, что при числе скан-цепей N_{max} возникла потеря тестового покрытия из-за недостаточной контролируемости и наблюдаемости внутренних узлов через блоки сжатия относительно полученных скан-схем с числом скан-цепей N_e. Для СБИС №1 предложенный метод позволил в 1,5 раза увеличить коэффициент компрессии и при этом в 10 раз сократить аппаратурные затраты. Коэффициент компрессии СБИС №3 увеличился на 9% и составил величину 103, а аппаратурные затраты сократились в 2,7 раза.

6. Заключение

Проведён теоретический анализ элементов скан-схем с компрессией с точки зрения наблюдаемости и контролируемости внутренних узлов СБИС при изменении числа внешних тестовых каналов и числа внутренних скан-цепей.

Предложен метод итерационного подбора параметров скан-схемы для достижения максимального коэффициента компрессии при минимальных аппаратурных затратах согласно предложенному критерию – функции эффективности компрессии. Экспериментальные результаты показывают, что данный метод оптимизации позволяет увеличить коэффициент компрессии до 1,5 раз, при этом снизить аппаратурные затраты до 10 раз. Данный метод может быть использован при разработке скан-схем с любым типом компрессии.

Iterative developing scan compression designs

M.S. Ladnushkin

Abstract: This paper investigates internal compression and decompression schemes for controllability and observability limits of different configurations. A new iterative method was introduced which could help the designer achieving highest compression ratio for current VLSI and compression architecture with minimal additional area. The results presented show the effectiveness of the proposed method.

Keywords: design-for-test, rejecting, scan compression, test pattern generation, modeling.

Литература

1. R. Kapur. Historical perspective on scan compression «IEEE Design & Test of Computers». Vol. 25 (2008), №2, 114-120.
2. A. Kumar, M. Kassab, E. Moghaddam, N. Mukherjee et al. Isometric Test Data Compression. “IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems”, V. 34 (2015), Issue: 11, 1847 – 1859.
3. P. Wohl, J.A. Waicukauski, F. Neuveux. Increasing Scan Compression by Using X-chains “IEEE Test Conference”, 2008, 1-10.
4. O. Sinanoglu, S. Almukhaizim. Predictive analysis for projecting test compression levels “IEEE International Test Conference (ITC)”, 2010, 1-10
5. S. M. Saeed, O. Sinanoglu, S. Almukhaizim. Predictive techniques for projecting test data volume compression. “IEEE Transactions on Very Large Scale Integration (VLSI) Systems”, vol. 21 (2013), no. 9, 1762-1766.
6. J. Saikia, P. Notiyath, Santosh, R. Kapur et al. Predicting Scan Compression IP Configurations for Better QoR “20th Asian Test Symposium (ATS)”, 2011, 261-266.
7. Ладнушкин М.С. Снижение аппаратурных затрат и увеличение коэффициента компрессии средств тестирования константных неисправностей КМОП цифровых СБИС «VII Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем – 2016». Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского», М., ИППМ РАН, 2016, Ч.2, 68-75.
8. N. A. Touba. Survey of test vector compression techniques “IEEE Design & Test of Computers”, July-Vol. 23 (2006), №4, 294–303.
9. A.Dutta, A.Touba. Using Limited Dependence Sequential Expansion for Decompressing Test Vectors “International Test Conference”, 2006, 1-9.
10. J.Rajski, J.Tyszer, M.Kasab et al. Embedded Deterministic Test “IEEE Transactions on Computer-Aided Design of Integrated Curcuits and Systems”, V.23. N.5. (2004), 776-792
11. S.Mitra, S.Lumetta. X-tolerant Test Response Compaction “IEEE Design & Test of Computers”, V.22 (2005), 566-574.
12. S. Mitra, S.S. Lumetta, M. Mitzenmacher. X-tolerant signature analysis “Proc. ITC”, 2004, 432-441.
13. P.Wohl, J.A.Waicukauski, S.Ramnath. Fully X-Tolerant Combinational Scan Compression “International Test Conference”, 2007, 1-10.
14. P.Wohl, J.A.Waicukauski, R.Kapur, S.Ramnath, E.Gizdarski, T.W.Williams, P.Jaini Minimizing the Impact of Scan Compression “VLSI Test Symposium”, 2007, 67-74.

Об эффективности использования канала связи между территориально удаленными суперкомпьютерными центрами

А.В.Баранов, Д.В.Вершинин, Д.Ю.Дербышев, Б.В.Долгов, С.А.Лещев, А.П.Овсянников, Б.М.Шабанов

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail: antbar@mail.ru

Аннотация: Статья рассматривает вопросы организации передачи данных между территориально удаленными суперкомпьютерными центрами для объединения вычислительных мощностей центров по высокоскоростному каналу связи. В работе приведены результаты экспериментов по исследованию защищённого канала связи 10 Гбит/с, организованном между ЦКП МСЦ РАН (г.Москва) и ЦКП ССКЦ СО РАН (г.Новосибирск).

Ключевые слова: сеть передачи данных, высокоскоростной канал связи, сетевой протокол, суперкомпьютерный центр, передача данных.

1. Введение

Одним из ключевых методов повышения доступности и эффективности использования ресурсов суперкомпьютерных центров коллективного пользования (СКЦ) является их объединение в единую распределённую сеть. Подобное объединение даёт возможность оперативного перераспределения нагрузки между ресурсами путём перенаправления пользовательских заданий из очереди одного СКЦ в очередь другого, менее загруженного в определённый момент времени. Научно-исследовательские работы в этом направлении в Межведомственном суперкомпьютерном центре РАН ведутся не первый год. Так, построение объединённой сети суперкомпьютерного центра на базе высокоскоростного канала длиной несколько километров рассматривалось в [1], а вопросы объединения высокопроизводительных вычислительных систем в распределённую грид-инфраструктуру представлены в [2,3].

В настоящее время появилась технологическая возможность использовать для связи удалённых СКЦ высокопроизводительные защищённые каналы связи, что позволяет гарантировано предсказывать время передачи данных. Настоящая работа посвящена возможностям использования такого высокоскоростного канала связи. Исследования проводились на канале связи 10 Гбит/с, которым были объединены Межведомственный суперкомпьютерный центр РАН (МСЦ РАН, г. Москва) и Сибирский суперкомпьютерный центр на базе Института вычислительной математики и математической геофизики Сибирского отделения РАН (ССКЦ СО РАН, г. Новосибирск). Расстояние между городами по прямой свыше 2800 км, что обусловило значительные задержки в линии связи (свыше 28 мс в одну сторону).

2. Выбор протокола

Прикладные протоколы и решения передачи данных могут основываться на протоколах TCP (FTP, SCP, HTTP, RCP, SFTP, RSYNC, PARSYNC и др.) или UDP (Tsunami UDP Protocol [4], UDT [5], Aspera [6], Signiant [7] и др.).

Передача информации на большие расстояния сталкивается с проблемой большого времени между отправкой сообщения по каналу и получением подтверждения о его прибытии – RTT (от англ. Round Trip Time). Связанная с этим временем величина BDP (от англ. Brandwith-delay Product) – произведение пропускной способности канала на RTT – равна максимальному значению объёма данных, который может физически содержаться в канале связи в каждый момент времени (иными словами, быть отправленным, но еще не получившим подтверждение).

При решении реальной задачи передачи файлов по сети, объёмы передаваемых в разных направлениях данных не равны между собой. Это, а также возможная асинхронность канала, дополнительно уменьшают скорость передачи и делают невозможной насыщение канала данными до величины BDP.

При передаче данных по протоколу TCP особенно важно значение RTT. Так как

в этом протоколе производится верификация получения отправляемых пакетов, то с увеличением RTT время на пересылку одного пакета информации и связанные с этим накладные расходы увеличиваются. Использование протоколов на базе UDP является существенно более сложной задачей из-за того, что в этом протоколе не происходит подтверждение получения пакетов и проверки порядка их получения. Протоколы передачи файлов на основе UDP имеют меньшее распространение, их установка и настройка существенно более трудоёмка. Особенно это касается нетривиальных протоколов вроде Tsunami, использующих гибридную схему с одновременной передачей данных по протоколу UDP и меньшего объёма метаданных по протоколу TCP. Кроме прочего, некоторые из подобных протоколов являются платными (Aspera, Signiant).

Авторами был выбран протокол ssh, преимущественно используемый пользователями для доступа к суперкомпьютерным ресурсам. В эксперименте измерялась скорость передачи файлов по каналу утилитой scp. Хотя формально производительность канала позволяла осуществить монтирование удалённой сетевой файловой системы, на данном этапе исследований авторы от этого отказались из-за несовместимости файловых систем в МСЦ РАН и ССКЦ СО РАН.

3. Постановка и результаты эксперимента

Исследуемая система включала два суперкомпьютерных центра: МСЦ РАН в Москве и ССКЦ СО РАН в Новосибирске, которые были объединены каналом 10 Гбит/с.

Канал непосредственно связывал сети передачи данных высокопроизводительных вычислительных кластеров, обеспечивая при этом автоматическое шифрование передаваемых данных.

В эксперименте производилась передача файлов с узлов вычислительного кластера МСЦ РАН на узлы вычислительного кластера ССКЦ и обратно.

Передача проводилась в njobs параллельных потоков, каждый из которых передавал nfile файлов размером size каждый. Параметры принимали следующие значения:

- size (размеры передаваемых файлов): 8K, 16K, 32K, 64K, 128K, 256K, 512K, 1M, 4M, 16M, 64M, 256M, 512M, 1G
- nfile (количество передаваемых файлов в одном потоке): 2, 10
- njobs (количество потоков): 1, 2, 4, 8, 16, 32, 48, 64.

Результаты измерений продемонстрированы на рисунках 1-4.

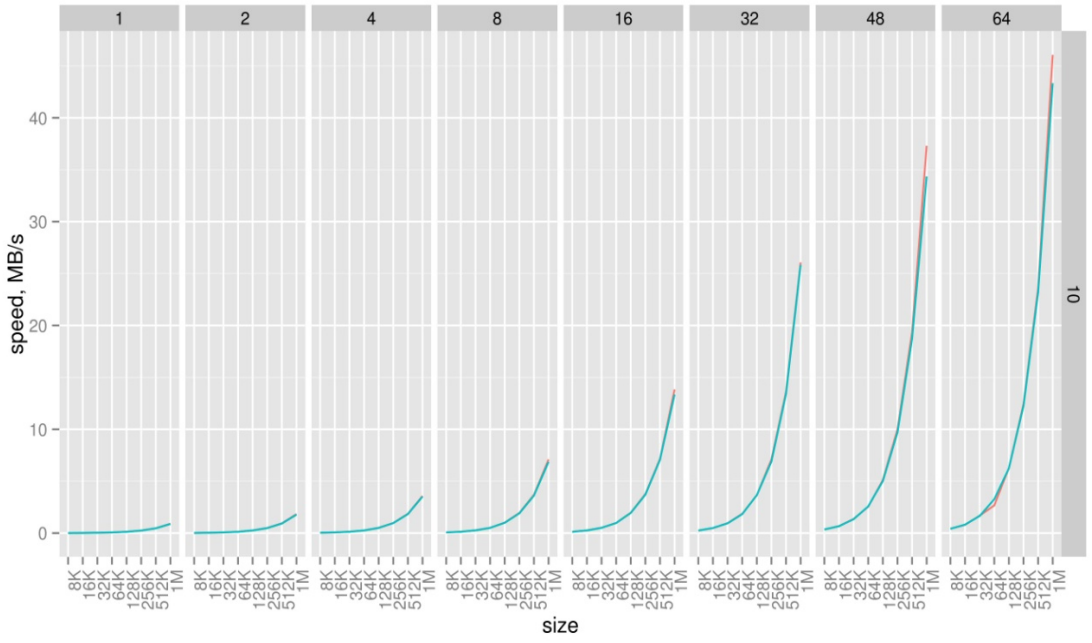


Рис. 1. Скорость передачи файлов в зависимости от размера для файлов размером менее 1 МБайта, 10 файлов в одном потоке

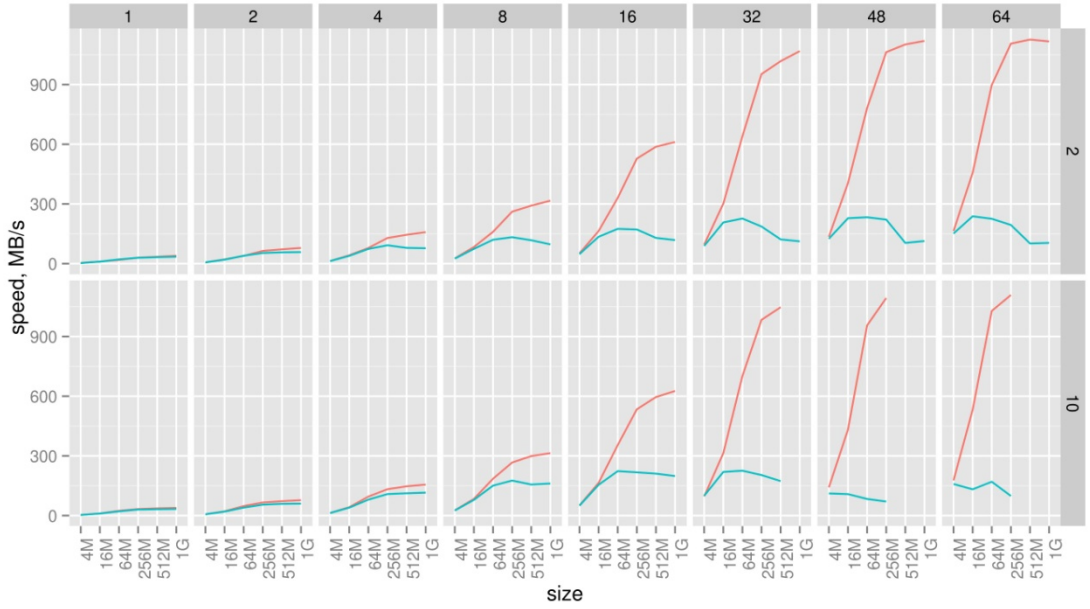


Рис. 2. Скорость передачи файлов в зависимости от размера для файлов размером от 1 МБайта до 1 ГБайта

Графики показывают зависимость суммарной скорости передачи файлов (speed) в мегабайтах/секунду от количества передаваемых файлов в одном потоке, количества потоков и размера передаваемых файлов. Количество передаваемых в одном потоке файлов указано на графиках справа; для файлов малого объёма (менее 1 МБайта) это количество всегда равнялось 10. Оставшиеся два параметра – количество потоков и размер передаваемых файлов – указаны, соответственно, сверху и снизу на графиках 1-2 и наоборот на графиках 3-4. Красные и синие линии обозначают передачу, соответственно, в прямом и обратном направлениях. В экспериментах передавались файлы суммарным объёмом не более 100 ГБ.

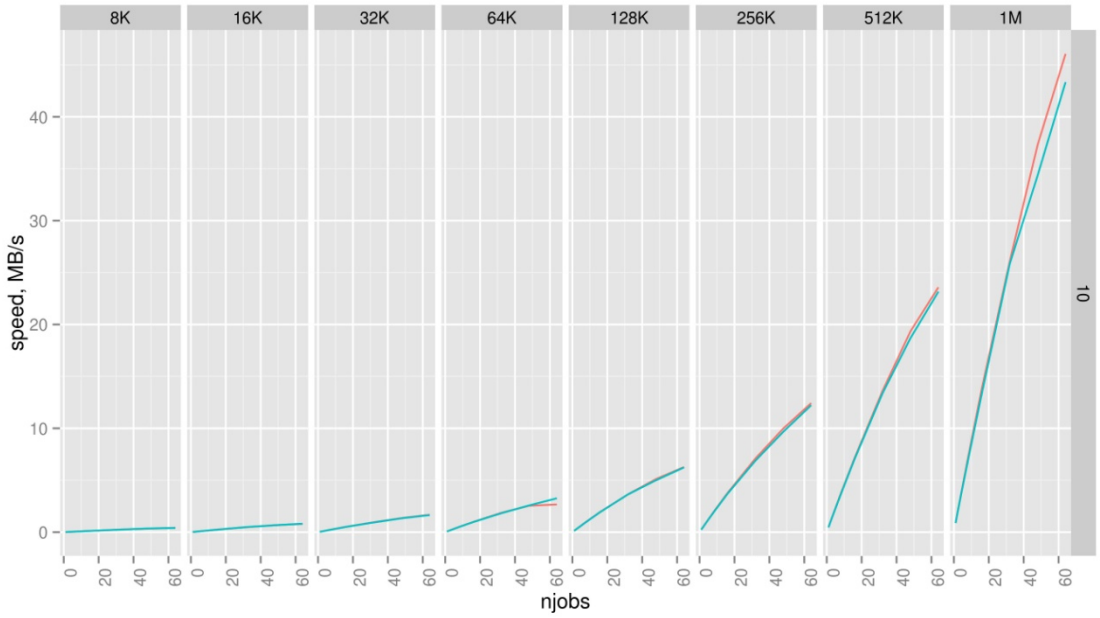


Рис. 3. Скорость передачи файлов в зависимости от числа потоков
для файлов размером менее 1 МБайта, 10 файлов в одном потоке

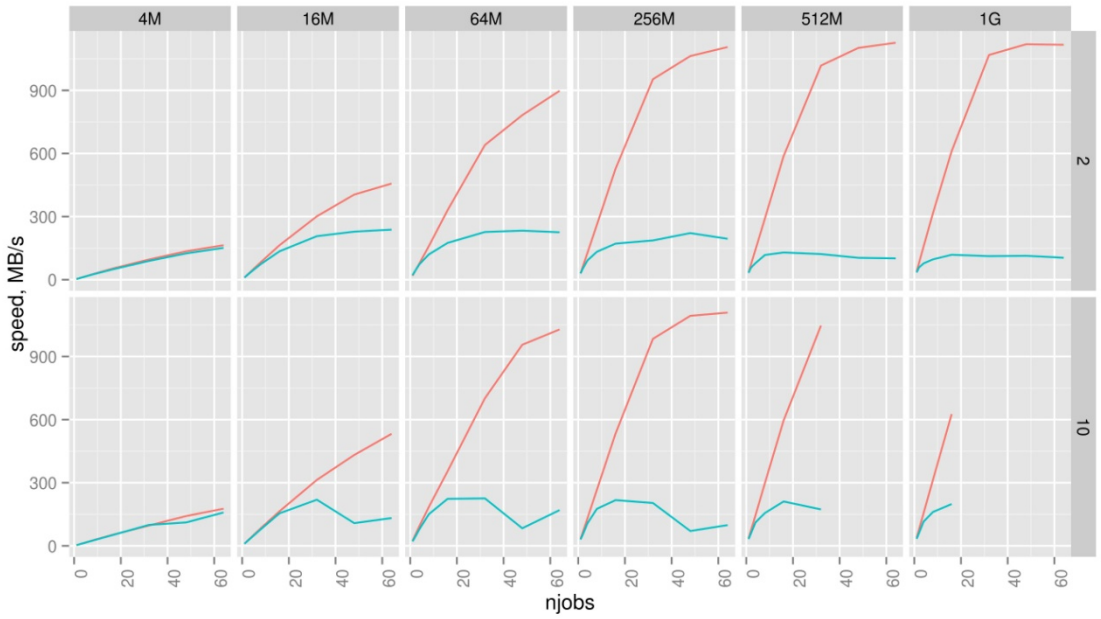


Рис. 4. Скорость передачи файлов в зависимости от числа потоков для файлов размером до 1 Гбайта

Как видно из приведённых графиков, с увеличением количества одновременно запущенных передач через протокол scp вначале происходит почти линейный рост скорости передачи до момента насыщения канала. Тесты продемонстрировали асимметричность канала, показали его пропускную способность при одновременной передаче нескольких больших файлов и латентность при передаче малых файлов. Латентность оказалась равна времени передачи файлов объёма до 10 Кбайт, поэтому графики для латентности отдельно не представлены. Для полной утилизации пропускной способности канала необходима одновременная передача нескольких файлов. После насыщения канала при дальнейшем увеличении количества одновременно установленных соединений наблюдается падение производительности. При асимметричном канале связи между кластерами становится важным направление передачи данных. Поскольку объединение суперкомпьютеров предполагает регулярный обмен данными сравнимых объёмов в обоих направлениях, именно меньшая пропускная способность в одном из направлений будет ограничивать производительность объединённой системы.

4. Способы повышения скорости передачи данных

Возможными способами увеличения скорости передачи файлов, являются:

- сжатие данных до передачи;
- объединение файлов небольшого объёма;
- увеличение размера данных, передаваемых в одном пакете.

Сжатие данных до их передачи с последующим разархивированием после получения позволяет эффективно передавать большие объёмы полезной информации при ограниченном объёме физически передаваемых по каналу данных. Однако эффективность такого подхода ограничивается двумя факторами: созданием дополнительной нагрузки на передающие и принимающие вычислительные узлы и зависимостью от типа передаваемых данных.

Последнее означает, что не все данные получится эффективно сжать, а при неудачном выборе алгоритма объём данных может возрасти. Ограничивающие факторы не позволяют рекомендовать метод сжатия для произвольных передаваемых данных.

При передаче большого количества файлов малого объёма могут создаваться дополнительные задержки.

К примеру, если протокол не поддерживает использование одного пакета данных для передачи более чем одного файла, то размеры файлов эффективно уменьшают размер пакетов.

Так как некоторые операционные системы и приложения не могут эффективно обрабатывать данные, представленные в виде большого количества файлов малого объёма, то рекомендуется избегать такого представления.

Рассмотрим влияние на скорость передачи данных по сети величины MTU (от

англ. Maximum Transmition Unit) – максимального размера пакета сетевого уровня, упаковываемого в кадр на канальном уровне. С увеличением MTU уменьшается число пакетов, необходимых для передачи данных фиксированного объёма. Каждый пакет помимо данных несёт служебную информацию: адрес получателя, адрес отправителя и др. сведения, необходимые для его доставки. За счёт уменьшения числа передаваемых пакетов уменьшается объём служебной информации, которую необходимо передать по сети, и, соответственно, объём фактически передаваемых данных. Но с увеличением MTU возрастает эффективная латентность сети, так как передача пакета большого объёма делает невозможной передачу иных пакетов до окончания передачи. В стандарте протокола Ethernet максимальным значением MTU является 1518 байт, при использовании оборудования, поддерживающего Jumbo Frames [8], – 9 КБ.

Хотя увеличение величины MTU и позволяет увеличить эффективную скорость передачи файлов, её поддержка может отсутствовать на сетевом уровне. Кроме того, неприемлемым может оказаться вызванное большим значением MTU увеличение латентности. В связи с этим нельзя рекомендовать такой подход для произвольных конфигураций.

Отметим, что повышению эффективности передачи данных способствует настройка параметров протокола TCP/IP на узлах, участвующих в информационном обмене [9].

5. Заключение

В настоящей работе предпринята попытка экспериментального исследования свойств высокоскоростного защищённого канала связи длиной более 2800 км.

Исследование показало возможности насыщения скорости передачи данных по каналу с использованием известного протокола передачи данных scp и выявило асимметричность канала связи.

Выявленные свойства канала при объединении с его помощью двух или более суперкомпьютерных систем определяют ограничения на пользовательские задания, выполняемые в объединённых СКЦ. Задания должны минимизировать объём передаваемых данных и частоту их передачи по каналу связи, а также учитывать асимметричность канала.

Работа выполнена в МСЦ РАН при поддержке программы III.1 Отделения математических наук РАН. При выполнении исследований использовался суперкомпьютер МВС-10П

Effective usage of the link between geographically distributed supercomputer centers

A.V.Baranov, D.Yu.Derbyshev, B.V.Dolgov, S.A.Leshchev, A.P.Ovsyannikov, B.M.Shabanov, D.V.Vershinin

Abstract: The article considers data transmission organization trough a high speed communication channel between geographically remote supercomputer centers for joining of computing resources. The paper reviews results of experiments with the 10 Gbps protected communication channel between JSCC RAS (Moscow) and SSCC SB RAS (Novosibirsk).

Keywords: telecommunication network, high speed link, network protocol, supercomputer center, data transmission

Литература

1. О.С. Аладышев, М.Р. Биктимиров, М.А. Жижченко, А.П. Овсянников, В.М. Опилек, Б.М. Шабанов, Н.Ю. Шульга. Особенности построения объединенной сети суперкомпьютерного центра. «Программные продукты и системы», (2008), № 2, С. 9-12.

2. Г.И. Савин, В.В. Корнеев, Б.М. Шабанов, П.Н. Телегин, Д.В. Семенов, А.В. Киселёв, А.В. Кузнецов, О.И. Вдовикин, О.С. Аладышев, А.П. Овсянников. Создание распределенной инфраструктуры для суперкомпьютерных приложений. «Программные продукты и системы», (2008), № 2, С. 2–7.

3. Г.И. Савин, Б.М. Шабанов, П.Н. Телегин, О.И. Вдовикин, А.П. Овсянников, И.А. Козырев, В.В. Корнеев, Д.В. Семёнов, А.В. Киселёв, А.В. Кузнецов. Инфраструктура грид для суперкомпьютерных приложений. «Известия высших учебных заведений. Электроника», (2011), № 1 (87), С. 51-56.

