ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

№ 35 (211)
2010

ISSN 2071-0216

СЕРИЯ

«МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОГРАММИРОВАНИЕ»

Выпуск 6

Содержание

Contents

УДК 519.677+517.544.8

АЛГОРИТМ ТОЧНОГО РЕШЕНИЯ ЧЕТЫРЕХЭЛЕМЕНТНОЙ ЗАДАЧИ ЛИНЕЙНОГО СОПРЯЖЕНИЯ С РАЦИОНАЛЬНЫМИ КОЭФФИЦИЕНТАМИ И ЕГО ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

В.М. Адуков, А.А. Патрушев

ALGORITHM OF EXACT SOLUTION OF GENERALIZED FOUR-ELEMENT RIEMANN – HILBERT BOUNDARY PROBLEM WITH RATIONAL COEFFICIENTS AND ITS PROGRAMM REALIZATION

V.M. Adukov, A.A. Patrushev

Предложен алгоритм точного решения четырехэлементной задачи линейного сопряжения с рациональными коэффициентами на единичной окружности. Алгоритм основан на сведении задачи к матричной краевой задаче Римана. Создана процедура, реализующая этот алгоритм в среде Maple. Используются вычисления в поле $\mathbb{Q}(i)$.

Ключевые слова: задача Маркушевича, четырехэлементная задача линейного сопряжения, матричная краевая задача Римана, явные и точные решения

An algorithm for an exact solution of the generalized four-element Riemann – Hilbert boundary problem with rational coefficients on unit cirle was suggested. An algorithm is based on a reduction of the problem to the matrix Riemann boundary problem. The Maple procedure for a realization of the algorithm is made. Calculations in the field $\mathbb{Q}(i)$ are used.

Keywords: Markushevich boundary problem, generalized four-element Riemann – Hilbert boundary problem, explicit and exact solutions

Введение

Пусть Γ – единичная окружность $|z| = 1$, D_+ – круг $|z| < 1$, D_- – дополнение замкнутого круга $|z| \leq 1$ в расширенной комплексной плоскости $\overline{\mathbb{C}} = \mathbb{C} \bigcup \{\infty\}$. Рассмотрим следующую четырехэлементную задачу линейного сопряжения аналитических функций [1]: найти функции $\varphi_+(z)$, $\varphi_-(z)$, $\varphi_-(\infty) = 0$, аналитические соответственно в областях D_+ и D_-, непрерывно продолжимые на контур Γ, граничные значения которых на Γ удовлетворяют условию сопряжения

$$a(t)\varphi_+(t) + b(t)\overline{\varphi_+(t)} = c(t)\varphi_-(t) + d(t)\overline{\varphi_-(t)} + f(t). \qquad (1)$$

Здесь коэффициенты $a(t)$, $b(t)$, $c(t)$, $d(t)$ и свободный член $f(t)$ – заданные на контуре Γ функции класса Гельдера; $\overline{\varphi(t)}$ – функция, комплексно сопряженная с $\varphi(t)$. Если $a(t) \equiv 1$,

$b(t) \equiv 0$, то получаем трехэлементную задачу, которую часто называют задачей Маркушевича. Задача, союзная к задаче Маркушевича, также имеет вид (1) при $c(t) \equiv 1$, $d(t) \equiv 0$. Классическая двухэлементная задача Римана является частным случаем задачи (1), если положить $a(t) \equiv 1$, $b(t) \equiv 0$, $d(t) \equiv 0$. Однако, далее мы считаем, что в граничном условии (1) обязательно содержится операция комплексного сопряжения, т.е. $b(t) \not\equiv 0$ или $d(t) \not\equiv 0$. В этом случае множество всех решений однородной четырехэлементной задачи является линейным простанством над полем действительных чисел $\mathbb{R}$.

Частные случаи сформулированной выше задачи имеют многочисленные приложения в теории фильтрации, электродинамике, теории оболочек [1] – [3].

Известно [4], что задача (1) нетерова, если

$$\delta(t) = \overline{a(t)}c(t) - b(t)\overline{d(t)} \neq 0$$

на контуре Г. При этом ее индекс $\varkappa$ совпадает с удвоенным индексом Коши функции $\delta(t)$:

$$\varkappa = 2\ \mathrm{ind}_\Gamma \delta(\mathrm{t}) = \frac{1}{\pi}\left[\arg \delta(\mathrm{t})\right]_\Gamma .$$

Для нетеровой задачи (1) пространство решений однородной задачи конечномерно, а неоднородная задача разрешима, если функция $f(t)$ удовлетворяет конечному числу действительных условий разрешимости. Если l – число линейно независимых решений однородной задачи, а p – число линейно независимых условий разрешимости, то $l - p = \varkappa$.

Полной теории задачи (1), как и задачи Маркушевича, в настоящее время нет. Характеристики l и p вычислены только в некоторых частных случаях, а решение в явном виде построено еще реже. Эффективный метод решения задачи Маркушевича, основанный на приведении ее к уравнению Фредгольма, разработан в статьях [5, 6].

Поскольку задача (1), вообще говоря, неустойчива, и нет эффективно проверяемых критериев ее устойчивости, то не разработаны и приближенные методы ее решения. Более того, даже если в каких-то частных случаях построен явный или эффективный алгоритм решения задачи (1), то это не гарантирует, в силу неустойчивости, что его можно использовать в вычислительных целях.

Задача (1) на окружности может быть сведена к матричной задаче Римана для двумерного вектора, а числа l и p выражаются через частные индексы матричного коэффициента этой задачи [1, 7]. Но теория матричной задачи Римана испытывает точно такие же трудности, и поэтому применить ее методы к нахождению явного решения задачи (1) до сих пор удавалось только в вырожденных случаях, когда $|a(t)| = |b(t)|$ или $|c(t)| = |d(t)|$.

Если коэффициенты $a(t)$, $b(t)$, $c(t)$, $d(t)$ – рациональные функции, то элементы матричного коэффициента соответствующей задачи Римана также будут рациональными функциями. Известно [4], что тогда задача Римана может быть решена эффективно, однако этот алгоритм решения вряд ли может быть реализован программно. В [8, 9] в более общем случае мероморфного коэффициента задачи Римана было получено ее явное решение, включающее и явное вычисление частных индексов средствами линейной алгебры [8, 9]. Была также найдена причина неустойчивости матричной задачи Римана в данной ситуации. Оказалось, что она обусловлена неустойчивостью вычислительных процедур нахождения ранга матриц и решения систем линейных однородных алгебраических уравнений. Это позволило довольно успешно бороться с неустойчивостью задачи Римана. В работе [10] в условиях, когда возможны точные вычисления в рациональной арифметике, алгоритм построения этого явного решения был реализован программно в среде Maple [10]. Кроме того, вычислительные эксперименты, проведенные в [10], показали, что использование сингулярного разложения матриц позволяет преодолеть неустойчивость и в случае приближенных вычислений.

Целью данной работы является применение этих результатов для разработки алгоритма нахождения точного решения четырехэлементной задачи Маркушевича. Под *точным*

решением данной задачи мы понимаем ее явное решение, которое использует конечное число операций точной арифметики, например, арифметики мнимого квадратичного поля $\mathbb{Q}(i)$.

Важность такого решения несомненна ввиду неустойчивости задачи. Разработанный алгоритм реализован в системе компьютерной математики Maple в виде процедуры ExactMarkushevich.

Необходимость проводить вычисления в рациональной арифметике налагает довольно обременительные условия на рациональные функции $a(t)$, $b(t)$, $c(t)$, $d(t)$, $f(t)$. Мы будем считать, что:

а) коэффициенты при степенях t в этих функциях принадлежат полю $\mathbb{Q}(i)$,

б) нули функции $\delta(t) = \overline{a(t)}c(t) - b(t)\overline{d(t)}$ и полюсы $a(t)$, $b(t)$, $c(t)$, $d(t)$, $f(t)$, лежащие в круге D_+, – также числа из $\mathbb{Q}(i)$.

В некоторых случаях, например, для классической трехэлементной задачи Маркушевича, эти требования могут быть ослаблены. Условие рациональности функций $a(t)$, $b(t)$, $c(t)$, $d(t)$, $f(t)$ не являются необходимым условием для реализации предложенного алгоритма, а вызваны неразработанностью программного обеспечения комплексного анализа в системах компьютерной математики. Возможно, это требование также в дальнейшем удастся ослабить.

1. Алгоритм точного решения

Перейдем к построению укрупненного алгоритма нахождения точного решения четырехэлементной задачи Маркушевича.

1 шаг. Сведение задачи к матричной задаче Римана [1, 7].

Решение $\varphi_+(z)$, $\varphi_-(z)$ задачи (1) мы будем рассматривать как кусочно аналитическую функцию

$$\Phi_1(z) = \begin{cases} \varphi_+(z), & z \in D_+, \\ \varphi_-(z), & z \in D_-, \end{cases}$$

заданную на $\mathbb{C} \setminus \Gamma$ и исчезающую на бесконечности. Окружность Γ является для нее линией разрыва. Функции $\varphi_\pm(z)$, аналитические в областях $D_\pm$, продолжим с помощью симметрии $z \to \dfrac{1}{\bar{z}}$ относительно окружности Γ в области $D_\mp$, определив аналитические в $D_\mp$ функции $\overline{\varphi_\pm\left(\overline{z}^{-1}\right)}$. Зададим кусочно аналитическую функцию

$$\Phi_2(z) = z^{-1}\overline{\Phi_1\left(\overline{z}^{-1}\right)} = \begin{cases} z^{-1}\overline{\varphi_-\left(\overline{z}^{-1}\right)}, & z \in D_+, \\ z^{-1}\overline{\varphi_+\left(\overline{z}^{-1}\right)}, & z \in D_-. \end{cases}$$

Множитель z^{-1} введен здесь для того, чтобы выполнялось условие $\Phi_2(\infty) = 0$. Для краткости обозначим $\Phi_1^*(z) = z^{-1}\overline{\Phi_1\left(\overline{z}^{-1}\right)}$. Ясно, что $\Phi_1(z) = \Phi_2^*(z)$.

Функция $\Phi_2(z)$ непрерывно продолжается на контур Γ из областей $D_\pm$, и для ее граничных значений $\Phi_2^\pm(t)$ справедливо

$$\Phi_2^+(t) = \overline{t\varphi_-(t)}, \quad \Phi_2^-(t) = \overline{t\varphi_+(t)}.$$

Присоединим к уравнению (1) уравнение, полученное применением к (1) операции комплексного сопряжения. Полученная система для функций $\Phi_1^\pm(t), \Phi_2^\pm(t)$ может быть записана в виде матричной задачи Римана

$$\Phi^+(t) = G(t)\Phi^-(t) + F(t), \tag{2}$$

где $\Phi(z) = \begin{pmatrix}\Phi_1(z)\\ \Phi_2(z)\end{pmatrix}$ – кусочно аналитический вектор, имеющий граничные значения $\Phi^{\pm}(t) = \begin{pmatrix}\Phi_1^{\pm}(t)\\ \Phi_2^{\pm}(t)\end{pmatrix}$,

$$G(t) = \frac{1}{\overline{\delta(t)}}\begin{pmatrix} |c(t)|^2 - |d(t)|^2 & t[\overline{a(t)}d(t) - \overline{c(t)}b(t)] \\ -\dfrac{a(t)\overline{d(t)} - \overline{b(t)}c(t)}{t} & |a(t)|^2 - |b(t)|^2 \end{pmatrix},$$

$$F(t) = \frac{1}{\overline{\delta(t)}}\begin{pmatrix} \overline{c(t)}f(t) - d(t)\overline{f(t)} \\ \dfrac{\overline{b(t)}f(t) - \overline{f(t)}a(t)}{t} \end{pmatrix}.$$

Для кусочно аналитического вектора $\Phi(z) = \begin{pmatrix}\Phi_1(z)\\ \Phi_2(z)\end{pmatrix}$ обозначим $\Phi^*(z) = \begin{pmatrix}\Phi_2^*(z)\\ \Phi_1^*(z)\end{pmatrix}$. Если $\Phi(z) = \Phi^*(z)$, то вектор $\Phi(z)$ будем называть *симметричным*. Таким образом, каждое решение $\Phi_1(z)$ задачи (1) порождает симметричное решение матричной задачи Римана (2).

Легко проверить, что матричный коэффициент $G(t)$ удовлетворяет условию

$$G(t)J\overline{G(t)}J = I, \tag{3}$$

где $J = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, I – единичная матрица второго порядка, а свободный член $F(t)$ – такой, что

$$G(t)J\overline{F(t)} + tF(t) = 0. \tag{4}$$

Эти два условия гарантируют существование симметричных решений задачи (2) в случае, когда эта задача разрешима. В самом деле, если $\Phi(z)$ – произвольное решение (2), то при выполнении условий (3) – (4) вектор $\Phi^*(z)$ – также решение (2), и тогда $\frac{1}{2}[\Phi(z) + \Phi^*(z)]$ – симметричное решение.

Ясно, что для любого симметричного решения $\Phi(z) = \begin{pmatrix}\Phi_1(z)\\ \Phi_1^*(z)\end{pmatrix}$ задачи (2) функция $\Phi_1(z)$ будет решением задачи (1).

Итак, четырехэлементная задача (1) эквивалентна задаче отыскания всех симметричных решений матричной задачи Римана (2).

2 шаг. Построение канонической матрицы для матричной задачи Римана.

Известно (см., например, [4]), что, если найдена так называемая *каноническая матрица* $X(z)$ задачи (2), то легко проверить условия разрешимости этой задачи и построить ее общее решение. В свою очередь, нахождение $X(z)$ равносильно построению левой факторизации матричного коэффициента $G(t)$:

$$G(t) = G_+(t)d(t)G_-(t).$$

Здесь $G_{\pm}(t)$ – матрицы-функции, аналитические и обратимые в областях $D_{\pm}$, напрерывно продолжимые на контур Γ, а $d(t) = \mathrm{diag}[t^{\kappa_1}, t^{\kappa_2}]$ – диагональная матрица-функция, $\kappa_1 + \kappa_2 = \kappa = 2\ \mathrm{ind}_{\Gamma}\delta(t)$. Целые числа $\kappa_1 \geq \kappa_2$ называются *частными индексами*. Они являются важными инвариантами задачи (2), а, следовательно, и (1). Каноническая матрица $X(z)$ в терминах факторизации строится следующим образом:

$$X(z) = \begin{cases} G_+(z), & z \in D_+, \\ G_-^{-1}(z)d^{-1}(z), & z \in D_-. \end{cases}$$

Мы предполагаем, что коэффициенты $a(t)$, $b(t)$, $c(t)$, $d(t)$ задачи (1) – рациональные функции. На единичной окружности выполняется условие $\overline{t} = \dfrac{1}{t}$. Поэтому функции, комплексно сопряженные к рациональным, также будут рациональными, и, следовательно, матричный коэффициент $G(t)$ задачи (2) – рациональная матрица-функция.

В этом случае задача факторизации может быть решена явно [8]. Алгоритм точного решения описан в работе [10]. Точное решение предполагает вычисления в рациональной арифметике. Это объясняет ограничения а), б), наложенные на $a(t)$, $b(t)$, $c(t)$, $d(t)$, $\delta(t)$. При данных ограничениях алгоритм реализован программно в среде Maple в виде процедуры ExactFactorization [10]. Воспользовавшись этой процедурой, мы можем построить каноническую матрицу $X(z)$, которая, очевидно, также является рациональной матрицей-функцией.

3 шаг. Построение общего решения однородной четырехэлементной задачи.

В соответствии с вышесказанным, для нахождения всех решений однородной задачи (1) требуется найти множество всех симметричных решений однородной задачи Римана (2). Легко видеть, что это множество является конечномерным пространством над полем $\mathbb{R}$, и, значит, нам необходимо построить базис $\Phi^1(z) = \begin{pmatrix}\varphi_1(z)\\ \varphi_1^*(z)\end{pmatrix}, \ldots, \Phi^l(z) = \begin{pmatrix}\varphi_l(z)\\ \varphi_l^*(z)\end{pmatrix}$ этого пространства. Тогда кусочно аналитические функции $\varphi_1(z), \ldots, \varphi_l(z)$ будут образовывать базис пространства решений однородной задачи (2).

Если частные индексы κ_1, κ_2 неположительны, то однородная задача (2), а, значит, и однородная задача (1) допускает в классе исчезающих на бесконечности кусочно аналитических функций только нулевое решение.

Если среди частных индексов имеются положительные, то размерность λ (над $\mathbb{C}$) пространства решений однородной задачи (2) совпадает с суммой положительных частных индексов. При этом, если $\kappa_1 > 0$, $\kappa_2 \le 0$, то базисом является система кусочно аналитических векторов $X^1(z), zX^1(z), \ldots, z^{\kappa_1-1}X^1(z)$, а при $\kappa_1 > 0$, $\kappa_2 > 0$ – система $X^1(z)$, $zX^1(z), \ldots, z^{\kappa_1-1}X^1(z), X^2(z), zX^2(z), \ldots, z^{\kappa_2-1}X^2(z)$. Здесь $X^1(z)$, $X^2(z)$ – соответственно первый и второй столбцы канонической матрицы $X(z)$. Далее этот базис мы обозначаем через $\Psi^1(z), \ldots, \Psi^\lambda(z)$, $\lambda = \kappa_1$ или $\lambda = \kappa_1 + \kappa_2$.

Условие (3) гарантирует нам, что векторы $(\Psi^1(z))^*, \ldots, (\Psi^\lambda(z))^*$ также являются решениями однородной задачи (2). Это позволяет нам построить 2λ симметричных решений этой задачи:

$$\frac{1}{2}\left[\Psi^1(z) + (\Psi^1(z))^*\right], \ldots, \frac{1}{2}\left[\Psi^\lambda(z) + (\Psi^\lambda(z))^*\right], \tag{5}$$
$$\frac{i}{2}\left[\Psi^1(z) - (\Psi^1(z))^*\right], \ldots, \frac{i}{2}\left[\Psi^\lambda(z) - (\Psi^\lambda(z))^*\right].$$

Легко видеть, что система (5) является полной в пространстве над $\mathbb{C}$ всех решений однородной задачи (2) и полной в пространстве над $\mathbb{R}$ всех симметричных решений однородной задачи (2). Поэтому ранг над $\mathbb{C}$ системы (5) равен λ. С другой стороны, легко проверить, что линейная независимость над $\mathbb{C}$ любой системы симметричных векторов равносильна линейной независимости над $\mathbb{R}$. Поэтому ранг системы (5) над $\mathbb{R}$ также равен λ. По этой причине мы можем далее не уточнять, над каким полем рассматривается линейная независимость любой подсистемы системы (5).

Таким образом, размерность l пространства всех симметричных решений однородной задачи (2), а, значит, и пространства всех решений однородной задачи (1) совпадает с суммой λ положительных частных индексов.

Осталось выбрать базис системы (5). Из условия сопряжения $\Phi^+(t) = G(t)\Phi^-(t)$ следует, что линейную независимость любой подсистемы системы (5) достаточно проверять только в области D_+, т.е. для «плюс»-компонент кусочно аналитических векторов. Эти компоненты являются рациональными вектор-функциями, не имеющими полюсов в $\overline{D}_+$. Поэтому,

умножив все векторы системы (5) на подходящий полином, мы сведем дело к проверке линейной независимости полиномиальных векторов. Пусть N – максимальная из степеней полиномов, входящих в преобразованную систему (5). Каждый полином из этой системы мы будем рассматривать как полином формальной степени N и поставим ему в соответствие $(N+1)$-мерный вектор, составленный из коэффициентов этого полинома. Таким образом, системе (5) мы поставим в соответствие систему, состоящую из $(2N+2)$-мерных векторов. Базис этой системы можно найти стандартыми методами линейной алгебры. В системе Maple для этих целей служит процедура Basis. Найдя номера векторов, входящих в базис, мы можем восстановить и базис системы (5). Таким образом, алгоритм нахождения базиса пространства решений однородной задачи (1) построен.

4 шаг. Построение общего решения неоднородной четырехэлементной задачи.

Чтобы найти общее решение неоднородной четырехэлементной задачи (1), нам нужно выяснить, когда эта задача разрешима, и отыскать ее частное решение. Как уже отмечалось при описании шага 1 алгоритма, неоднородная задача (1) разрешима тогда и только тогда, когда разрешима соответствующая матричная задача Римана (2). Известно (см., например, [4]), что задача (2) разрешима безусловно (т.е. при любой правой части $F(t)$), только если индексы κ_1, κ_2 являются неотрицательными. В этом случае кусочно аналитический вектор $\Psi(z) = X(z)\Omega(z)$ является решением неоднородной задачи (2). Здесь вектор $\Omega(z)$ находится по формуле

$$\Omega(z) = \frac{1}{2\pi i}\int_\Gamma \frac{G_+^{-1}(t)F(t)}{t-z}dt.$$

В нашем случае, когда элементы матрицы $G_+^{-1}(t)$ и вектора $F(t)$ являются рациональными функциями, мы можем найти компоненты $\Omega_\pm(z)$ кусочно аналитического вектора $\Omega(z)$, не используя интегралы типа Коши. Пусть $R(z)$ – рациональная функция, имеющая полюсы в точках $z_1,\dots,z_m$ области D_+, и $G_1(z),\dots,G_m(z)$ – главные части рядов Лорана $R(z)$ в окрестности этих точек. Тогда $R_-(z) = -G_1(z) - \cdots - G_m(z)$ – аналитическая в D_- функция, исчезающая на бесконечности, а $R_+(z) = R(z) + R_-(z)$ – функция, аналитическая в D_+. Таким образом, разложение $R(z) = R_+(z) - R_-(z)$ является разложением по формулам Сохоцкого и

$$\frac{1}{2\pi i}\int_\Gamma \frac{R(t)}{t-z}dt = \begin{cases} R_+(z), & z\in D_+,\\ R_-(z), & z\in D_-. \end{cases} \tag{6}$$

Этот способ построения кусочно рациональной функции реализован в виде процедуры Sokhotckii. Применив ее к элементам вектора $G_+^{-1}(t)F(t)$, мы найдем кусочно аналитический вектор $\Omega(z)$ и получим решение неоднородной задачи Римана в виде

$$\Psi(z) = \begin{cases} G_+^{-1}(z)\Omega_+(z), & z\in D_+,\\ G_-^{-1}(z)d^{-1}(z)\Omega_-(z), & z\in D_-. \end{cases} \tag{7}$$

Теперь мы можем отыскать симметричное решение

$$\frac{1}{2}\left[\Psi(z)+\Psi^*(z)\right] = \frac{1}{2}\begin{pmatrix}\Psi_1(z)+\Psi_2^*(z)\\ \Psi_2(z)+\Psi_1^*(z)\end{pmatrix}$$

задачи (2) и решение $\frac{1}{2}\left[\Psi_1(z)+\Psi_2^*(z)\right]$ неоднородной задачи (1).

Если среди индексов κ_1, κ_2 есть отрицательные, то неоднородная задача (2) разрешима тогда и только тогда, когда правая часть $F(t)$ удовлетворяет некоторым условиям разрешимости. При $\kappa_1 \geq 0$, $\kappa_2 < 0$ эти условия состоят в том, что элемент $\Omega_2(z)$ кусочно аналитического вектора $\Omega(z)$ должен иметь в бесконечно удаленной точке нуль порядка не ниже, чем $|\kappa_2|+1$. Если же $\kappa_1 < 0$, $\kappa_2 < 0$, то к этому условию добавляется требование, что $\Omega_1(z)$ имеет в бесконечности нуль порядка не ниже, чем $|\kappa_1|+1$. Таким образом, число p условий

разрешимости задачи (2) равно сумме отрицательных частных индексов, взятой с обратным знаком. Для проверки выполнимости данных условий мы разлагаем рациональные функции $\Omega_j^-(z)$ в ряд Лорана в окрестности бесконечности и проверяем равенство нулю коэффициентов при $\frac{1}{z}, \ldots, \frac{1}{z^{|\kappa_j|}}$, $j = 1, 2$. При выполнении условий разрешимости вектор-функция (7) по-прежнему является решением неоднородной задачи (2), а, значит, $\frac{1}{2}\left[\Psi_1(z) + \Psi_2^*(z)\right]$ будет решением исходной задачи (1).

2. Описание процедуры

Опишем теперь процедуру ExactMarkushevich, реализующую вышеуказанный алгоритм. Ей передается пять параметров a, b, c, d, f – коэффициенты задачи (1) и ее свободный член. Перед обращением к процедуре требуется подключить пакет LinearAlgebra. Приведем пример обращения к процедуре

```
>with(LinearAlgebra)
>a:=1; b:=0; c:=t; d:=(1+3*t^3)/t^4; f:=1/(2*t+1)^4;
>ExactMarkushevich(a,b,c,d,f).
```

Поскольку точные вычисления предполагают использование рациональной арифметики, то должны быть выполнены условия а), б). Выполнимость а) обеспечивает пользователь. Процедура проверяет выполнение условия б) и условие нетеровости $\delta(t) \neq 0$. Если они нарушены, то процедура выдает соответствующее сообщение и заканчивает работу.

Процедура возвращает числа l, p и базис пространства решений однородной задачи, если это пространство нетривиально. Кусочно аналитические функции, входящие в базис, представлены двумерными строками, в которых первый элемент есть компонента, аналитическая в области D_+, а второй – компонента, аналитическая в D_-.

Условие б) для свободного члена $f(t)$ и условия разрешимости задачи процедура проверяет после решения однородной задачи. При невыполнении их процедура выдает соответствующие сообщения и заканчивает работу. В случае разрешимости задачи процедура возвращает частное решение неоднородной задачи в виде двумерной строки.

Пример 1. Приведем пример точного решения неустойчивой неоднородной задачи Маркушевича с помощью процедуры ExactMarkushevich. Зададим

$$a(t) = 1,\ b(t) = 0,\ c(t) = t,\ d(t) = \frac{1+3t^3}{t^4},\ f(t) = \frac{1}{(2t+1)^4}.$$

В результате работы процедуры мы получаем значения $l = 4$, $p = 2$ и базис

$$\left(-\frac{3}{2} + \frac{3t^3}{2},\ \frac{1}{2t^4} - \frac{1}{2t}\right), \left(-\frac{3t}{2} + \frac{3t^2}{2},\ \frac{1}{2t^3} - \frac{1}{2t^2}\right),$$

$$\left(-\frac{3i}{2} - \frac{3it^3}{2},\ \frac{i}{2t^4} + \frac{i}{2t}\right), \left(-\frac{3it}{2} - \frac{3it^2}{2},\ \frac{i}{2t^3} + \frac{i}{2t^2}\right)$$

пространства решений однородной задачи Маркушевича. Далее процедура проверяет условия разрешимости неоднородной задачи Маркушевича, определяет, что данная задача разрешима, и возвращает ее частное решение

$$\left(-\frac{t(2+3t^3)}{2(t+2)^4},\ \frac{3t^3-2}{2t(2t+1)^4}\right).$$

Работа проводилась при финансовой поддержке РФФИ-Урал, грант №07-01-96010

Литература

1. Литвинчук, Г.С. Краевые задачи и сингулярные интегральные уравнения со сдвигом / Г.С. Литвинчук. – М.: Наука, 1977. – 448 с.
2. Емец, Ю.П. Точное решение задачи о формировании тока в двоякопериодической гетерогенной системе / Ю.П. Емец, Ю.В. Обносов // Доклады АН СССР. – 1990. – Т. 309, № 2. – С. 319 – 322.
3. Векуа, И.Н. Обобщенные аналитические функции / И.Н. Векуа. – М.: Физматгиз, 1959. – 628 с.
4. Векуа, Н.П. Системы сингулярных интегральных уравнений / Н.П. Векуа. – М.: Наука, 1970. – 380 с.
5. Расулов, К.М. Об одном методе решения граничной задачи Маркушевича в классе аналитических функций / К.М. Расулов // Исследования по краевым задачам комплексного анализа и дифференциальным уравнениям: межвуз. сб. науч. тр. – Смоленск, 2001. – Вып. 3. – С. 98 – 108.
6. Расулов, К.М. О решении обобщенной граничной задачи Маркушевича в классе аналитических функций / К.М. Расулов // Системы компьютерной математики и их приложения: сб. тр. Междунар. науч. конф. – Смоленск, 2002. – С. 137 – 142.
7. Чибрикова, Л.П. К решению одной общей задачи линейного сопряжения аналитических функций в случае алгебраических контуров / Л.И. Чибрикова, Л.Г. Салехов // Тр. семинара по краев. задачам. – Казань, 1968. – Вып. 5. – С. 224 – 249.
8. Адуков, В.М. Факторизация Винера – Хопфа мероморфных матриц-функций / В.М. Адуков // Алгебра и анализ. – 1992. – Т. 4, вып 1. – С. 54 – 74.
9. Адуков, В.М. Факторизация Винера – Хопфа кусочно мероморфных матриц-функций / В.М. Адуков // Матем. сборник. – 2009. – Т. 200, № 8. – С. 3 – 24.
10. Адуков, В.М. О точном и приближенном решении задачи факторизации Винера – Хопфа для мероморфных матриц-функций / В.М. Адуков // Вестн. Юж.-Урал. гос. ун-та, серия «Математика, физика, химия». – 2008. – № 7(107), вып. 10. – С. 3 – 12.

Адуков Виктор Михайлович, доктор физико-математических наук, кафедра дифференциальных уравнений и динамических систем, Южно-Уральский государственный университет, avm@susu.ac.ru.

Патрушев Алексей Алексеевич, кафедра общей математики, Южно-Уральский государственный университет, avm@susu.ac.ru.

Поступила в редакцию 22 июня 2010 г.

УДК 681.5.015.23

ОПЫТ РЕШЕНИЯ ЗАДАЧИ ПАРАМЕТРИЧЕСКОГО ОЦЕНИВАНИЯ ЦИФРОВЫХ МОДЕЛЕЙ НЕФТЯНОГО МЕСТОРОЖДЕНИЯ

А.В. Гагарин, Г.А. Макеев, Р.А. Байков, В.Г. Волков

AN EXPERIENCE OF PARAMETRIC ESTIMATION OF DIGITAL OIL RESERVOIR MODELS

A.V. Gagarin, G.A. Makeev, R.A. Baikov, V.G. Volkov

В статье рассматривается автоматизированная система идентификации параметров цифровых моделей нефтяного месторождения. Исследуется применение методов оптимизации общего назначения, разработанных интеллектуальных алгоритмов оптимизации и других инструментов для поиска решения, анализа чувствительности и взаимозависимостей между искомыми параметрами. Предложенная система может быть использована в любых инженерных приложениях, где значение целевой функции зависит от результатов ресурсоемких расчетов.

Ключевые слова: идентификация параметров, оптимизация, генетические алгоритмы, нейронные сети, гидродинамическое моделирование

A system that is designed to identify parameters of digital oil reservoir models is presented in this paper. An application of common optimization methods, proposed intelligent optimization algorithms and other tools for finding solution, making sensitivity analysis and searching for hidden dependencies between parameters of interest is investigated. The system developed is suited for solving optimization tasks with fitness function value depending on results of resource consuming calculations.

Keywords: parameters' identification, an optimization, genetic algorithms, neural networks, hydrodynamic models

Введение

Решение задач параметрического оценивания и анализа чувствительности геолого-гидродинамических моделей (ГДМ) принципиально требует проведения массовых расчетов моделей с привлечением всех доступных вычислительных ресурсов, что сопряжено со значительными временными затратами и проведением большого объема рутинной работы специалистов по моделированию. Общий поток работ включает в себя, в частности:

- генерацию множества вариантов некоторой модели;
- распределение задач по доступным вычислительным мощностям;
- сбор и обработку результатов выполнения моделей.

Для формализации задачи параметрического оценивания предположим, что объект исследования (ОИ) характеризуется контролируемыми выходными сигналами $\mathbf{y}(k) = (y_1(k), y_2(k), \ldots, y_m(k))$, которые регистрируются в ходе натурного эксперимента в дискретные моменты времени t_k, $k = 1, 2, \ldots$ [11, 12]. Им соответствуют выходные сигналы $\hat{\mathbf{y}}(k \mid \mathbf{x}) = (\hat{y}_1(k \mid \mathbf{x}), \hat{y}_2(k \mid \mathbf{x}), \ldots, \hat{y}_m(k \mid \mathbf{x}))$ цифровой модели ОИ, определяемой вектором параметров $\mathbf{x}$. Невязкой $\varepsilon(k, \mathbf{x})$ называется разность между выходными сигналами ОИ $\mathbf{y}(k)$ и настраиваемой модели $\hat{y}_i(k \mid \mathbf{x})$: $\varepsilon(k, \mathbf{x}) = \mathbf{y}(k) - \hat{\mathbf{y}}(k \mid \mathbf{x})$. Пусть имеются данные наблюдения $\mathbf{y}(1), \ldots, \mathbf{y}(N)$ за ОИ в N различных моментов времени. Тогда задача параметрического оценивания модели заключается в выборе из всего множества параметров $\mathbf{x} \in D_M$ некоторого оптимального вектора $\mathbf{x}^{\text{opt}}$ таким образом, чтобы невязка $\varepsilon(k, \mathbf{x}^{\text{opt}})$, $k = 1, \ldots, N$ была по возможности мала.

Как правило, данная задача сводится к задаче минимизации целевой функции (ЦФ) вида:

$$f(\mathbf{x}) = \sum_{k=1}^{N} \varepsilon(k, \mathbf{x})^T \cdot \mathbf{W} \cdot \varepsilon(k, \mathbf{x}), \tag{1}$$

где $\mathbf{W}$ — матрица весовых коэффициентов.

Данная функция является мерой близости настраиваемой модели к реальному ОИ и отражает качество идентификации ее параметров. Соответственно,

$$\mathbf{x}^{\text{opt}} = \underset{\mathbf{x} \in D_M}{\operatorname{argmin}} f(\mathbf{x}).$$

На практике ищется минимум $f(\mathbf{x})$ с заданным порогом δ, позволяющим контролировать точность получаемого результата. Иными словами, необходимо найти решение $\mathbf{x}^*$ такое, что:

$$f(\mathbf{x}^*) < f\left(\mathbf{x}^{\text{opt}}\right) + \delta. \tag{2}$$

Для нахождения минимума $f(\mathbf{x})$, удовлетворяющего условию (2), обычно применяются итерационные методы оптимизации, генерирующие последовательность приближений $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N_o)}\right\}$ такую, что $\mathbf{x}^{(N_o)} \in D_M^*$ (то есть $\mathbf{x}^{(N_o)}$ удовлетворяет условию (2)). Значение индекса N_o показывает количество приближений, требуемых для нахождения оптимума ЦФ с заданным порогом. На практике, в дополнение к критерию (2) часто применяется ограничение по времени, когда оптимизация останавливается по условию $N_o = N_o^{\max}$.

В зависимости от сложности модели и величины N, ресурсоемкость вычисления $\hat{\mathbf{y}}(k \mid \mathbf{x})$, $k = 1, \ldots, N$, может быть велика. Например, время численного моделирования течения многофазной смеси жидкости и газа в пористой среде, широко используемое в моделях нефтяных месторождений, даже на современных суперкомпьютерах с применением коммерческих симуляторов, таких как Shlumberger ECLIPSE, Roxar Tempest MORE, NGT BOS[1] и т. д., может исчисляться сутками. Задача оптимизации ГДМ является примером некорректно поставленной обратной задачи с ЦФ вида (1). У нее может не существовать решения, может существовать больше одного решения, и неизвестна устойчивость решения. В таких условиях одним из методов решения могут быть методы глобальной и локальной оптимизации общего назначения, такие как градиентные и генетические алгоритмы [1–3, 6]. Среди особенностей задачи моделирования можно выделить большое количество искомых параметров и возможную неявную зависимость между этими параметрами.

В связи с этим большую актуальность приобретает задача автоматизации процесса многовариантного моделирования, то есть полного цикла генерации вариантов моделей, их рас-

[1]Разработка ООО «РН-УфаНИПИнефть»

чета и обработки результатов для решения различных прикладных задач. Результатом работы должна стать единая универсальная среда выполнения возможных сценариев для оптимизации моделей, пользуясь которой специалисты смогут сосредоточиться на решении только принципиальных задач моделирования.

1. Описание разработанной системы

Разрабатываемая система автоматизированной оптимизации ГДМ основана на совместном использовании вычислительной среды MATLAB и гидродинамического симулятора NGT BOS Core. Типичный цикл оптимизации модели с помощью этой системы включает следующие этапы (рис. 1).

На первом этапе производится выбор варьируемых параметров модели, определяется вид ЦФ, и способ преобразования аргументов целевой функции в варьируемые параметры модели. Далее под параметрами понимаются именно аргументы ЦФ.

На втором этапе определяется область поиска оптимальных значений параметров (границы) и шаги дискретизации области поиска по всем параметрам.

На третьем этапе анализируется чувствительность ЦФ к искомым параметрам, определяются однородные и неоднородные области параметрического пространства.

На четвертом этапе с помощью алгоритмов оптимизации общего назначения (генетических, градиентных и т.д.) система самостоятельно выполняет цикл, в котором генерируются ГДМ, соответствующие приближениям $\mathbf{x}^{(\nu)}$, производится численное моделирование, необходимое для вычисления ЦФ и обрабатываются результаты расчета.