4. Tsunami UDP Protocol // [В Интернете] http://tsunami-udp.sourceforge.net/ (дата обращения: 16.10.2017)

5. UDT: Breaking the Data Transfer Bottleneck // [В Интернете] http://udt.sourceforge.net/ (дата обращения: 16.10.2017)

6. Aspera FASP High Speed Transport. A Critical Technologiy Comparison. White paper. // Aspera, an IBM company, 2017

7. Flight Product White Paper // Signant, 2017

8. Ethernet Jumbo Frames, Version 0.1, November 12, 2009 // Ethernet Alliance, 2009

9. Enabling High Performance Data Transfers. System Specific Notes for System Administrators (and Privileged Users) // [В Интернете] https://www.psc.edu/index.php/projects/641-tcp-tune (дата обращения: 16.10.2017)

Векторный потоковый процессор и многопроцессорная система с общей памятью на его основе

Н.И. Дикарев[1], Б.М. Шабанов[2], А.С. Шмелёв[3]

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail's: [1] nic@jscc.ru, [2]shabanov@jscc.ru, [3]guest8993@rambler.ru

Аннотация. Принцип работы процессора с архитектурой управления потоком данных (потокового процессора) позволяет достичь значительно более высокой производительности за счет децентрализованной схемы управления и поиска готовых к выполнению команд в окне из десятков тысяч команд, а не из 200 команд, как в лучших современных процессорах. Однако известные проекты разработки потокового процессора не смогли составить конкуренцию процессорам традиционной архитектуры из-за сложности работы с массивами данных и необходимости вести ассоциативный поиск в памяти большой ёмкости. Показано, что в разрабатываемом векторном потоковом процессоре за счет использования оригинального метода хранения массивов можно значительно уменьшить требуемую ёмкость ассоциативной памяти и достичь более высокой производительности одного процессорного ядра. Приводятся результаты моделирования программы перемножения матриц на одном процессоре и её ускорение в системе из нескольких ядер разрабатываемого процессора с общей памятью.

Ключевые слова: векторный процессор, архитектура управления потоком данных, многопроцессорная система с общей памятью, оценка производительности.

Введение

Замедлившийся в последние годы рост степени интеграции СБИС говорит о том, что совершенствование КМОП технологии, по-видимому, подходит к концу, и следует искать другие способы повышения производительности суперЭВМ помимо простого увеличения числа процессорных ядер на кристалле СБИС и числа таких кристаллов в суперЭВМ. Это тем более справедливо, если учесть, что при числе ядер в суперЭВМ превышающем сотни тысяч их и сейчас трудно эффективно использовать в задачах моделирования, поскольку простое увеличение размеров сетки уже не приводит к заметному повышению его точности.

Однако точность моделирования можно повысить, если вместо упрощенных математических моделей реальных систем использовать более сложные модели, учитывающие важные детали этих систем. При этом модели становятся нелинейными и со связанными структурами, что приводит к необходимости работы с мелкоструктурным и нерегулярным параллелизмом, на котором современные процессоры и ускорители имеют низкую эффективность работы.

Разрабатываемый в МСЦ РАН векторный процессор с архитектурой управления потоком данных (потоковый процессор) имеет не только значительно более высокую производительность одного процессорного ядра, но и способность её сохранения при работе с мелкоструктурным параллелизмом [1], что обеспечивает ему существенные преимущества при использовании в суперЭВМ.

Цель данной работы – показать за счет чего в разрабатываемом векторном потоковом процессоре можно достичь значительно более высокую производительность по сравнению с фон-неймановским процессором, оценить оптимальную производительность одного ядра и её ускорение в системе из нескольких ядер с общей памятью.

Принцип работы потокового процессора и сложности его реализации

Процессор с архитектурой управления потоком данных потенциально может достичь значительно более высокой производительности за счет того, что параллелизм выполнения команд в программе закладывается уже при составлении графа программы, и в отличие от фон-неймановского процессора его не требуется выявлять с помощью сложной аппаратуры из последовательности команд, заданной счетчиком команд.

Программой в потоковом процессоре является ориентированный граф, узлами которого являются команды, а информация по дугам передается в виде токенов, содержащих значение операнда и контекст. Любая команда в

графе выдаётся на выполнение по прибытию на вход последнего из токенов операндов с одинаковым контекстом. После вычисления результата в исполнительном устройстве новые токены со значением результата отправляются на входы последующих команд согласно графу программы, а использованные токены операндов уничтожаются. Тем самым принцип работы потокового процессора основан на памяти поиска пар (ППП) готовых операндов, а не на линейно адресуемой памяти, как у фоннеймановского процессора, что исключает конфликты информационной зависимости, связанные с порядком обращения по чтению и записи к слову в обычной памяти. При этом в потоковом процессоре отсутствует центральное устройство управления (счетчик команд), а параллелизм выполняемых команд определяется в динамике по приходу операндов на входы команд в децентрализованной схеме, поскольку ППП можно выполнить в виде многих работающих в параллель модулей (Рис. 1). Такой принцип работы позволяет одновременно выполнять в потоковом процессоре команды из нескольких программ, и в отличие от фоннеймановского процессора у него нет аппаратуры, отвечающей за выполнение каждого из потоков команд. Однако необходимо чтобы команды из разных программ умещалась в модулях памяти команд (ПК), которые для каждого ИУ по номеру команды в графе читают её код операции и информацию о том, сколько токенов со значением результата нужно сформировать и на входы каких команд их отправить. Кроме того, необходимо чтобы не переполнялись модули ППП, в которых токен с одним операндом команды ожидает прихода второго токена операнда (для двухвходовых команд).

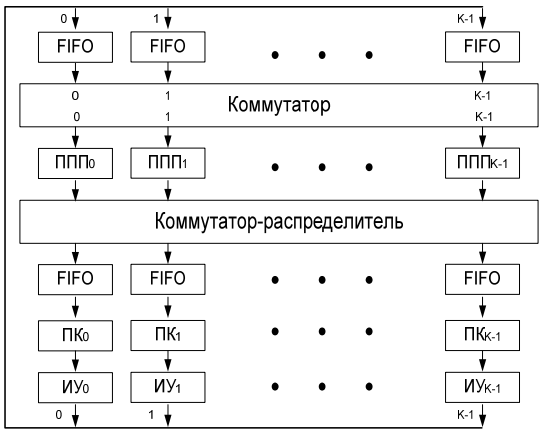


Рис. 1. Структурная схема потокового процессора

Заметим, что модули ПК и ППП должны иметь высокое быстродействие, поскольку вносимая ими задержка вместе со временем вычисления результата в ИУ и задержкой прохождения токенов через два коммутатора определяет время выполнения команды в цепочке команд, связанных зависимостью данных. Если среднюю задержку выборки данных (командного слова) из ПК большой ёмкости при её реализации в виде кэш памяти можно приблизить к времени выборки из блока памяти малой ёмкости, то к ППП такой приём не применим. Дело в том, что эффективная работа кэш памяти основана на многократной выборке одних и тех же данных, например команд в цикле, а в ППП такая повторяемость обращений к одной и той же ячейке памяти отсутствует. Так, после записи токена в ППП будет лишь одно обращение к той же ячейке памяти по приходу второго токена операнда той же команды.

Таким образом, повысить быстродействие модулей ППП можно только за счет уменьшения их ёмкости, при этом нельзя допустить переполнения даже одного модуля ППП в потоковом процессоре, поскольку это приведёт к неработоспособности процессора. Более высокую производительность обеспечивает динамическая модель потокового процессора, в которой поиск готовых команд в ППП осуществляется не только по совпадению номера команды у токенов операндов, но и ещё нескольких полей, например, номера индекса, итерации и запуска подпрограммы. Это позволяет на одном и том же графе программы выполнять в параллель итерации нескольких вложенных циклов и запусков процедур, то есть выявлять больше параллелизма в программе [2]. Однако это приводит к сложности реализации ППП, поскольку модули ППП должны осуществлять ассоциативный поиск, иметь большую ёмкость, чтобы не допускать их переполнения, а также высокое быстродействие, что несовместимо с большой ёмкостью.

Для снижения требуемой ёмкости ППП нужно, во-первых, не использовать её для хранения массивов данных, которые нужно хранить в обычной линейно адресуемой памяти большой ёмкости. Во-вторых, нельзя допускать чрезмерного параллелизма вычислений в программе, с которым не сможет справиться ограниченное число ИУ в процессоре. Избыточный параллелизм проявляется в увеличении числа токенов операндов, ожидающих в ППП прихода парного операнда, и числа готовых команд, выданных из ППП и ожидающих освобождения нужного ИУ в буфере FIFO перед ПК на входе ИУ (см. рис. 1). Наконец, для снижения ёмкости ППП нужно уменьшать число команд, выполняемых в программе. Тогда меньше токенов будет циркулировать в схеме процессора и, следовательно, ожидать парных токенов в ППП.

Проблема избыточного параллелизма в потоковых ЭВМ тесно связана с обработкой и

хранением массивов данных. В общем случае потоковому процессору не нужны другие устройства памяти кроме ППП. При создании массива A его элементы A(i) поступают в ППП в виде токенов с различными значениями индекса i в поле контекста. Каждый из этих элементов либо находит себе пару в ППП, и команда выдается на выполнение, либо записывается в ППП и ждет второго операнда. Однако при увеличении размера обрабатываемых массивов естественный параллелизм у многих задач может быть столь велик, что обрабатывать все их элементы в параллель невозможно, поскольку для этого не хватит ресурсов ни в какой реальной системе [3]. Поэтому потоковый процессор часто содержит обычную память для хранения массивов - память структур данных, и эта функция снимается с ППП, но тогда для исключения конфликтов информационной зависимости необходима синхронизация обращений к его отдельным элементам по записи и чтению, что увеличивает аппаратные затраты и число выполняемых команд.

Для ограничения параллелизма обычно используется система авторегулирования с отрицательной обратной связью, как например, в Манчестерском проекте потоковой машины (MDFM) [4]. Система авторегулирования оценивает уровень нагрузки в процессоре по числу команд в буфере FIFO, ожидающих выполнения в ИУ, и разрешает или задерживает активацию новых итераций цикла либо вызовов подпрограмм. В MDFM эта функция возложена на «глобальный распределитель», который помимо регулировки нагрузки занимается также выделением места под создаваемые массивы в памяти структур данных, поскольку активация нового программного блока часто требует ещё и выделения места в памяти для записи данных из этого блока.

Произвольная длина создаваемых массивов обусловила сложность алгоритма работы «глобального распределителя» в MDFM и его медленную работу, что, в свою очередь, привело к большому размеру активируемых им «гранул» параллельных блоков. Например, во вложенных циклах «гранулами» в MDFM были внешние итерации цикла. В результате большая инерционность системы авторегулирования приводит к появлению в MDFM избыточного параллелизма. Так, на программе перемножения матриц размером 35x35 число команд в буферах на входе ИУ в MDFM достигало 6764, и число токенов в ППП - 39500 [4].

Отмеченные выше сложности аппаратной реализации потокового процессора привели к тому, что ни в одном из многочисленных проектов по его созданию, пик которых за рубежом пришелся на 80-е годы прошлого века, не удалось достичь более высокой производительности по сравнению фон-неймановским процессором, и к концу 2000-х годов таких проектов практически не осталось.

Векторный потоковый процессор

Наличие векторной обработки в разрабатываемом в МСЦ РАН векторном потоковом процессоре (ВПП) позволяет во много раз уменьшить число выполняемых команд, поскольку одна векторная команда заменяет собой цикл с независимыми итерациями с числом итераций VL, где VL – длина вектора. Для записи результатов векторных команд и хранения массивов в ВПП используется линейно адресуемая память, содержащая два уровня - память векторов (ПВ) большой ёмкости, реализованную на микросхемах динамической памяти, и быструю локальную память векторов (ЛПВ) значительно меньшей емкости, размещенную на процессорном кристалле. Выделение свободного места в ПВ и ЛПВ производится фрагментами, равными аппаратной длине вектора VL_{max}=256 слов, и служит для записи результата одиночных векторных команд. Соответственно и ограничение параллелизма в ВПП осуществляется с минимально возможным размером «гранул» - уровня отдельных векторных команд. Выделение места в каждом из двух уровней памяти фрагментами постоянной длины производится чисто аппаратно за счет ведения списков свободных векторов в ПВ и ЛПВ (без вмешательства операционной системы). Это позволяет быстро и гибко распределять адресное пространство в ПВ и ЛПВ как между блоками одной программы, так и между программами, и свести к минимуму избыточный параллелизм. Так при выполнении той же программы перемножения матриц, но с размером не 35x35, как в MDFM [4], а 128x128 суммарное число команд в буферах на входе ИУ в ВПП не превышает 25 и токенов в ППП – 550, то есть на 2 порядка меньше чем в MDFM. Причем векторные ИУ в ВПП на этой программе работают без простоев, обеспечивая близкую к пиковой производительность.

Однако при такой фрагментации памяти большие массивы в ВПП приходится хранить в виде «векторов-указателей», то есть векторов, элементами которых являются указатели векторов подмассивов, как это показано на рис. 2. Соответственно массивы, например матрицу размером больше чем 256x256, приходится разбивать на подматрицы с числом элементов по каждому измерению не более VL_{max}, и обрабатывать такие подматрицы последовательно. При этом в программу нужно вводить дополнительные циклы, что, безусловно, усложняет программирование.

Однако подобный приём, а именно, разбиение массива на подмассивы с локализацией вычислений в них широко используется в настоящее время для снижения требуемой пропускной способности процессора к памяти. Это блочные методы, которые используют ёмкость нескольких уровней кэш памяти процессора и локализацию вычислений в них для достижения высокой производительности. Сложность программирования при использовании блочных методов в современных процессорах возрастает многократно. Так если исходный алгоритм программы перемножения матриц состоит из 10 строк кода, то оптимизированный алгоритм, рассчитанный на блочное размещение по двум уровням кэш памяти и в регистровом файле, содержит уже порядка 1000 строк кода [5].

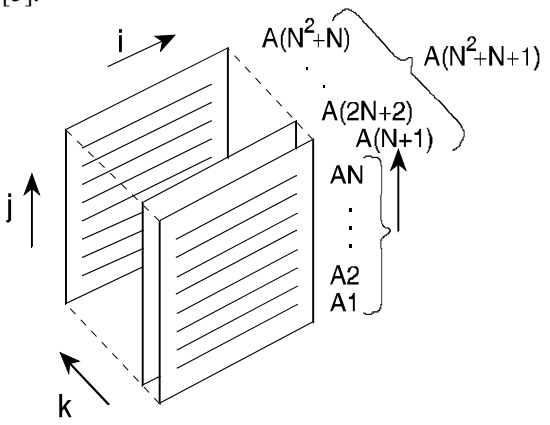


Рис. 2. Представление трехмерного массива векторами-указателями.

В ВПП использование векторов-указателей для хранения больших массивов также представляет собой блочный метод, и его, как будет показано ниже, легко применить для снижения требуемой пропускной способности процессора к памяти. Причем поблочное копирование массива из ПВ в ЛПВ и его последующая обработка требует меньших затрат программиста по сравнению с копированием блоков разного размера для размещения по двум уровням кэш памяти и в регистровом файле, как это требуется в процессорах традиционной архитектуры. Кроме того, использование векторов-указателей позволяет значительно уменьшить число выполняемых скалярных команд при выполнении программы и, следовательно, повысить степень векторизации и производительность векторной обработки в ВПП.

Для иллюстрации рассмотрим программу перемножения матриц, как это было сделано в [1].

Основной объем вычислений в программе перемножения матриц A и B приходится на подпрограмму SGENV [5] по вычислению элементов j-го столбца матрицы результата C:

```
      DO 20 k=1, N
       DO 20 i=1, N
20     C(i,j)=C(i,j)+A(i,k)*B(k,j).
```

Внутренний цикл в этой подпрограмме исключается за счет использования векторных команд, и подпрограмму SGENV можно представить в следующем виде:

```
      DO 20 k=1, N
20   Cj = Cj + Ak * B(k,j).
```

Граф рассматриваемой подпрограммы для ВПП при использовании метода векторов указателей показан на рис. 3 в предположении, что размер матрицы $N<VL_{max}$.

На вход графа поступают подлежащий вычислению указатель вектора столбца Cj с нулевыми значениями элементов, указатель матрицы A – вектор A(N+1), содержащий указатели векторов столбцов A1,A2,... AN, и указатель j-го столбца матрицы B - Bj.

Подпрограмма SGENV в программе перемножения матриц вызывается N раз со значениями j от 1 до N, и полный граф этой программы содержит еще внешний цикл, команды которого отправляют токены на входы графа на рис. 3 и принимают токен результат с его выхода.

Эти команды, в частности, читают указатель Bj из вектора указателя матрицы B(N+1) и записывают указатель Cj с выхода графа в j-й элемент вектора указателя C(N+1).

Как уже отмечалось выше, для возможности параллельного выполнения команд из разных итераций цикла контекст токена в ВПП имеет поля индекса (И), итерации (Т) и подпрограммы (П).

Поле И содержит номер итерации самого внутреннего цикла (k в формуле выше), поэтому токены указателей, приходящие на вход графа на рис. 3, имеют поле И=1.

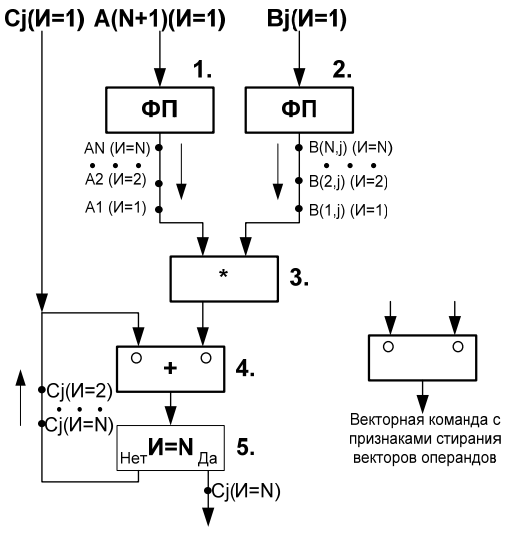


Рис. 3. Граф подпрограммы SGENV для выполнения в ВПП

Команды 1 и 2 "Формирование Потока" (ФП) в графе на рис. 3 читают в ЛПВ элементы вектора указателя A(N+1) матрицы A и в ПВ вектора столбца Bj соответственно, формируя последовательность токенов со значениями элементов A1,A2,... AN и B(1,j),B(2,j),... B(N,j) для всех итераций цикла, выполняемого подпрограммой. Поэтому в каждой из N итераций цикла в ВПП выполняются лишь две векторные команды с плавающей запятой (команды 3 и 4 на рис. 3) и одна скалярная команда (команда 5). Команда 5 сравнивает номер текущей итерации k (поле И) в токене вектора Cj с длиной вектора N и осуществляет выход из цикла при k=N, а при невыполнении этого условия увеличивает на 1 значение И в токене результата для передачи Cj на вход команды сложения 4 в следующей итерации цикла. Признаки стирания на входах команды 4 указывают, что оба указателя вектора используются в качестве операнда последний (единственный) раз и после выполнения команды должны быть отправлены в список свободных векторов ЛПВ. Освобождение ресурса памяти путем установки признаков стирания или выполнения команд "Удаление Вектора" возложено на компилятор.

Заметим, что вектора, представляющие промежуточные и коротко живущие результаты команд, как для команд 3 и 4 в графе, должны быть направлены компилятором на запись в ЛПВ. Так вектор Cj с нулевыми значениями элементов, поступающий на вход подпрограммы (см. рис. 3), при его создании записывается в ЛПВ. Матрицы A и B изначально находятся в ПВ, и поскольку вектора столбцы матрицы A используются в качестве операндов командой 3 при каждом вызове подпрограммы SGENV, то есть N раз, то столбцы матрицы A сначала копируются из ПВ в ЛПВ и записываются в качестве элементов нового указателя матрицы A(N+1), который и подаётся на вход подпрограммы. Тем самым копирование матрицы A в ЛПВ позволяет уменьшить число обращений в ПВ при вычислении матрицы результата уменьшается в N раз. Заметим лишь, что копировать матрицу B из ПВ в ЛПВ не имеет смысла, так как рассматриваемая подпрограмма читает разные столбцы Bj, и нет повторных обращений к одним и тем же элементам этой матрицы.

Сравним число команд, необходимых для выполнения подпрограммы SGENV, в ВПП с векторным процессором традиционной архитектуры, например, CRAY Y-MP. В CRAY Y-MP в каждой итерации цикла этой подпрограммы выполняется 8 команд, из них 5 – скалярных, а в ВПП – 3, и только 1 – скалярная. Значительное уменьшение общего числа выполняемых команд и особенно скалярных команд в ВПП достигается за счет формирования токенов операндов команды 3 с установкой индекса И от 1 до N векторными командами ФП для всех итераций цикла. Это позволило исключить одиночные команды обращения к памяти по чтению векторов Ak и элементов B(k,j), команды вычисления адресов чтения векторов Ak, а также увеличения индекса, необходимые в традиционном процессоре. Тем самым удается уменьшить число выполняемых команд в 2,7 раза и фактически векторизовать не только самый внутренний, но и следующий из вложенных циклов в программе.

Как показали результаты моделирования программы перемножения матриц, полученные на VHDL модели ВПП и приведенные в [1], программа автоматически распараллеливается на вычисление нескольких столбцов матрицы результата, так чтобы обеспечить полную загрузку сумматоров и умножителей в векторном АЛУ. Дело в том, что цикл, выполняемый в подпрограмме SGENV, имеет зависимость по данным, когда вектор, созданный сумматором в предыдущей итерации цикла, используется в качестве его операнда в следующей итерации. Поэтому итерации данного цикла приходится выполнять последовательно, что приводит к большому периоду следования команд векторного сложения в ВВП и, как следствие, его низкой производительности. Здесь проявляется известный недостаток потоковой архитектуры, заключающийся в низкой производительности при выполнении последовательных команд.

Однако в ВПП этот недостаток компенсируется возможностью осуществлять поиск готовых команд в гораздо большем окне (в 100 раз) по сравнению с лучшими процессорами традиционной архитектуры, что позволяет не останавливать работу умножителя, поставляющего вектора операнды на другой вход сумматора. В результате происходит инициирование следующей подпрограммы SGENV, затем ещё одной и так до тех пор, пока векторный сумматор не станет выполнять команды из нескольких подпрограмм, то есть вычислять в параллель несколько столбцов матрицы результата, что устраняет простои в его работе, и производительность ВПП приближается к пиковой. При этом число команд, ожидающих парного операнда у сумматора в ВПП, растёт пропорционально его производительности, и при производительности ВПП равной 256 флоп в такт полная загрузка векторного сумматора достигается при одновременном выполнении 20 подпрограмм SGENV, а число токенов в ППП достигает 3300. Кроме того, в [1] было показано, что в ВПП можно не только повысить производительность процессора до 256 флоп в такт, что в 4 раза выше, чем у последнего векторного процессора NEC SX-ACE, но

сохранять ее при значительно меньших размерах обрабатываемых массивов. Последние изменения, внесенных в схему и VHDL модель ВПП, заключались в замене раздельных сумматоров и умножителей с плавающей запятой в векторном блоке процессора на сдвоенные сумматоры и умножители (FMA) и были направлены на уменьшение числа портов доступа от этих АЛУ к ЛПВ. Это позволило уменьшить число конфликтов занятого банка в ЛПВ и повысить производительность ВПП на программе перемножения матриц размером 512x512 с 222 до 243 флоп в такт или до 95% от пиковой производительности в 256 флоп в такт.

Многопроцессорная система на основе ВПП и результаты её моделирования

Высокая производительность векторной обработки в ВПП достигается за счет реализации векторных ИУ в виде 32 идентичных колец (lanes) обработки, что позволяет обрабатывать вектор страницами по 32 элемента в такт. Локализация вычислений в пределах каждого кольца позволяет уменьшить среднюю длину связей на кристалле и, как следствие, снизить потребляемую мощность либо повысить тактовую частоту процессора. Каждое из этих колец векторной обработки в ВПП содержит модуль ЛПВ ёмкостью 128 Кбайт, а также 4 конвейерных FMA ИУ, которые, работая в параллель, обеспечивают пиковую производительность ВПП равную 256 флоп в такт. Для чтения трех операндов и записи результата каждому из FMA ИУ требуется 4 порта доступа к ЛПВ. Расслоение ЛПВ на 16 двухпортовых банков вместе с коммутатором в каждом кольце дают возможность задействовать до 32 портов доступа к ЛПВ. Векторное АЛУ из четырех FMA ИУ используют 16 портов, и ещё через три порта к ЛПВ обращается блок выполнения специальных операций (БВСО). БВСО может работать одновременно с векторным АЛУ, выполняя команды чтения и записи одиночных элементов вектора, команды редукции, среди которых вычисление суммы элементов вектора и поиска максимального (минимального) элемента, а также команды сборки – разбрасывания элементов вектора.

Производительность одного ядра ВПП на векторной обработке можно наращивать как за счет увеличения числа колец векторной обработки, так и за счет числа FMA ИУ в кольце. Однако и в том и другом случае аппаратные затраты растут быстрее пиковой производительности из-за наличия коммутаторов. С увеличением числа FMA ИУ в кольце пропорционально растет число портов n в коммутаторе, подключающем входы и выходы FMA ИУ к банкам ЛПВ, а затраты на коммутатор растут гораздо быстрее - почти как n^2. Также быстро увеличиваются и затраты на коммутатор в составе БВСО от числа колец в ВПП, поскольку с его помощью осуществляется доступ к ЛПВ в любом из колец. Кроме того, с ростом пиковой производительности процессора его реальная производительность растёт не так быстро. Так на программе перемножения матриц у ВПП с 32 кольцами и 4 FMA ИУ в кольце производительность равна 243 флоп в такт или 95% от пиковой производительности в 256 флоп в такт. Моделирование той же программы при увеличении числа FMA ИУ в кольце до 8 показало, что отношение реальной производительности ВПП к пиковой уменьшается до 83%.

Быстрый рост аппаратных затрат при снижении отдачи в виде реальной производительности говорит о том, что не имеет смысла увеличивать производительность одного ядра ВПП сверх определённого предела. В [7] было показано, что на программе перемножения матриц максимальная производительность с единицы площади кристалла достигается у ВПП при 32 кольцах, 4 FMA ИУ в кольце и расслоении ЛПВ на 16 двухпортовых модулей. При 22 нм технологии изготовления СБИС этот экстремум составляет 2,19 флоп в такт с $мм^2$, что позволяет на кристалле площадью 450 $мм^2$ разместить многопроцессорную систему из 4 ядер ВПП с суммарной производительностью 1024 флоп в такт. Для моделирования такой системы исходная VHDL модель ВПП уровня регистровых станций была использована для построения модели многопроцессорной системы из нескольких ядер этого процессора с общей памятью, причём число ядер в модели можно задать в качестве параметра.

В качестве общей памяти в модели многопроцессорной системы использовалась ПВ – память большой ёмкости, реализованная на микросхемах динамической памяти, причем пропускная способность к ПВ (256 ГБ/с) осталась той же, что была в модели одного ядра ВПП. Каждое ядро ВПП в модели при обращении к ПВ получало доступ сразу ко всем каналам динамической памяти через схему разрешения конфликтов. Меняющийся приоритет у процессорных ядер в этой схеме обеспечивал симметричный доступ к ПВ между всеми ядрами кроме нулевого ядра, которое имело более высокий приоритет, поскольку оно помимо пользовательских программ выполняет управляющую программу. Аналогичная схема разрешения конфликтов использовалась и для доступа к устройству распределения памяти, которое выделяет адрес свободного вектора для записи результата векторной команды, если он должен быть записан в ПВ, а не в ЛПВ.

Для распределения задач по процессорам был разработан граф управляющей программы, которая ведёт список свободных ядер в многопроцессорной системе и формирует токены с исходными данными для запуска очередного процесса на выполнение. Обрабатываемые процессом массивы данных должны находиться в ПВ, и управляющая программа посылает ядру токены с указателями этих массивов. Разработанная VHDL модель многопроцессорной системы и управляющая программа отлаживались на программе перемножения матриц. Использовался блочный вариант программы, при котором матрицы A и B размером 512x512 состоят из 4 блоков размером 256x256, и в соответствии с описанным ранее методом хранения массивов хранятся в ПВ в виде трехмерного массива размером 4x256x256, как показано на рис. 2. Одна и та же программа перемножения матриц загружается в память команд каждого из процессорных ядер, а в память команд нулевого ядра - ещё и управляющая программа.

Программа перемножения матриц в процессорном ядре начинает вычисление подматрицы результата размером 256x256, получив от управляющей программы 3 токена, два из которых содержат вектора указатели матриц A и B размером 512x512, и третий токен передаёт номер вычисляемой подматрицы. В процессе вычисления по переданному номеру подматрицы результата программа перемножения матриц вначале копирует из ПВ в ЛПВ ядра нужную подматрицу A размером 256x256 и затем 256 раз выполняет подпрограмму SGENV. После чего делается ещё один проход вычислений с копированием другой подматрицы A и суммированием с накопленными в первом проходе промежуточными результатами. По окончании второго прохода вычисленная подматрица C переписывается из ЛПВ ядра в общую ПВ, и токен с вектором указателем вычисленной подматрицы отправляется управляющей программе в нулевое ядро.

Управляющая программа записывает полученный указатель подматрицы в качестве элемента вектора указателя матрицы C и отмечает приславшее токен ядро как свободное, чтобы отправить ему задание на вычисление следующей подматрицы C, если такие остались.

Моделирование выполнения программы перемножения матриц на системе из двух процессорных ядер показало, что при обработке матриц размером 512x512 каждое из ядер вычисляет по 2 подматрицы результата. При этом на нулевом ядре одновременно выполняются управляющая программа и программа перемножения матриц.

Они не мешают друг другу, поскольку управляющая программа загружает работой скалярные ИУ процессорного ядра, а перемножения матриц - векторные.

При использовании в модели ядер с производительностью 256 флоп в такт (32 кольца с 4 FMA ИУ) время вычисления ядром подматрицы результата составило 279 тысяч тактов, что соответствует производительности 240,9 флоп в такт (94,1% от 256).

Перерыв в работе каждого ядра между вычислением первой и второй подматриц результата, и обусловленный работой управляющей программы, составляет меньше 200 тактов, то есть пренебрежимо мал.

Большее влияние оказывает задержка в 2200 тактов начала копирования первым ядром подматрицы A из ПВ в ЛПВ из-за конфликта занятого тракта обращения к ПВ, поскольку нулевое ядро выполняет аналогичное копирование.

В результате, время выполнения программы перемножения матриц размером 256x256 на системе из двух ядер ВПП составило 571 тысячу тактов, что соответствует производительности 478,7 флоп в такт или 93,5% от пиковой производительности.

Если учесть, что блочный вариант той же программы на одном ядре выполняется с производительностью 243 флоп в такт, то ускорение составляет 1,97, что можно считать близким к линейному ускорению.

Заключение

Таким образом, результаты моделирования ВПП показали, что в нём действительно можно достичь значительно более высокой производительности за счет в 2 – 3 раза меньшего числа выполняемых команд и способности распараллеливать код на десятки мелких гранул.

Одновременное выполнение до 20 подпрограмм SGENV в программе перемножения матриц достигается аппаратно, и позволяет компенсировать существенный недостаток потоковых процессоров, заключающийся в низкой производительности при выполнении цепочек из последовательных команд.

Моделирование системы из 2 – 4 ядер ВПП с общей памятью подтвердило возможность достижения близкого к линейному ускорения.

Работа выполнена в МСЦ РАН в рамках Государственного задания. В исследованиях использовался суперкомпьютер МВС-10П.

Vector dataflow processor and shared-memory multiprocessor built on its base

N.I. Dikarev, B.M. Shabanov, A.S. Shmelev

Abstract: Dataflow processor can offer significantly higher performance in comparison with conventional processor due to decentralized control and searching for ready instructions in a much wider window. However, the well-known projects of dataflow processor failed to compete with von-Neumann processor because of array processing complexity and the need to perform associative search in large capacity memory. In this paper we show that the developing Vector Dataflow Processor (VDP) is able to significantly reduce capacity requirements of associative memory due to original method of storing arrays and to achieve higher performance of a single processor core. The results of matrix multiplication program simulation on a single core VDP and its acceleration on a shared memory multicore VDP are presented.

Keywords: vector processor; dataflow architecture; shared-memory multiprocessor; performance evaluation.

Литература

1. Н.И.Дикарев, Б.М.Шабанов, А.С.Шмелёв. Векторный потоковый процессор: оценка производительности. Известия ЮФУ. Технические науки, 2014, № 12, 36–46.
2. Dataflow Computers: Their History and Future. Wiley Encyclopedia of Computer Science and Engineering, edited by Benjamin War, 2008, John Wiley & Sons, Inc.
3. D.E.Culler and Arvind. Resource Requirements of Dataflow Programs. Proc. 15-th Ann. Symp. on Computer Architecture, 1988, 141–150.
4. J.Gurd et.al. Performance Issues in Dataflow Machines. Future generations computer systems, 1987, vol. 3, № 4, 285-297.
5. Л.Гервич, Б.Я.Штейнберг, М.Юрушкин. Программирование экзафлопсных систем. Открытые системы, № 8, 2013, 26-29.
6. J.F.Hake and W.Homberg. The Impact of Memory Organization on the Performance of Matrix Calculations. Parallel Computing, 1991, V. 17, № 2/3, 311–327.
7. Н.И.Дикарев, Б.М.Шабанов, А.С.Шмелёв. Выбор оптимальной производительности ядра векторного потокового процессора. «Суперкомпьютерные технологии (СКТ-2016) (19 - 24 сентября 2016 г.), Материалы 4-й Всероссийской научно-технической конференции: в 2 т.» – Ростов-на-Дону: изд-во ЮФУ, 2016, Т.1, 36-41.

Особенности современных сетей компьютерной памяти в суперкомпьютерных центрах

С.А.Лещев [1], О.С.Аладышев [2]

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail's: [1] sergey.leshchev@jscc.ru, [2] O.Aladyshev@jscc.ru

Аннотация: Системы хранения данных, применяемые в современных суперкомпьютерных центрах, имеют ряд особенностей, существенно отличающих их от использующихся в обычных и «облачных» центрах обработки данных. Причиной служат существенные отличия в характере решаемых задач, применение для высокопроизводительных вычислений свободного программного обеспечения и недорогого оборудования. Но есть и сходства, т.к. во всех центрах обработки данных решаются схожие задачи, такие как долгосрочное хранение и защита данных самих пользователей/клиентов и проектных/корпоративных данных, хранение программных образов операционных систем, баз данных и баз знаний, данных информационных систем. В статье описываются динамика развития систем хранения данных, прогресс изменения требований к ним и анализируются причины сложившихся особенностей систем хранения данных для суперкомпьютерных центров обработки данных.

Ключевые слова: сеть хранения данных, система хранения данных, суперкомпьютер, внешняя память, параллельная файловая система, Lustre

1. Введение

Прогресс в развитии вычислительных технологий, используемых в центрах обработки данных, вынуждает нас к пересмотру требований, предъявляемых к системам хранения данных в суперкомпьютерных центрах (СКЦ).

Как и в обычных центрах обработки данных, в СКЦ необходимо обеспечивать общий функционал: авторизацию пользователей, организацию баз данных, информационных систем и баз знаний, защиту данных и систем, различных вспомогательных функций и др.

И для этого применяются схожие по характеру системы хранения данных. Основной же отличительной особенностью СКЦ является характер решаемых суперкомпьютерных задач и применяемые для этого программное и аппаратное обеспечение.

Таким образом, со временем, меняются особенности нагрузки на сети компьютерной памяти – системы хранения данных (СХД) СКЦ.

Целью данной работы является выработка подхода к созданию перспективной СХД СКЦ МСЦ РАН на основе результатов анализа существующих решений по построению систем хранения данных и характера операций ввода-вывода вычислительных задач.

2. Устройство современных сетей компьютерной памяти

Система хранения данных вычислительного центра обычно включает в себя устройства хранения данных, узлы ввода-вывода, вычислительные и вспомогательные узлы, объединённые сетью, и строится по иерархическому принципу. На первом уровне хранения информация размещается на быстрых дисках узлов хранения данных, например на SSD, имеющих относительно небольшую ёмкость, но при этом обладающих высоким быстродействием операций ввода-вывода. Быстрые диски предназначены для хранения т.н. "горячих данных". На втором уровне хранения информация размещается на не таких быстрых дисках, но более ёмких например на HDD, предназначенных для использования в качестве хранилища т.н. "тёплых данных", но не на столько часто. На третьем уровне данные располагаются на магнитных лентах, использующихся для долговременного хранения т.н. “холодных данных”, редко используемых, например архивов проектов и резервных копий, хранить которые на втором уровне становится невыгодно с экономической точки зрения.

Устройство хранения СХД может быть сервером с размещённой в корпусе и/или вне него дисковой подсистемой, специализированным дисковым массивом,

ленточной библиотекой или другим самоуправляемым устройством хранения данных.

В серверах и дисковых массивах используются как накопители на жестких магнитных дисках (HDD), так и накопители на твердотельных дисках (SSD), при этом ёмкость хранения универсального сервера ограничивается десятками терабайт, в то время как специализированные дисковые массивы могут масштабироваться до десятков петабайт [1]. Специализированные дисковые массивы характеризуются более высокой производительностью и более низким уровнем задержки операций чтения и записи данных, чем универсальные серверы. В режиме высоких нагрузок вероятность сбоя устройств хранения с HDD заметно возрастает [2]. Это обстоятельство вынуждает производить плановую замену дисков раз в 3-5 лет. Ленточные библиотеки могут масштабироваться до сотен петабайт дискового пространства [3], характеризуются крайне высоким уровнем задержек операций случайного чтения и записи, высокой производительностью упорядоченного чтения и записи данных, и 30-летним декларируемым временем хранения данных [4]. Ленточные библиотеки используются, как правило, для архивного хранения данных. Несмотря на высокую активность работ ([5], [6], [7] и [8]) по использованию кварцевых дисков в качестве материала для долговременного надежного хранения информации, новые технологии до сих пор не получили промышленной реализации и не представляют альтернативы магнитным лентам по плотности записи и надежности хранения.

В современных СКЦ, как для решения суперкомпьютерных, так и общих задач, используются коммуникационные сети с низким уровнем задержки, низкой латентностью, объединяющих вычислительное поле – память вычислительных узлов, такие как InfiniBand, OmniPath.

В то время как в обычных центрах обработки данных суперкомпьютерные задачи не решаются, и поэтому используются сети Ethernet, латентность которых существенно выше, а стоимость ниже.

Сети передачи данных, построенные на основе Fibre Channel (FC), Data Center Bridging Ethernet (DCB), InfiniBand или Omni-Path, служат для связи устройств хранения данных и вычислительных узлов (ВУ).

Программные средства управления СХД предназначены для управления всеми физическими элементами сети хранения данных, организации логического пространства хранения данных, автоматизации перемещения данных между различными логическими уровнями и устройствами хранения, управления уровнем качества доступа к данным (Quality of Service, QoS), а также архивированием и резервным копированием данных - система резервного копирования СКЦ предназначена для защиты от потери данных не только при выходе из строя физических элементов СХД, но в случае совершения ошибок пользователями СКЦ.

В зависимости от сценария взаимодействия элементов СХД между собой выделяют пять основных архитектур систем хранения данных [9]:

0) локальная - ввод-вывод производится на локальные диски вычислительных узлов;

1) централизованная – ввод-вывод производится на центральный файловые сервер, расположенный в транспортной сети вычислительного кластера;

2) распределённая – ответственность за разделение доступа к данных распределяется между узлами ввода-вывода (параллельная файловая система GPFS и др.);

3) разделённая - ответственность за разделение доступа к данных закрепляется за специальными узлами метаданных (параллельные файловые системы Lustre, CephFS и др.);

4) архитектура прямого блочного доступа – сеть хранения данных, в которой разделение доступа к данным производится на вычислительных узлах.

3. Функциональные и конструктивные различия между системами хранения данных суперкомпьютерного центра и обычного центра обработки данных

Любая потеря данных, хранимых на СХД в центре обработки данных (ДЦ) влечёт за собой утрату престижа и большие убытки для ДЦ, по причине того, что данные обычно являются ценными. Данные же, хранимые на СХД СКЦ, делятся на классы по их ценности и возможности их восстановления. Обычно в СКЦ преобладают данные, которые могут быть восстановлены из других источников или заново получены пересчётом. Ценность таких данных определяют затраты, которые необходимо понести для их восстановления

(затраты на процессорное временя, электроэнергию, амортизацию оборудования, оплату работы персонала СКЦ и пр.). В других случаях, потеря данных, хранимых на СХД в СКЦ, как и данных, хранимых на СХД ДЦ, недопустима. Поэтому с точки зрения обеспечения отказоустойчивости и надёжности хранения данных СХД ДЦ и СКЦ не имеют больших различий.

С первой половины 2000-х годов наметились тенденции к уменьшению размеров вычислительного узла (ВУ) и, соответственно, увеличению плотности размещения ВУ в единице площади. Естественным шагом на этом пути миниатюризации ВУ высокопроизводительной вычислительной стало появление blade-систем и вынос за периметр ВУ, помимо систем питания и охлаждения, и устройств внешней памяти. В результате для СКЦ стали более востребованы внешние системы хранения (ВСХД) [9]. Следует отметить, что современные ДЦ, построенные на конвергентных решениях, таких как Cisco UCS или HPE ConvergedSystem, рассчитанные на виртуализацию ресурсов и предоставление облачных сервисов, становятся архитектурно похожи на СКЦ, но всё же отличаются большим отношением объёма оперативной памяти ВУ к количеству вычислительных ядер, меньшей плотностью упаковки вычислительных узлов и использованием высокоскоростных низколатентных коммуникационных сетей с гарантированной доставкой (как правило DCB или InfiniBand) в качестве сетей передачи данных.

Нагрузки, испытываемые СХД ДЦ, создаются в основном процессами обработки транзакций SQL серверов и поддержки принятия решений, приложениями баз данных, веб- и почтовыми серверами, виртуальными рабочими станциями и пр. Для профилей таких нагрузок характерны операции случайного чтения и записи блоками небольшого размера.

Для СХД СКЦ характерен ряд специфических нагрузок, заметно отличающихся от типовых нагрузок на СХД ДЦ [9]. К таким нагрузкам относят в первую очередь запись пользовательских контрольных точек, для которых характерны массированная параллельная запись группой ВУ в несколько (как правило по одному на процессор) файлов, либо в один файл (посредством MPI/IO), блоков данных большого размера. При этом, выполнение вычислительного задания группой узлов возобновляется только после окончания записи всеми узлами группы. Специфическую нагрузку с похожим на запись контрольной точки характером создаёт и ряд научных задач, порождающих крупные объемы результирующих данных – задачи биоинформатики, геномики, и пр. Ряд научных задач по обработке крупных массивов данных: моделирование в целях разведки природных ископаемых, атмосферных процессов, океанических течений, и пр. характеризуется массированным параллельным чтением группой ВУ одного или нескольких файлов из проектной директории большими по размеру блоками.

Следует отметить, что для СХД ДЦ активно применяются механизмы блочной дедупликации, кэширования на быстрых дисках, автоматической миграции данных (autotiering), создания моментальных снимков файловой системы (snapshotting), дублирования данных на территориально распределённых площадках, и прочие.

Пользовательские контрольные точки (КТ) и результаты вычислительных экспериментов как правило записываются и хранятся последовательными сериями, и редко бывают востребованы, что позволяет записывать «горячие» данные на быстрые диски и позднее автоматически мигрировать их на более «холодные» уровни. Система управления вычислительными заданиями (СУЗ) может оптимизировать процесс миграции, предугадывая характер операций ввода-вывода, выполняющихся и находящихся в очереди заданий, для чего в паспорте заданий можно указать место расположения данных, характер и время работы с ними.

Во время эксплуатации СХД СКЦ МСЦ РАН установлено:

1) что последовательные серии КТ и результатов вычислительных экспериментов могут содержать большое количество дублирующих блоков данных;

2) для разделов хранения данных вычислительных задач, обрабатывающих большие объемы данных, использование блочной дедупликации оказывается неэффективно;

3) для раздела хранения данных профилей пользователей не характерны массированные операции ввода-вывода.

Опираясь на результаты анализов состава оборудования, используемого в современных ДЦ, приведённых в работах [10], [11] и [12], произведённых в 2012, 2014 и 2017 годах, можно отметить, что для СХД ДЦ характерно использование СХД архитектур 1-го и 4-го типов классификации,

разработанной Аладышевым О.С. [9]. В современных ДЦ (например, Selectel) стали использоваться системы 3-го типа (CephFS) и даже 2-го.

Анализ СХД первого десятка высокопроизводительных систем из списка TOP500 по состоянию на 2014 год, произведенный в [13] и [14], а также информация из текущей версии Top500 показывает неуклонный рост объемов хранения – с 0,16 PB в 2001 году, до 1,6 PB в 2006 г., 20 PB в 2011 г., 55 PB в 2014 г., а также рост производительности файловой системы – со 100 GB/s в 2006 году до 500 GB/s в 2011 г. и 965 GB/s в 2014 г.

Таким образом с 2001 года по настоящее время для СХД крупнейших СКЦ мы можем наблюдать следующие тенденции - увеличение объемов хранения в среднем в 10 раз за 5 лет, замедление роста производительности файловых систем в среднем с пятикратного до двухкратного за 5 лет. Фактически, место в TOP10 делят параллельные файловые системы Lustre (разделенная архитектура), увеличившая свою долю с 0 до 80%, и GPFS (распределенная архитектура), причём на долю СХД с GPFS приходятся только суперкомпьютеры с архитектурой BlueGene /Q, также разработанной IBM.

Опираясь на функциональные различия между СХД ДЦ и СКЦ, специфические нагрузки, характерные для СХД СКЦ, а также подходы к построению СХД десяти наиболее производительных суперкомпьютерных систем мира за последние 15 лет можно сделать вывод, что для СХД СКЦ устоявшейся практикой является использование параллельной файловой системы Lustre с архитектурой 3-го типа.

4. Современные подходы к построению СХД СКЦ

В настоящий момент прослеживается два основных подхода к построению систем хранения данных в СКЦ.

Для небольших кластерных вычислительных установок используют файловые серверы (NAS) под управлением свободного программного обеспечения, либо коммерческие системы хранения уровня предприятия (enterprise level) низкого ценового сегмента.

При сопоставимой стоимости более предпочтительной является коммерческая система, как более надёжная и производительная. По мере роста размеров вычислительного кластера приоритеты смещаются от низкого ценового сегмента в средний, и далее к использованию параллельных файловых систем.

По мере роста плотности размещения ВУ, количества ядер на ВУ и соответственно объема ОЗУ, производительности имеющейся сети на базе Ethernet, стало недостаточно.

В то же время, в связи с миниатюризацией, не стало пространства внутри ВУ для установки высокопроизводительного интерфейса ввода-вывода, что обусловило использование интерфейса сети межпроцессорного обмена для обеспечения операций ввода-вывода.

Такой подход оправдан в случае, когда вычислительные процессы и процессы ввода-вывода на вычислительных узлах разделены по времени.

Для СКЦ характерно поэтапное развитие от опытной установки до полноценного суперкомпьютера.

Для обеспечения этого обычно выбирают СХД с модульной архитектурой либо в процессе модернизации данные переносятся со старого на вновь установленное оборудование. СХД на основе Lustre состоит из блоков хранилищ объектов и сервера метаданных, которые соединены сетью передачи данных (Ethernet, InfiniBand, Omni-Path, и пр.).

С помощью организации LNet router можно организовать доступ к данным из другой транспортной сети.

Масштабируемый блок хранилища объектов представляет собой два блочных дисковых массива уровня предприятия, соединённых с двумя серверами хранения данных таким образом, что каждый из двух модулей ввода-вывода каждого дискового массива соединяется напрямую с каждым из двух серверов высокоскоростной сетью.

Предназначение масштабируемого блока – хранение данных, поэтому дисковые массивы обычно состоят из SATA дисков большой емкости.

В Lustre блок хранилища метаданных не масштабируется, но в некоторых других системах с архитектурой 3-го типа, таких как PANASAS, блоки метаданных и данных могут быть линейно добавлены.

У всех крупных современных производителей оборудования: блочных дисковых массивов; низколатентных сетей; серверов, как правило существуют типовые архитектуры ([15], [16]) для реализации масштабируемых блоков СХД на основе Lustre, что облегчает её планирование и развёртывание.

5. Заключение

Проведенное исследование показывает, что построение перспективной системы хранения данных СКЦ МСЦ РАН дополнительно должно основываться на следующих подходах:

1) СХД должна быть рассчитана на массированную параллельную запись и чтение группой узлов одного или многих файлов в единой директории;

2) СХД должна состоять из нескольких различных разделов хранения данных, отличающихся своим функционалом работы с данными (блочной дедупликацией, выполняющейся в режиме по требованию (on-demand) или в реальном времени (inline), др.) в зависимости от класса хранимых данных;

3) для раздела хранения данных профилей пользователей не требуется система хранения высокой производительности и может использоваться централизованная архитектура, в т.ч. коммерческая система хранения уровня предприятия низкого ценового сегмента;

4) СХД должна быть интегрирована с системой управления вычислительными заданиями для повышения эффективности ввода-вывода путем автоматизированного перемещения данных между уровнями хранения и эффективности хранения путём блочной дедупликации;

5) самой популярной параллельной файловой системой, реализующей распределённую архитектуру, и предпочтительной для построения СХД СКЦ является Lustre;

6) СХД должна строится на основе модульной архитектуры, позволяющей наращивать ее производительность путем добавления новых блоков хранения данных и метаданных.

Оптимизация ввода-вывода данных путем автоматизации их перемещения между уровнями хранения данных в СХД с использованием современных систем управления заданиями вычислительного кластера (СУППЗ, Slurm, PBS, Moab и др.), выбор предпочтительных файловых систем для реализации разделов хранения и модернизация системы сбора статистики производительности существующей СХД МСЦ РАН для интеграции с используемыми СУЗ вычислительного кластера являются темами для дальнейших исследований и разработок.

Работа выполнена в МСЦ РАН в рамках Государственного задания. В исследованиях использовался суперкомпьютер МВС-10П.