На пятом этапе анализируются найденные решения и кластеры решений, определяется их устойчивость.

И на шестом этапе строятся отчеты по лучшим найденным решениям и интегральные отчеты в разрезе всех рассчитанных решений.

1.1. Матрица «параметры-оценки»

Параметры и соответствующие им рассчитанные значения ЦФ в системе формируют матрицу параметров и оценок (расширенный и отформатированный вид ее приведен на рис. 2). Матрица имеет $n+l$ столбцов и e строк, где n — количество параметров, а l — количество оценок. Каждая строка соответствует одному эксперименту (одной ГДМ). Все модели различаются только значениями n варьируемых параметров, а в остальном идентичны.

Добавление новой строки в матрицу «параметры-оценки» приводит к автоматическому

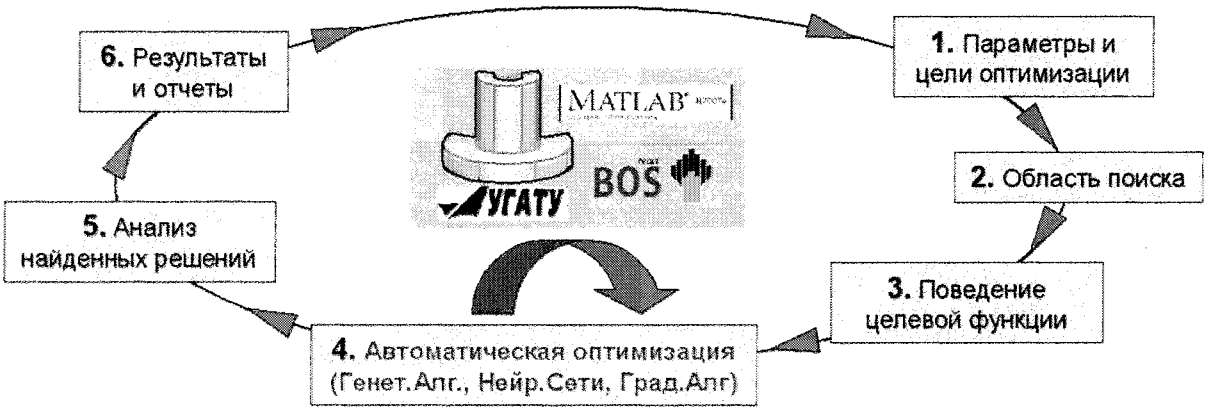


Рис. 1. Общая структура системы

	Параметры					Оценки			
	P1	P2		PN	Состояние	E1	E2	...	EM
N эксп.	Забойное давление там-то	Длина трещины такой-то	...	Признак установки эквалайзера		Общая добыча нефти	Газовый фактор на скважине такой-то		Средняя обводненность
1	250	50		1	[illegible]	120	1		0.7
2	300	75		1	[illegible]	130	1		0.7
					...				
E	250	50		1	**рассчитывается**				

Рис. 2. Пример матрицы параметров и оценок

созданию новой гидродинамической модели. Система позволяет произвести численное моделирование как на локальном компьютере, так и на кластере. После проведения численного моделирования вычисляется значение ЦФ, соответствующее данной ГДМ.

Поскольку матрица «параметры-оценки» доступна в среде MATLAB, то добавляться новые строки могут:

- на основе приближений $x^{(\nu)}$, генерируемых алгоритмом оптимизации;
- вручную, если специалисту требуется проверить модель с заданными набором параметров;
- функцией группового добавления: например, в соответствии с алгоритмом планирования эксперимента (экспериментальный дизайн).

Важной особенностью матрицы «параметры-оценки» является то, что в ней не может быть двух моделей, все параметры которых совпадают. Это означает, что численное моделирование с уникальным набором параметров модели производится только один раз.

1.2. Параметризация модели

Система позволяет использовать в качестве аргумента ЦФ любое значение, прямо или опосредованно влияющее на параметры ГДМ. С точки зрения системы любой аргумент ЦФ имеет свою область определения и шаг дискретизации. С помощью языка MATLAB можно производить сложные преобразования аргументов ЦФ в параметры ГДМ. Формально, преобразование аргументов ЦФ в параметры модели в разработанной системе выполняется с помощью двух механизмов:

- подстановка на место параметра ГДМ аргумента целевой функции;
- вызов некоторой функции MATLAB, осуществляющей преобразование аргументов ЦФ в параметры ГДМ.

1.3. Применяемые алгоритмы оптимизации

Поскольку задача параметрического оценивания модели формулируется в виде задачи оптимизации, важнейшую роль в системе играет используемый алгоритм оптимизации, в значительной степени определяющий способность исследования пространства поиска, скорость поиска решения, возможность нахождения глобального оптимума и т.д.

Разработанная система построена по модульному принципу и позволяет использовать различные оптимизационные блоки, обладающие одинаковым программным интерфейсом (API). В частности, на данный момент система включает реализации градиентного поиска, усовершенствованных генетических алгоритмов (ГА), эвристического алгоритма, основанного на методе кригинга (Kriging), алгоритма Монте-Карло. Кроме того, имеются также

модули планирования эксперимента, факторного анализа, анализа чувствительности и зависимостей между параметрами и оценками.

Поскольку нахождение параметров адекватной модели, обладающей хорошими прогнозными свойствами — нетривиальная задача, которую невозможно полностью автоматизировать, специалист по моделированию может на различных уровнях настраивать поведение системы. В частности, именно он определяет оптимизационное ядро для решения прикладных задач. Однако, на практике в общем случае ЦФ имеет очень сложную форму, и область локализации ее глобального оптимума неизвестна, поэтому целесообразно на начальном этапе использовать алгоритмы с возможностями глобальной оптимизации и уже на этапе уточнения решения переходить к методам локального поиска.

1.3.1. Гибридный генетический нейросетевой алгоритм

Хорошо известно, что использование стандартного ГА предполагает на каждой итерации для каждого набора параметров (хромосомы в терминах ГА) вычисление значений ЦФ, включающее в себя моделирование ОИ (то есть расчет значений $\hat{\mathbf{y}}(k \mid \mathbf{x})$). Разработанный *гибридный генетический нейросетевой алгоритм* (ГА+НС) является модификацией стандартного ГА, в дальнейшем называемого *главным*, на каждой итерации которого создается нейросетевая аппроксимация $\widetilde{f}(\mathbf{x})$ целевой функции $f(\mathbf{x})$, предназначенная для получения прогноза $\widetilde{\mathbf{x}}^*$ оптимального решения функции $f(\mathbf{x})$. В большинстве случаев $\widetilde{f}(\mathbf{x})$ будет вычисляться значительно быстрее $f(\mathbf{x})$, поскольку не требует предварительного расчета модели. Поиск $\widetilde{\mathbf{x}}^*$ осуществляется с помощью дополнительного ГА, в дальнейшем называемого *вспомогательным*, для которого в качестве ЦФ используется $\widetilde{f}(\mathbf{x})$ (то есть значения, вычисляемые нейронной сетью). Найденный прогноз затем добавляется в популяцию потомков главного ГА на текущей итерации, и цикл повторяется.

Адекватный прогноз $\widetilde{\mathbf{x}}^*$ оптимального решения может значительно ускорить эволюционный поиск основного ГА. Если прогноз окажется неудачным, эволюционные механизмы обеспечат стабильность оптимизации. Приближения $\mathbf{x}$, генерируемые основным ГА на протяжении всего процесса оптимизации, вместе с соответствующими значениями ЦФ $f(\mathbf{x})$ участвуют в формировании обучающей выборки для НС (стандартный ГА хранит лишь текущую популяцию фиксированного размера). Вспомогательный ГА может выполнять значительно более интенсивный поиск по сравнению с главным ГА (большой размер популяции, много итераций и т. п.) и с большой вероятностью находить глобальный оптимум $\widetilde{f}$, так как данная функция вычисляется сравнительно быстро.

В современной литературе методы идентификации систем, основанные на замене сложной модели реального ОИ на значительно менее ресурсоемкую модель, приближенно воспроизводящую отклик исходной модели, называются суррогатным моделированием или метамоделированием [8,10]. Суррогатные модели ОИ обычно строятся с минимальным привлечением данных из предметной области на основе имеющихся вычислительных экспериментов с исходной моделью или данных наблюдения за реальным ОИ с помощью различных методов построения аппроксимации многомерных зависимостей (аппроксимирующие полиномы, сплайны, радиально-базисные функции, кригинг, нейронные сети и т. д.). Прямое применение данного подхода с использованием НС для построения суррогатной модели путем обучения на заранее подготовленной обучающей выборке, и дальнейшего применения ГА для идентификации параметров суррогатной модели описано, например, в работах [5,7,9]. Такую схему можно рассматривать как одну итерацию нейросетевого контура предлагаемого алгоритма ГА+НС. Отличительной особенностью ГА+НС является то, что НС используется не для построения отображения $(\mathbf{x}, k) \mapsto \hat{\mathbf{y}}(k \mid \mathbf{x})$, а для аппроксимации ЦФ (которая, в свою очередь зависит от $\mathbf{y}(k)$ и $\hat{\mathbf{y}}(k \mid \mathbf{x})$). Кроме того, предлагаемый алгоритм не требует

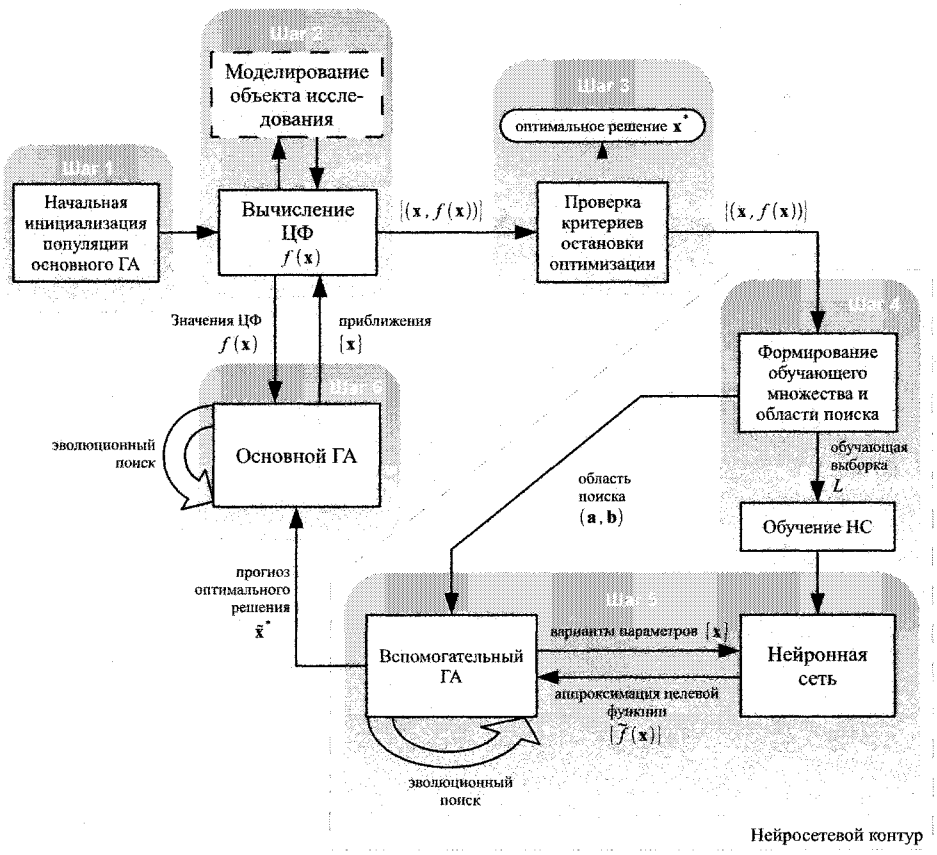


Рис. 3. Схема алгоритма ГА+НС

наличия априорно заданной обучающей выборки, нейросетевая аппроксимация $\tilde{f}$ строится динамически на каждой итерации главного ГА на основе имеющегося на данный момент множества пар $\{(\mathbf{x}, f(\mathbf{x}))\}$.

Наиболее близкий аналог ГА+НС предлагается в работе [4]. Гибридный алгоритм в [4] не содержит вспомогательного ГА, поскольку на вход НС подается значение ЦФ, а на выходе получается прогноз соответствующего вектора искомых параметров. Однако такая схема имеет очевидный недостаток, заключающийся в некорректности обратной задачи, решаемой НС.

1.3.2. Кригинг-оптимизатор

Кригинг – семейство геостатистических алгоритмов интерполяции неизвестной случайной величины в некоторой точке по известным реализациям этой случайной величины в некоторых других точках. В общем виде значение $Z^*(u)$ в интерполируемой точке u определяется следующим образом: через известные реализации случайных величин в точках u_α, $\alpha = \overline{1,n}$:

$$Z^*(u) = m(u) + \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u) \left(Z(u_\alpha) - m(u_\alpha)\right),$$

где $Z(u_\alpha), Z(u)$ — реализации случайных величин в точках u и u_α;
$m(u_\alpha), m(u)$ — математические ожидания этих случайных величин;
$\lambda_\alpha(u)$, $\alpha = \overline{1,n}$ — веса известных реализаций , меняющиеся при переходе к следующей точке u.

Основная идея кригинга в том, что веса $\lambda_\alpha(u)$, $\alpha = \overline{1,n}$ выбираются так, чтобы интерполируемое значение $Z^*(u)$ соотносилось с истинным (неизвестным) значением $Z(u)$ в этой же точке следующим образом:

$$E\left(Z^*(u) - Z(u)\right) = 0,$$
$$D\left(Z^*(u) - Z(u)\right) \to \min.$$

Данные, которыми оперирует кригинг, включают не только значения $Z(u_\alpha)$, но и априорную информацию о моделируемых случайных величинах в виде ковариационной функции $Cov\left(Z(u), Z(u+h)\right) = C_R(h)$ при допущении, что последняя не зависит от положения u, а зависит только от расстояния h между точками.

Кригинг-оптимизатор — набор параллельно работающих эвристик, основанных на кригинг-интерполяции параметрического пространства и принятии решений о новых точках на основании результата интерполяции.

Использование кригинг-интерполяции в данном случае не имеет под собой твердой математической основы, так как в случае параметрического пространства целевой функции нельзя говорить о каких-либо случайных величинах, тем более об их ковариациях. Однако использование кригинг-интерполяции как еще одного алгоритма интерполяции, обладающего свойствами сохранения значений в известных точках и удовлетворительного управляемого сглаживания, вполне допустимо.

Кригинг получает на вход все имеющиеся наборы аргументов и значений ЦФ, рассматриваемые как реализации некоторой случайной величины, и строит:

- оценку мат. ожидания E в требуемых точках — прогноз значения ЦФ в неизвестной точке;
- оценку дисперсии D в требуемых точках — в некотором смысле, меру «неизвестности» параметрического пространства в неизвестной точке (она же мера удаленности неизвестной точки от всех известных);
- оценку «локальной изменчивости» вида $LD = \frac{\sum_{i=1}^{N} \lambda_i (y_i - E)^2}{N}$ — в некотором смысле меру неоднородности параметрического пространства в неизвестной точке.

Эвристические стратегии поиска, используемые кригинг-оптимизатором, выбирают место в параметрическом пространстве для создания новой точки так, чтобы:

- мат. ожидание $E \to \max$ — попытка воспользоваться интерполяцией для поиска оптимального значения;
- дисперсия $D \to \max$ — исследование неизвестных областей параметрического пространства;
- локальная изменчивость $LD \to \max$ — исследование областей параметрического пространства с высокой изменчивостью;
- $D \cdot LD \to \max$ — исследование неизвестных областей параметрического пространства с высокой изменчивостью.

Кригинг-оптимизатор обладает следующими достоинствами:

- одновременно использует все эвристики в определенном соотношении;
- не имеет «истории», может быть остановлен в любой момент и запущен опять;
- равномерно заполняет параметрическое пространство, но в отличие от регулярных методов, делает это при любом имеющемся количестве «попыток».

1.3.3. Модули планирования экспериментов

Модули планирования экспериментов предназначены для исследования параметрического пространства путем заполнения его множеством точек в соответствии с некоторым алгоритмом планирования эксперимента:

- случайный засев;
- полный факторный анализ;
- алгоритм Box-Behnken Design;
- алгоритм Central Composite Design.

1.3.4. Модули анализа чувствительности и независимости параметров

Выделение независимых аргументов или групп аргументов ЦФ, основанное на определении чувствительности ЦФ, а также контролируемых параметров модели $\mathbf{y}(k)$ к аргументам ЦФ, является эффективным способом уменьшения размерности пространства поиска.

Анализ чувствительности в разработанной системе основан на оценке частных производных по каждому параметру в некоторых (заданных пользователем или выбираемых вручную) точках. Результатом работы блока оценки чувствительности является матрица, где каждой строке соответствует аргумент ЦФ, а каждому столбцу соответствует контролируемый параметр или значение ЦФ. Элемент матрицы содержит значение чувствительности контролируемого параметра к соответствующему аргументу ЦФ. Такая матрица позволяет выявить разбиение аргументов ЦФ на независимые группы.

2. Опыт применения для задач моделирования

В этом разделе приводится описание задач оптимизации ГДМ, для которых была применена разработанная система. В общем виде все задачи следуют одной и той же схеме. Раз система оптимизации умеет менять только числа, и с помощью алгоритмов параметрической оптимизации находить оптимальные их значения для некоторой определенной функции, то для любой фактической задачи моделирования требуется:

- параметризовать модель, то есть определить набор изменяемых численных параметров со своими областями определения, и определить процесс трансформации определенных значений параметров в конкретные (часто достаточно большие и сложные) изменения в модели;
- определить целевую функцию, которая для рассчитанной модели возвращает некоторое число, характеризующее то, насколько текущая модель «близка» к поставленной цели;
- запустить систему параметрической оптимизации, которая решает задачу оптимизации путем расчета большого количества ГДМ.

Далее приведено несколько примеров, которые должны проиллюстрировать разнообразие задач, которые могут быть решены в рамках разработанной системы.