Features of a Network of Data Storages for Supercomputer Center

S.A.Leshchev, O.S.Aladyshev

Abstract: Specific features of data storage systems in supercomputer centers are significantly different from ones in conventional and "cloud" data centers. The cause is in essential difference in the solved problems, usage of free software and cost of DSS. But there are similarities. Both data and supercomputer centers solve similar auxiliary tasks. It is necessary to store and protect user/client and project/corporate data for a long period, and to deal with problems of storing disks of virtual machines, databases and knowledge bases. The article demonstrates the evolution of data storage systems, the progress of requirements change for them, and analyzes the emerge reasons of data storage systems specific features in supercomputer centers.

Kewords: storage network, data storage system, supercomputer, auxiliary memory, clustered file system, Lustre.

Литература

[1] «FAS Hybrid Flash Arrays», 09 10 2017. [В Интернете]. Available: http :// www.netapp.com/us/ products/storage-systems/hybrid-flash-array/index.aspx. [Дата обращения: 09 10 2017].

[2] E. Pinheiro, W.-D. Weber и L. A. Barroso. «5th USENIX Conference on File and Storage Technologies (FAST'07)», в *Failure Trends in a Large Disk Drive Population*, San Jose, 2007.

[3] «IBM TS4500 Tape Library», [В Интернете]. Available: https://www.ibm.com/usen/ marketplace/ts4500/details. [Дата обращения: 09 10 2017].

[4] «HPE LTO Ultrium Storage Supplies», [В Интернете]. Available: https://www.hpe.com/h20195/v2/getpdf.aspx/c04154430.pdf. [Дата обращения: 09 10 2017].

[5] «Лаборатория лазерного наноструктурирования стекла», Фонд перспективных исследований, [В Интернете]. Available: http://fpi.gov.ru/activities/lab/rxtu. [Дата обращения: 12 01 2017].

[6] E. Kee. «Hitachi claims to have "forever" data solution», 24 09 2012. [В Интернете]. Available: http://www.ubergizmo.com/2012/09/hitachi-claims-to-have-forever-data-solution/. [Дата обращения: 12 01 2017].

[7] J. Zhang, M. Gecevičius, M. Beresna и P. G. Kazansky. «5D data storage by ultrafast laser nano-structuring in glass», *Conf. on Lasers and Electro-Optics (CLEO)*, San Jose, US, 09 - 14 May 2013.

[8] Zhang Jingyu, Cerkauskaite Ausra, Drevinskas Rokas, Patel,Aabid, Beresna Martynas and Kazansky Peter G. «Eternal 5D data storage by ultrafast laser writing in glass,» в *SPIE Photonics West: Laser-based Micro- and Nanoprocessing X*, San Francisco, US, 13 - 18 Feb 2016.

[9] А. О.С., «Летняя суперкомпьютерная академия МГУ» Москва, 2012.

[10] «Функциональное и экономическое сравнение российских операторов предоставляющих облачные услуги», 07 03 2012. [В Интернете]. Available: https://habrahabr.ru/post/139511/. [Дата обращения: 10 10 2017].

[11] «Функциональное и экономическое сравнение российских операторов предоставляющих облачные услуги. Версия 2.0» 01 07 2014. [В Интернете]. Available: https://habrahabr.ru/post/228157/. [Дата обращения: 10 10 2017].

[12] «Сравнение российских операторов предоставляющих облачные услуги», 28 08 2017. [В Интернете]. Available: https://habrahabr.ru/post/334044/. [Дата обращения: 10 10 2017].

[13] Е. О. Тютляева и А. А. Московский. «Анализ основных тенденций в области хранения данных», *Информационные технологии и вычислительные системы,* т. 2, pp. 64-75, 2012.

[14] Е. О. Тютляева и М. М. Тютляев. «Системы хранения данных лидирующих суперкомпьютеров» *СуперКомпьютеры,* № 4 (20), pp. 31-34, зима 2014.

[15] N. Robert Lai «TR-3997 NetApp High-Performance Computing Solution for Lustre: Solution Guide,» 08 2012. [В Интернете]. Available: https://www.netapp.com/us/media/tr-3997.pdf. [Дата обращения: 10 10 2017].

[16] «Implementing Storage in Intel Omni-Path Architecture Fabrics» [В Интернете]. Available: https://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-storage-white-paper.html. [Дата обращения: 11 10 2017].

Производительность современных вычислительных платформ в расчетах молекулярной динамики белок-мембранных систем

Д.Е.Нольде[1], Н.А.Крылов[2], П.Н.Телегин[3], Р.Г.Ефремов[4], Б.М.Шабанов[5]

[1,4] *ФГБУН Институт биоорганической химии им. академиков М.М. Шемякина и Ю.А. Овчинникова РАН,*

[2,3,5] *ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,*

E-mail's : [1] *nolde@nmr.ru,* [2] *krylovna@gmail.com,* [4] *efremov@nmr.ru,* [3,5] *antbar@mail.ru*

Аннотация: На примере программного пакета Gromacs проведено исследование скорости расчета классической молекулярной динамики на различных компьютерных системах: настольных компьютерах, кластеров на основе процессоров x86_64, многоядерных процессоров архитектуры MIC, а также гетерогенных систем с использованием «игровых» видеокарт или графических ускорителей. Рассмотрен вопрос выбора оптимальной платформы для проведения расчетов молекулярной динамики.

Ключевые слова: молекулярная динамика, высокопроизводительные кластеры, графические ускорители, технология Cuda, процессоры Xeon Phi

1. Введение

Молекулярное моделирование полимеров и биополимеров систем является в настоящее время мощным теоретическим инструментом исследования молекулярных систем, вносящим вносить существенный вклад в развитие физико-химической биологии и биотехнологии. Наряду с квантово-механическими расчетами, которые позволяют получать результаты только для систем небольшого размера (обычно до сотен атомов), метод молекулярной динамики (МД) является основным инструментом атомистического моделирования биомолекулярных систем. Суть данного метода заключается в решении классических уравнений движения Ньютона. В современных задачах с помощью МД изучают молекулярные системы размером 104-107 атомов на интервалах времени от 100 нс до 10 мкс при шаге интегрирования 1-2 фс. Наиболее информативной величиной, которую можно получить в результате таких расчетов, является свободная энергия образования биомолекулярного комплекса (белок-лиганд, белок-белок, белок-мембрана и др.). Однако для решения задачи в каждом случае требуется расчет нескольких десятков траекторий МД и, следовательно, необходимы огромные вычислительные ресурсы. Важно, что для данного класса задач биологически значимая информация может быть получена лишь по мере накопления некой «критической» массы данных МД, что делает особенно актуальной проблему повышения скорости расчета МД, как за счет повышения производительности вычислительных систем, так и за счет оптимизации программ и параметров запуска МД.

Основное процессорное время при решении уравнений Ньютона затрачивается на расчет сил парных взаимодействий атомов системы: электростатических и ван-дер-ваальсовых.

Для уменьшения времени расчета сил невалентных взаимодействий (при сохранении необходимой точности) применяется следующий метод: электростатические и ван-дер-ваальсовы взаимодействия на расстояниях меньше r_{cut} (обычно, 12-15 □) считаются напрямую, а на расстояниях, больших r_{cut}, считаются только электростатические взаимодействия с помощью метода Эвальда [1] (суммирование в обратном пространстве Фурье).

Характерная плотность атомов в ячейке МД составляет ~100 шт.*нм-3, и напрямую считается взаимодействие каждого атома с 700-1400 соседями.

В результате прямые расчеты невалентных взаимодействий занимают около 50% процессорного времени, а расчет электростатических взаимодействий с помощью метода Эвальда — 25%.

2. Программа Gromacs

В настоящее время программный пакет Gromacs [2,3] □ одна из самых эффективных программ расчета МД. Начиная с версии 5.0 (2014 год) в пакете используется многоуровневая параллелизация: разбивка системы на домены и параллельный расчет с помощью MPI; параллельный расчет внутри одного MPI-процесса с помощью директив OpenMP; векторизация вычисления невалентных взаимодействий с использованием расширенного набора инструкций процессора (SSE 4.2, AVX-128, AVX2-256, AVX-512).

На сегодняшний момент вычислительное ядро Gromacs адаптировано и оптимизировано для множества микроархитектур, в частности, поддерживаются современные расширения наборов команд x86_64, реализована возможность расчета сил парных взаимодействий на базе графических процессоров (технологии CUDA и OpenCL). Реализована поддержка процессоров Intel Xeon Phi в нативном (native) режиме (KNC с версии 5.0, KNL с версии 2016). В некоторых версиях поддерживалась работа с процессорами Intel Xeon Phi архитектуры KNC в оффлоад (offload) режиме, но из-за проблем с быстродействием начиная с версии 2016 г., от этого режима отказались.

3. Методика измерения

В настоящей работе мы исследовали производительность программного пакета Gromacs (версии 2016.3) на различных вычислительных системах, от персональных компьютеров до суперкомпьютерных кластеров. В качестве критерия производительности использовали скорость расчета траектории молекулярной динамики в нс/сутки для одной системы. Для тестирования применяли уравновешенную систему белок TRPV1 в гидратированном липидном бислое [4]. Размер системы (~330 тыс. атомов) является средним, по современным меркам. Измерения времени расчета проводились на участке траектории длиной 25000 шагов (50 пкс) через 5000 шагов после старта. Для каждого запуска проводили 5 измерений с усреднением результата по трем измерениям после отбрасывания крайних значений.

4. Процессоры x86_64

В Таблице 1 приведена скорость расчета МД на одиночных компьютерах архитектуры x86_64. Как и ожидалось, скорость расчета возрастает как с увеличением числа ядер, так и с ростом частоты процессора. При этом использование процессоров с большим числом ядер выгоднее, чем с большей частотой (например, i7-5930 выгоднее i7-6700k). Обращает на себя внимание существенный рост производительности при переходе от процессора Xeon E5-2667v2 к v3, что связано с поддержкой набора команд AVX. Также следует отметить относительно невысокую производительность на процессорах AMD Opteron семейства Piledriver.

Процессор	Частота (ГГц)	Число процессоров (ядер)	Скорость счета (нс/сутки)
i7-6700k	4	1 (4)	1,2
i7-5930k	3,5	1 (6)	1,5
Xeon E5-2667v2	3,3	2 (16)	2,4
Xeon E5-2667v3	3,2	2 (16)	3,3
Opteron 6378	2,4	4 (64)	2.9
Xeon E5-2697v3	2,6	2 (28)	4,2
Xeon E5-2697Av4	2,6	2 (32)	6,2

Таблица 1. Скорость расчета МД программой Gromacs 2016.3 на одиночных компьютерах.

Стандартный способ увеличения скорости расчета □ использование кластерных систем. Следует заметить, что при использовании метода Эвальда для расчета электростатических взаимодействий, необходимо использовать кластерные системы с низколатентной сетью для межпроцессорного обмена (например, Infiniband).

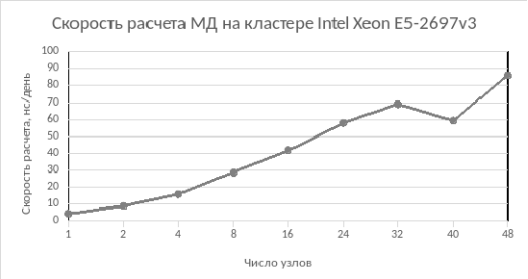


Рис. 1. Скорость расчета МД на кластере, состоящем из 2-процессорных узлов на базе ЦПУ Intel Xeon E5-2697v3, соединенных низколатентной сетью Infiniband, в зависимости от числа узлов.

Как видно из Рис. 1, расчет МД рассматриваемой системы хорошо масштабируется на кластерах до 48 узлов (1344 ядра), как минимум. В редких случаях наблюдаются небольшие “провалы” в производительности, как в случае с числом узлов, равным 40.

5. Графические процессоры

Наряду с использованием кластерных вычислительных систем для повышения скорости расчета МД использовали графические процессоры. При этом на них выполняется расчет невалентных взаимодействий, занимающий большую часть процессорного времени. Изначально использовали специализированные графические ускорители (например Nvidia Tesla), но в последние годы также активно используют и более дешевые “игровые” видеокарты, несмотря на то, что у них отсутствует коррекция ошибок памяти, что может приводить к более частым остановкам расчетов МД.

Процессор	Видеокарта	Скорость счета (нс/сутки) с числом видеокарт			Прирост (одна/две видеокарты) в %
		0	1	2	
i7-6700k	GTX 980	1,2	5,1	-	330/-
i7-5930k	GTX 980	1,5	7	-	370/-
Xeon E5-2667v2	GTX 1080	2,4	8,6	-	260/-
Xeon E5-2667v3	GTX 1080	3,3	9,5	-	190/-
Xeon E5-2697v3	Tesla K40X	4,2	-	8,5	-/100
Xeon E5-2697Av4	GTX 1080	6,2	13,8	16,8	123/170

Таблица 2. Скорость расчета МД программой Gromacs 2016.3 на одиночных компьютерах с использованием видеокарт Nvidia.

В таблице 2 представлены результаты тестирования производительности систем из табл. 1 с видеокартами. Учитывая, что стоимость одной видеокарты составляет от 15% (для серверов) до 30% (для настольных компьютеров) от стоимости системы, полученный прирост производительности является очень хорошим результатом.

6. Архитектура MIC

Первоначально планировалось, что процессоры архитектуры MIC (Intel Xeon Phi) будут работать в двух режимах: процессоры используются совместно с ЦПУ в режиме ускорителей (как видеокарты) или расчет выполняется только на процессорах Xeon Phi. К сожалению, в первом случае не удалось достичь приемлемого быстродействия в задачах МД и не только. Косвенно, проблему с быстродействием в режиме сопроцессора признала компания Intel: процессоры нового поколения KNL уже не позиционируются как сопроцессоры (ускорители).

Сравнение скорости расчета на кластерах, собранных на процессорах Xeon Phi двух поколений, приведены на Рис. 2. Как видно из Рис. 2, быстродействие процессоров поколения KNC низкое и они не могут использоваться в реальных задачах. Также видно, что эффективное распараллеливание наблюдается только в диапазоне до 6-8 процессоров на задачу, по-видимому, из-за большого числа ядер в процессоре.

Так в случае процессора KNL, на один процессор приходилось 72 MPI-процессов по 4 OpenMP “нитки” на процесс. Стоит отметить, что для процессоров KNL максимум производительности наблюдается при 432 ядер на задачу, для KNC — при 480, а на кластерах с обычными процессорами производительность росла в диапазоне до 1344 ядер на задачу. Таким образом, можно ожидать, что в ходе оптимизации кода можно повысить эффективность использования процессоров Intel Xeon Phi KNL. В настоящее время их применение для расчета МД ограниченно.

Для задач среднего размера неэффективно применение большого числа процессоров (>6), а система, построенная на шести процессорах Xeon Phi по быстродействию примерно соответствует более дешевой системе, построенной на двух процессорах Intel Xeon E5-2697Av4 и двух видеокартах Nvidia GTX 1080.

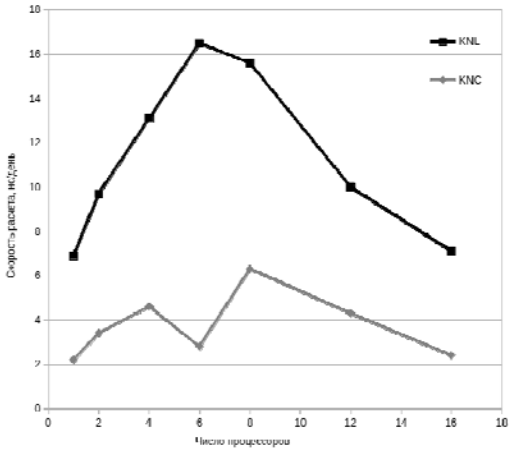


Рис. 2. Скорость расчета МД на процессорах Intel Xeon Phi 7120D (KNC) и 7290 (KNL) в зависимости от числа процессоров.

7. Выводы

Выбор оптимальной конфигурации компьютера для МД представляет нетривиальную задачу.

Для решения задач небольшой размерности можно использовать системы относительно низкого ценового уровня, такие как настольный компьютер с производительным процессором последнего или предпоследнего поколений (6 и более ядер, частота не менее 3,5 ГГц, поддержка AVX2) и “игровой” видеокартой (например, GTX 10 серии). На таких компьютерах для систем среднего размера можно получать относительно короткие траектории (длиной несколько десятков нс). Использование двухпроцессорных серверов с двумя «игровыми» видеокартами позволяет увеличить скорость расчета в 3-4 раза, но все равно это не позволяет надеяться на получение траекторий длительностью хотя бы 1 мкс.

Использование вместо «игровых» видеокарт специализированных графических ускорителей ведет к удорожанию системы при заметном отсутствии выигрыша в производительности.

Для расчета микросекундных траекторий в настоящее время необходимо использовать большие кластерные системы, доступные в суперкомпьютерных центрах.

В настоящее время предпочтительно использование кластеров на двухпроцессорных платформах с серверными процессорами Intel Xeon (например, 16-ядерный E5-2697Av4).

При этом рост производительности таких систем в задачах МД ограничен лишь числом узлов кластера.

Результаты тестирования производительности процессоров архитектуры MIC (Xeon Phi) показывают, что их потенциал пока еще не ипользуется в полной мере и требуется дополнительная оптимизации кода программы.

Работа выполнена в МСЦ РАН при поддержке программы Президиума РАН I.33П «Фундаментальные проблемы математического моделирования. Фундаментальные проблемы факторизационных методов в различных областях. Алгоритмы и математическое обеспечение для вычислительных систем сверхвысокой проводимости».

В расчетах использовался суперкомпьютер МВС-10П.

The performance of modern computing platforms in molecular dynamics simulation of protein-membrane systems

D.E.Nolde, N.A.Krylov, P.N.Telegin, R.G.Efremov, B.M. Shabanov

Abstract: The performance of molecular dynamics software package Gromacs was measured on various hardware: desktop computers, clusters based on x84_64 processors or many integrated core processors, and heterogeneous system with gaming graphic cards or general purpose GPU systems. The optimal choice of hardware for molecular dynamics simulations is discussed.

Keywords: molecular dynamics, high-performance computing, GPU, Cuda, MIC, Xeon Phi

Литература

1. U.Essmann, L.Perera, M.L.Berkowitz. A smooth particle mesh Ewald method. «The Journal of Chemical Physics», vol. 103 (1995), pp. 8577-8593.
2. B.Hess, C.Kutzner, D.van der Spoel, E.Lindahl. GROMACS 4:□ Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. «J. Chem. Theory Comput.», vol. 4 (2008), № 3, pp. 435–447.
3. M.J.Abraham, T. Murtola, R.Schulz, S.Páll, J.C.Smith, B.Hess, E.Lindahl. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. «SoftwareX», vol. 1-2 (2015), pp. 19-25.
4. A.O.Chugunov, P.E.Volynsky, N.A.Krylov, D.E.Nolde & R.G.Efremov. Temperature-sensitive gating of TRPV1 channel as probed by atomistic simulations of its trans- and juxta-membrane domains. «Scientific Reports», vol. 6 (2016), 33112, http://doi.org/10.1038/srep33112

О реализации российского сегмента Eduroam

Д.В.Вершинин, Ю.Н.Морин, А.П.Овсянников, Б.М.Шабанов

ФГУ ФНЦ НИИСИ РАН, Межведомственный суперкомпьютерный центр, Москва, Россия,

E-mail: ovsyanikov@jscc.ru

Аннотация: Статья рассматривает вопросы организации российского сегмента Eduroam. Описывается реализация федеративного контроля доступа для научных сетей на основе технологии Eduroam. Технология предоставляет пользователям возможность безопасной аутентификации для доступа к сети и использования сетевых ресурсов в любой организации, поддерживающей Eduroam, с использованием единственного набора учетных данных. Рассматриваются особенности организации российского сегмента Eduroam на базе отраслевых научно-образовательных телекоммуникационных сетей RASNet и RUNNet, эксплуатируемых МСЦ РАН и ФГАУ ГНИИ ИТТ «Информика». Описаны реализованные технические решения, обеспечивающие взаимодействие регистраторов и развитие российского сегмента Eduroam.

Ключевые слова: eduroam, аутентификация, сети науки и образования, RADIUS, беспроводные сети, wifi.

1. Введение

Сегодня научные исследования и образование становятся все более глобальными, международное сотрудничество работает во всех областях исследований. Исследователи и студенты, научные коллективы со всего мира кооперируются в проведении научных исследований. Исследования и образование больше не ограничиваются классом или лабораторией – исследователи и студенты ожидают, что смогут получить доступ в Интернет и работать там, где и когда им нужно. Eduroam помогает обеспечить мобильность исследователям и студентам, предоставляя возможность получать доступ к бесплатным услугам Wi-Fi в научных организациях и учебных заведениях в 89 странах [1], включая большинство европейских стран, страны Юго-Восточной Азии, Северную и Южную Америки, Австралию, и ряд стран Африки (см. рисунок 1).

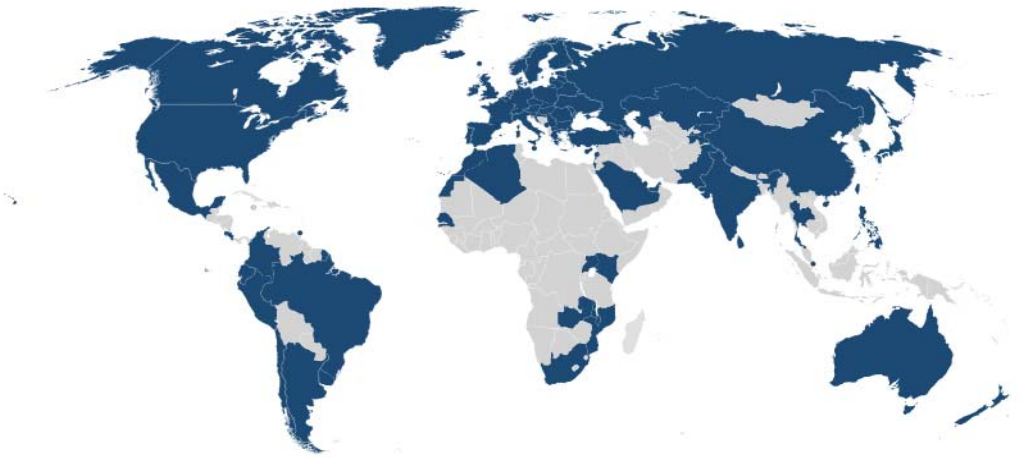


Рис. 1. Страны, поддерживающие сервис Eduroam

Суть сервиса Eduroam состоит в том, чтобы обеспечить моментальный доступ к сети с помощью мобильного устройства сотрудника или студента с одними и теми же настройками сети и учетными данными в любой организации, поддерживающей сер-

вис. Организация, предоставляющая доступ – провайдер сервиса (SP - Service Provider) - обращается для проверки учетных данных к институту/университету, в котором работает/учится пользователь – провайдеру идентификации (IdP – Identity Provider). Таким образом, реализуется технология единого входа для беспроводных сетей научных и образовательных организаций – Eduroam (Educational Roaming).

Сервис Eduroam был реализован в России в 2011 году [2,3] Межведомственным суперкомпьютерным центром Российской академии наук. На момент написания статьи сервис использовали 19 организаций науки и образования в России.

2. Технология Eduroam

Технически сервис Eduroam заключается в аутентификации сотрудника сервером аутентификации его домашнего института в сети любой организации, поддерживающей сервис.

Сервис реализуется в сетях с контролем доступа к портам 802.1X [4] с использованием системы аутентификации на основе иерархической системы прокси-серверов RADIUS. Для аутентификации пользователей используется Расширяемый Протокол Аутентификации (Extensible Authentication Protocol - EAP) [5] . Аутентификационные данные передаются по локальной сети (EAP over LAN - EAPoL) и инкапсулированными в протокол RADIUS [6]. При подключении к сети мобильное устройство обращается к устройству контроля доступа, направляя через него инкапсулированный запрос к серверу аутентификации (RADIUS). Собственно устройство контроля доступа лишь передает сообщения клиента к серверу аутентификации и предоставляет либо запрещает доступ к сети на основе его ответа. Протокол 802.1X позволяет использовать различные методы аутентификации на основе EAP, что позволяет каждому провайдеру идентификации самостоятельно выбрать наиболее удобный для себя метод, обеспечивающий взаимную аутентификацию клиента и сервера аутентификации: EAP-TLS, EAP-TTLS или EAP-PEAP. При использовании любого из методов, клиент может проверить полученный от сервера аутентификации сертификат с помощью предустановленной копии открытого ключа удостоверяющего центра и, возможно, списка отзыва сертификатов. Хотя в процессе аутентификации клиент и сервер аутентификации обмениваются зашифрованными сообщениями, после аутентификации клиента данные передаваемые по сети могут быть незашифрованными. Для шифрования трафика авторизованного клиента используется один из методов шифрования: WEP, WPA или WPA2. При этом 802.1X, вкупе с одним из методов аутентификации, может быть использован для распространения ключей, которыми будет шифроваться данные клиента.

Запрос к серверу аутентификации сервис-провайдера (SP - Service Provider), то есть института, предоставляющего доступа к своей сети, содержит шифрованную часть, включающую имя и пароль пользователя, и адресную часть - доменное имя провайдера идентификации (IdP - Identity Provider), то есть института (университета), в котором работает (учится) пользователь.

К серверу аутентификации этого института через иерархическую систему прокси-серверов RADIUS и будет направлен запрос на аутентификацию пользователя.