2.1. Настройка относительных фазовых проницаемостей в виде степенных функций

Относительная проницаемость указывает на способность нефти и воды одновременно течь в пористой среде и определяется как отношение эффективной проницаемости фазы

к абсолютной (когда порода заполнена только одной фазой). ОФП задается в виде кривой зависимости относительной проницаемости данной фазы (нефть, вода, газ) от водонасыщенности, нормированной к диапазону [0; 1]. Форма кривых ОФП воды и нефти аппроксимировалась степенными функциями, форма которых задавалась с помощью восьми параметров. Задача заключалась в поиске таких значений этих параметров, чтобы результаты расчета полученной гидродинамической модели, минимизировали следующую ЦФ:

$$f(\mathbf{x}) = \sum_{k=1}^{N} \sum_{i=1}^{m} \sum_{j=1}^{u} \frac{(y_{i,j}(k) - \hat{y}_{i,j}(k \mid \mathbf{x}))^2}{\sigma_i^2},$$

где k — номер временного шага;

i — индекс интересующего контрольного параметра;

j — номер скважины;

σ_i — дисперсия наблюдаемых значений i-го параметра.

Учитывались следующие контрольные величины: мгновенные и суммарные дебиты воды и нефти, дебит жидкости, суммарный объем закачки, обводненность скважин. Исторические значения этих величин были предварительно получены из результатов расчета модели с экспертно заданными ОФП. Эти данные были приняты в качестве эталонных и использовались для сравнения с расчетными значениями в процессе оптимизации. В качестве ядра оптимизации применялся алгоритм ГА+НС.

В ходе оптимизации было сделано 10 итераций алгоритма ГА+НС, на каждой из которых оценивалась популяция из 10 моделей. Результаты эксперимента представлены на рисунках 4 и 5.

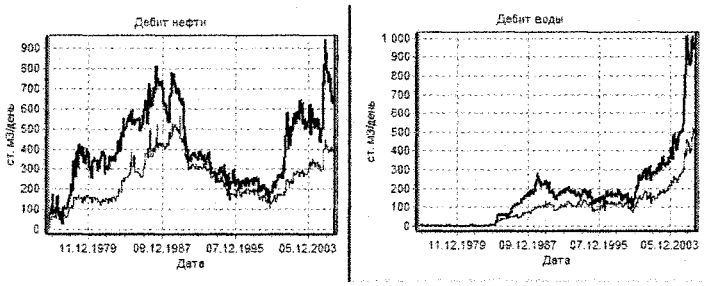


Рис. 4. Начальное приближение контрольных величин

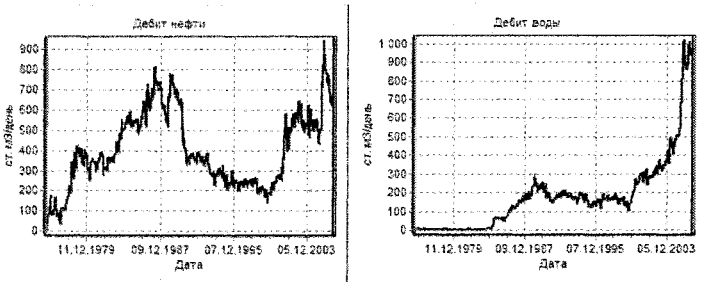


Рис. 5. Контрольные величины после оптимизации

2.2. Подбор петрофизической зависимости между проницаемостью и пористостью

Трехмерные массивы, определяющие пористость и проницаемость в каждой ячейке модели месторождения представляют собой часть численной модели, с которой работает симулятор. Исходными данными для построения этих кубов являются разнообразные исследования керна скважин при бурении и в процессе работы скважины и прочие источники информации, обладающие различной достоверностью.

Принято считать, что статистически пористость m и проницаемость k связаны соотношением $m = a + b \cdot \log_{10} k$, где a и b — константы для некоторого региона. Эти константы, как правило, находятся каким-нибудь методом линейной регрессии по облаку точек $(m, \log_{10} k)$ (рисунок 6, слева).

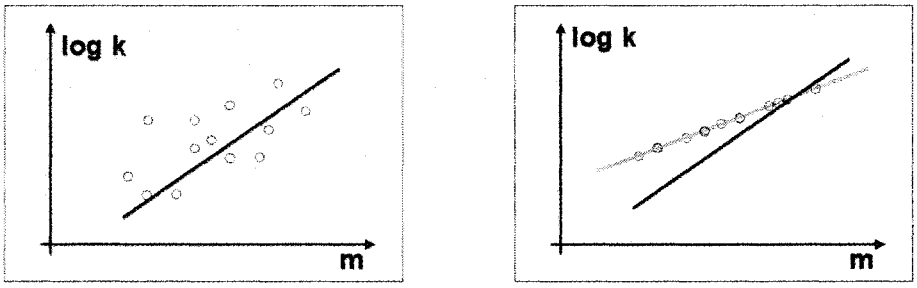


Рис. 6. Зависимость между пористостью и проницаемостью

Была решена задача в следующей постановке. Куб пористости модели был зафиксирован, как обладающий достаточной достоверностью. Выбирались некоторые значения a и b, и по приведенному выше петрофизическому соотношению из куба пористости рассчитывался куб проницаемости. Фактически, в облаке точек проводилась новая прямая, и все точки «сажались по вертикали» на эту прямую: значение m оставалось одно и то же, значение $\log_{10} k$ менялось (правый рисунок).

Полученная модель рассчитывалась на симуляторе, и далее рассчитывалась невязка между историческими и расчетными значениями добычи нефти и воды. Таким образом, задача сводилась к задаче параметрической оптимизации петрофизических параметров a и b для минимизации численного значения невязки. И уже эта задача параметрической оптимизации решалась стандартными средствами разработанной системы. Таким образом, система оптимизации варьировала целый куб проницаемости ГДМ путем параметризации этого куба всего двумя параметрами.

2.3. Адаптация карты проницаемости

Данные трехмерного куба проницаемости в модели, как правило, являются вторичными, пересчитанными из данных куба пористости по принятым (пусть и в результате эксперимента) петрофизическим зависимостям. Принято считать, что эти данные можно менять в процессе оптимизации модели, однако изменения не должны приводить к очевидно нефизичным результатам. Одним из таких изменений, например, является повышение или понижение проницаемости в ячейках куба, прилегающих к скважинам, однако такое изменение должно быть «плавным».

Для адаптации куба проницаемости был реализован следующий алгоритм. Выбирались некоторые значения множителей на проницаемость на некоторых скважинах, полученные значения добавлялись в пустую «карту множителей», которая после этого заполнялась це-

ликом интерполяцией добавленных множителей. Каждый горизонтальный слой куба проницаемости затем умножался на гладкую карту множителей. Полученная модель рассчитывалась на симуляторе, и вычислялась невязка между историческими и расчетными значениями добычи нефти и воды.

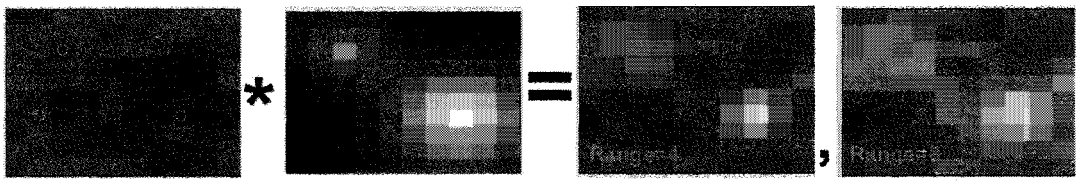


Рис. 7. Преобразование карты проницаемости

На рисунке 7 показан фрагмент одного горизонтального слоя начального куба проницаемости и четыре скважины. На левой нижней и правой верхней скважинах множитель устанавливался равным 1. На левой верхней скважине он был равен 2, а на правой нижней множитель был равен 3. На среднем рисунке показан фрагмент интерполированной карты множителей. На правом рисунке показан фрагмент произведения слоя начальной пористости на карту множителей. Задача заключалась в параметрической оптимизации некоторого ограниченного числа множителей на проницаемость на некоторых скважинах для минимизации численного значения невязки. И, опять-таки, после параметризации задачи она была решена стандартными средствами разработанной системы. Таким образом, система оптимизации варьировала целый куб проницаемости ГДМ путем параметризации этого куба значениями множителей на некотором наборе скважин.

2.4. Оптимизация траектории скважины

Задачи, так или иначе связанные с оптимизацией траекторий новых скважин, могут быть поставлены множеством различных способов. Это может быть выбор местоположения устья скважины, подбор схемы и параметров размещения групп скважин на месторождении (системы разработки), выбор направления и угла наклона сегмента горизонтальной скважины и т.д.

Во всех этих постановках требуется, как и во всех предыдущих задачах оптимизации, выбрать правильную параметризацию, чтобы система оптимизации вносила изменения в модель, меняя только малое количество численных параметров.

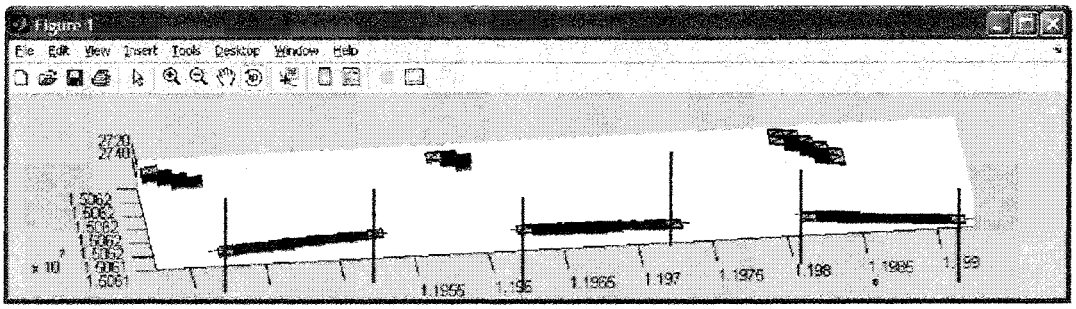


Рис. 8. Оптимизация траектории скважины

Например, в случае, если траектория скважины задается в модели путем определения координат всех точек траектории, фиксация X и Y координат начала и конца некоторого сегмента, но варьируя Z координату этих точек в некоторых пределах (красные отрезки на

рисунке), мы получаем задачу выбора угла наклона сегмента горизонтальной скважины. Данная задача решалась одновременно с выбором марки устройства управления лабиринтным притоком, которое устанавливается на скважину для предотвращения преждевременного прорыва газа. Целевой функцией в этом случае была максимизация добычи нефти при минимизации (или контроле непревышения некоторого порога) добычи газа.

Выводы

1. Использование кластерных вычислений позволяет эффективно решать разнообразные обратные задачи при идентификации параметров ГДМ путем решения множества прямых задач.

2. Использование разработанных алгоритмов оптимизации позволяет резко сократить количество численных экспериментов.

3. Разработанная система оптимизации ГДМ активно и успешно используется сотрудниками РН-УфаНИПИнефть при адаптации ГДМ.

Статья рекомендована к печати программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010» http://agora.guru.ru/pavt.

Литература

1. Ballester, P. J. A parallel real-coded genetic algorithm for history matching and its application to a real petroleum reservoir / P. J. Ballester, J. N. Carter //J. of Petroleum Science and Engineering. – 2007. – V. 59. – P. 157 – 168.

2. Evolutionary algorithms applied to history matching of complex reservoirs / R. Schulze-Riegert, J. Axmann, O. Haase, et al. // SPE Reservoir Evaluation & Engineering. – 2002. – V. 5, №. 2. – P. 163 – 173.

3. Gomez, S. Gradient-based history-matching with a global optimization method / S. Gomez, O. Gosselin, J. Barker // Society of Petroleum Engineering J. – 2001. – V. 6. – P. 200 – 208.

4. Javadi, A.A. A hybrid intelligent genetic algorithm / A.A. Javadi, R. Farmani, T.P. Tan // Advanced Engineering Informatics. – 2005. – V. 19, №. 4. – P. 255 – 262.

5. Kuo, J.-T. A hybrid neural-genetic algorithm for reservoir water quality management. / J.-T. Kuo, Y.-Y. Wang, W.-S. Lung // Water Res. – 2006. – Apr. – V. 40, №. 7. – P. 1367 – 1376.

6. Soleng, H. Oil reservoir production forecasting with uncertainty estimation using genetic algorithms / H. Soleng // Proc. Congress on Evolutionary Computation CEC 99. – V. 2. – 1999. – 6 – 9 July.

7. Srinivas, V. An integrated approach for optimum design of bridge decks using genetic algorithms and artificial neural networks / V. Srinivas, K. Ramanjaneyulu // Advances in Engineering Software. – 2006. – №. 38. – P. 475 – 487.

8. Wang, G.G. Review of metamodeling techniques in support of engineering design optimization / G.G. Wang, S. Shan // J. of Mechanical Design. – 2007. – V. 129, №. 4. – P. 370 – 380.

9. Wang, L. A hybrid genetic algorithm-neural network strategy for simulation optimization / L. Wang // Applied Mathematics and Computation. – 2005. – V. 170. – P. 1329 – 1343.

10. Кулешов, А.П. Когнитивные технологии в основанных на данных адаптивных моделях сложных объектов / А.П. Кулешов // Информационные технологии и вычислительные системы. – 2008. – № 1. – С. 18 – 29.

11. Льюнг, Л. Идентификация систем. Теория для пользователя: пер. с англ. / Л. Льюнг; Под ред. Я.З. Цыпкина. – М.: Наука; Гл. ред. физ.-мат. лит., 1991. – 432 с.

12. Цыпкин, Я.З. Информационная теория идентификации / Я.З. Цыпкин. – М.: Наука. Физматлит, 1995. – 336 с.

Гагарин Александр Владимирович, научный сотрудник ООО «РН-УфаНИПИнефть», unientity@gmail.com.

Макеев Григорий Анатольевич, доцент кафедры ВМиК, Уфимский государственный авиационный технический университет, г.Уфа, grigorym@ufanet.ru.

Байков Рафаель Анварович, доцент кафедры ВМиК, Уфимский государственный авиационный технический университет, г.Уфа, baikovRA@ufanipi.ru.

Волков Владимир Григорьевич, директор департамента ООО «РН-УфаНИПИнефть», volkovvg@ufanipi.ru.

Поступила в редакцию 7 апреля 2010 г.

УДК 537.86

ВОССТАНОВЛЕНИЕ ПОТЕНЦИАЛА В ОБРАТНОЙ СПЕКТРАЛЬНОЙ ЗАДАЧЕ ДЛЯ ОПЕРАТОРА ЛАПЛАСА С КРАТНЫМ СПЕКТРОМ

Г.А. Закирова

POTENTIAL'S RESTORE IN THE INVERSE SPECTRAL PROBLEM FOR LAPLACE OPERATOR WITH MULTIPLE SPECTRUM

G.A. Zakirova

Исследуются обратные спектральные кратного спектра. Построен алгоритм численного нахождения приближенного решения.

Ключевые слова: оператор Лапласа, обратная спектральная задача, приближенное решение, кратный спектр

Inverse spectral problems with multiple spectrum are investigated. A numerical algorithm of the approximate solution finding is obtained.

Keywords: Laplace operator, an inverse spectral problem, an approximate solution, a multiple spectrum

Введение

Обратные спектральные задачи в различных постановках играют фундаментальную роль в различных разделах математики и имеют множество приложений в естествознании. Большинство работ в этом направлении связаны с обыкновенными дифференциальными операторами. Что касается операторов в частных производных, то здесь, в основном, рассматривается степень оператора Лапласа с простым спектром.

В настоящей работе исследуется обратная задача для степени оператора Лапласа порожденного краевой задачей Дирихле в случае непростого спектра. Основным методом исследования является так называемый резольвентный метод, теоретически обоснованный в работах [1, 2]. Получены теоремы существования решения поставленной обратной задачи, позволившие разработать вычислительный алгоритм восстановления потенциала по спектру и создать программу, определяющую возмущение по заданной последовательности собственных чисел.

Постановка обратной задачи

Пусть

$$\Pi = \{x = (x_1, x_2, \ldots, x_N) \ : \ 0 \le x_j \le a_j, j = 1, \ldots, N\}, \ a_j > 0.$$

В пространстве $L_2(\Pi)$ рассмотрим дискретный самосопряженный оператор T_0, определенный краевой задачей Дирихле

$$-\Delta v = \lambda v, \quad v|_{\partial\Pi} = 0, \tag{1}$$

где Δ — оператор Лапласа, $\partial\Pi$ — граница Π.

Рассмотрим оператор $T = \int_0^\infty \lambda^\beta dE(\lambda)$, являющийся степенью оператора T_0, где $E(\lambda)$ — спектральное разложение единицы оператора T_0, $\beta \geq 1$, $\lambda^\beta > 0$ при $\lambda > 0$.

Очевидно, спектр $\sigma(T)$ оператора T неоднократный. Иногда, для удобства будем нумеровать упорядоченные по возрастанию собственные числа $\lambda_m = \lambda_{(m_1,m_2,\ldots,m_N)}$ оператора T и связанные с ними спектральные объекты одним нижним натуральным индексом и одним верхним, при этом верхний индекс будет отвечать за кратность ν_t собственного числа λ_t, т.е. $\lambda_t = \lambda_t^k = \lambda_t^j$, $k = \overline{1,\nu_j}$.

Пусть P — оператор умножения на вещественную функцию $p \in L_2(\Pi)$, называемую потенциалом.

Обозначим через μ_t собственные числа оператора $T + P$, занумерованные в порядке возрастания действительных частей с учетом алгебраической кратности, а через u_t — соответствующие им ортонормированные в $L_2(\Pi)$ собственные функции.

Рассмотрим следующую обратную задачу спектрального анализа.

Пусть дана последовательностью комплексных чисел $\{\xi_t\}_{t=1}^\infty$, близкая к спектру оператора T. При различных степенях $\beta \geq 1$ требуется доказать существование оператора P, такого что спектр $\sigma(T+P)$ совпадает с последовательностью $\{\xi_t\}_{t=1}^\infty$.

Степень оператора Лапласа с потенциалом в N-мерном параллелепипеде

Сформулируем вспомогательные утверждения, на которых базируется доказательства основных результатов данной статьи. Доказательства самих вспомогательных утверждений приведены в работе [3].

Обозначим:

$R_0(\lambda) = (T - \lambda E)^{-1}$, $R(\lambda) = (T + P - \lambda E)^{-1}$;

$a_t = \{\lambda : \mathrm{Re}\ \lambda = \frac{\lambda_{t+1} + \lambda_t}{2}\}$, $\Gamma_t = \{\lambda \in \mathbb{C} : \mathrm{Re}\lambda \in a_t\}$, $\gamma_t = \{\lambda \in \mathbb{C} : |\lambda_t - \lambda| = r_0\}$,

$r_t = \frac{1}{2}\min\{\lambda_{t+1} - \lambda_t; \lambda_t - \lambda_{t-1}\}$, $r_0 = \inf\limits_t r_t$;

$\Omega_t = \{\lambda : |\lambda_t - \lambda| \geq r_0\}$, $\Omega = \bigcap\limits_{t=1}^\infty \Omega_t$, $V = \prod\limits_{j=1}^n a_j$.

Лемма 1.

Если $\|P\| < r/2$, где $0 < r \leq r_0$, то оператор $T + P$ — дискретен, причем

(i) если $R_0(\lambda) \in \mathfrak{S}_q$, то $R(\lambda) \in \mathfrak{S}_q$, $1 \leq q < \infty$,

(ii) если $\lambda_t \in \mathbb{C} \setminus \Omega_t$, то $\mu_t^s \in \mathbb{C} \setminus \Omega_t$, $s = \overline{1,\nu_t}$, ν_t—кратность собственного числа λ_t.

Лемма 2.

Если $\beta > \frac{3N}{4}$, то ряд $\sum\limits_{t=1}^\infty r_t^2 \max\limits_{\lambda\in\gamma_t} \|R_0(\lambda)\|_2^4$ сходится.

Обозначим сумму ряда через s^2.

Теорема 1. *Пусть $\beta > \frac{3N}{4}$, $r \in (0, \min\{r_0, \frac{1}{s\sqrt{2^N}}\})$. Если для комплексной последовательности $\{\xi_t^k\}$ выполняется неравенство:*

$$\sqrt{2^N V}\left(\sum_{t=1}^\infty \sum_{k=1}^{\nu_t} |\xi_t^k - \lambda_t|^2\right)^{\frac{1}{2}} < \frac{r}{2}(1-\omega),$$

где $\omega = \sqrt{2^N}sr < 1$, *то существует потенциал* $p \in L_2(\Pi)$ *такой, что для любого* $t \in \mathbb{N}$

$$\sum_{k=1}^{\nu_t} \xi_t^k = \sum_{k=1}^{\nu} \mu_t^k, \tag{2}$$

где $\sigma(T+P) = \{\mu_t^k\}$.

Замечание 1. Оператор $T+P$, обладающий свойством (2), неединственен.

Объединим в один класс P все операторы, спектр которых обладает свойством (2). Если мы не будем различать представителей этого класса, то можем говорить о единственности решения обратной задачи.

Приближенное восстановление потенциала. Численный эксперимент

Введем в рассмотрение следующую систему функций:

$$\varphi_m(x) = \sqrt{\frac{2^{N+q}}{V}} \prod_{j=1}^{N} \cos\left(\frac{2\pi m_j x_j}{a_j}\right),$$

где $m = (m_1, \ldots, m_N)$, $m_j \in \{0\} \cup \mathbb{N}$, q—число ненулевых индексов в мультииндексе m. При $m_j \in \mathbb{N}$ эту систему будем нумеровать нижним и верхним индексами так же, как и систему $\{v_m\}$, т.е. в соответствии с нумерацией собственных чисел λ_t^k.

Можно показать, что уравнение

$$p = \alpha_0 - \alpha(p),$$

где $\alpha_0 = (-1)^N\sqrt{2^N V}\sum_{t=1}^{\infty}\sum_{k=1}^{\nu_t}(\xi_t^k - \lambda_t)\varphi_t^k$, $\alpha(p) = (-1)^N\sqrt{2^N V}\sum_{t=1}^{\infty}\sum_{k=1}^{\nu_t}\frac{\alpha_t(p)}{\nu_t}\varphi_t^k$,

$$\alpha_t(p) = \frac{1}{2\pi i}\int_{\gamma_{r_t}} \lambda\, \mathrm{S}\,[R(\lambda)(PR_0(\lambda))^2]d\lambda = \frac{1}{2\pi i}\int_{\gamma_{r_t}} \mathrm{S}\,[\sum_{k=2}^{\infty} R_0(\lambda)(PR_0(\lambda))^k]d\lambda,$$

имеет единственное решение p.

Пусть

$p_0 \equiv 0$, тогда $p_1 = \alpha_0 - \alpha(p_0) = \alpha_0$,

$p_2 = \alpha_0 - \alpha_(p_1) = \alpha_0 - \alpha(\alpha_0)$, $p_3 = \alpha_0 - \alpha(p_2), \ldots, \lim\limits_{t\to\infty} p_t = p$.

Методом последовательных приближений найдено решение $\widetilde{p}$ уравнения

$$\widetilde{p} = \alpha_0 - \widetilde{\alpha}(\alpha_0), \tag{3}$$

где

$$\widetilde{\alpha}(p) = \sum_{t=1}^{\infty}\widetilde{\alpha_t}(p)\varphi_t,\ \widetilde{\alpha_t}(p) = \frac{1}{2\pi i}\int_{\gamma_{r_t}} \lambda\, \mathrm{S}\,[R_0(\lambda)(PR_0(\lambda))^2]d\lambda.$$

$$\widetilde{p} = \alpha_0 - \sqrt{2^N V}\sum_t(\sum_{j\neq t}\sum_{k=1}^{\nu_t}\frac{(\alpha_0\varphi_t, \varphi_j)^2}{(\lambda_t - \lambda_j)})\varphi_t^k.$$

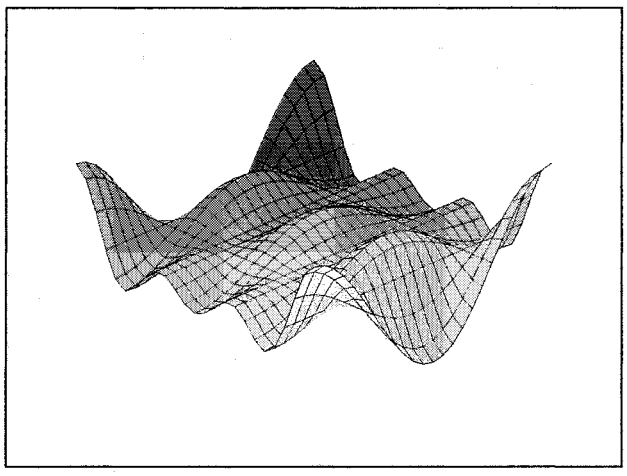
В пакете Maple 6.0 разработана программа, которая по заданной последовательности собственных чисел определяет в явном виде приближенный потенциал, такой, что спектр

возмущенного оператора будет совпадать в заданном смысле с введенной последовательностью.

Далее приведем пример, иллюстрирующий работу программы.
Пусть T — степень оператора T_0 ($\beta = 2$), определенного краевой задачей Дирихле (1) на прямоугольнике Π со сторонами $a = 1$, $b = 4$. Пусть далее, $\xi_{mn} = \lambda_{mn} + 0.0001$, $m, n \leq 3$. По теореме (1) существует потенциал $p \in L_2(\Pi)$ такой, что для любого $t \in \mathbb{N}$

$$\sum_{k=1}^{\nu_t} \xi_t^k = \sum_{k=1}^{\nu_t} \mu_t^k, \quad \{\mu_{mn}\} = \sigma(T + P). \tag{4}$$

Приближенное решение исследуемой обратной спектральной задачи, найденной в предложенной программе по первым трем членам последовательности $\{\xi_{mn}\}$, имеет вид



Приближенный потенциал, восстановленный программой

Литература

1. Дубровский, В.В. К обратной задаче для степени оператора Лапласа с непрерывным потенциалом / В.В. Дубровский, А.В. Нагорный // Дифференц. уравнения. – 1990. – Т. 26, № 9. – С. 1563 – 1567.
2. Дубровский, В.В. Обратная задача для степени оператора Лапласа с потенциалом из L_2 / В.В. Дубровский, А.В. Нагорный // Дифференц. уравнения. – 1992. - Т. 28, № 9. – С. 1552 – 1561.
3. Седов, А.И. Обратная задача спектрального анализа для степени оператора Лапласа на равнобедренном прямоугольном треугольнике / А.И. Седов, Г.А. Закирова // Вестн. СамГУ. Естественнонаучная серия. – 2008. – № 2(61). – С. 34 – 42.

Закирова Галия Амруловна, кандидат физико-математических наук, кафедра уравнений математической физики, Южно-Уральский государственный университет, zakirova81@mail.ru.

Поступила в редакцию 2 августа 2010 г.

УДК 519.6

ЭКЗАПРОБЛЕМЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

В.П. Ильин

THE EXAPROBLEMS OF MATHEMATICAL MODELING

V.P. Iliyn

Рассматриваются вопросы, связанные с разработкой программного обеспечения для решения задач математического моделирования на МВС. Проводится анализ и классификация математических постановок из различных прикладных областей, а также методов и технологий для основных этапов компьютерного эксперимента. Предлагаются концепция и структура базовой системы моделирования (БСМ), включающей проблемно-независимые компоненты для геометрического и функционального моделирования, генерации сеток, аппроксимации начально-краевых задач, алгебраических решателей, постобработки и визуализации результатов, а также управления вычислительным процессом, из которых могут формироваться прикладные программные продукты.

Ключевые слова: экстремальные вычисления, междисциплинарные прямые и обратные задачи, математическое моделирование, вычислительные методы и технологии, базовая система моделирования

The questions of the development of scientific software for solving mathematical modeling problems on the multi-processor computational systems are considered. The analysis and classification of the mathematical statements from various applications, as well as methods and technologies for the main stages of numerical experiments are presented. The conception and the structure of the Basic System of Modeling (BSM), which includes the problem-independent components for geometric and functional modeling, grid generation, approximation of the initial-boundary value problems, algebraic solvers, post-processing and visualization of the results, and control of computational process, are proposed. The applied program products can be constructed from the described components.

Keywords: extremal computing, interdisciplinary direct and inverse problems, mathematical modeling, computational methods and technologies, basic system of modeling

1. Введение

Математическое моделирование окончательно обретает статус третьего пути человеческого познания, наряду с теоретическими и экспериментальными исследованиями. Его развитие в начале 21-го века происходит под воздействием трех основных факторов: появление новых актуальных практических задач, в том числе междисциплинарных, обратных и оптимизационных, требующих экстремальных ресурсов; достижения теоретической и вычислительной математики по созданию и обоснованию современных моделей и эффективных алгоритмов; разработка высокопроизводительных компьютерных технологий на базе бурно прогрессирующих многопроцессорных вычислительных систем (МВС).

Стоящие на повестке дня проблемы изучения динамических процессов и явлений включают понятия хаоса, странных аттракторов, фрактальных сред, синергетической самоорганизации сложных систем, инициирующие новые принципы моделирования и требующие сверхвысокого разрешения.

Последние десятилетия ознаменованы созданием новых математических подходов и постановок реализуемых или ожидающих своего воплощения в вычислительных алгоритмах и технологиях, см. [1, 2]. Здесь можно назвать комплексные согласованные модели механики сплошных сред, единообразно описывающие свойства упругих, пластических, жидких, газообразных и многофазных сред, а также переходы из одних состояний в другие, использование теории групп, дифференциальных форм, распределений, дискретных эйлеровых и лагранжевых представлений, и т.д.

Кардинально меняется понятие «большой» задачи, предполагающей уже применение сеток с миллионами, десятками и сотнями миллионов узлов, для решения которой численные методы становятся принципиально параллельными и ориентированными на МВС с сотнями и тысячами процессоров. Проблема обеспечения высокой производительности и масштабируемости крупномасштабных вычислений требует углубленного узучения структуры алгоритмов и их отображения на архитектуру ЭВМ, а также создание и эффективное использование современных программных технологий, см. [3 – 7]. Хотя первый в мире компьютер петафлопной производительности запущен только в 2008 г., уже объявлено грядущее появление в 2019 г. «экзафлопника» с быстродействием операций в секунду (1000 петафлоп!, [5]). Данный прогноз представляется вполне реальным, что означает создание в недалеком будущем МВС с миллиардами вычислительных ядер. В связи с этим по инициативе Дж.Донгарры и П.Бекмана сформирован международный Exascale Software Project, миссией которого является выработка новых концепций и моделей программирования для компьютеров экстремального и экзафлопного уровня (авторами [5] такое программное обеспечение названо X-stack). Именно глобальные проблемы программного обеспечения, включая операционные системы, компиляторы, базы данных и всевозможные инструментарии, являются главными на ближайшее десятилетие, поскольку существующее «софтверное» оснащение вычислительных систем своими корнями связано с последовательными вычислениями на фон-неймановской машине, а наступающая эра массового параллелизма неизбежно требует перехода количества в качество. Основной заявленный принцип построения нового программного обеспечения – создание открытых (Open Source) библиотек для виртуальных машин. Эта стратегия естественным образом соответствует исторически складывающейся методологии развития прикладного программного обеспечения задач математического моделирования. Чтобы понять возникающие на текущем этапе проблемы, необходимо оценить тенденции возникающих практических задач, эволюцию современных вычислительных методов и новые технологические требования, возникающие из особенностей развития компьютерных архитектур.

2. Прикладные задачи и математические модели

Актуальные практические проблемы из различных сфер человеческой деятельности, требующие все более глубокого понимания протекающих процессов и явлений, являются главным двигателем прогресса в математическом моделировании. Если говорить о естественно-научных направлениях, то практически все они охватываются следующим списком: материаловедение и нанотехнологии, энергетика, машиностроение и металлургия, физика плазмы и высоких энергий, электроника и связь, астрофизика и астрономия, науки о Земле (атмосфера, океан, твердь), биология и медицина.

В терминах уравнений математической физики, или механики сплошных сред (или тео-

ретической физики) эти задачи составляют физику твердого тела, т.е. упругости, пластичности и разрушения, классическую и магнитную гидрогазодинамику, тепломассоперенос в многофазных и пористых средах, электромагнетизм. Каждая из этих дисциплин представляет собой фундаментальное научное направление, изучение которого требует глубокого погружения и профессионализма.

Однако особенность реальных требований в серьезных технических разработках, в наукоемких производствах и в исследовании природных явлений заключается в необходимости междисциплинарного подхода к моделированию с обязательным учетом факторов самой различной природы. Примером может служить проблема долгосрочного прогноза погоды и климата [8], которая включает общую циркуляцию атмосферы и океана, учет мезометеорологических структур и морфологии приземной поверхности, солнечной радиации и ионосферных процессов, распространения примесей и антропогенных воздействий. Математическая модель такой суперзадачи представляет собой начально-краевую задачу для сложнейшей системы нелинейных дифференциальных уравнений или эквивалентных вариационных соотношений со многими неизвестными функциями, имеющими многочисленные разномасштабные во времени и пространстве особенности. Зачастую коэффициенты решаемых уравнений, определяющие важные физические факторы (вязкость, теплопроводность и т.д.), известны с большой долей неопределенности, которую надо устранить путем сопоставления расчетных данных с результатами натурных измерений. Таким образом возникает проблема идентификации параметров модели – одна из типичных обратных задач, для нахождения ответа которой, как правило, требуется многократное решение прямых задач (которые проще хотя бы потому, что в них задается вся информация, необходимая для получения искомого результата). В целом моделирование динамики атмосферы, океана и климата – это работа для команды, которая является коллективным естествоиспытателем.

Иллюстрация совсем другого рода – это комплексное моделирование технологических процессов в алюминиевом электролизере [9], представляющем собой в определенном смысле сложный организм, длительное и успешное функционирование которого требует тщательного внимания и управления. Здесь подаются огромные токи и загружается сырье, из которого выплавляется периодически отгружаемый жидкий металл. Конечная практическая проблема в данном случае – оптимизация производственного режима, обеспечивающего минимизацию целевого функционала, т.е. отношение эксплуатационных расходов (стоимость электроэнергии, материалов, ремонтных работ и т.д.) к стоимости получаемого продукта, характеризуемой итоговым объемом алюминия определенного качества. При этом необходимо удовлетворять различным эксплуатационным ограничениям, например, на температурные напряжения, во избежание разрушения конструкции. Такая обратная задача включает решение множества прямых задач, каждая из которых описывается джентльменским набором уравнений математической физики: Максвелла (электромагнетизм), Ламе (упругость и прочность), Навье – Стокса (циркуляции жидкого металла и электролита), теплопереноса с фазовыми переходами, химической кинетики. Решать эти взаимосвязанные уравнения необходимо в реальных трехмерных областях сложной конфигурации. Достаточно сказать, что компьютерная модель электролизера (среднего уровня детализации) насчитывает более 500 геометрических объектов и 30 различных материалов (анодные и катодные узлы, графитовые блоки, футировочные материалы, корпус и т.п.). К этому следует добавить, что в заводском цехе работает «линейка» из электрически взаимосвязанных более чем 100 электролизеров, магнитные поля которых существенно влияют друг на друга.

Можно привести много других больших задач, решение которых с необходимой точностью (на практике постоянно ужесточающейся) требует привлечения вычислительных ресурсов пета- и экзафлопного масштаба (будем их называть «экстремальными», или X-ресурсами). Подчеркнем еще раз общую для них тенденцию: серьезные задачи являются

междисциплинарными и обратными, а расчет какого-то одного поля представляет собой частный или методический интерес.

Важно отметить, что при всем необозримом многообразии актуальных приложений, их математические постановки могут быть строго классифицированы и составлять конечный набор основных формулировок, который может при необходимости расширяться и углубляться, не выходя при этом за рамки унифицированной технологии. В целом формальное описание достаточно абстрактной начально-краевой задачи может быть представлено следующим образом.

Пусть $\vec{u} = \{u_\mu, \mu = 1, ..., m_f\}$ есть вектор-функция, у которой каждая скалярная составляющая u_μ зависит от вектора пространственных координат $\vec{x} = (x_1, x_2, x_3)$ и, возможно, от времени t. Система координат может быть декартовая, цилиндрическая или сферическая (или какая-то специальная), а количество независимых пространственных переменных d (размерность задачи) может уменьшаться с трех до двух или одного.

Обозначим через $\bar{\Omega} = \Omega \bigcup \Gamma$ замкнутую ограниченную область евклидового пространства с границей Γ, в которой определена функция $\vec{u}(\vec{x}, t)$ для каждого момента времени $0 < t \leq T < \infty$. Предполагается, что $\vec{u}(\vec{x}, t)$ принадлежит некоторому функциональному пространству, в котором имеют смысл операторы L и l. Требуется в расчетной области $\bar{\Omega}$ найти решение системы дифференциальных уравнений с известной функцией $\vec{f} = \{f_\mu, \mu = 1, ..., m_f\}$

$$L\vec{u} = \vec{f}(\vec{x}, t), \quad \vec{x} \in \bar{\Omega}, \quad 0 < t \leq T < \infty, \tag{1}$$

удовлетворяющее заданным граничным и начальным условиям

$$l\vec{u} = \vec{g}(\vec{x}, t), \quad \vec{x} \in \Gamma, \quad \vec{u} = (\vec{x}, 0) = \vec{u}^0(\vec{x}). \tag{2}$$

В качестве примера можно привести дифференциальный оператор второго порядка

$$L = A\frac{\partial}{\partial t} + \nabla B \nabla + C\nabla + D, \tag{3}$$

где ∇ есть градиент, а A, B, C, D – некоторые квадратные матрицы порядка m, элементы которых суть или постоянные, или функции $\vec{x}, t$, или зависят от $\vec{u}$. В зависимости от конкретного вида матричных коэффициентов, математическая постановка (1) – (3)формально описывает стационарные или нестационарные, линейные или нелинейные процессы и явления в огромном числе приложений. Расчетную область будем всегда считать связной (хотя формально это и не обязательно), иначе задача (1) – (2) распадается на независимые подзадачи.