Верхний уровень иерархии прокси-серверов RADIUS составляют так называемые сервера верхнего уровня (TLRS - Top-Level RADIUS Server), на уровне страны - так называемые сервера RADIUS федеративного уровня (FLRS - Federation Level Radius Server), нижний уровень образуют сервера организаций: сервера авторизации сервис-провайдеров (SP) и провайдеров аутентификации (IdP).

TLRS содержит базу данных всех серверов RADIUS федеративного уровня (FLRS), каждый FLRS содержит базу всех серверов своего домена первого уровня. Передача данных между серверами RADIUS защищается шифрованием.

Имя пользователя в Eduroam состоит из двух частей: префикса, являющегося именем пользователя в его домашней организации, формат которого определяется этой организацией, и отделенного символом @ суффикса, называемого realm пользователя. Realm совпадает с доменным именем организации в терминах системы доменных имен DNS.

]Например, пользователь из МСЦ РАН может иметь имя user@jscc.ru. Маршрутизация запросов производится на основании realm пользователя (см. рис. 2). Так, в случае, если пользователь user@jscc.ru отправляет запрос, находясь за пределами Российской Федерации, он будет направлен сервис-провайдером (SP), к которому подключается пользователь, на FLRS страны SP, далее через серверы конфедерации TLRS (домен «.») на российский FLRS (домен «ru.»), и, наконец, на сервер идентификации (IdP) домашней организации пользователя (домен «jscc.ru.» - МСЦ РАН). В случае, если пользователь подключается к SP, принадлежаще-

му к той же федерации, что и IdP пользователя, запрос будет направлен через FLRS соответствующей федерации к IdP пользователя.

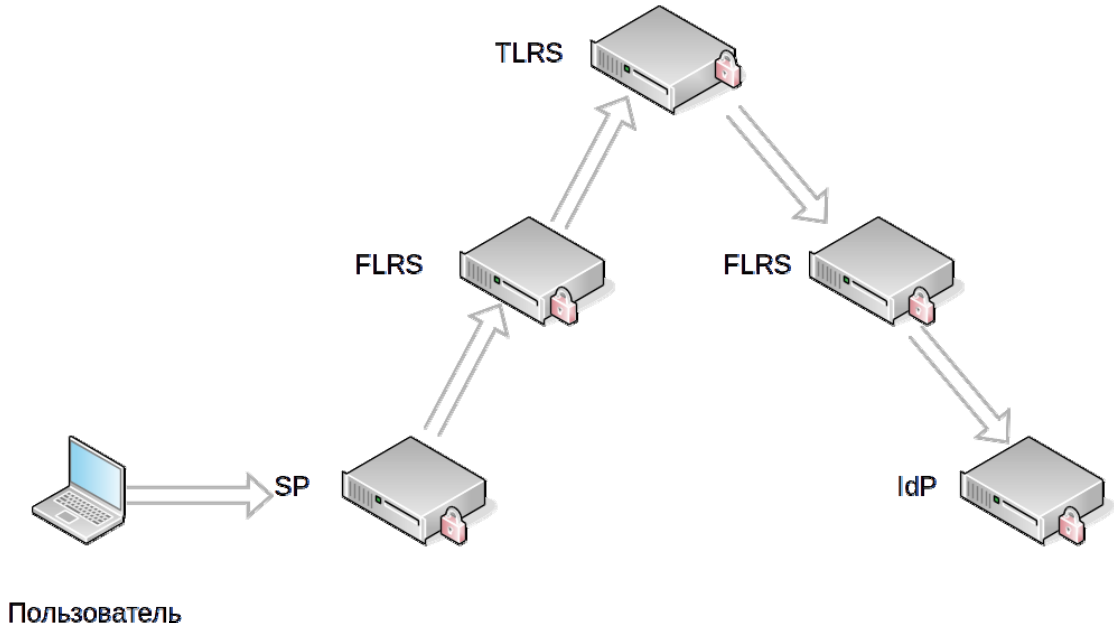


Рис. 2. Авторизация пользователя в Eduroam

На вершине национальной иерархии Eduroam находится RADIUS-сервер федерального уровня (FLRS), который содержит информацию обо всех подключенных организациях и их ролях. Для надежности используется резервирование FLRS (см. рисунок 3).

Институт может выступать как в роли провайдера сервиса так и в роли провайдера аутентификации (обычно эти роли совмещаются). Для присоединения к Eduroam организации необходимо передать оператору FLRS информацию о своих RADIUS-серверах и настроить их.

3. Организация российского сегмента Eduroam

В связи с большим числом организаций науки и образования в России, их разной ведомственной принадлежностью и источниками финансирования, для облегчения вхождения организаций в Eduroam и их технической поддержки российский сегмент Eduroam включает специальные организации – регистраторы [7], уполномоченные обеспечивать технические мероприятия и поддержку по подключению организаций науки и образования к Eduroam, а также управления группами доступа в национальной инфраструктуре [3].

В 2017 году МСЦ РАН и ФГАУ ГНИИ ИТТ «Информика» заключили соглашение о сотрудничестве в области создания и развития интегрированной инфраструктуры авторизации и аутентификации международных проектов Eduroam и EduGAIN на базе эксплуатируемых отраслевых научно-образовательных телекоммуникационных сетей RUNNet и RASNet.

Соглашение предусматривает организацию резервирования ключевых элементов создаваемой инфраструктуры; содействие в продвижении сервисов удостоверяющих систем сети Интернет в научных и образовательных организациях Российской Федерации; распространение лучших практик использования служб и сервисов удостоверяющих систем в научных и образовательных организациях; выработку методик внедрения элементов удостоверяющих систем в научных и образовательных организациях; взаимодействие с общественными организациями и экспертным сообществом в целях улучшения качества предоставления сервисов и расширения круга участников проектов.

Таким образом, с 2017 года в российском сегменте Eduroam функционируют два регистратора: МСЦ РАН и ФГАУ ГНИИ ИТТ «Информика», что позволило более точно адресовать Eduroam целевым аудиториям разных ведомств: организациям ФАНО России и высшим учебным заведениям.

Для реализации организационного взаимодействия регистраторов был разрабо-

тан ряд технических решений по резервированию инфраструктуры национального уровня (FLRS) и технического взаимодействия.

Маршрутизация запросов серверов аутентификации RADIUS организаций — участников Eduroam может производиться как через прокси-сервер RADIUS регистратора, так и напрямую через прокси-сервер национального уровня FLRS (см. рис. 3).

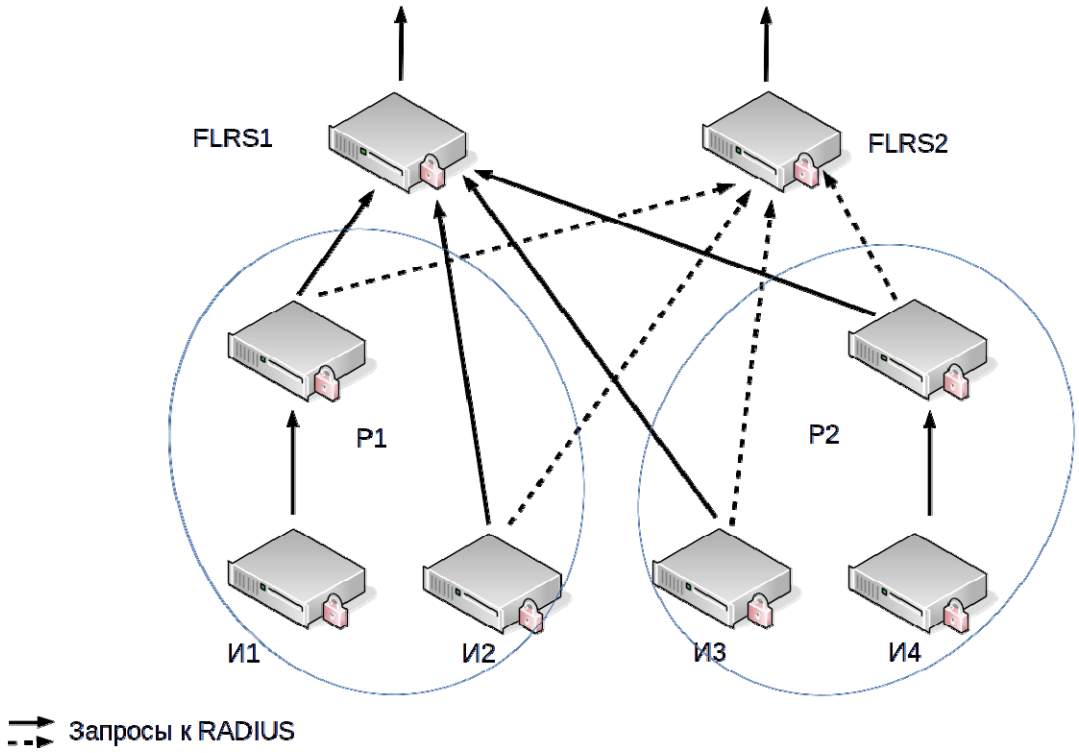


Рис. 3. Маршрутизация запросов институтов при использовании регистраторов

Институты И1 и И2, изображенные на рисунке 3, обслуживаются регистратором P1, институты И3 и И4 — регистратором P2. При этом запросы институтов И1 и И3 маршрутизируются через прокси-сервер регистратора, а И2 и И4 минуя регистратора маршрутизируются напрямую на FLRS.

На рисунке 3 изображены два FLRS – основной и резервный, запросы к резервному обозначены пунктиром, хотя на практике запросы через сервера могут балансироваться.

Маршрутизация через регистраторов не означает, что институты И1 и И3 используют домены третьего уровня, достаточно в настройках FLRS указать, что домены И1 и И3 маршрутизируются на P1 и P2 соответственно.

В настройках маршрутизации прокси-сервера RADIUS маршрутизация организации INSTITUTE описывается строками вида (freeRADIUS, [8]):

```
#  INSTITUTE servers
client INSTITUTE {
    host A.A.A.A
    type UDP
    fticksVISCOUNTRY RU
    secret XXXXXXXXXXXXXXXXX
}

server INSTITUTE {
    host A.A.A.A
    type UDP
    secret XXXXXXXXXXXXXXXXX
}

# INSTITUTE
realm /institute\.ru$ {
    server INSTITUTE
}
```

Первые фрагменты определяют сервера аутентификации сервис-провайдера и ключи шифрования при обмене информации с ними, последний предписывает запросы с realm «institute.ru» направлять на сервер, описанный во фрагменте «server INSTITUTE {}».

При подключении институтов регистратор должен внести изменения в базы данных (БД) корневых серверов (FLRS) (см. рис. 4).

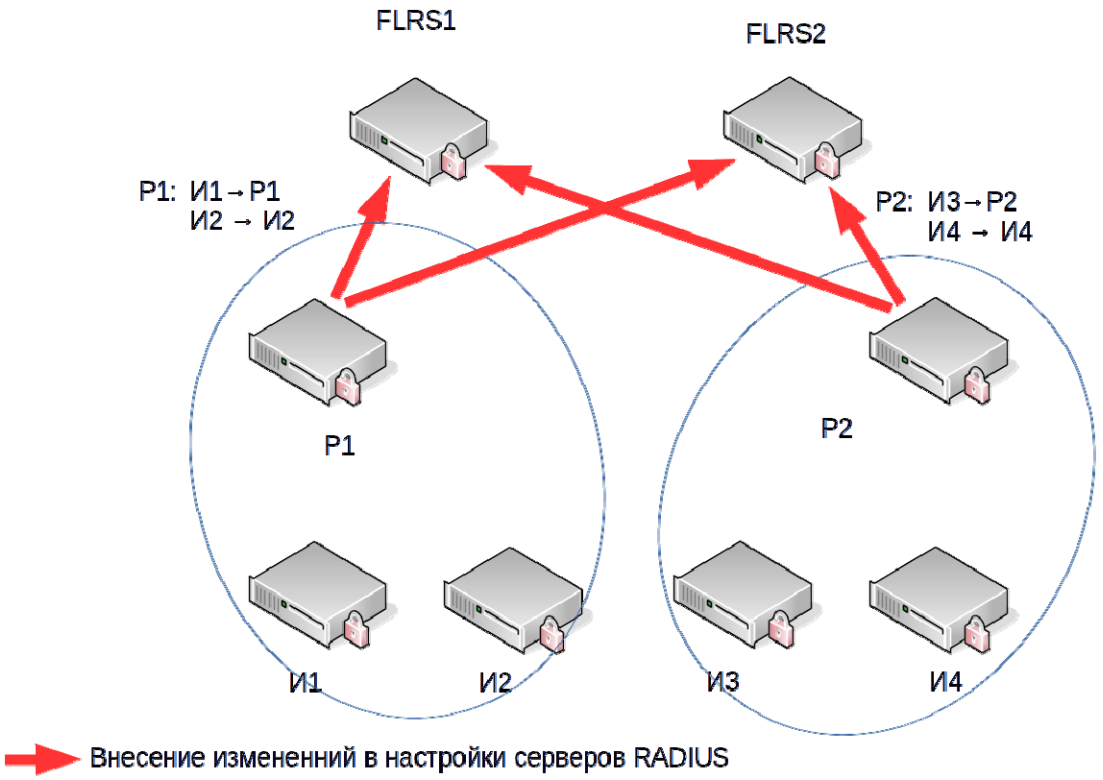


Рис. 4. Внесение изменений в настройки корневых прокси-серверов RADIUS национального уровня

P1 должен сообщить FLRS1 и FLRS2, что маршрут запросов к И1 идет через P1, а И2 – непосредственно к И2. Соответственно, P2 должен сообщить FLRS1 и FLRS2, что маршрут запросов к И3 идет через P2, а И4 – непосредственно к И4.

То есть в файлы настроек прокси-серверов FLRS, для организаций И1 и И3 должны быть внесены записи:

```
realm И1 { server P1 }
realm И3 { server P2 }
```

а для институтов И2 и И3 записи:

```
client И2 { … }
server И2 { … }
realm И2 { server И2 }
client И4 { … }
server И4 { … }
realm И4 { server И4 }
```

Для нормальной эксплуатации добавление лишнего шага маршрутизации через регистратора избыточно: и задержку вносит при обработке запросов, и в файлы настроек корневых серверов надо вносить записи обо всех подключаемых организациях. Но для процесса отладки первоначального подключения организации прохождение запросов через регистратора полезно.

Благодаря тому, что файлы настроек программного пакета freeRADIUS, используемого на корневых серверах FLRS, допускают использование директивы include, регистратор готовит подключаемые файлы, описывающие настройки подведомственных организаций (маршрутизируемых как через регистратора, так и напрямую), а FLRS синхронизирует их с серверов регистраторов утилитой rsync.

Поскольку за каждую подключенную организацию отвечает единственный регистратор импортируемые таким образом в FLRS файлы настроек не могут содержать взаимоисключающих данных.

5. Заключение

Таким образом, разработаны технические решения, обеспечивающие развитие российского сегмента Eduroam и взаимодействия МСЦ РАН и ФГАУ ГНИИ ИТТ «Информика» на базе эксплуатируемых отраслевых научно-образовательных телекоммуникационных сетей RUNNet и RASNet, что дает новый импульс к развитию Eduroam в России.

Для полноценного контроля российской системы Eduroam и осуществления технической поддержки, необходим централизованный сбор журналов серверов FLRS (с резер-

вированием), обеспечение доступа к журналам регистраторов, а также разработка средств доступа организаций-пользователей российского сегмента Eduroam к касающимся их пользователей записям журналов прокси-серверов FLRS и регистраторов.

Работа выполнена в МСЦ РАН в рамках Государственного задания.

The Implementation of Eduroam in Russia

Yu.N.Morin, A.P.Ovsyannikov, B.M.Shabanov, D.V.Vershinin

Abstract: The article considers the implementation of eduroam in Russia. The federative access control for research networks based on eduroam is described. Eduroam provides safe network authentication based on single user account in any institute participated in eduroam. The paper considers the features of the eduroam infrastructure based on Russian research and education networks RASNet and RUNNet, operated by JSCC and Informika. The implemented engineering solutions for the registrars interaction and the development of eduroam in Russia are described.

Keywords: eduroam, authentication, research and education networks, RADIUS, wifi networks.

Литература

1. Where can I eduroam? // [В Интернете] https://www.eduroam.org/where/ (дата обращения 20.10.2017 г.)
2. А.П.Овсянников, Г.И.Савин, Б.М.Шабанов. Удостоверяющие федерации научнообразовательных сетей // Программные продукты и системы. – 2012, № 4, с. 3-7
3. А.П.Овсянников, Т.В.Овсянникова, С.А.Овчаренко. Механизм прав на основе групп пользователей в eduroam - федеративной системе управления доступом к сетевым ресурсам научно-образовательных сетей // Программные продукты и системы. – 2012, № 4, с. 10-18
4. 802.1X Port-Based Network Access Control // IEEE Computer Society. 2004. [В Интернет] http://standards.ieee.org/getieee802/download/802.1X-2004.pdf (дата обращения: 20.10.2017).
5. B. Aboba, L.Blunk, J.Vollbrecht, J.Carlson. RFC3748 – Extensible Authentication Protocol (EAP) // IETF, 2004. [В Интернет] http://www.ietf.org/rfc/rfc3748.txt (дата обращения: 20.10.2017).
6. C. Rigney, W.Willats, P.Calhoun. RFC 2869 - RADIUS Extensions // IETF, 2000. [В Интернет] http://www.ietf.org/rfc/rfc3748.txt (дата обращения: 20.10.2017)
7. Регламент Российской Удостоверяющей Федерации Eduroam // МСЦ РАН, 2017 [В Интернет] http://new.jscc.ru/wp-content/uploads/2017/06/eduroam-order.pdf (дата обращения: 20.10.2017)
8. FreeRADIUS // [В Интернет] https://freeradius.org/ (дата обращения: 20.10.2017)

Высокопроизводительные вычисления по принципу SIMD в гиперэллиптических полях

М.С. Аристов

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mais: maristov@niisi.ras.ru

Аннотацияю: В статье рассматриваются вычислительные аспекты, связанные с проблемой поиска и построения S-единиц в гиперэллиптических полях с использованием высокопроизводительных вычислений на различных графических ускорителях.

Ключевые слова: S-единицы, гиперэллиптические поля, непрерывные дроби, графические ускорители, GPGPU, CUDA, OpenCL.

Одной из важных современных проблем алгебры и теории чисел является проблема существования и построения фундаментальных S-единиц в гиперэллиптических полях.

В основополагающей работе [2] были предложены два новых метода для поиска и построения S-единиц в гиперэллиптических полях.

В первом из этих методов – так называемом методе матричной линеаризации – впервые продемонстрирован оригинальный подход с использованием ганкелевых матриц к поиску S-единиц в гиперэллиптических полях. Второй метод – функциональных непрерывных дробей – дал возможность рассматривать с теоретико-числовой точки зрения проблему существования нетривиальных S-единиц в гиперэллиптическом поле.

Идеи работы [2] получили развитие в работах [3] и [4], результатом которых было нахождение S-единиц новых степеней в гиперэллиптических полях рода 2 над полем рациональных чисел в качестве поля констант с помощью указанных методов и компьютерных вычислений.

Основные достижения, полученные до недавнего времени в направлении исследований проблемы существования и построения S-единиц в гиперэллиптических полях содержатся в фундаментальной работе [1].

Дальнейшее развитие этих методов связано с оптимизацией алгоритмов и разработанных программных систем. Были построены новые эффективные алгоритмы поиска фундаментальных S-единиц заданной степени для гиперэллиптических полей рода 2, определяемых многочленами нечётных степеней, и для S, состоящего из бесконечного нормирования и линейного нормирования.

Указанные алгоритмы основаны на эффективных соотношениях, связанных с непрерывными дробями, а также на новом локально-глобальном принципе для функциональных полей и свойствах редукций якобианов гиперэллиптических кривых в конечные поля.

В очень упрощённой форме алгоритм представляет из себя перебор всевозможных комбинаций коэффициентов многочленов в определённом диапазоне. По каждому многочлену строится непрерывная дробь и проверяется наличие свойства периодичности этой непрерывной дроби, что позволяет сделать вывод о наличии нетривиальной S-единицы заданной степени.

В силу того, что свойство наличия нетривиальной S-единицы заданной степени в гиперэллиптическом поле является редким явлением, поиск многочлена, определяющего такое поле, является вычислительно трудной задачей. Например, наличие S-единицы степени 29 для S, состоящего из двух бесконечных нормирований, для конечного поля констант с характеристикой 1019 характерно только для каждого миллионного многочлена. В силу этого обстоятельства, применение только центрального процессора для вышеуказанных вычислений становится недостаточно эффективным.

В отличие от вычислений на центральном процессоре неспециализированные вычисления на графических процессорах (General-purpose computing for graphics processing units, далее по тексту GPGPU) как правило имеют дело с принципиально другой аппаратной архитектурой, которая позволяет параллельно выполнять простые одинаковые инструкции над большим потоком различных данных.

Если вычисления построены таким образом, что их можно разбить на

независимые друг от друга вычислительные блоки, в которых выполняются одни и те же действия, но над разными наборами данных, то такие вычисления эффективно могут использовать особенности архитектуры графических ускорителей.

На текущий момент в мире превалируют 2 подхода к организации подобных вычислений с использованием архитектур OpenCL и CUDA. Эти подходы развивались параллельно и программы, написанные с их помощью имеют существенные различия. А именно, программа написанная с использованием одной

из технологий не может быть без существенной переработки запущена с использованием другой технологии.

Задача поиска S-единиц в гиперэллиптических полях может быть реализована на независимых вычислительных блоках. Она требует исключительно целочисленные вычисления, использует сравнительно небольшое число операций и задействует объём памяти на вычислительное ядро. Это позволяет эффективно использовать GPGPU подход для данной задачи.

Первым для реализации был выбран подход на базе платформы OpenCL, который имеет преимущество в возможности запуска программ, построенных на этой платформе, на различных графических и центральных процессорах без привязки к производителю. Однако, при полной утилизации ресурсов GPU NVIDIA и в целях адаптации программного комплекса, для запуска вычислений на суперкомпьютерах (как правило, построенных на базе ускорителей NVIDIA - одним из примеров которых является суперкомпьютер «Ломоносов»), было необходимо задействовать другую технологию.

В рамках указанной адаптации был разработан и оптимизирован распределенный многоступенчатый программный комплекс на базе программно-аппаратной архитектуры Compute Unified Device Architecture (далее по тексту, CUDA). В ходе работ по программной оптимизации были реализованы упомянутые алгоритмы с учётом особенностей архитектуры одиночный поток команд, множественный поток данных (single instruction, multiple data, далее SIMD), а также с учётом оптимизационных техник по ускорению арифметических операций поля.

Как указано выше, в процессе поиска решения задачи были использованы две различные технологии, построенные по принципу SIMD, для вычислений с использованием различных аппаратных платформ: OpenCL (для запуска на центральных процессорах и различных GPU) и CUDA (для запуска на GPU Nvidia). Программная реализация была создана и оптимизирована для запуска на следующих графических ускорителях:

- AMD Radeon HD 7970 (OpenCL)
- Nvidia GTX 1080 (Cuda)

Для того, чтобы оценивать оптимизационный эффект при адаптации программного комплекса необходимо оценить разницу в производительности аппаратных платформ. Для сравнения общей производительности графических адаптеров использовалась эталонная задача целочисленных вычислений – массивно-параллельное вычисление результата хэш-функции SHA256. Известны эффективные реализации данной задачи, оптимизированные для обеих технологий. Кроме того, вычислительно и математически алгоритм SHA256 схож с рассматриваемой нами задачей.

Результаты запуска тестов приведены в таблице: * *Мх/с – миллионов хэшей в секунду.*

Количество операций с плавающей точкой в секунду и пропускная способность памяти обозначены в спецификации GPU.

В ходе работ по адаптации для исполнения на графических ускорителях NVIDIA и ускорению программной реализации были использованы различные оптимизационные техники. Кроме улучшения математического алгоритма, были применены вычислительные подходы к ускорению программно-аппаратного комплекса. Был применён подход использования константной памяти для хранения часто используемых данных.

Вычисления в рамках GPGPU подразумевают загрузку данных в память графического ускорителя и работу вычислительных ядер с этими данными.

Одна из техник оптимизации заключается в параллельной передаче новых данных и обработке уже загруженных в память.

Данная техника была применена в реализации программного комплекса. В силу особенностей архитектуры SIMD метод раскрытия циклов и раскладывания часто используемых данных в регистры позволил добиться существенного ускорения.

Согласно данным, приведенным в таблице, производительность перебора SHA-256 на графическом ускорителе NVIDIA GTX 1080 в 1.2 раза выше по сравнению с производительностью перебора на графическом ускорителе AMD Radeon HD 7970.

GPU	Кол-во операций с плавающей точкой в секунду	Пропускная способность памяти	Производительность поиска значения хэш-функции
AMD Radeon HD 7970	3.79 Gflops	264 Гб/с	17.5 Мх/с
Nvidia GTX 1080	8.8 Gflops	320 Гб/с	21 Мх/с

Эталонная реализация с использованием аппаратной платформы OpenCL на графическом ускорителе AMD Radeon HD 7970 продемонстрировала скорость перебора 16 миллионов многочленов в секунду. Реализованный программный комплекс с использованием CUDA на графическом ускорителе NVIDIA GTX 1080 - 81 миллион многочленов в секунду.