Простейшей иллюстрацией граничных условий является смешанная краевая задача, в которой граница области Γ состоит из частей Γ_D, Γ_N, на которых заданы условия разных типов:

$$u = g_D, \quad x \in \Gamma_D; \quad D_N u + A_N \nabla_n u = g_N, \quad x \in \Gamma_N, \tag{4}$$

где D_N и A_N – матрицы (в общем случае прямоугольные), а g_D, g_N – вектор-функции, элементы которых или известны, или зависят от искомых решений (∇_n – оператор дифференцирования по направлению внешней нормали к границе).

Вместо классического дифференциального описания (1) – (4) исходная задача может быть представлена в эквивалентной (в каком-то смысле) вариационной постановке для обобщенного решения. Кроме того, вместо дифференциальных могут фигурировать интегральные уравнения, а в общем случае – системы дифференциально-интегральных равенств и/или неравенств.

Приведенная выше формулировка определяет прямые задачи математического моделирования. Исходные данные каждой из них можно определить зависящими от вектора параметров $\vec{p} = (p_1, ..., p_{m_0})$, компоненты которого надо оптимизировать по условию минимума описанного заранее целевого функционала от определенного на параметризованном решении $\vec{u}(\vec{x}, t\vec{p})$ поставленной задачи (см. [10] и цитируемые там работы):

$$\Phi_0(\vec{u}(\vec{x}, t\vec{p}_{opt})) = \min_{\vec{p}} \Phi_0(\vec{u}(\vec{x}, t\vec{p})), \tag{5}$$

при заданных линейных или функциональных ограничениях ($m_1 + m_2 = m_0$):

$$p_k^{\min} \le p_k \le p_k^{\max}, \quad k = 1, ..., m_1, \quad \Phi_l(\vec{u}(\vec{x}, t\vec{p})) \le \delta_l, \quad l = 1, ..., m_2. \tag{6}$$

В этом случае описание прямой задачи фигурирует формально как дополнительное ограничение в виде уравнения состояния

$$L\vec{u}(\vec{p}) = \vec{f}, \quad \vec{p} = \{p_k\}. \tag{7}$$

Таким образом, мы приходим к оптимизационной постановке обратной задачи, заключающейся в идентификации параметров модели математической проблемы, которая может включать и комплексное моделирование различного вида полей, когда рассматривается одновременно совокупность физических эффектов, а целевой функционал Φ_0 включает отклонение натурных и расчетных данных для всех видов измеряемых полей.

Расчетную область можно представить как объединение $N_D \ge 1$ замкнутых подобластей, т.е. $\bar{\Omega} = \bigcup\limits_k \bar{\Omega}_k$, $\Omega_k \bigcap \Omega_{k'} = 0$, при $k \ne k'$, где $k, k' = 1, ..., N_D$, в каждой из которых могут задаваться свои коэффициенты или даже различные типы решаемых уравнений. Граница k-ой подобласти может быть представлена как объединение смежных границ с примыкающими подобластями:

$$\bar{\Omega}_k = \Omega_k \bigcap \Gamma_k, \quad \Gamma_k = \bigcup_{k'} \Gamma_{k,k'}, \quad \bar{\Omega}_k \bigcap \bar{\Omega}_{k'} = \Gamma_{k,k'} = \Gamma_{k',k} \ne 0. \tag{8}$$

Если внешнее пространство (дополнение) по отношению к Ω обозначить формально как подобласть $\Omega_0 = R^d/\bar{\Omega}$, то внешняя граница расчетной области представляется в виде $\Gamma = \Gamma^{(e)} = \bigcup\limits_k \Gamma_{k,0}$, а граница каждой подобласти может быть разбита на ее внешнюю и внутреннюю части, т.е.

$$\Gamma_k = \Gamma_k^{(e)} \bigcap \Gamma_k^{(i)}, \quad \Gamma_k^{(e)} = \Gamma_{k,0}, \quad \Gamma_k^{(i)} = \bigcup_{k' \ne 0} \Gamma_{k,k'}. \tag{9}$$

Если подобласть – трехмерный объект, имеющий свой ненулевой объем, то ее граница в качестве меры измерения имеет площадь. Каждый из граничных фрагментов внешней или внутренней границы (на каждом из которых ставится какое-то краевое условие) будем считать кусочно-гладкой поверхностью, т.е. состоящей из совокупности гладких поверхностных сегментов, характеризуемых своим уравнением. Граница расчетной области является в общем случае многосвязной, или конечносвязной, т.е. состоящей из конечного числа связных множеств. Граница является односвязной, если сама расчетная область не имеет «дыр».

Каждый поверхностный сегмент имеет свою граничную линию, состоящую из конечного числа криволинейных (пространственных в общем случае) отрезков, характеризуемых парой своих концевых точек, длиной и уравнениями двух поверхностей, пересечением которых он является. Такие граничные отрезки будем называть ребрами расчетной области, а их концевые точки, лежащие на пересечении ребер, будем называть вершинами. Совокупность

ребер и вершин назовем каркасом области, который формально можно представить в виде многомерного графа.

В качестве итога обсуждения постановок задач отметим, что при всех различиях их технического или естественно-научного происхождения, соответствующие постановки могут быть формально описаны с помощью обозримого количества функциональных и геометрических объектов.

3. Вычислительные методы и технологии

При всем разнообразии математических задач и моделей, поддающихся тем не менее вполне обозримой систематизации, количество методов их решения, естественно, во много раз больше. Однако этапы вычислительного эксперимента также могут быть четко выделены и классифицированы, а алгоритмы их реализации – фрагментированы по модульному принципу, в зависимости от их назначения, применяемых подходов, ресурсоемкости, параллелезуемости и т.д.

а. Дискретизация математической модели

Задачи математического моделирования в подавляющем своем большинстве описываются в терминах интегро-дифференциального исчисления над функциями непрерывного аргумента. Исключения составляют процессы масштаба межатомных расстояний, которые актуальны, например, в нанотехнологиях, где могут быть даже неприменимы понятия производных, но мы на них останавливаться не будем.

Дискретизация расчетной области, т.е. построение сетки, представляет особо сложную проблему в многомерных случаях с кусочно-гладкими криволинейными и, возможно, движущимися границами (для последних специальный класс составляют т.н. свободные границы, положение которых заранее неизвестно). Сама сетка определяется совокупностью своих объектов различной размерности: узлы, ребра, грани, конечные объемы,– а также топологическими связями между ними. Между геометрической структурой расчетной области с подобластями и сеточной структурой существует большая аналогия, и они содержат по сути однотипные наборы объектов макро- и микро-уровня. Если математическая постановка является дифференциальной или в форме граничных интегральных уравнений, то отличие дискретизации заключается в том, что в последнем случае сетка строится только на граничных поверхностях расчетной области.

Существует большое количество критериев качества сетки, которые в значительной степени определяют точность и экономичность численного решения. Одно из главных требований сетке – адаптивность, означающая в том или ином смысле учет особенностей конфигурации границы и свойств искомого решения, которые определяются или теоретически априори, или апостериори экспериментально, т.е. на основе предварительных расчетов.

В первую очередь требуется, чтобы вершины, ребра и граничные поверхности расчетной области составлялись из соответствующих сеточных объектов, или, в крайнем случае, аппроксимировались ими с возможно малой погрешностью. Второй аспект связан с возникающей сингулярностью, т.е. сильным ростом производных, в окрестности углов и ребер границы, что требует специального сгущения узлов в таких подобластях. А при наличии сильно меняющихся пространственно-временных особенностях решения сетки должны быть динамическими, т.е. регулярно или периодически перестраиваемыми.

Распространенные сеточные технологии включают триангуляции Делоне, ячейки Дирихле – Вороного и различные приемы контроля вырожденных случаев. Важным вычислительным средством являются многосеточные методы, основанные на построении последовательности вложенных сеток или на их локальном сгущении.

Наиболее просто конструируемыми и одновременно обеспечивающими больший порядок точности являются равномерные сетки, которые могут иметь различные типы конечных

элементов: кубы или параллелепипеды, призмы, тетраэдры и т.д. Однако в реальных задачах сетки приходится строить неравномерные и нерегулярные, или неструктурированные, у которых для каждого узла номера его ближайших соседей можно задать только перечислением. Существуют компромиссные квазиструктурированные сетки, когда расчетная сеточная область состоит из сеточных подобластей, в каждой из которых сетка строится по своим правилам и может быть структурированной. Такие сетки могут быть несогласованными или согласованными - в последних случаях узлы на смежных границах раздела соседних сеточных подобластей являются совпадающими.

Методы построения сеток отличаются большим разнообразием и имеют обширную профессиональную литературу. Здесь используются и квазиконформные отображения, и вариационные принципы, и специальные метрические пространства, и многочисленные эмпирические приемы. Соответствующее программное обеспечение, или генераторы сеток, существует в широком ассортименте, или в свободном доступе через Интернет, или в качестве коммерческих продуктов. Зачастую процедуры построения сеток являются неотъемлемой частью проблемно-ориентированных пакетов прикладных программ (ППП), но они также существуют и в автономной форме, позволяющей их встраивать в различные приложения.

Рассмотренные выше принципы дискретизации, т.е. перехода от функций непрерывного аргумента к сеточным функциям, связанным с узлами, ребрами и другими объектами, базируются на эйлеровом подходе. Однако при моделировании движущейся среды оказывается более удобно лагранжевая система координат, порождающая многочисленные варианты методов больших частиц. Широко распространены алгоритмы «частиц в ячейках», использующие сочетание эйлеровых сеток с дискретизацией потоков субстанции. Имеются также и «бессеточные» методы с чисто лагранжевым описанием процессов.

Все рассматриваемые выше подходы относятся к детерминистским алгоритмам, альтернативу которым составляют статистическое моделирование и методы Монте-Карло. Основанные на вероятностном принципе, они являются незаменимым инструментом исследования процессов и явлений со случайными исходными данными или описываемыми многомерными дискретными соотношениями. Они обладают высокой универсальностью и формально могут быть применены к решению практически любых дифференциальных и/или интегральных уравнений, однако вопрос о целесообразности их выбора – это уже компетенция конкретного пользователя.

б. Алгоритмы аппроксимации

Строго говоря, генерация сетки является первым этапом дискретизации, конечной целью которой является переход от исходных функциональных соотношений к конечномерным уравнениям, неравенствам или рекурсиям. При этом фактически задача алгебраизируется, что достигается путем аппроксимации функций, производных и интегралов. Основные подходы к построению сеточных соотношений – это методы конечных разностей, конечных объемов, конечных элементов (МКР, МКО, МКЭ), коллокаций и спектральные методы, связанные с разложениями в ряды Фурье. Не пытаясь делать обзора имеющегося огромного материала по данным вопросам, мы коротко остановимся главным образом на технологических аспектах наиболее универсальных МКО и МКЭ, позволяющих конструировать сеточные аппроксимации высоких порядков точности на различных типах конечных объемов для широкого класса задач математического моделирования.

Важным моментом этих двух близких подходов является поэлементная технология вычисления локальных матриц и векторов правых частей с последующей сборкой (ассемблированием) глобальных матриц и систем алгебраических уравнений (линейных или нелинейных – СЛАУ или СНАУ). Этот прием значительно упрощает программную реализацию и естественным образом обеспечивает эффективное распараллеливание данного вычислительного этапа.

В нестационарных проблемах пространственная аппроксимация осуществляется на каждом временном шаге, а при наличии нелинейностей такая процедура повторяется итерационно для всех шагов. Наиболее трудоемкими оказываются задачи с динамическими сетками, если их приходится перестраивать на каждой итерации для каждого шага по времени.

Совокупность аппроксимационных алгоритмов для типовых задач математической физики может быть систематизирована и классифицирована по следующим признакам:

- по типу аппроксимируемого члена дифференциального уравнения (градиент, дивергенция и т.д.) или его механического смысла; примером могут быть матрицы жесткости и матрицы масс в МКЭ;
- по виду конечных элементов или объемов: тетраэдры, параллелепипеды, призмы и т.д.; при этом могут выделяться частные случаи, которые наиболее экономично реализуются (например, правильные фигуры);
- по характеру базисных функций, которые могут отличаться своими носителями, порядками и структурными свойствами (лагранжевые и эрмитовые, скалярные и векторные, и т.п.);
- по конфигурации сеточного шаблона, т.е. совокупности узлов, участвующих в одном уравнении, получаемом в итоге формирования алгебраических систем; наиболее экономичными оказываются компактные схемы, в которых участвуют только наиболее близкие геометрически соседние узлы; как правило, повышение порядка аппроксимации ведет к «растягиванию» сеточного шаблона и расширению ленточной структуры итоговых матриц, так что с точки зрения общей эффективности алгоритмов приходится искать «золотую середину».

Следует подчеркнуть, что процедуры аппроксимации носят локальный характер, реализация которых использует свойства только топологически ближайших сеточных объектов, но никак не использует их дальние связи.

в. Задачи вычислительной линейной алгебры

Если решаемая проблема в целом является нестационарной и нелинейной, все равно после этапов временной аппроксимации и квазилинеаризации мы приходим к задачам линейной алгебры, из которых наиболее типичны – это решение СЛАУ или проблемы собственных значений (как правило, частичной, т.е. вычисление нескольких собственных чисел и векторов).

Характерные матрицы, возникающие в сеточных методах решения дифференциальных задач, являются разреженными, ленточными и большими. Это означает, во-первых, что порядки N достигают десятков и сотен миллионов, а ненулевые элементы сосредоточены в некоторой полосе ширины m около главной диагонали, причем величина m и число ненулевых элементов в каждой строке не зависят от N. Дискретизированные алгебраические системы, возникающие из аппроксимации интегральных уравнений (определенных на границе или в объеме расчетной области) являются, наоборот, плотными, но зачастую имеют специальные структурные свойства (например, являются теплицевыми, квази- или блочно-теплицевыми).

Вычислительная линейная алгебра – хорошо продвинутая математическая дисциплина – и содержит большое количество алгоритмов для решения задач с самыми разными типами матриц: вещественными и комплексными, квадратными и прямоугольными, эрмитовыми и неэрмитовыми, положительно определенными и знаконеопределенными. Имеется также соответствующее обширное программное обеспечение в виде библиотек или прикладных пакетов, как свободно распространяемых в Интернете, так и коммерческих. Данные алгоритмы и программы делятся в основном на два класса: ориентированные на плотные матрицы и

на разреженные. Первый из них содержит в основном прямые методы, основанные на точной факторизации матрицы, а второй – предобусловленные итерационные алгоритмы или же быстрые прямые процедуры для специальных матричных структур (например, происходящих из многомерных краевых задач с частично или полностью разделяющимися переменными, где эффективно работают знаменитые методы быстрого преобразования Фурье, циклической редукции и некоторые другие).

Алгебраические задачи – «узкое горлышко» математического моделирования, поскольку потребляемые вычислительные ресурсы нелинейно растут с увеличением порядка N. Поэтому здесь особенно актуально распараллеливание алгоритмов на МВС с общей, разделенной и гибридной памятью. Ключевым принципом является в данном случае алгебраическая декомпозиция областей, или «разделяй и властвуй».

Существенным моментом программного обеспечения для разреженных матриц является оптимизация кода, поскольку здесь для хранения ненулевых элементов неизбежно применение сжатых форматов данных, которые позволяют кардинально сокращать объем хранимой информации, но существенно усложняют реализацию доступа к матричным элементам, что особенно критично при многоуровневой неоднородности кэша и оперативной памяти.

г. Методы оптимизации и решения нелинейных уравнений

Алгоритмы оптимизации решения обратных задач, сводящихся к проблеме условной минимизации функционала, являются бурно развивающейся в последние десятилетия областью вычислительной математики. Здесь развиты модификации методов множителей Лагранжа и внутренних точек, являющихся развитием подходов со штрафными функциями, использование доверительных интервалов для регулировки последовательности шагов, а также варианты алгоритмов Ньютона и последовательного квадратичного программирования для решения возникающих на промежуточных этапах СНАУ.

В данной тематике имеется много актуальных и далеко не решенных вопросов, связанных с поиском глобального минимума и оптимального управления, применения методов теории возмущений и сопряженных уравнений, нахождения градиентов функционалов или функций чувствительности к вариациям исходных данных. Зачастую постановки требуют «штучного» исследования устойчивости и корректности задачи для поиска соответствующего регуляризационного подхода. К этой же области можно отнести изучение нелинейных динамических систем, связанных с эффектами бифуркаций, самоорганизации и хаоса, странных аттракторов и т.д., где зачастую еще остаются открытыми методологические принципы математического моделирования.

д. Постобработка и визуализация результатов, геометрическое и функциональное моделирование

Непосредственная реализация наукоемких алгоритмов в многомерных задачах может составлять отнюдь не главную долю общего вычислительного процесса, поскольку анализ получаемых результатов требует их презентабельной визуализации с предварительной обработкой сеточных функций, что требует выполнения ресурсоемких технологических операций по формированию сечений, изолиний и изоповерхностей, различных графиков с возможной анимацией многоцветных изображений. Такие технологические проблемы особенно актуальны в системах принятия решений по результатам математического моделирования.

Рассмотренные технологические аспекты касаются одной стороны интерфейса – от компьютера к человеку. Не менее актуальна, особенно в интеллектуальном плане, и вторая сторона, связанная со способами представления, варьирования и ввода в компьютер математических моделей, а также с управлением вычислительным экспериментом. В информационном смысле главными здесь являются полнота, универсальность и гибкость средств описания геометрических и функциональных объектов, где к последним относятся спецификации коэффициентов и основных интегро-дифференциальных операторов, а также граничных и

начальных условий для искомых решений систем дифференциальных и/или интегральных уравнений. Имеется еще и третья сторона интерфейса - это взаимодействие с внешним окружением, включающим или сторонние программные комплексы (например, типа САПР) или системы автоматизации экспериментов, производств, природного мониторинга и т.д.

4. О концепциях прикладного программного обеспечения

Традиционные формы программного обеспечения математического моделирования – это библиотеки алгоритмов и пакеты прикладных программ (ППП), ориентированные на решение конкретного класса задач определенным кругом пользователей, которые определяют требования к интерфейсу и функциональным характеристикам.

С точки зрения концепции современного прикладного математического и программного обеспечения кардинальным представляется создание автономных библиотек алгоритмов и инструментариев, поддерживающих различные технологические этапы моделирования (генераторы сеток, аппроксиматоры и т.д.), взаимодействующих между собой на основе разработки согласованных структур данных (геометрических, сеточных, алгебраических, графических). Такой модульный принцип позволит осуществить независимую реализацию, развитие и переиспользование продуктов коллективов разработчиков, а также их адаптацию на вновь проявляемым платформам МВС. Кроме того, эта модель ориентирована на сборку конкретных приложений из представительного набора программных блоков наподобие детского конструктора, что должно обеспечить такие трудно совместимые категории, как универсальность и эффективность. Разумеется, достижение этой цели требует координации усилий вычислительного сообщества, но примеры успешного сотрудничества такого рода уже имеются: в области линейной алгебры общепринятыми являются форматы и стили пакетов BLAS, SPARSE BLAS, а также коллекции типовых матриц для сравнительного тестирования.

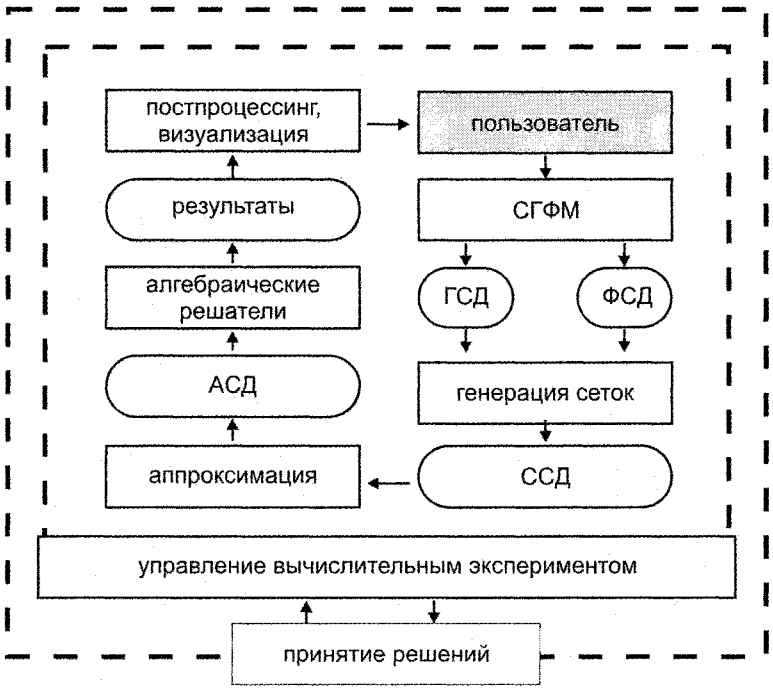
В качестве унифицированной технологической среды решения широкого круга прикладных математических задач можно рассмотреть Базовую Систему Моделирования (БСМ), состоящую из следующих основных компонент, охватывающих все главные этапы крупномасштабного вычислительного эксперимента:

- система геометрического и функционального моделирования (СГФМ), реализующая непосредственный входной пользовательский интерфейс и формирующая в результате своей работы структуры геометрических и функциональных данных (ГСД и ФСД), причем последние являются в определенном смысле вторичными, так как задаются «с привязкой» к объектам ГСД; ряд соответствующих вопросов геометрического и функционального моделирования рассмотрены в [11];
- библиотека генераторов квазиструктурированных сеток (БГС), функционирующая на основе анализа ГСД, а также ФСД, и формирующая, в свою очередь, сеточную структуру данных (ССД), которая полностью и однозначно определяет итоги дискретизации расчетной области, включая взаимосвязи сеточных объектов с функциональными;
- система программной реализации различных аппроксимационных алгоритмов исходной задачи (САЗ), на основе уже построенных ССД и ФСД, результатом исполнения которой является алгебраическая структура данных (АСД), представляющая дискретизованную модель исходной задачи;
- библиотека алгебраических методов (БАМ), осуществляющая на базе ССД решение СЛАУ или СНАУ, проблемы собственных значений или вычисление рекуррентных последовательностей в нестационарных задачах (некоторые проблемы разработки высокопроизводительных алгоритмов решения СЛАУ рассмотрены в [12, 13]);

- постобработка и визуализация результатов моделирования (ПВР), ориентированная на совокупность конкретных способов представления сеточных функций в рамках БСМ и использующая имеющиеся в широком распространении внешние графические редакторы, а также системы автоматизации проектирования (САД, CAM, CAE, PLM);

- набор оптимизационных процедур для решения обратных задач идентификации параметров модели (условная минимизация целевых функционалов) на основе реализации последовательности прямых задач; средства управления сложными вычислительными процессами.

Схема взаимодействия функциональных и информационных компонент БСМ представлена на рисунке.



Структура функциональных и информационных компонент БСМ

Важно отметить, что данные компоненты являются самодостаточными, достаточно автономными и могут разрабатываться независимыми группами специалистов в различных алгоритмических областях на основе согласования множества форматов данных. Структура разрабатываемых программных и информационных компонент предусматривает гибкое взаимодействие с внешними прикладными и системными разработками, в том числе включение в библиотеки БСМ алгоритмических процедур других авторов, для чего создается совокупность конверторов («переходников») структур различных данных. Помимо расширяемого набора вычислительных инструментариев, БСМ включает средства конфигурации (сборки) проблемно-ориентированных ППП, а также пакеты программ для конкретных прикладных областей.

Описанные принципы построения базовой системы математического моделирования касаются в основном только содержательных аспектов. При этом многие актуальные вопросы требуют дальнейших исследований: поддержка коллективной разработки, сопровождения, эксплуатации и развития программных комплексов с длительным жизненным циклом,

обеспечение крупномасштабируемого параллелизма на МВС экстремальной производительности, оптимизация программного кода на алгоритмическом и системном уровне, организация массовых экспериментов на центрах коллективного пользования (Data Centers) по технологиям «облачных» вычислений (Cloud Computing), см. [4 – 7], и т.д.

Работа поддержана грантами РФФИ №008-01-00526 и ОМН РАН №1.3.4

Статья рекомендована к печати программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010» http://agora.guru.ru/pavt.

Список литературы

1. Годунов, С.К. Элементы механики сплошных сред и законы сохранения / С.К. Годунов, Е.И. Роменский. – Новосибирск: Науч. кн., 1998.
2. Hiptmair, R. Finite elements in computational electromagnetism / R. Hiptmair. – Acta Numerica, Cambridge Univ. Press., 2002. – С. 237 – 339.
3. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: БХВ-Петербург, СПб, 2002.
4. Asanovic, K. The Parallel Computing Laboratory at U.C.Berkeley: A Research Agenda Based on the Berkeley View. Techn. Rep. UCB/EECS – 2008-23.
5. Dongarra, J., Beckman, P., et. al. – IESP: International Exascale Software Project. Road Map, 18 Nov., 2009, www.exascale.org.
6. Armbrust, M. et al. About the Clouds: A Berkeley View of Cloud Computing.–Technical Report No. UCB/EECS – 2009-28 http://www.eecs.berkeley.edu/Pubs.
7. Amarasinghe, S., et. al. Exascale Software Study: Software Challenges in Exstreme Scale System. – DARPA report, 2009, www.users.ece.gatech.edu.
8. Марчук, Г.И. Математическое моделирование общей циркуляции атмосферы и океана / Г.И. Марчук, В.П. Дымников и др. – Л.: Гидрометеоиздат, 1984.
9. Комплексное моделирование технологических процессов в алюминиевом электролизере / И.С. Голосов, Н.И. Горбенко, Я.Л. Гурьева, и др. // Актуальные научно-технические проблемы алюминиевой промышленности России. – М., 2003. – С. 72 – 80.
10. Ильин, В.П. О численном решении прямых и обратных задач электромагнитной георазведки / В.П. Ильин // Сиб. журн. выч. мат. – 2003. – Т. 6, № 4. – С. 381 – 394.
11. Ильин, В.П. Геометрическая информатика моделей сплошных сред / В.П. Ильин, Д.Ю. Трибис // Вычислительные методы и программирование. – 2009. – Т. 10, №1 – С. 306 – 313.
12. Ильин, В.П. Проблемы высокопроизводительных технологий решения больших разреженных СЛАУ / В.П. Ильин // Вычислительные методы и программирование. – 2009. – Т. 10, № 1. – С. 130 – 136.
13. Krylov: библиотека высокопроизводительных алгоритмов для решения разреженных СЛАУ / В.П. Ильин, Е.А. Ицкович, А.В. Петухов и др. – Современные проблемы математического моделирования: тр. XIII Всерос. конф.-шк. – Ростов-на-Дону, 2009. – С. 110 – 118.

Ильин Валерий Павлович, доктор физико-математических, профессор, лаборатория вычислительной физики, Институт вычислительной математики и математической геофизики СО РАН, ilin@sscc.ru.

Поступила в редакцию 26 апреля 2010 г.

УДК 004.72

ОТЕЧЕСТВЕННАЯ КОММУНИКАЦИОННАЯ СЕТЬ 3D-ТОР С ПОДДЕРЖКОЙ ГЛОБАЛЬНО АДРЕСУЕМОЙ ПАМЯТИ

А.А. Корж, Д.В. Макагон, А.А. Бородин, И.А. Жабин, Е.Р. Куштанов, Е.Л. Сыромятников, Е.В. Черемушкина

RUSSIAN 3D-TORUS INTERCONNECT WITH GLOBALLY ADDRESSABLE MEMORY SUPPORT

A.A. Korzh, D.V. Makagon, A.A. Borodin, I.A. Zhabin, E.R. Kushtanov, E.L. Syromyatnikov, E.V. Cheryomushkina

В статье рассматриваются детали реализации и первые результаты макетирования разработанной в НИЦЭВТ межузловой коммуникационной сети с топологией 3D-тор. Данная сеть может эффективно применяться как в вычислительных кластерах небольшого и среднего размера, так и в суперкомпьютерах транспетафлопсного уровня производительности. Особое внимание в статье уделено библиотеке параллельного программирования SHMEM, посредством которой программисту предоставляется доступ к глобально адресуемой памяти.

Ключевые слова: коммуникационная сеть, суперкомпьютер, 3D-тор, глобально адресуемая память

This paper gives the overview and early results of prototyping of the 3d-torus interconnect developed in NICEVT, Moscow. This interconnect was designed to be equally effective in small-size computing clusters and petascale systems. The key features of the interconnect are high fault-tolerance, high message rate per core supported by host adapter and hardware support for globally addressable memory provided via SHMEM parallel programming library.

Keywords: interconnection network, supercomputer, 3D-torus, globally addressable memory

Введение

Межузловая коммуникационная сеть является одной из важнейших частей суперкомпьютера, определяющей его производительность и масштабируемость. Разрыв тактовых частот процессора и подсистемы локальной и удалённой памяти постоянно увеличивается, поэтому именно время выполнения обращений в память удалённого узла становится «узким местом» при распараллеливании большинства задач на кластерах и суперкомпьютерах.

Коммуникационные сети современных суперкомпьютеров можно разделить на два класса: коммерческие (Ethernet, Infiniband, Quadrics, Myrinet) и заказные (IBM BlueGene, Cray XT, SGI Altix UV).

Коммерческие сети разрабатываются для широкого спектра применений, использование их в качестве коммуникационной сети суперкомпьютеров — лишь одно из них. В связи с этим в жертву универсальности приносится производительность в ряде режимов, в частности,

в коммерческих сетях не оптимизирована передача коротких сообщений, нет адаптивной передачи, аппаратно не поддержана отказоустойчивость при отказах линков.

Для систем среднего уровня (до 1000 узлов), а также для вычислительных кластеров, используемых в режиме счета нескольких небольших задач, умещающихся по потреблению памяти в одной стойке, чаще всего именно фактор цены оказывается важнее возможности эффективно считать одну задачу на всем суперкомпьютере. Как правило, для таких систем применяются коммерчески доступные коммуникационные сети, среди которых в последнее время с огромным отрывом лидируют сети стандарта Infiniband.

Для суперкомпьютеров с большим числом узлов (10–100 тысяч), достигших в настоящее время петафлопсного уровня производительности, вопрос сбалансированности производительности сети и процессоров является критическим, поэтому применение коммерческих технологий, таких как Infiniband, оказывается неприемлемым.

В связи с этим крупнейшие производители суперкомпьютеров, такие как Cray, IBM и SGI, используют коммуникационные сети собственной разработки: Cray Seastar2+ и будущая сеть машин серии Cray Baker — Gemini High-Speed Network, IBM Bluegene Torus Network, SGI NUMAlink. Такие сети недоступны для коммерческой продажи и используются исключительно в машинах высшего эшелона производительности.

В таких системах в настоящее время используются преимущественно сети с топологией 3D-тор [4]. Основные причины этого — трёхмерность коммуникационного шаблона многих физических задач, простая расширяемость системы, регулярность упаковки сети по стойкам с учетом экономии на межстоечных кабелях, хорошая масштабируемость и т.д.

При всех достоинствах топологии 3D-тор, она становится малоэффективной, когда число вычислительных узлов измеряется многими десятками тысяч (не говоря уже о сотнях тысяч). Переход к торам большей размерности хотя и увеличивает производительность (вместе со стоимостью), однако требует заметно более сложного способа упаковки и плохо соотносится с коммуникационными шаблонами физических задач.

В связи с этим в последнее время широко исследуются альтернативные топологии, в частности, различные варианты модифицированных деревьев (fat-tree) и сетей Клоса. Так, в числе последних разработок фирмы Cray — коммутатор YARC (для суперкомпьютера Cray X2), в ближайшем будущем планируется выпуск коммутатора Cray Aries (предположительно, с топологией dragonfly [5]).

Важным аспектом, определяющим применение заказных сетей, является поддержка глобально адресуемой памяти как основного режима программирования стратегических суперкомпьютеров. Данная парадигма может эффективно применяться при использовании обычных коммерческих микропроцессоров [1].

На отечественном рынке на данный момент присутствуют только зарубежные коммерческие разработки, не подходящие для суперкомпьютеров высшего эшелона производительности. Более того, каналы импорта компонентов зарубежных коммерческих сетей сравнительно легко могут быть перекрыты, чего с процессорами или памятью сделать практически невозможно. В связи с этим в последние несколько лет НИЦЭВТ ведет разработку заказной высокоскоростной коммуникационной сети в рамках проекта разработки суперкомпьютера стратегического назначения (СКСН).

Разрабатываемой коммуникационной сети уделяется особое внимание, как одному из критических компонентов СКСН. В настоящее время изготовлен, налажен и находится в опытной эксплуатации шестиузловой вычислительный кластер, построенный на основе второго поколения макетных образцов маршрутизаторов высокоскоростной коммуникационной сети с топологией 2D-тор на основе ПЛИС. Его тестирование специалистами ряда организаций показало хорошие результаты, сопоставимые с образцами коммерческой сети Infiniband DDR. При этом в наиболее тяжёлом режиме работы с интенсивным обменом короткими па-

кетами сообщений достигнутые характеристики существенно превосходили сеть Infiniband, в том числе и последнего поколения Infiniband QDR 4x.

В настоящее время изготовлен и находится в стадии отладки макетный образец третьего поколения, построенный на основе наиболее современных ПЛИС фирмы Xilinx; его характеристики в 1,8 раза лучше, достигаемый темп передачи сообщений — 14,9 миллионов пакетов в секунду.

В маршрутизаторе на аппаратном уровне поддерживаются четыре операции с удалённой памятью: асинхронное чтение (get), асинхронная запись (put), атомарное сложение (add), атомарное исключающее или (xor), а также различные методы синхронизации и контроль возврата выданных запросов на асинхронное чтение.

Реализована стандартная библиотека Cray SHMEM с дополнениями, позволяющими более эффективно выполнять типовые операции, и библиотека MPI на основе MPICH.

Далее в статье приводится описание разработанного макетного образца маршрутизатора и методология написания параллельных программ на языках C/C++ и Fortran с использованием аппаратно поддерживаемой в макете глобально адресуемой памяти. В заключении приведены основные полученные на текущий момент результаты и сформулированы планы на будущее.

1. Архитектура маршрутизатора

Разработка высокоскоростной коммуникационной сети в НИЦЭВТ ведётся уже несколько лет в рамках проекта создания СКСН. В 2007-м году был изготовлен полнофункциональный макет M2 с топологией 2D-тор, состоявший из шести узлов. Пропускная способность межузловых линков — 6,25 Гбит/с в каждую сторону, интерфейс с процессором — PCI-Express x4 (10 Гбит/с).

Обобщённая структурная схема разработанного маршрутизатора приведена на рис. 1. В его состав входят 4 линка (Link) — по два на каждое измерение тора, кроссбар (Crossbar), интерфейс с процессором вычислительного узла (NI+PCI) и сервисный процессор (Service PowerPC).

Каждый линк содержит в ПЛИС блок сериализатора-десериализатора и состоит из входной и выходной частей. Входная часть включает блоки виртуальных каналов и арбитры.

В маршрутизаторе реализован алгоритм бездедлоковой детерминированной маршрутизации, основанный на правилах «пузырька» и «порядка направлений» [3].

С помощью виртуальных каналов реализовано две независимых подсети для запросов и ответов, чтобы избежать дедлоков «процессор — сеть» (запросы не должны задерживать доставку ответов). Аппаратно поддерживается обход отказавших узлов (алгоритм «нестандартного первого и последнего шага») [2].

Выходная часть реализует надёжный протокол доставки сообщений, обеспечивающий непрерывную передачу пакетов и гарантирующий повтор в случае ошибок на линии.