Таким образом, нам удалось добиться совокупного ускорения адаптированной программной реализации задачи поиска и построения S-единиц в гиперэллиптических полях, использующей технологию по CUDA, более чем в 4 раза по сравнению с реализацией на высокопроизводительном программно-аппаратном комплексе НИИСИ РАН, использующего технологию OpenCL. Это, в свою очередь, позволило существенно расширить диапазон исследуемых гиперэллиптических полей.

Работа выполнена при поддержке гранта РНФ (проект № 16-11-10111).

High-performance computing by the SIMD principle in hyperelliptical fields

M.S. Aristov

Abstract: In the article, some computational aspects associated with the problem of searching and constructing У-units in hyperelliptic fields using high-performance computations on various graphic accelerators.

Keywords: S-unit, hyperelliptic fields, continued fraction, graphics accelerator, GPGPU, CUDA, OpenCL

Литература

1. В.П. Платонов «Теоретико-числовые свойства гиперэллиптических полей и проблема кручения в якобианах гиперэллиптических кривых над полем рациональных чисел». Успехи математических наук, 69:1(415) (2014), 3–38.
2. В.П. Платонов, В.В. Беняш-Кривец, «Группы S-единиц в гиперэллиптических полях и непрерывные дроби». Математический Сборник, 200:11 (2009), 15–44.
3. В.П.Платонов, М.М.Петрунин. «О проблеме кручения в якобианах кривых рода 2 над полем рациональных чисел». Доклады РАН. 2012, Т. 446, № 3, 263–264.
4. В.П.Платонов, М.М.Петрунин. «Новые порядки точек кручения в якобианах кривых рода 2 над полем рациональных чисел». Доклады РАН, 2012, Т. 443, № 6, 664–667.

Об одной схеме исследования случайного процесса в многокомпонентной резервированной системе

Д.В. Гулуа

Грузинский Технический Университет, Тбилиси, Грузия, E-mail: d_gulua@gtu.ge

Аннотация: В работе рассмотрена многокомпонентная резервированная система в которой протекает марковский случайный процесс. Для моделирования и исследования системы, рассмотрена графическая схема исследования двумерного случайного процесса.

Ключевые слова: резервирование, замещение, восстановление, марковский процесс, мнемоническая схема.

Введение

Система в которой протекает марковский случайный процесс (переходы системы из состояния в состояние происходят под воздействием стационарных пуасоновских потоков событий) (см.напр. [1]), представляет собой систему со случайными переходами из одного состояния в другое. Для исследования системы с дискретными состояниями $s_1, s_2, \dots, s_n$ и непрерывным временем t, удобно пользоваться наглядной схемой – размеченным графом состоянии системы, которая, в свою очередь, является удобной иллюстрацией матрицы интенсивностей системы (см.например [2]). Зная поток вероятности $\lambda_{i,j} p_i(t)$ (произведение вероятности $p_i(t)$ состояния s_i в момент времени t, на интенсивность $\lambda_{i,j}$ потока событий переводящих систему из состояния s_i в состояние s_j), уравнения Колмогорова (см.напр. [3]) составляются по следующему мнемоническому правилу: производная вероятности любого состояния равна сумме потоков вероятности, переводящих систему в это состояние, минус сумма всех потоков вероятности, выводящих систему из этого состояния

$$\frac{d}{dt} p_i(t) = \sum_{j=1}^{n} p_j(t)\lambda_{ji} - p_i(t)\sum_{j=1}^{n} \lambda_{ij} \quad (i = \overline{1,n}).$$

Зная начальное состояние системы, уравнения Колмогорова дают возможность найти все вероятности состояний как функции времени. Из уравнений Колмогорова непосредственно получаем систему линейных алгебраических уравнений для определения финальных (предельных) вероятностей состояний системы $p_i = \lim_{t\to\infty} p(i,t)$. Как извесно (см.например [2]), можно и не писать уравнений Колмогорова, а прямо по графу состояний системы записать линейные алгебраические уравнения для определения финальных вероятностей состояний системы. Для этого используем следующее мнемоническое правило записи алгебраического уравнения: справа стоит финальная вероятность данного состояния p_i умноженная на сумарную интенсивность всех потоков ведущих из данного состояния, а справа - сумма произведений интенсивностей всех потоков, входящих в i -е состояние, на вероятности тех состояний, из которых эти потоки исходят.

Использование размеченного графа состояний упрощает процесс моделирования и анализа систем в которых протекает одномерный марковский случайный процесс. Хотя хорошо известно, что сложности, связанные с изучением таких процессов, с увеличением размерности растут в огромной степени.

В начале 90-х годов прошлого столетия шведские специалисты, по поручению Международного Союза Электросвязи (МСЭ), разработали рекомендацию Е.862 «Надежностное планирование телекоммуникационных сетей» [4]. В указанной рекомендации, из двух возможных подходов к надежностному планированию – интуитивному и аналитическому, предпочтение вполне убедительно отдается – аналитическому. Особое внимание уделяется аналитическим моделям сложных резервированных систем, моделям и методам надежности планирования, проектирования, функционирования и технического обслуживания телекоммуникационных сетей и вопросам использования этих методов к разнообразным услугам в международной сети. Рекомендуется применение аналитических методов в класической теории резервирования, а также разработка новых методов, с учетом тех факторов,

которые являются существенными для современных технических систем, в частности – телекоммуникационных. Одним из наиболее важных факторов такого типа является учет длительности замещения отказавших элементов в резервированных системах. Среди важнейших направлений надежностного планирования выделяется задача определения оптимального количества резервных элементов и средств технического обслуживания для тех случаев, когда элементы системы подвергаются отказам, теряют работоспособность и требуют замены.

В работе [5], на основе уравнений Колмогорова, постраены математические модели для резервированных систем. В [6,7], для исследования резервированных систем использованы графические схемы.

В представленной работе рассмотрена многокомпонентная резервированная система обслуживания очередей замещений и восстановлений с временем замещения отличным от нуля. Для исследования рассматриваемой резервированной системы, в статье предложено построение графической схемы - «карты состояний системы» (КСС), которая позволяет наглядно изображать двумерные случайные процессы. Кроме наглядного изображения возможных состоянии системы и переходов из одного состояния в другое, предлагаемая «карта» позволяет, без записи уравнений Колмогорова, непосредственно записывать линейные алгебраические уравнения для вероятностей состояний системы. Кроме того, предложенная графическая схема позволяет выявить т.н. "линии состояния системы", благодаря которым упрощается нахождение средных значений состоянии системы. Учитывая сказанное, а также ряд преимуществ использования КСС для анализа систем, думаем, что предложенная схема может быть полезна исследователям сложных стохастических систем, в том числе при рассмотрении задач принятия решений при интервальной и стохастической неопределенности [8].

По нашему мнению, предложеная в статье модель анализа стохастической системы, может быть полезна при исследования многих систем и, тем самым является, в определенном смысле, универсальным. Эта универсальность основывается на изоморфизме процессов, протекающих в системах различной природы, абстрактным процессам в их математических моделях.

1. Постановка задачи

Рассмотрим систему состоящую из m основных и n резервных элементов (Рис. 1). Все элементы идентичны. Для нормального функционирования системы, необходимо поддерживать все m основные элементы в рабочем состоянии. Система продолжает функционировать и в случае сокращения количества основных элементов, но эффективность функционирования системы падает. В результате необходимо обратиться к резервному элементу и осуществить замещение отказавшего основного элемента резервным. Интенсивность отказа основного элемента α, а резервного - β. Отказавший основной элемент заменяется работоспособным резервным элементом при первой же возможности. При этом интенсивность замещения равна λ. Интенсивность ремонта элемента принимается равной μ. Количество органов осушествляющих замещение и восстановление (ремонт) элементов равно r. При этом операция замещения приоритетна по отношению к операции восстановления. Очевидно, что при $r \geq m+n$ вопрос приоритета несушественнен.

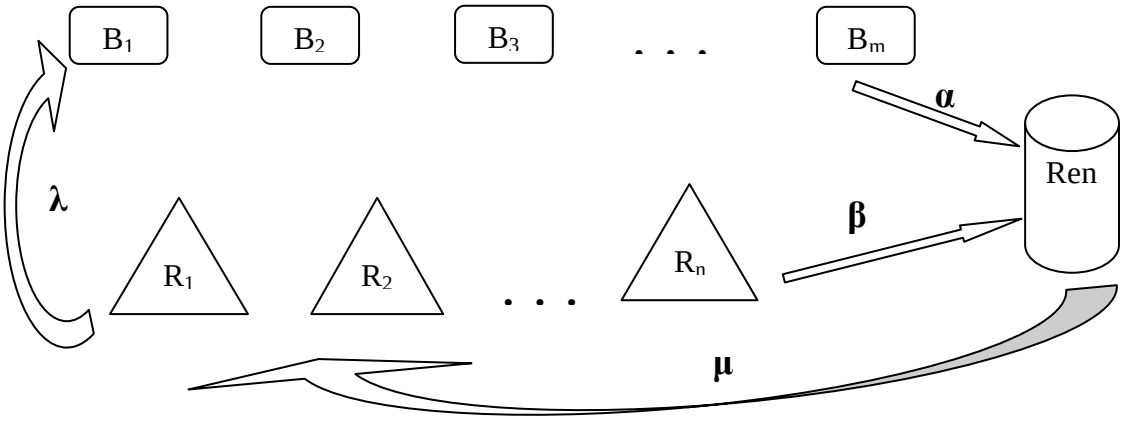


Рис.1. Схема функционирования резервированной системы (B_i –основные элементы; R_i -резервные элементы; Ren -орган восстановления)

Для составления математической модели функционирования рассматриваемой системы введем понятие состояния системы. Скажем,

что система находится в состоянии $s_{i,j}$ ($i=\overline{0,n}$, $j=\overline{0,m+n}$), если количество недостающих основных элементов равен i, в то время как количество вышедших из строя элементов (основных и резервных) равен j. Процесс функционирования системы опишем функциями p(i,j,t) - вероятность того, что в момент времени t система находится в состоянии $s_{i,j}$.

Анализ возможных состояний рассматриваемой системы показывает, что существенными являются только состояния $s_{i,j}$ ($i=\overline{0,m}$, $j=\overline{0,n+i}$). Вначале рассмотрим систему в случае $r=m+n$

Придадим t малое приращение h и найдем вероятность p(i,j,t+h) того, что за время h система перейдет в состоянии $s_{i,j}$. Используя формулу полной вероятности, будем рассуждать следующим образом. В момент времени t+h система может оказаться в состоянии $s_{i,j}$, если: 1) в момент времени t она находилась в этом же состоянии $s_{i,j}$ и за время h она не вышла из него, т.е. не произошло отказа ни рабочего, ни резервного элементов, а также замещения и восстановления; 2) в момент времени t система находилась в состоянии $s_{i-1,j-1}$, а за время h произошел отказ одного из рабочих элементов, при этом не произошло отказа резервного элемента, а также замещений и востановлений; 3) система находилась в состоянии $s_{i,j-1}$ и за время h произошел отказ резервного элемента при этом не происходит отказа основного элемента, а также восстановления и замещения; 4) находясь в момент t в состоянии $s_{i+1,j}$ произошло замещение, при этом выхода из строя элемента, а также восстановления не происходит; 5) система в момент времени t находится в состоянии $s_{i,j+1}$ и за время h произошло восстановление, при этом выхода из строя или замещения элемента не происходит. Любое другое состояние системы к моменту времени t никоим образом не обеспечивает ее состояние $s_{i,j}$ к моменту t+h. Следует заметить, что вероятность наличия двух операций за время h есть величина порядка o(h), которой в нашей модели можно пренебречь. (Под одной операцией понимаем одно из событий: отказ рабочего элемента, отказ резервного элемента, замещение, либо восстановление элемента).

Исходя из вышеприведенных рассуждений, пользуясь формулой полной вероятности и отбросив члены порядка o(h), получим уравнения для вероятностей p(i,j,t+h):

$$p(i,j,t+h) = p(i,j,t) + p(i,j,t)\,(-\alpha(m-i)-\beta(n-(j-i))-\mu\, j-\xi_{i,j})h + p(i-1,j-1,t)\alpha(m-i+1)h+p(i,j-1,t)\beta(n-j+1)h + p(i,j+1,t)\,\mu\,(j+1)h + p(i+1,j,t)\,\lambda\,(n-j+i+1)h+o(h),$$

где $\xi_{i,j} = \lambda\,(n-j+i)$, при $i\neq 0$ и $\xi_{i,j}=0$, если $(i=0)\square(j=n+i)$; $p(-1,j,t)= p(i,-1,t)= p(m+1,j,t)= p(i,n+i+1,t)=0$; $i=\overline{0,m}$, $j=\overline{0,n+i}$.

Из полученных уравнений, сделав простые преобразования и устремляя h к нулю, приходим к дифференциальным уравненям (уравнения Колмогорова):

$$\frac{d}{dt}p(i,j,t) = -(\alpha(m-i)+\beta(n-(j-i))+\mu\, j+\xi_{i,j})\,p(i,j,t) + \alpha\,(m-i+1)\,p(i-1,j-1,t)+\beta\,(n-j+i+1)\,p(i,j-1,t) + \mu\,(j+1)p(i,j+1,t) + \lambda\,(n-j+i+1)\,p(i+1,j,t),$$

$i=\overline{0,m}$, $j=\overline{0,n+i}$.

Приведенные дифференциальные уравнения записаны для каждого состояния $s_{i,j}$ ($i=\overline{0,m}$, $j=\overline{0,n+i}$). Количество уравнений совпадает с количеством состояния $s_{i,j}$ и соответственно с количеством искомых вероятностей p(i,j,t). Многообразие уравнений объясняется многообразием состояний системы в зависимости от i и j.

Рассмотрим вопрос финальных вероятностей, т.е. вероятностей $p_{i,j}=\lim_{t\to\infty} p(i,j,t)$. Число состояний рассматриваемой системы конечно и из каждого состояния, за конечное число шагов возможен переход в любое другое состояние. Как известно, этого достаточно [см.напр. 8] для существования финальных вероятностей. При $t\to\infty$ в системе устанавливается стационарный режим, для которого вероятности состояний уже не заисят от времени. В стационарном режиме система продолжает переходить из одного состояния в другое, но вероятности нахождения системы в этих состояниях уже не зависят от времени. Эти вероятности можно исталковать, как среднее относительное время пребывания системы в соответствующих состояниях. Так как при этом $\frac{d}{dt}p(i,j,t)=0$, для определения $p_{i,j}$ (финальных вероятностей, уже не зависящих от t) получим систему линейных алгебраических уравнений

$$(\alpha(m\text{-}i) + \beta(n\text{-}(j\text{-}i)) + \mu\, j + \xi_{i,j})\, p_{i,j} =$$
$$\alpha\,(m\text{-}i+1)\; p_{i-1,j-1} + \beta\,(n\text{-}j+i+1)\, p_{i,j-1} \qquad (1)$$
$$+\; \mu\;(j+1)\, p_{i,j+1} + \lambda\,(n\text{-}j+i+1)\, p_{i+1,j},$$

где $p_{-1,j} = p_{i,-1} = p_{m+1,j} = p_{i,n+i+1} = 0$;
$i = \overline{0,m}$, $j = \overline{0,n+i}$.

2. Графическая схема функционирования резервированной системы

Использование размеченного графа состояний упощает процесс моделирования и анализа систем в которых протекает одномерный марковский случайный процесс. Что касается сложных технических систем для описания которого необходимо рассматривать двумерные случайные процессы, такой подход достаточно сложен, а результат не всегда очевиден. Ниже приведено построение, "привязанной" к координатной плоскости, графической схемы -"карты состояния системы" (КСС), которая, на наш взгляд, позволяет наглядно представлять двумерные случайые процессы. Особенно хотим отметить, что "привязанность" состояний системы к координатам позволяет проследить закономерности по которым "движутся" потоки событий. Кроме наглядного представления состояний $s_{i,j}$ системы и перехода из одного состояния в другое, предложенная "карта" позволяет, без записи уравнений Колмогорова, записывать

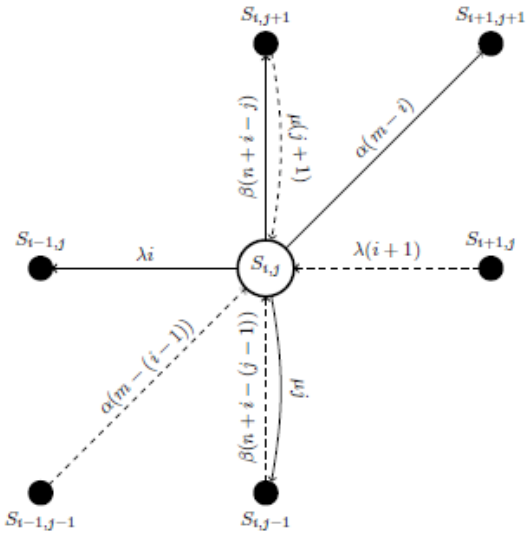


Рис. 2. Схема потока событий для состояния $s_{i,j}$

систему линейных алгебраических уравнений для финальных вероятностей состояний системы $p_{i,j}$.

На (Рис.2) схематически изображены потоки переводящие систему в состояние $s_{i,j}$ (пунктирная линия) и потоки выводящие систему из состояния $s_{i,j}$ (сплошная линия). Приведённая схема получается в результате следующего рассуждения. В состоянии $s_{i,j}$ ($i = \overline{1,m-1}$, $j = \overline{1,n-1}$) система переходит в одном из следующих случаев, a) из состояния $s_{i-1,j-1}$ в результате выхода из строя одного из работаспособных m-(i-1) основных элементов системы; b) из состояния $s_{i,j-1}$ в результате выхода из строя одного из работаспособных n+i-(j-1) резервных элементов системы; c) из состояния $s_{i+1,j}$ в результате замещения одного из i+1 недостающих основных элементов системы; d) из состояния $s_{i,j+1}$ в результате восстановления одного из $j+1$ неработоспособных элементов системы. Некакое другое состояние не обеспечивает, за один шаг, переход системы в состоянии $s_{i,j}$.

Отметим на координатной плоскости iQj точки $s_{i,j}$ (назовём их узламы), с координатами (i, j), символизирующие состояние $s_{i,j}$ резервированной системы. Тогда рассуждения приведённые при построении схемы (Рис.2), позволяют построить всю "карту состояний системы" (Рис.3). Как было отмечено, существенными состояниями рассматриваемой резервированной системы (Рис.1), являются $s_{i,j}$ ($i = \overline{0,m}$, $j = \overline{0,n+i}$). Без ограничения общности допустим, что m > n. Тогда, на координатной плоскости iOj, функционирование системы, схематически можно представить в виде трапеции с вершинами $s_{0,0}, s_{0,n}, s_{m,m+n}, s_{m,0}$ (Рис.3). Стрелки на КСС символизируют потоки событий переводящие резервированную систему из состояния в состояние. Заметим, что при построение КСС, мы строим стрелки (потоки) переводящие систему в определённое состояние $s_{i,j}$. При этом стрелки выхода системы из состояний вычерчиваются автоматически.

Заметим, что на КСС есть узлы (граничные ...) в которых полностью не реализуются все четыре возможные переходы. Например, рассуждения приведённые выше для узлов $s_{i,j}$ ($i = \overline{0,m}$, $j = \overline{0,n+i}$), упрошаются для узла $s_{0,0}$. В самом деле, количество состояний системы из которых можно перейти в

состояние $s_{0,0}$, не четыре, а всего два (из состояния $s_{0,1}$ и $s_{1,0}$). Соответственно на построенной схеме в узел $s_{0,0}$ входят две стрелки (два патока событий переводящие систему в положение $s_{0,0}$). Или к примеру, если $j=\overline{n+1,m+n-1}$, то в случае (c)интенсивность замещения равна λ(n+(i+1)-j), а не λ(i+1). Это вызвано количеством работоспособных резервных элементов в данном состоянии системы $s_{i+1,j}$. Таким образом, при построении КСС, надо учитывать "однотипность" узлов (состоянии) в зависимости от их расположения на схеме.

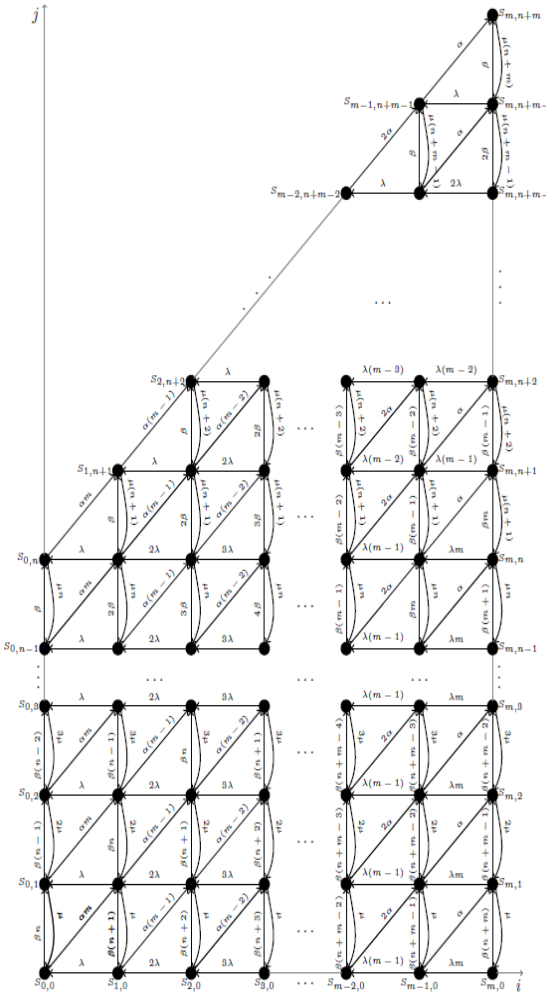


Рис.3. Карта состоянии резервированной системы

Примечательно, что в результате чётких закономерностей, которые легко заметить на КСС, как построение самой "карты", так же анализ процессов протекающих в резервированной системе упрощается. Применяя "карту состояний", можно непосредственно записать как уравнения Колмогорова, также уравнения для определения финальных вероятностей системы. Например, запись уравнений для определения финальных вероятностей системы происходит по следующему мнемоническому правилу: сумма входных потоков равна сумме выходных. Конкретно, для каждого узла КСС уравнение составляется следующим образом: в правой части равенства ставим произведение $p_{i,j}$ -вероятности что система находится в состоянии $s_{i,j}$ на сумму интенсивностей потоков выводящих систему из этого состояния (стрелки на КСС выходящие из узла $s_{i,j}$), а в левой части равенства сумму произведений вероятностей состояний из которых система переходит в состоянии $s_{i,j}$ на интенсивность потока, осуществляющего этот переход (стрелки на КСС входящие в узел $s_{i,j}$). В результате получим следующую систему линейных алгебраических уравнений:

$(\alpha m+\beta n)\ p_{0,0}=\lambda\ p_{1,0}+\mu\ p_{0,1}$

$(\alpha m+\beta(n-j)+\mu j)\ p_{0,j}=\beta\ (n-j+1)\ p_{0,j-1}$
$+\lambda\ p_{1,j}+\mu\ (j+1)\ p_{0,j+1},\qquad j=\overline{1,n-1}$

$(\alpha m+\mu n)\ p_{0,n}=\beta\ p_{0,n-1}+\lambda\ p_{1,n}$

$(\alpha(m-i)+\beta(n+i)+\lambda i)\ p_{i,0}=\alpha\ (i+1)\ p_{i+1,0}$
$+\mu\ p_{i,1},\qquad i=\overline{1,m-1}$

$(\beta(n+m)+\lambda m)\ p_{m,0}=\mu\ p_{m,1}$

$(\alpha(m-i)+\beta(n-(j-i))+\lambda i+\mu j)\ p_{i,j}=$
$\alpha\ (m-i+1)\ p_{i-1,j-1}+\beta\ (n-j+i+1)\ p_{i,j-1}$
$+\lambda\ (i+1)\ p_{i+1,j}+\mu\ (j+1)\ p_{i,j+1},$
$i=\overline{1,m-1},\quad j=\overline{1,n}$

$(\alpha(m-i)+\mu\ (n+i))\ p_{i,n+i}=\alpha\ (m-i+1)\ p_{i,n+i-1}$
$+\lambda\ p_{i+1,n+i},\qquad i=\overline{1,m-1}$

$\mu\ (n+m)\ p_{m,n+m}=\alpha\ p_{m-1,n+m-1}+\beta\ p_{m,n+m-1}$

$(\alpha(m-i)+\beta(i-j))+\lambda\ (i-j)+\mu\ (n+j))\ p_{i,n+j}$
$=\alpha\ (m-i+1)\ p_{i-1,n+j-1}+\beta\ (i-j+1)\ p_{i,n+j-1}$
$+\lambda\ (i-j+1)\ p_{i+1,n+j}+\mu\ (n+j+1)\ p_{i,n+j+1},$
$i=\overline{2,m-1},\quad j=\overline{1,i-1}$

$(\beta(n+m-j)+\lambda m+\mu j)\ p_{m,j}=\alpha\ p_{m-1,j-1}$
$+\beta\ (n+m-j+1)\ p_{m,j-1}+\mu\ (j+1)\ p_{m,j+1},\ j=\overline{1,n}$

$(\beta(m-j)+\lambda\ (m-j)+\mu\ (n+j))\ p_{m,n+j}$
$=\alpha\ p_{m-1,n+j-1}+\beta\ (m-j+1)\ p_{m,n+j-1}$
$+\mu\ (n+j+1)\ p_{m,n+j+1},\qquad j=\overline{1,m-1},$

которая, естественно, совпадает с системой уравнений (1).

Для решения полученной системы линейных однородных алгебраических уравнении необходимо воспользоваться нормирующим условием $\sum_{i=0}^{m}\sum_{j=0}^{n+i} p_{i,j} = 1$, после чего трудности ее решения носят чисто вычислительный характер.

В рассмотренной нами резервированной системе и соответственно при построениии КСС, мы не ограничивали количество органов замещения и ремонта, т.е. мы рассмотрели случай $r = m+n$. Как было отмечено, в этом случае вопрос приоритетности несущественен. Конечно, основной интерес представляет случай $0 < r < n+m$ (при r=0, система имеет единственное концевое (поглашающее) состояние $s_{n,n+m}$ и этот случай не представляет практического интереса). Приведём алгоритм преобразования построенной КСС для случая $1 \le r < m+n$.

Для удобства формулировки алгоритма указанного преобразования, обозначим через $K_{i,j}$ и $L_{i,j}$ коэфициенты, стоящие соответственно при λ и μ на КСС над стрелками выходяшими из узла $s_{i,j}$ в случае когда $r = m+n$ (Рис.3), а через $K_{i,j}^r$ и $L_{i,j}^r$ - коэфициенты которые будут стоять при λ и μ на КСС в случае, если количество органов замещения-восстановления равно $r < m+n$. Подчеркнём, что в случае $r < n+m$, существенен вопрос приоритета операции. Мы рассматриваем случай, когда операция замещения имеет абсолютный приоритет. Это означает, что ремонт элемента осуществляется только в случае ненадобности (все основные элементы в рабочем состоянии) или в слечае невозможности (нет работоспособных резервных элементов для замещения) осуществить замещение. Предварительный анализ показывает: a) для любого $s_{i,j}$, $K_{i,j}^r + L_{i,j}^r \le r$; b) коэффициенты при α и β не зависят от r. Учитывая введённые обозначения, сформулируем шаги преобразования "карты состояний" для случая $r = n+m$ (Рис.3) в "карту состояний" для случая когда количество органов замещения-восстановления равно $r < n+m$.

1. Базисом (основанием) для построения КСС при $r < n+m$, служит построенная нами КСС для $r = n+m$ (см. Рис.3);
2. Любые состояния $s_{i,j}$ ($j < i-(r-1)$) не существенны, т.е. область КСС лежащая ниже прямой $j = i-(r-1)$ "убирается";
3. Коэффициенты при λ и μ на КСС (Рис.3) (на оставшихся стрелках) преобразуются следующим образом: $K_{i,j}^r = \min(r, K_{i,j})$, $L_{i,j}^r = r - K_{i,j}^r$, $j \ge i-(r-1)$.

На (Рис.4) изображены области существенных состояни системы для различных значений r.

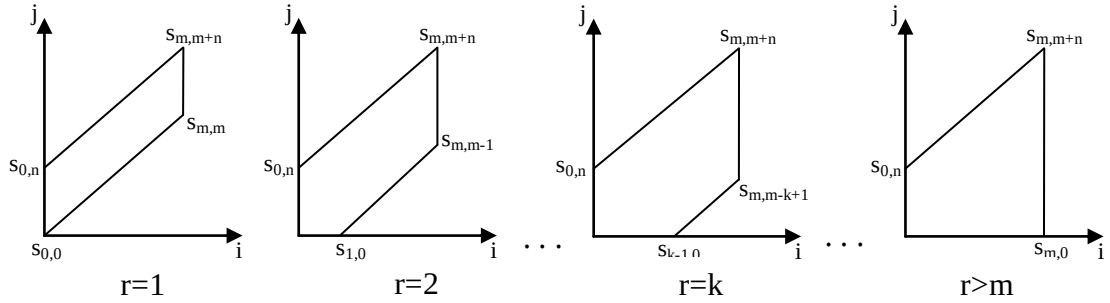


Рис.4. Области существенных состояний системы для различных r

В заключении этой главы отметим, что при разных условиях и дисциплинах обслуживания рассматриваемой резервированной системы (Рис.1), получаем соответствущие КСС. Хотя принцип составления "карты" и соответственно принципы исследование технической системы по построенной карте во всех случаях аналогичны.

3. Явная формула для определения финальных вероятностей резервированной системы

Рассмотрим резервированную систему (Рис.1) с приоритетом замещения, в случае когда она обслуживается одним органом замещения-восстанвления ($r = 1$).

В этом случае карта состояния системы имеет вид (Рис.5). Следуя мнемонической схеме, получим систему линейных алгебраических уравнений для определения предельных вероятностей резервированной системы:

$$(\alpha m + \beta n)\, p_{0,0} = \mu\, p_{0,1}$$

$$
\begin{aligned}
&(\alpha(m\text{-}i) + \beta n + \lambda)\, p_{i,i} \\
&\quad = \alpha\,(m\text{-}i+1)\, p_{i-1,i-1}, \qquad i = \overline{1,m} \\
&(\alpha m + \beta(n\text{-}j) + \mu)\, p_{0,j} \\
&\quad = \beta\,(n\text{-}j+1)\, p_{0,j-1} + \lambda\, p_{1,j} \\
&\qquad + \mu\, p_{0,j+1}, \quad j = \overline{1,n-1} \\
&(\alpha m + \mu)\, p_{0,n} = \beta\, p_{0,n-1} + \lambda\, p_{1,n} \\
&(\alpha(m\text{-}i) + \mu)\, p_{i,n+i} = \alpha\,(m\text{-}i+1)\, p_{i-1,n+i-1} \\
&\quad + \beta\, p_{i,n+i-1} + \lambda\, p_{i+1,n+i}, \qquad i = \overline{1,m-1} \\
&\mu\, p_{m,m+n} = \alpha\, p_{m-1,n+m-1} + \beta\, p_{m,n+m-1} \\
&(\alpha(m\text{-}i) + \beta(n\text{-}(j\text{-}i)) + \lambda)\, p_{i,j} \\
&\quad = \alpha\,(m\text{-}i+1)\, p_{i-1,j-1} \\
&\quad + \beta\,(n\text{-}j+i+1)\, p_{i,j-1} + \lambda\, p_{i+1,j} \\
&\qquad i = \overline{1,m}\ ,\ j = \overline{i+1, n+i-2} \qquad (2) \\
&(\alpha(m\text{-}i) + \beta + \lambda)\, p_{i,n+i-1} = \alpha\,(m\text{-}i+1)\, p_{i-1,n+i-2} \\
&\quad + 2\beta\, p_{i,n+i-2} + \lambda\, p_{i+1,n+i-1} + \mu\, p_{i,n+i}, \\
&\qquad\qquad i = \overline{1,m-1} \\
&(\beta + \lambda)\, p_{m,n+m-1} = \alpha\, p_{m-1,n+m-2} \\
&\quad + 2\beta\, p_{m,n+m-2} + \mu\, p_{m,n+m}\,.
\end{aligned}
$$

Очевидно, что учитывая нормирующее условие $\sum_{i=0}^{m}\sum_{j=i}^{n+i} p_{i,j} = 1$, решение полученной системы носит чисто вычислительный характер.

Покажем, что наглядность КСС (Рис.5) позволяет найти решение полученной системы линейных алгебраических уравнений в явном виде.

Заметим, что уравнения системы (2), не является разностным в чистом виде. Так что её решение методом решения разностных уравнений затруднительно (если вообще возможно). Для решения этой системы воспользуемся наглядностью КСС.

Согласно мнемоническому правилу, в каждом уравнении записанном в узле $s_{i,j}$, присутствуют неизвесные $p_{k,l}$ ($s_{k,l}$ -узлы состояния системы, из которых исходят стрелки (потоки) входящие в узел $s_{i,j}$) и неизвестная $p_{i,j}$ -вероятность нахождения системы в состоянии $s_{i,j}$. Заметим так же, что (см. схему (Рис.5)): 1) в узел $s_{1,1}$ входит единственная стрелка из $s_{0,0}$; 2) в каждый узел $s_{i,j}$, $(i = \overline{1,m},\ j = i)$ входит единственная стрелка из узла $s_{i-1,j-1}$; 3) в $s_{0,0}$ входит единственная стрелка из $s_{0,1}$; 4) в узел $s_{1,2}$ входят стрелки из узлов $s_{0,1}, s_{1,1}$ и $s_{2,2}$; и т.д. Учитывая эти особенности функцианирования рассматриваемой системы, которые проявляются при анализе "карты", можно последовательно выражать все $p_{i,j}$ через $p_{0,0}$. Далее используя нормирующее условие $\sum_{i=0}^{m}\sum_{j=i}^{n+i} p_{i,j} = 1$, определяем значение $p_{0,0}$ и далее все предельные вероятности состояний системы. В таблице (Таб.1) показана последовательность выражения неизвестных $p_{i,j}$ через $p_{0,0}$.

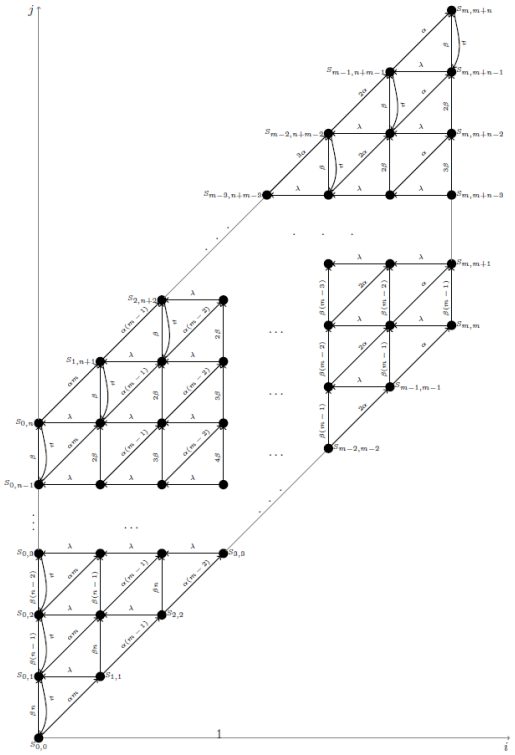


Рис.5. Карта состоянии системы (r=1)

В приведённой таблице: столбец I -номер шага в последовательности нахождения неизвестных $p_{i,j}$; столбец II -номер узла в которой записано алгебраическое уравнение системы; столбец III -неизвестная $p_{i,j}$, которая на данном шаге выражается через $p_{0,0}$. Следуя указанной таблице, получаем формулы для определения значения неизвестных $p_{i,j}$ $(i = \overline{0,m}\ ,\ j = \overline{i, n+i})$.

Таб.1. Последовательность вычислений вероятностей $p_{i,j}$ состояний системы

I	II	III
1	s(1,1)	p(1,1)
2	s(2,2)	p(2,2)
…	…	…
m	s(m,m)	p(m,m)
m+1	s(0,0)	p(0,1)
m+2	s(1,2)	p(1,2)
m+3	s(2,3)	p(2,3)
…	…	…
2m+1	s(m,m+1)	p(m,m+1)
2m+2	s(0,1)	p(0,2)
2m+3	s(1,3)	p(1,3)
…	…	…
(n-1)m+n-2	s(m,m+n-2)	p(m,m+n-2)
(n-1)m+n-1	s(0,n-2)	p(0,n-1)
(n-1)m+n	s(0,n-1)	p(0,n)
(n-1)m+n+1	s(0,n)	p(1,n)
(n-1)m+n+2	s(1,n)	p(1,n+1)
…	…	…
(n-1)m+n+2m-1	s(m-1,m+n-1)	p(m,m+n-1)
(n-1)m+n+2m	s(m,m+n-1)	p(m,m+n)

$$p_{i,i+j} = q(j,i)\ p_{0,0}, \qquad (3)$$

$$p_{0,0} = \left(\sum_{i=0}^{m} \sum_{j=0}^{n} q(i,j) \right)^{-1},$$

$$i = \overline{0,m},\ j = \overline{1,n},$$

где

$$q(0,i) = \frac{\alpha^i m!}{(m-i)!} \left(\prod_{k=1}^{i} (\alpha (m-k) + \beta n + \lambda) \right)^{-1},$$

$$i = \overline{1,m}, \qquad q(0,0)=1,$$

$$q(j,i) = k_{ij}(\alpha(m-i+1)q(j,i-1) + \beta(n-j+1)q(j-1,i) + \lambda\, q(j-1,i+1)), \qquad j = \overline{1,n-2},\ i = \overline{1,m},$$

где

$k_{ij} = (\alpha(m-i)+\beta(n-j)+\lambda)^{-1}$, $\quad q(j,m+1) = 0$

$$q(j,0) = \mu^{-1}((\alpha m+\beta(n-j+1)+\mu)q(j-1,0) - \beta(n-j+2)q(j-2,0) - \lambda\, q(j-2,1)), \qquad j = \overline{2,n}$$

$$q(1,0) = \mu^{-1}(\alpha m+\beta n)$$

$$q(n-1,i) = \lambda^{-1}((\alpha(m-i+1)+\mu)q(n,i-1) - w\cdot\alpha(m-i+2)q(n,i-2) - \beta q(n-1,i-1)), \quad i = \overline{1,m}$$

где, если i=1 тогда w=0, иначе w=1.

$$q(n,i) = \mu^{-1}((\alpha(m-i)+\beta+\lambda)q(n-1,i) - \alpha(m-i+1)q(n-1,i-1) - 2\beta q(n-2,i) - \lambda\, q(n-2,i+1)), \quad i = \overline{1,m}$$

здесь q(-1,i)=0.

Имея явные формулы для определения предельных вероятностей состояния системы, исследование системы (вопросы надёжности системы, оптимизация по экономическому критерию и т.д.) упрощается. Меняя параметры технической системы мы не заботимся о виде системы алгебраических уравнений (2), а анализируем поведение технической системы, опираясь на формулы (3) для определения предельных
вероятностей для различных значений параметров системы.

4. Линии состояний системы. Расчёт экономической эффективности функционирования резервированной системы

Построение модели и определение предельных вероятностей $p_{i,j}$ состояний $s_{i,j}$ системы не представляет самостоятельного интереса. Как правило, эти данные нужны для расчёта определённых характеристик систем (надёжность, экономическая эффективность и т.д.). Для определения этих характеристик, часто необходимо прослеживать за изменением состояний технических систем. Обычно, в сложных системах, эти состояния бывают достаточно многообразны и довольно "запутаны". Например, для определения средных значений величин (например количества работающих основных элементов), надо проследить: состояния системы где эти величины присутствуют, размер этой величины и вероятность нахождения системы в этом состоянии. Производить подобные действия, при стандартном, аналитическом исследовании систем, процесс достаточно трудоёмкий, а результат не совсем наглядный. Вопрос упрощается при использо-

вании, предложенной в статье, "карты состояния системы".

Назавём линиями состояний технической системы, линии на КСС, на котором расположены узлы состояний системы $s_{i,j}$ в которых определённый параметр системы сохраняет постоянство.

Рассмотрим поподробнее линии основных элементов .

На (Рис. 6) сплошными линиями изображены линии на которых расположены узлы $s_{i,j}$ соответствующие состоянию системы, когда количество основных элементов постоянно и равно m, m-1, m-2, …, 1, 0 соответственно (на оси Oi помечены количество основных элементов функционирующих в состояниях $s_{i,j}$, которые соответствуют узлам расположенным на соответствующей сплошной линии). Например, на крайней левой сплошной линии расположены узлы $s_{0,j}$ ($j=\overline{0,n}$) соответствующие тем состояниям технической системы, когда работают все m основные элементы системы, а на крайней правой сплошной линии расположены узлы $s_{m,j}$ ($j=\overline{0,m+n}$) соответствующие тем состояниям технической системы, когда система "осталась" без основных элементов.

На (Рис.7) приведены схематические изображения следующих линии состояний, соответственно: линии резервных элементов (a); линии органов занятых замещением (b); линии органов занятых восстановлением (c).

Линии состояний изображённые на (Рис.6), (Рис.7) соответствуют резервированной системы, меняются (все или частично) линии операции. Например, если количество органов занимающихся замещением и восстановлением ограничены и равны соответственно k и l, то естественно одновременно можно замещать не более k элементов и соответственно восстанавливать не более l неработоспособных элементов. Это конечно должно найти своё отражение при постраении соответствующих линии состояни.

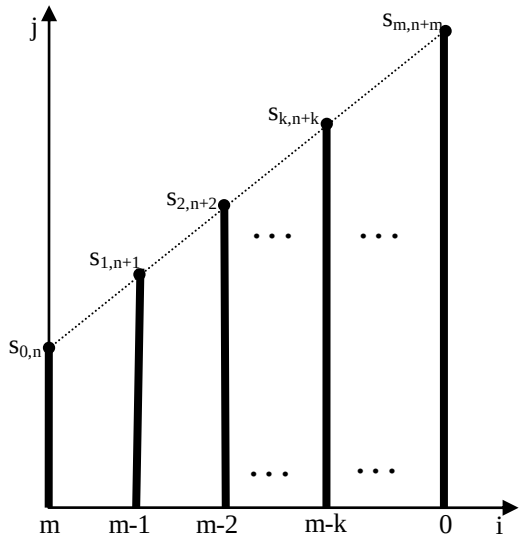


Рис.6. Линии основных элементов

Как было отмечено вначале главы, мы считаем, что введение линии операции системы упрощает анализ и определение разных характеристик системы.

Рассмотрим вопрос экономической эффективности системы. В качестве целевой функции введем показатель экономической эффективности:

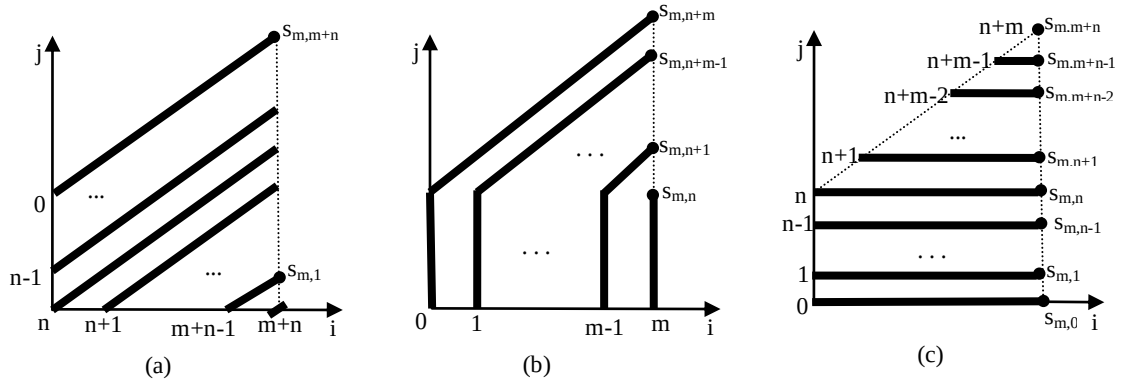


Рис.7. Линии состояний резервированной системы

ной системе описанной в первой главе, т.е для случая когда органы замещения-восстановления неограничены $r=m+n$. Естественно при изменении параметров резер-

$$F(m,n) = (z_1-c_1)E_1 + (z_2-c_2)E_2 - \sum_{k=3}^{6} c_k E_k \ , \quad (4)$$

Здесь: z_1- доход в единицу времени от одного работоспособного (РС) основного элемента; z_2 - доход в единицу времени от одного РС резервного элемента; c_1- затраты в единицу времени на одного РС основного элемента; c_2 - затраты в единицу времени на одного РС резервного элемента; c_3 - затраты в единицу времени на одного неработоспособного (НРС) элемента; c_4 - затраты в единицу времени на одного работающего органа восстановления (ОВ); c_5 - затраты в единицу времени на одного работающего органа замещения (ОЗ); c_6 - затраты в единицу времени на одного не работающего органа замещений и восстановлений (ОЗВ); E_1 – среднее количество РС основных элементов; E_2 – среднее количество РС резервных элементов; E_3 – среднее количество НРС элементов; E_4 –среднее количество работающих ОВ; E_5 –среднее количество работающих ОЗ; E_6 – среднее количество не работающих ОЗВ.

Для определения значения целевой функции (4) необходимо найти средние значения E_i (i=1,…,6). Отметим, что для их вычисления удобно пользоваться картой состояний системы (Рис.1), точнее линиями операции.

Для определения среднего количества основных элементов E_1, рассмотрим случайную величину – количество РС основных элементов. Заметим, что на карте состояний системы каждому конкретнему значению рассматриваемой случайной величины соответствуют узлы, которые расположены вдоль прямых параллельных оси Oj (Рис.6). Распределение случайной величины имеет вид:

m	m-1	m-2	...
$\sum_{j=0}^{n} p_{0,j}$	$\sum_{j=0}^{n+1} p_{1,j}$	$\sum_{j=0}^{n+2} p_{2,j}$	...

...	2	1	0
...	$\sum_{j=0}^{n+m-2} p_{m-2,j}$	$\sum_{j=0}^{n+m-1} p_{m-1,j}$	$\sum_{j=0}^{n+m} p_{m,j}$

Отсюда определяем среднее значение количества РС основних элементов

$$E_1 = \sum_{i=0}^{m} (m-i) \sum_{j=0}^{n+i} p_{i,j} .$$

Случайная величина - количество неработоспособных элементов имеет следующее распределение (см.Рис.6(b)):

m+n	m+n -1	m+n -2	...
$p_{m,m+n}$	$\sum_{i=m-1}^{m} p_{i,m+n-1}$	$\sum_{i=m-2}^{m} p_{i,m+n-2}$	...

...	N	...	1	0
...	$\sum_{i=0}^{m} p_{i,n}$	...	$\sum_{i=0}^{m} p_{i,1}$	$\sum_{i=0}^{m} p_{i,0}$

Отсюда, среднее значение количества НРС элементов

$$E_3 = \sum_{j=0}^{n} j \sum_{i=0}^{m} p_{i,j} + \sum_{j=n+1}^{n+m} j \sum_{i=j-n}^{m} p_{i,j} .$$

Несложно заметить, что среднее значение случайной вличины - количества РС резервных элементов можно определить непосредственно E_2=n+m- E_1- E_3.

Учитывая, что в нашем случае, количество органов обслуживающих замещение и восстановление элементов (ОЗВ) равно $r = m+n$, можем записать

$$E_4 = \sum_{j=0}^{n} j \sum_{i=0}^{m} p_{i,j} + \sum_{j=n+1}^{n+m} j \sum_{i=j-n}^{m} p_{i,j} ,$$

т.е. паралельно восстанавливаются все НРС элементы.

Случайное число - количество органов занятых замещением имеет следующее распределение (Рис.6(c)):

0	1	2	...
$\sum_{j=0}^{n} p_{0,j} +$ $\sum_{i=1}^{m} p_{i,n+i}$	$\sum_{j=0}^{n} p_{1,j} +$ $\sum_{i=1}^{m-1} p_{i+1,n+i}$	$\sum_{j=0}^{n} p_{2,j} +$ $\sum_{i=1}^{m-2} p_{i+2,n+i}$	...

...	k	...	m-1	m
...	$\sum_{j=0}^{n} p_{k,j} +$ $\sum_{i=1}^{m-k} p_{i+k,n+i}$	...	$\sum_{j=0}^{n} p_{m-1,j} +$ $p_{m,n+1}$	$\sum_{j=0}^{n} p_{m,j}$

Следовательно, можем записать

$$E_5 = \sum_{k=0}^{m} k \left(\sum_{j=0}^{n} p_{k,j} + \sum_{i=1}^{m-k} p_{i+k,n+i} \right).$$

Очевидно, что среднее количество не работающих ОЗВ E_6= r-(E_4+ E_5).

5. Результаты численных экспериментов. Случай специализации органов обслуживания

Для начала (в продолжении главы 4), рассмотрим конкретный вопрос оптимизации резервированной системы. Допустим, для заданных значений параметров технической системы α=0.01, β=0.001, λ=5, μ=2, требуется определить количество необходимых резервных элементов для достижения максимальной экономической эффективности функционирования системы, при условии, что система должна состоять из пяти основных элементов. Т.е. требуется определить количество n, при котором достигается maxF(5,n), где целевая функция определяется по формуле (4).

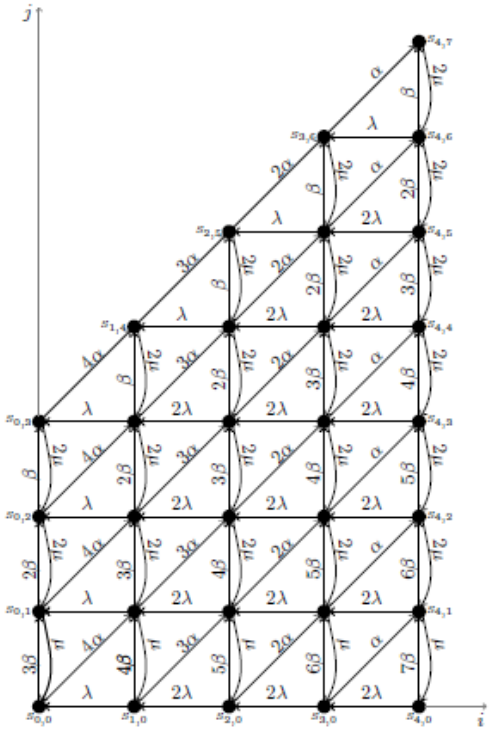


Рис.8. Карта состояний системы

Для решения вышеприведенной задачи оптимизации, используем, предложенный в четвёртой главе, алгоритм определения целевой функции F(m,n). В результате простого перебора получим оптимальное количество резервных элементов n=2. При этом достигается экономическая эффективность

$$\max F(5,n) = F(5,2) = 44{,}2771.$$

Перйдём теперь к составлению математической модели резервированной системы с четырмя основными (m=4) и тремя резервными (n=3) элементами. Рассмотрим случай, когда ограны обслуживапия имеют конкретную специализацию, а именно два органа занимаются замещением и два - ремонтом. Очевидно, что в виду специализации органов обслуживания, вопрос приоритета несущественен.

При построение системы линейных алгебраических уравнений, для описания поведения системы, используем карту состояния рассматриваемой резервированной системы (Рис.8).

Следуя мнемоническому правилу (сумма входных, равна сумме выходных потоков), по КСС получаем следующую систему линейных алгебраических уравнений для определения финальных вероятностей $p_{i,j}$ $(i=\overline{0,4},\ j=\overline{0,3+i})$:

(4α+3β) $p_{0,0} = \lambda p_{1,0} + \mu p_{0,1}$

(3α+4β+λ) $p_{1,0} = 2\lambda p_{2,0} + \mu p_{1,1}$

(2α+5β+2λ) $p_{2,0} = 2\lambda p_{3,0} + \mu p_{2,1}$

(α+6β+2λ) $p_{3,0} = 2\lambda p_{4,0} + \mu p_{3,1}$ P(3,0)

(7β+2λ) $p_{4,0} = \mu p_{4,1}$

(4α+2β+μ) $p_{0,1} = 3\beta p_{0,0} + \lambda p_{1,1} + 2\mu p_{0,2}$

(3α+3β+λ+μ) $p_{1,1} = 4\alpha p_{0,0} + 3\beta p_{1,0} + 2\lambda p_{2,1} + 2\mu p_{1,2}$

(2α+4β+2λ+μ) $p_{2,1} = 3\alpha p_{1,0} + 5\beta p_{2,0} + 2\lambda p_{3,1} + 2\mu p_{2,2}$

(α+5β+2λ+μ) $p_{3,1} = 2\alpha p_{2,0} + 6\beta p_{3,0} + 2\lambda p_{4,1} + 2\mu p_{3,2}$

(6β+2λ+μ) $p_{4,1} = \alpha p_{3,0} + 7\beta p_{4,0} + 2\mu p_{4,2}$

(4α+β+2μ) $p_{0,2} = 2\beta p_{0,1} + \lambda p_{1,2} + 2\mu p_{0,3}$

(3α+2β+λ+2μ) $p_{1,2} = 4\alpha p_{0,1} + 3\beta p_{1,1} + 2\lambda p_{2,2} + 2\mu p_{1,3}$

(2α+3β+2λ+2μ) $p_{2,2} = 3\alpha p_{1,1} + 4\beta p_{2,1} + 2\lambda p_{3,2} + 2\mu p_{2,3}$

(α+4β+2λ+2μ) $p_{3,2} = 2\alpha p_{2,1} + 5\beta p_{3,1} + 2\lambda p_{4,2} + 2\mu p_{3,3}$

(3β+2λ+2μ) $p_{4,2} = \alpha p_{3,1} + 6\beta p_{4,1} + 2\mu p_{4,3}$

(4α+2μ) $p_{0,3} = \beta p_{0,2} + \lambda p_{1,3}$

(3α+β+λ+2μ) $p_{1,3} = 4\alpha p_{0,2} + 2\beta p_{1,2} + 2\lambda p_{2,3} + 2\mu p_{1,4}$

(2α+2β 2λ+2μ) $p_{2,3} = 3\alpha p_{1,2} + 3\beta p_{2,2} + 2\lambda p_{3,3} + 2\mu p_{2,4}$

$(\alpha+3\beta+2\lambda+2\mu)\, p_{3,3} = 2\alpha\, p_{2,2} + 4\beta\, p_{3,2} + 2\lambda\, p_{4,3} + 2\mu\, p_{3,4}$

$(4\beta+2\lambda+2\mu)\, p_{4,3} = \alpha\, p_{3,2} + 5\beta\, p_{4,2} + 2\mu\, p_{4,4}$

$(3\alpha+2\mu)\, p_{1,4} = 4\alpha\, p_{0,3} + \beta\, p_{1,3} + \lambda\, p_{2,4}$

$(2\alpha+\beta+\lambda+2\mu)\, p_{2,4} = 3\alpha\, p_{1,3} + 2\beta\, p_{2,3} + 2\lambda\, p_{3,4} + 2\mu\, p_{2,5}$

$(\alpha+2\beta+2\lambda+2\mu)\, p_{3,4} = 2\alpha\, p_{2,3} + 3\beta\, p_{3,3} + 2\lambda\, p_{4,4} + 2\mu\, p_{3,5}$

$(3\beta+2\lambda+2\mu)\, p_{4,4} = \alpha\, p_{3,3} + 4\beta\, p_{4,3} + 2\mu\, p_{4,5}$

$(2\alpha+2\lambda)\, p_{2,5} = 3\alpha\, p_{1,4} + \beta\, p_{2,4} + \lambda\, p_{3,5}$

$(\alpha+\beta+\lambda+2\mu)\, p_{3,5} = 2\alpha\, p_{2,4} + 2\beta\, p_{3,4} + 2\lambda\, p_{4,5} + 2\mu\, p_{3,6}$

$(2\beta+2\lambda+2\mu)\, p_{4,5} = \alpha\, p_{3,4} + 3\beta\, p_{4,4} + 2\mu\, p_{4,6}$

$(\alpha+2\mu)\, p_{3,6} = 2\alpha\, p_{2,5} + \beta\, p_{3,5} + \lambda\, p_{4,6}$

$(\beta+\lambda+2\mu)\, p_{4,6} = \alpha\, p_{3,5} + 2\beta\, p_{4,5} + 2\mu\, p_{4,7}$

$2\mu\, p_{4,7} = \alpha\, p_{3,6} + \beta\, p_{4,6}$

Нетрудно заметить, что определитель системы равен нулю. Для определения предельных вероятносткей $p_{i,j}$, заменим одно уравнении полученной системы нормирующим условием $\sum_{i=0}^{4}\sum_{j=i}^{i+3} p_{i,j} = 1$.

Рассмотрим вопрос экономической эффективности системы. В качестве целевой функции введем показатель экономической эффективности:

$$F(m,n,k,r) = \sum_{k=1}^{9} c_k E_k \qquad (5)$$

Здесь: c_1 - доход в единицу времени от одного основного элемента; c_2 - доход в единицу времени от одного резервного элемента; c_3 - доход в единицу времени от одного резервного злемента, находящегося в процессе замещения; c_4 -доход в единицу времени от одного злемента, находящегося в процессе ремонта; c_5 - затраты в единицу времени на одного неработоспособного элемента (не в ремонте); c_6 - затраты в единицу времени на одного работающего органа замещения; c_7 - затраты в единицу времени на одного не работающего органа замещения;c_8 - затраты в единицу времени на одного работающего органа восстановления (ремонта); c_9 - затраты в единицу времени на одного не работающего органа восстановления (ремонта);

По линии операции системы, можно легко записать распределение случайной величины – количество работоспособных основных элементов.

Тогда,

$$E_1 = \sum_{i=0}^{6} p_{3,i} + 2\sum_{i=0}^{5} p_{2,i} + 3\sum_{i=0}^{4} p_{1,i} + 4\sum_{i=0}^{3} p_{0,i}$$

0	1	2	3	4
$\sum_{i=0}^{7} p_{4,i}$	$\sum_{i=0}^{6} p_{3,i}$	$\sum_{i=0}^{5} p_{2,i}$	$\sum_{i=0}^{4} p_{1,i}$	$\sum_{i=0}^{3} p_{0,i}$

Аналогично:

$E_2 = [\, p_{0,2} + p_{1,2} + p_{2,2} + p_{3,3} + p_{4,4}\,]$
$+2[\, p_{0,1} + p_{1,1} + p_{2,1} + p_{3,2} + p_{4,3}\,]$
$+ 3[\, p_{0,0} + p_{1,0} + p_{2,0} + p_{3,1} + p_{4,2}\,]$
$+ 4[\, p_{3,0} + p_{4,1}\,] + 5\, p_{4,0}$,

$$E_3 = \left[\sum_{i=0}^{3} p_{1,i} + \sum_{i=2}^{4} p_{i,i+2}\right] + 2\sum_{i=2}^{4}\sum_{j=0}^{i+1} p_{i,j},$$

$$E_4 = \sum_{i=0}^{4} p_{i,1} + 2\left(1 - \sum_{j=0}^{1}\sum_{i=0}^{4} p_{i,j}\right),$$

$E_5 = E_5^1 - E_4$,

где E_5^1 - среднее количество неработоспособных элементов и

$$E_5^1 = \sum_{i=0}^{4} p_{i,1} + 2\sum_{i=0}^{4} p_{i,2} + 3\sum_{i=0}^{4} p_{i,3} + 4\sum_{i=1}^{4} p_{i,4} + 5\sum_{i=2}^{4} p_{i,5} + 6\sum_{i=3}^{4} p_{i,6} + 7\, p_{4,7}.$$

Далее очевидн, что

$E_6 = E_3$, $E_7 = 2 - E_6$, $E_8 = E_4$, $E_9 = 2 - E_8$.

Подставляя полученные значения E_i ($i=\overline{1,9}$) в (5), можно определить экономическую эфективность функционирования исследуемой системы (m=4, n=3, k=r=2) для заданных параметров α,β,λ,μ. Имея возможность вычислить экономическую эфективность системы, мы можем решать задачу оптимизации системы по входным (управляемым) параметрам.

Построенная модель реализована и среде MatLab. Экспериментальные входные данные:
α=0,01; β=0,001; λ=1; μ=0,1;
c_1=10; c_2=2; c_3=0; c_4=-3; c_5=-3;
c_6=-5; c_7=-1; c_8=-15; c_9=-3.

В результате вычисления получаем F(4,3,2,2) = 30,1572. На столбиковой диаграмме (Рис.9) изображены величины значений компонентов вектора p = ($p_{0,0}$, $p_{0,1}$, $p_{0,2}$, $p_{0,3}$, $p_{1,0}$, $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, $p_{1,4}$, $p_{2,0}$, … , $p_{4,7}$). Диаграмма указывает на стабильную работу основных элементов. Если среди входных параметров изменить значение α=0,01 на α=0,1 (т.е. надёжность работы оснавных элементов ухудшить на порядок), то в резельтате расчётов получим F(4,3,2,2) = -31,9787 (т.е. система работает на убыль), а диаграмма (Рис.10) указывает на резкое ухудшение стабильности работы оснавных элементов.

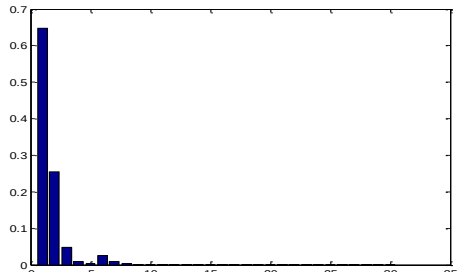


Рис.9: Диаграмма вектора p , (α=0,01)

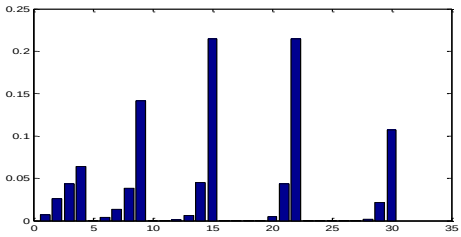


Рис.10: Диаграмма вектора p , (α=0,1)

About one investigation scheme for random process in a multicomponent standby system

D.V.Gulua

Abstract: The paper considers the multicomponent standby system with Markov random process. For modeling and investigating the system, graphic scheme is constructed for two-dimensional random processes

Keywords: standby, replacement, renewal, Markov processes, mnemonic scheme.

Литература

1. А.Я.Хинчин. Математические методы теории массового обслуживания,Труды Матем. инта им В.А.Стеклова, т.49, изд. АН СССР, 1963.
2. Е.С. Вентцель, Л.А. Овчаров. Теория случайных процессов и её инжинерные приложения. М.: Нука, 1991.
3. И.И.Гихман, А.В.Скороход. Введение в теорию случайных процессов. М.: Нука, 1977.
4. Recommendation E.862 (rev.1) Dependability of Telecommunication Networks. Geneva, 1992.
5. Д.В. Гулуа, З.А. Баиашвили, С.Р. Мания, М.Г. Кимостели. Математическая модель системы приоритетного обслуживания с двумя типами операции, Грузинский Технический Университет, Труды, #2(460), 2006.
6. R. Khurodze, R. Kakubava, D. Gulua. Priority queuing system for replacement and renewal. Bulletin of the Georgian Academy of Sciences, 172, #1, 2005.
7. D. Gulua. A General stochastic model of the priority service system with two types of operations, PROCEEDINGS of I.Vekua Institute of Applied Mathematics, Vol. 57, 2007.
8. А.Г. Мадера. Интервально стохастическая неопределенность оценок в многокритериальных задачах принятия решений // Искусственный интеллект и принятие решений. 2014. №3. С. 105-115.
9. Б.В. Гнеденко, И.Н.Коваленко. Введение в теорию массового обслуживания. М: Нука, 1966.

О некоторых условиях на неполные частные непрерывных дробей, связанных с гиперэллиптическими полями и S-единицами

Ю.В. Кузнецов[1], Ю.Н. Штейников[2]

[1,2] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]yukuz61@rambler.ru, [2]yuriisht@gmail.com

Аннотация: Пусть Q - поле рациональных чисел, Q(x) - поле рациональных функций от одной переменной, f - свободный от квадратов многочлен нечетной степени равной 2g+1, g>0. Пусть для многочлена h степени 1 дискретное нормирование определяемое этим многочленом имеет два неэквивалентных продолжения на квадратичное расширение поля L=Q(x)(√f). Имеется связь между свойствами разложения специальных элементов поля L в непрерывную дробь и S-единицами, задаваемыми конечным и бесконечным нормированием. Для некоторых небольших значений длины периода получены оценки на степени соответствующих фундаментальных S-единиц, а также некоторые необходимые условия, которым должны удовлетворять элементы указанных дробей, чтобы при заданных параметрах существовали нетривиальные S– единицы.

Ключевые слова: непрерывные дроби, гиперэллиптические поля, S-единицы, нормирование.

Введение

Пусть k - поле характеристики отличной от 2, f - многочлен нечетной степени 2g+1 и f свободен от квадратов. Обозначим через L поле k(x) (√f). Напомним, что для неприводимого над k многочлена h дискретное нормирование ν_h (элемента поля k(x)) задается равенством

$$\nu_h (h^m a/b)=m,$$

где многочлены a,b не делятся на h.

Пусть далее h - многочлен степени 1 такой, что дискретное нормирование ν_h поля k(x), задаваемое этим многочленом, имеет два продолжения на поле k(x) (√f). Одно из этих нормирований обозначим через ν'_h и определим $S=\{ \nu'_h , \nu_\infty \}$.

Обозначим ещё через O_s кольцо S-целых элементов L, то есть таких элементов z, что $\nu(z)\geq 0$ для всех нормирований ν поля L, не принадлежащих S.

Отметим сразу, что поскольку многочлен f имеет нечетную степень, то нормирование ν_∞ поля k(x) имеет одно продолжение на поле L. Тот факт, что ν_h имеет два продолжения на поле L равносилен тому, что свободный коэффициент многочлена f при разложении f по степеням h есть квадрат из k^*.

Мультипликативная группа кольца O_s - S-целых элементов поля L, называется группой S-единиц. Если существует хотя бы одна нетривиальная S-единица (то есть отличная от константы поля k), то в описанном нами случае группа S-единиц является прямым произведением k^* и бесконечной циклической группы. Образующие этой циклической группы называются фундаментальными S-единицами.

Имеется важная связь между существованием нетривиальных S-единиц и разложением $\sqrt{f}\ /h^{g+1}$ в непрерывную дробь в поле L. Именно, В.П. Платоновым и М.М. Петруниным см [1],[2],[3] было показано, что нетривиальная S-единица в L существует тогда и только тогда, когда бесконечная непрерывная функциональная дробь, в которую раскладывается элемент $\sqrt{f}\ /h^{g+1}$, является периодичной.

Сформулируем основную цель данной работы. Мы подробно изучим специальный вид периодической непрерывной дроби для элемента $\sqrt{f}\ /h^{g+1}$ и, пользуясь дополнительной информацией о виде такого разложения, найдём оценки на степени соответствующих фундаментальных S-единиц. Кроме того, мы получим необходимые условия, которым должны удовлетворять элементы указанных непрерывных дробей.

Вспомогательные утверждения

Приведём основные утверждения о S-единицах и непрерывных дробях, которые понадобятся нам в дальнейшем.

Но сначала напомним понятия периода и квазипериода непрерывной функциональной дроби.

Пусть задана непрерывная дробь $\propto=[a_0, a_1 ,a_2,\ldots]$. Обозначим через $\propto_n$ - полное частное с номером n, то есть

$\propto_n$= [a_{n},a_{n+1},….].
Разложение в непрерывную дробь элемента $\propto$ называется периодичным (квазипериодичным), если существует такие i>j ≥ 0, что $\propto_i=\propto_j$ (соответственно $\propto_i=c\propto_j$, где c – некоторая константа поля k. Наименьшее такое i-j называется тогда периодом (квазипериодом) рассматриваемой непрерывной дроби. Всюду далее в качестве базового поля будем рассматривать поле Q.

Теорема 1 из работы [2] устанавливает связь между существованием нетривиальных S-единиц и периодичностью непрерывной дроби для элемента $\sqrt{f}$ /h^{g+1}. Для более полных формулировок мы отсылаем читателя к работе [2]. Формулировка этой теоремы такова.

Теорема 1. *Непрерывная дробь для элемента* $\sqrt{f}$ /h^{g+1} -- *периодична тогда и только тогда, когда существует нетривиальная S-единица поля L.*

Разложение в непрерывную дробь элемента $\sqrt{f}$ /h^{g+1} имеет важную особенность - оно обладает симметрией.

Следуя работам [2,3,4], введём понятие степени фундаментальной S-единицы.

Пусть p+q$\sqrt{f}$ - фундаментальная S-единица и число m>0 определяется из следующего норменного уравнения

$$P^2 - q^2 f = bh^m,$$

где b – константа поля k. Тогда m называется степенью фундаментальной S-единицы величины p+q$\sqrt{f}$.

Обозначим через a_i неполные частные для $\sqrt{f}$ /h^{g+1}. Имеется связь между степенью фундаментальной S–единицы (m), суммы «нормирований неполных частных по периоду/квазипериоду» (t) и степенью монома специального много-члена L (r):

$$m = 2t + r.$$

За подробностями мы отсылаем читателя к работе [2]

Доказательство основных утверждений

Пусть рассматривается непрерывная дробь для $\propto = \sqrt{f}$ /h^{g+1}. В последующей лемме мы вычислим выражение для f через разложение $\propto$ в непрерывную дробь.

Теорема 2. *Если*

$$\propto = [\, a_0, \overline{a_1, a_2, a_1, 2a_0}\ ],$$

то

$$f = h^{2g+2}\,(a^2_0 + (a_2+2a_0+2a_0a_1a_2)/(2a_1+a^2_1a_2)).$$

Доказательство этого факта напрямую следует из вычислений.

Теперь перейдем к необходимым и достаточным условиям существования нетривиальных S-единиц. Для этого лишь требуется найти условия, при которых так определенные выражения для f в действительности задают многочлен 5 степени, и убедиться, что свободный коэффициент многочлена f при разложении по степеням h есть полный квадрат из поля k. При формулировке нижеследующего основного утверждения полагается, что $\propto = \sqrt{f}$ /h^{g+1} , а величины m, t, r, g определены также как и раньше.

Теорема 3. *Пусть*

$$\propto = [\, a_0, \overline{a_1, a_2, a_1, 2a_0}\ ],$$

и степень фундаментальной S-единицы поля L есть число нечётное.

Тогда 2g+3≤ m ≤ 4g+3. Кроме того $2a_1 + a_1^2 a_2$ *делит* $a_2 + 2a_0 + 2a_0a_1a_2$ *как элемент* $Q[h^{-1}]$.

Доказательство.

1. Во-первых, согласно работе [2] получаем, что в случае нечётного m имеем

$$m = 2t+2g+1,\ t = -\nu(a_1).$$

Ясно, что t≥1, и в этом случае m≥ 2g +3.

В случае же чётного m имеем

$$m=2t+2g+2,\quad t= -2\nu(a_1)- \nu(a_2),$$

и тогда m≥2g+8.

2. Далее, поскольку

$$\nu(a_0) = -g-1,\ a_0^{\,2} \in Q[h^{-1}]\ ,$$

то отсюда заключаем, что выражение

$$(\, a_2 + 2a_0 + 2a_0a_1a_2)/(\, 2a_1 + a_1^{\,2}a_2)$$

в своём разложении не имеет членов с положительными степенями h. Поэтому имеет место второе утверждение.

3. Из условия делимости

$a_2 + 2a_0 + 2a_0a_1a_2$ на $2a_1 + a_1^{\,2}a_2$

и сравнения нормирований, заключаем, что g+1 ≥ t . А это равносильно неравенству

$$m \le 4g+3.$$

4. Для сравнения степеней необходимо выполнение соотношения

$$A_0^{\,2} + (A_2 + 2A_0 + 2A_0A_1A_2) / (2A_1 + A_1^{\,2}A_2) = 0,$$

где A_1, A_2, A_3 – младшие коэффициенты разложения a_0, a_1, a_2 по степеням h^{-1}. Это условие гарантирует, что так определённое

выражение для f в действительности является многочленом степени не выше равно 2g+1.

5. Найдем теперь условие того, чтобы это выражение было бы многочленом степени в точности 2g+1. Для этого выпишем в явном виде коэффициент при h^{2g+1}. Этот коэффициент равен в точности $2A_0A_1 + C \neq 0,$ где A_1 - коэффициент при h^{-1} у элемента a_0, C – коэффициент при h^{-1} у элемента

$$(a_2 + 2a_0 + 2a_0a_1a_2)/(2a_1 + a_1^2a_2).$$

Обозначим последний элемент через w. Сравнивая коэффициенты при h^{-1} у элементов

$$a_2 + 2a_0 + 2a_0a_1a_2 \quad \text{и} \quad w(2a_1 + a_1^2a_2),$$

можно однозначно выразить константу C.

6. Наконец, должно выполняться условие, заключающееся в том, что свободный коэффициент у многочлена f при разложении по степеням h являлся бы квадратом.

Но такое условие будет автоматически выполняться, так как оно вытекает из того, что нормирование по h величины a_0^2 строго меньше чем

$$(a_2 + 2a_0 + 2a_0a_1a_2)/(2a_1 + a_1^2a_2),$$

и нужный коэффициент всегда окажется квадратом.

А это требование равносильно тому, что соответствующее нормирование ν_h имеет два продолжения на поле L.

Работа выполнена при поддержке гранта РНФ (проект № 16-11-10111).

On some conditions on incomplete partial continued fractions associated with hyperelliptic fields and S-units

Yu. V. Kuznetsov, Yu. N. Shteynikov

Abstract: Let Q be the field of rational numbers, Q (x) be the field of rational functions of one variable, f be a square-free polynomial of odd degree equal to 2g + 1, g> 0. Suppose that for a polynomial h of degree 1 the discrete valuation defined by this polynomial has two nonequivalent extensions to the quadratic extension of the field L = Q (x) (√f). There is a connection between the properties of the decomposition of special elements of L into a continued fraction and S-units defined by finite and infinite valuation. For some small values of the length of the period, estimates are obtained for the degree of the corresponding fundamental S-units, as well as some necessary conditions that the elements of these fractions must satisfy, so that for non-trivial S-units there exist non-trivial S-units.

Keywords: continuous fractions, hyperelliptic fields, S-units, valuation.

Литература

1. В.П. Платонов, М.М. Петрунин. Фундаментальные S-единицы в гиперэллиптических полях и проблема кручения в якобианах гиперэллиптических кривых. ДАН, 2015, 465, N 1, 23-25.
2. В.П. Платонов, М.М. Петрунин. S-единицы и периодичность в квадратичных функциональных полях. УМН , 2016, Т 71, выпуск 5(431), 181–182.
3. В.П. Платонов, М.М. Петрунин. S-единицы в гиперэллиптических полях и периодичность непрерывных дробей. ДАН, 2016, Т 470, N 3, 260-265.
4. В.П. Платонов. Теоретико-числовые свойства гиперэллиптических полей и проблема кручения в якобианах гиперэллиптических кривых над полем рациональных чисел. УМН, 2014, Т. 69, N 1 (415), 3-38.
5. В.В. Беняш-Кривец, В.П. Платонов. Группы S-единиц в гиперэллиптических полях и непрерывые дроби. Мат. Сборник, 2009. Т. 200, N 11, 15--44.