Управление потоком данных осуществляется с помощью передачи кредитной информации (кредитов). Пакет передаётся следующему узлу, только если в принимающем буфере соответствующего виртуального канала гарантировано наличие необходимого свободного места. Такая гарантия предоставляется с помощью отсылки «кредитов»: как только в некотором входном буфере маршрутизатора освобождается место (один пакет передаётся в кроссбар и далее — соседнему узлу), по линку, связывающему данный входной буфер с выходным буфером другого маршрутизатора, отсылается «кредит» — информация об освободившемся месте. Основываясь на этой информации, в каждом маршрутизаторе определяется возможность передачи пакетов по линкам в каждый момент времени. «Кредиты»

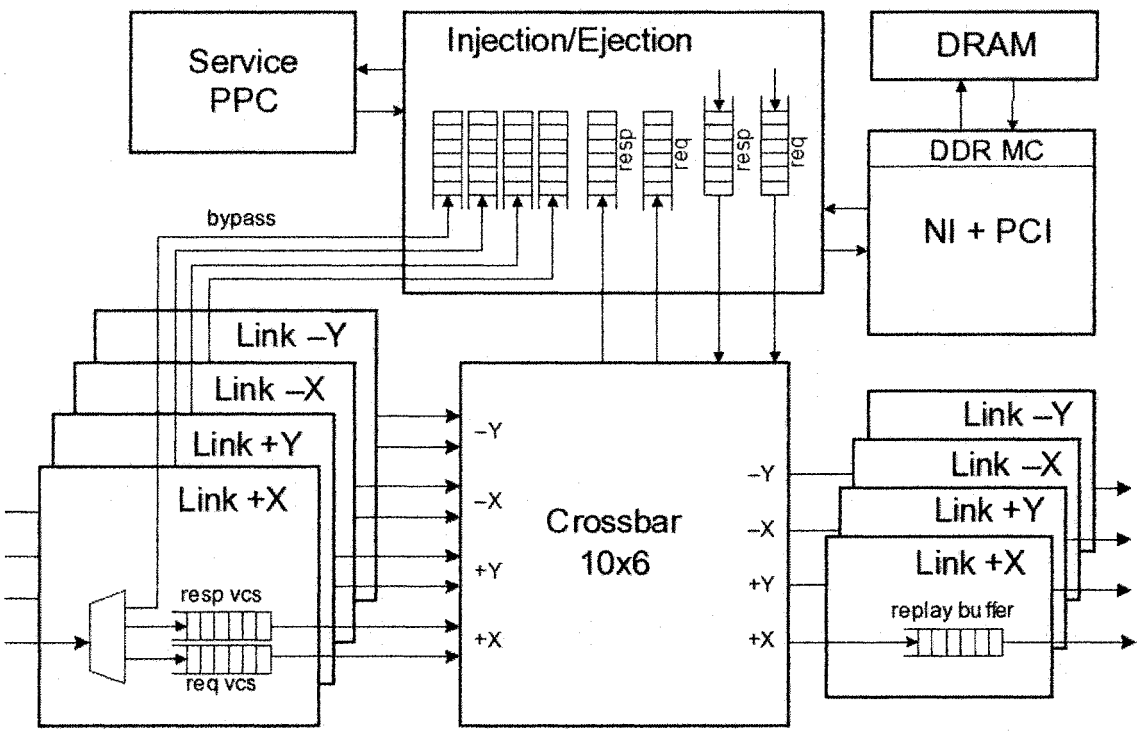


Рис. 1. Обобщённая структурная схема разработанного в НИЦЭВТ маршрутизатора

гарантируют наличие определённого количества свободного места (флитов), при этом за время передачи кредита свободное место может увеличиться (но гарантированно не уменьшится).

Предотвращение сбоев при передаче пакетов по линку обеспечивается собственным протоколом гарантированной передачи, реализованным поверх Xilinx Aurora link-layer protocol [6]. В случае несовпадения контрольной суммы при передаче пакета от одного маршрутизатора другому, пакет запрашивается повторно. В случае долговременного выхода из строя ряда линков возможна перенастройка таблиц маршрутизации сети и последующая перемаршрутизация пакетов. В сетевом интерфейсе содержится настраиваемая сервисной подсистемой таблица маршрутизации, позволяющая задавать различные маршруты между узлами.

Интерфейс с процессором вычислительного узла включает IP-ядро стандартного интерфейса PCI Express или HyperTransport, в зависимости от варианта исполнения маршрутизатора. Основной функцией интерфейса является инжекция пакетов в сеть. Для каждого пакета в автоматическом режиме на основании данных из встроенной таблицы маршрутизации определяется физический адрес и направление передачи. При приёме пакетов из сети интерфейс с процессором извлекает из полученных пакетов данные и записывает их напрямую в оперативную память вычислительного узла с использованием технологии прямого доступа в память (DMA). В некоторых случаях для повышения эффективности пакеты сначала накапливаются в памяти маршрутизатора, а уже затем записываются в память узла.

Для реализации подсистемы отказоустойчивости, позволяющей корректно обрабатывать отказы отдельных линков и даже вычислительных узлов, использовано встроенное в ПЛИС процессорное ядро, выполняющее функции коммуникационного процессора, следящего за состоянием сети и, при отказах, осуществляющего изменение данных в таблице маршрутизации.

Макетный образец третьего поколения, изготовленный в 2009 году, имеет пропускную способность линка 13 Гбит/с, интерфейс с процессором PCI Express x8. Поддерживается топология трёхмерный тор (используются 6 линков), адаптивная маршрутизация и агрегация коротких сообщений от разных узлов. Задержка между соседними узлами не превышает 1,2 мкс.

На аппаратном уровне поддерживаются четыре операции с удалённой памятью: асинхронное чтение (get), асинхронная запись (put), атомарное сложение (add), атомарное исключающее ИЛИ (xor), а также аппаратный контроль возврата выданных запросов на асинхронное чтение.

Для эффективной реализации операций с удалённой памятью потребовалось снизить накладные расходы, вносимые трансляцией адресов для поддержки виртуальной памяти. Было решено статически выделить область физической памяти, с которой может работать как маршрутизатор, так и пользовательская задача. Это позволило упростить интерфейс маршрутизатора, убрав громоздкие и неэффективные схемы трансляции адресов, присутствующие в коммерческих коммуникационных сетях, таких как Infiniband.

Одной из главных задач, поставленных перед разработчиками маршрутизатора для СКСН, было получение высокой пропускной способности на коротких пакетах. Для этого были использованы несколько виртуальных каналов; при передаче через PCI Express был применён режим Write Combining при посылке пакетов. При приёме пакетов была добавлена возможность агрегации коротких сообщений во встроенной памяти на плате маршрутизатора (через шину PCI данные сбрасываются большими блоками). На задачах типа умножения разреженной матрицы на вектор (SpMV) данные решения оказались довольно эффективными.

2. Программное обеспечение

Коммуникационная сеть, разработанная в НИЦЭВТ, имеет развитую программную поддержку, включающую реализацию стандартной библиотеки SHMEM (версии фирмы Cray с дополнениями, позволяющими более эффективно выполнять типовые операции) для языков C/C++ и Fortran. Также реализована версия библиотеки MPI 1.1 (MPICH), отлажена базовая реализация MPI-2, проведено функциональное тестирование с помощью пакета Intel MPI Benchmarks 3.2 (с проверкой на корректность доставки сообщений).

Поддерживается написание параллельных программ, использующих одновременно и MPI, и OpenMP, и SHMEM. Рекомендуется гибридный режим программирования SHMEM+OpenMP и SHMEM+OpenMP+MPI.

2.1. Библиотека SHMEM

Модели параллельного программирования можно разделить на два класса, в зависимости от того, на коммуникациях какого типа они основаны: двусторонних (передача сообщений) либо односторонних (обращения к удалённой памяти).

В двусторонних коммуникациях активное участие принимают две стороны: одна отправляет записываемое слово, а вторая — ждёт прихода слова, после чего копирует полученное слово из буфера приёма в нужную ячейку памяти. Адрес, куда записать слово в памяти второго узла, указывается самим получателем.

В односторонних коммуникациях активное участие принимает лишь инициатор: при записи он отсылает записываемое слово, а то, достигнув по сети адресата, напрямую записывается в память второго узла (при этом процессорное время на ожидание и запись вторым узлом не тратится). Адрес, куда записать слово в памяти второго узла, указывается отправителем.

Программы на MPI используют двусторонние коммуникации Send/Recv; переход к односторонним коммуникациям Put/Get (поддерживаемым не только интерфейсом SHMEM, но и стандартом MPI-2) потенциально способен повысить продуктивность разработки и сопровождения параллельных программ и эффективность их выполнения.

Такому переходу мешает в основном тот факт, что большая часть имеющихся библиотек написаны на MPI, большая часть разработанных параллельных алгоритмов созданы для двухсторонней парадигмы Send/Recv, большая часть программистов и математиков, занимающиеся распараллеливанием научных задач, привыкли думать именно в парадигме Send/Recv и не хотят переходить на другие парадигмы, не видя существенных преимуществ.

Система программирования SHMEM была разработана фирмой Cray более 15 лет назад, как интерфейс односторонних коммуникаций, способный стать эффективной альтернативой и дополнением к MPI и PVM. Интерфейс SHMEM поддерживается всеми MPP-системами фирмы Cray (Cray T3E, Cray XT5), Silicon Graphics (SGI Altix), интерконнектами Quadrics (QsNetIII). Также библиотека SHMEM (с некоторыми дополнениями) была реализована в системе МВС-Экспресс (под руководством А.О. Лациса).

По сути SHMEM реализует простейший вариант программирования в стиле PGAS (partitioned global address space). У каждого узла есть локальная память; каждому узлу также доступна удалённая память: узел может напрямую обращаться к локальной памяти любого узла системы. Поскольку обращения к удалённой памяти происходят через коммуникационную сеть, время их выполнения заметно больше, а темп — меньше, чем у обращений к локальной памяти. Ожидать выполнения каждой одиночной операции крайне дорого, поэтому требуется, чтобы программист явно выделял удалённые обращения.

В отличие от других PGAS-языков (например, UPC) SHMEM заставляет программиста явно выделять внешние обращения с помощью функций `shmem_put`, `shmem_get`, при этом дальнейшая группировка обращений и оптимизация выполняются аппаратно.

Большинство прикладных задач, использующих MPI, основано на простом коммуникационном шаблоне: preposted `MPI_Recv` + `MPI_Send`; однако такой шаблон гораздо естественнее записывается с использованием SHMEM как асинхронная запись (`shmem_put`) с подтверждением прихода сообщений. Подобным образом авторами статьи были переписаны на SHMEM тесты NPB CG и NPB UA.

Основу SHMEM составляют две операции: `shmem_put` — запись в память удалённого узла и `shmem_get` — чтение из памяти удалённого узла. Синхронизация происходит с помощью встроенной функции `shmem_barrier`. Возможность напрямую обращаться к удалённой памяти даёт большинство преимуществ работы с «общей памятью», не накладывая никаких дополнительных ограничений на то, как память распределена физически.

Главная проблема односторонних операций — необходимость при отправке указывать адрес в памяти другого узла. Пересылать адреса между узлами — неэффективно, поэтому было предложено следующее решение: на всех узлах использовать симметричные массивы (симметричные указатели), имеющие одинаковые адреса на всех узлах. Если в программе объявлен симметричный массив, то у каждого процесса будут свои локальные данные, однако адреса элементов и размер массива у всех процессов будут одинаковые. Благодаря этому процессы могут обращаться к массивам других узлов, используя локально известные адреса. Симметричное выделение памяти происходит с помощью функции `shmem_alloc`.

Помимо операций записи (`shmem_put`) и чтения (`shmem_get`), реализованы атомарные операции сложения (`shmem_add`) и исключающее ИЛИ (`shmem_xor`).

В макетах М2/М3 реализован SHMEM с поддержкой эффективной передачи коротких сообщений, атомарных операций и быстрого барьера, поэтому большинство алгоритмов (CG, FFT, UA) на SHMEM не только проще записываются, но и эффективнее выполняются.

2.2. Порядок сообщений и синхронизация узлов

При односторонних коммуникациях особое значение приобретают функции синхронизации и порядок сообщений.

В макете M2 используется модель программирования, основанная на односторонних коммуникациях без подтверждений; все пакеты при этом передаются детерминировано.

Данная модель предполагает стиль программирования, в котором каждая итерация алгоритма делится на две фазы: подкачка и счёт. Например: асинхронная подкачка данных для (i+1)-ой итерации; счёт i-ой итерации; ожидание завершения подкачки для (i+1)-ой итерации; асинхронная подкачка данных для (i+2)-ой итерации; счёт (i+1)-ой итерации и т.д.

Обычно подкачка выполняется с помощью операций чтения, однако стандартным приёмом программирования в стиле PGAS является замена подкачки с помощью чтений на посылку узлом-владельцем элементов с помощью операций записи. На множестве реальных задач (CG, UA) этот приём даёт значительный выигрыш, так как снижает нагрузку на сеть, потому как операция чтения посылает по сети два пакета (запрос и ответ), а операция записи на многих системах реализована посылкой пакета только в одну сторону.

Ожидание завершения подкачки (с помощью операций `shmem_put`, `shmem_get`, `shmem_add` и `shmem_xor`) может осуществляться стандартным методом — с помощью вызова общего барьера (`shmem_barrier`). При этом также реализованы несколько специальных методов, функционально заменяющих барьер для контроля выполнения выданных асинхронных записей и чтений, позволяющих достичь намного большей производительности.

Контроль возврата выданных запросов на асинхронное чтение реализуется блокирующей функцией `shmem_syncreads` (используется аппаратно реализованный счётчик в сетевом интерфейсе: при отправке запроса на чтение он увеличивается на единицу, при получении ответа — уменьшается на единицу).

Контроль порядка выданных асинхронных записей реализуется функцией `shmem_fence`: гарантируется, что все отосланные до `shmem_fence` сообщения узлом A узлу B достигнут адресата строго до сообщений, отосланных после `shmem_fence`. Таким образом, порядок гарантируется только для пакетов, имеющих одинаковых отправителей и получателей. При детерминированной маршрутизации такой порядок соблюдается по определению и не требует какой-либо синхронизации узлов.

Контроль выполнения выданных асинхронных записей осуществляется с помощью блокирующей функцией `shmem_quiet`. Стандартный метод реализации — потребовать подтверждений для всех операций. Если пакеты были отосланы детерминировано, то достаточно запросить подтверждения только последней отправленной операции для каждого узла. Часто именно с помощью этой функции гарантируется глобальный порядок сообщений: ни одно сообщение, отосланное до `shmem_quiet`, не должно обогнать сообщение, отосланное после `shmem_quiet`.

Функция `shmem_barrier` гарантирует, что каждый процесс продолжит работу только после того, как все узлы дойдут до места вызова барьера в коде программы. Данная функция может быть реализована в две фазы: `shmem_barrier_notify` (начало барьера — неблокирующая) и `shmem_barrier_wait` (ожидание окончания барьера — блокирующая).

В большинстве реализаций SHMEM барьер выполняет важную упорядочивающую функцию: узлы выйдут из барьера только тогда, когда каждый узел получит все адресованные ему сообщения.

Однако данное условие является слишком строгим и во множестве (даже в большинстве) задач совсем не требуется. Вполне достаточно, чтобы узел выходил из барьера, как только он получит все отосланные ему до барьера сообщения. При таком условии каждый узел может

продолжить счёт сразу, как только получит все адресованные ему данные, не дожидаясь остальных узлов (вплоть до следующего барьера). Такой «быстрый» барьер в среднем, как минимум, в два раза быстрее; именно он реализован в функции `shmem_barrier` библиотеки SHMEM для макета M2/M3. Обычный «медленный» барьер реализуется выполнением двух «быстрых» барьеров подряд.

Помимо глобального барьера, в котором участвуют все процессы данной задачи, поддерживаются барьеры по заданной группе, где явно указывается (с помощью маски, шага) множество узлов, участвующих в барьере.

3. Результаты тестирования, производительность

3.1. Intel MPI Benchmarks

Intel MPI Benchmarks (IMB) 3.2 — свободно распространяемый набор тестов, предназначенный для оценки эффективности выполнения операций MPI (с проверкой на корректность доставки сообщений) [7]. Ранее данный набор тестов был известен как Pallas MPI Benchmarks (PMB).

На макете M2 на тесте PingPong на сообщениях размером до 2 килобайт пропускная способность находится на уровне Infiniband DDR 4x. На больших сообщениях из-за более медленного линка M2 отстает. Макет M3 превосходит Infiniband QDR 4x на коротких сообщениях и несколько уступает на больших сообщениях (рис. 2).

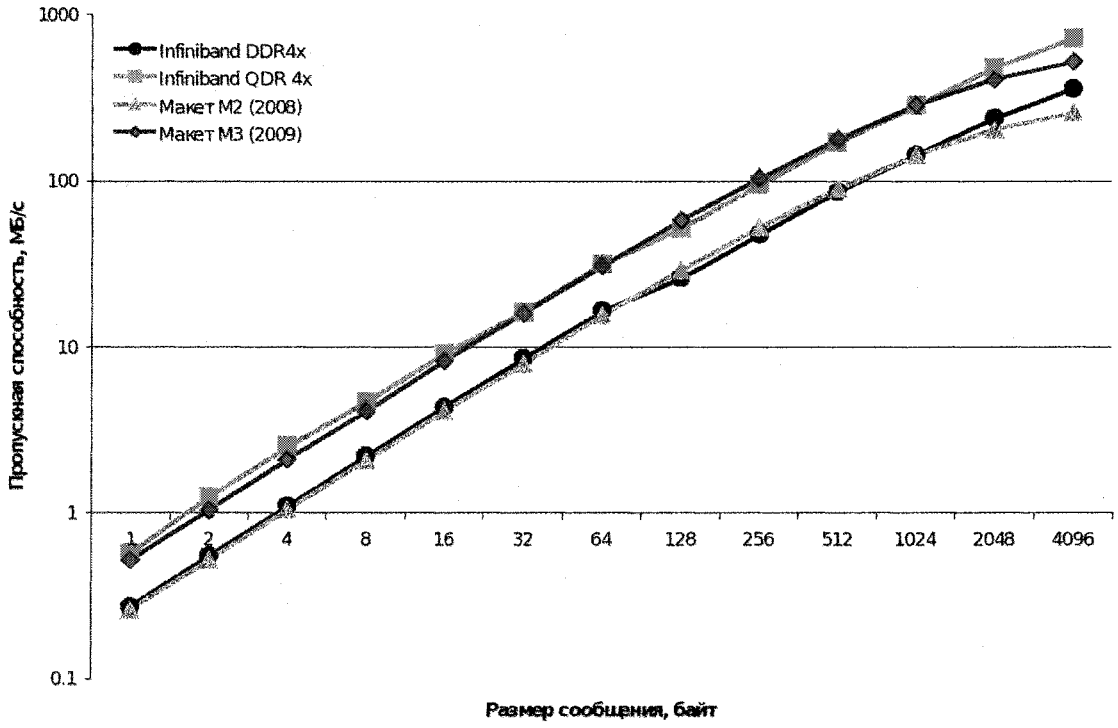


Рис. 2. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте IMB PingPong

На тесте Bcast коммуникационная задержка при сообщениях размером до 1 килобайт на макете M2 на 70% меньше, чем на Infiniband DDR 4x. На больших сообщениях задержка на M2 превосходит уровень Infiniband (рис. 3).

По характеристикам (на всех тестах из пакета IMB) на пакетах до 2КБ реализация MPI на M2 не отстает от реализации MPI на сети Infiniband DDR 4x. На больших пакетах производительность ниже по двум причинам: у M2 существенно меньшая пропускная способность сетевых линков, а также отсутствует поддержка передачи больших сообщений в режиме DMA.

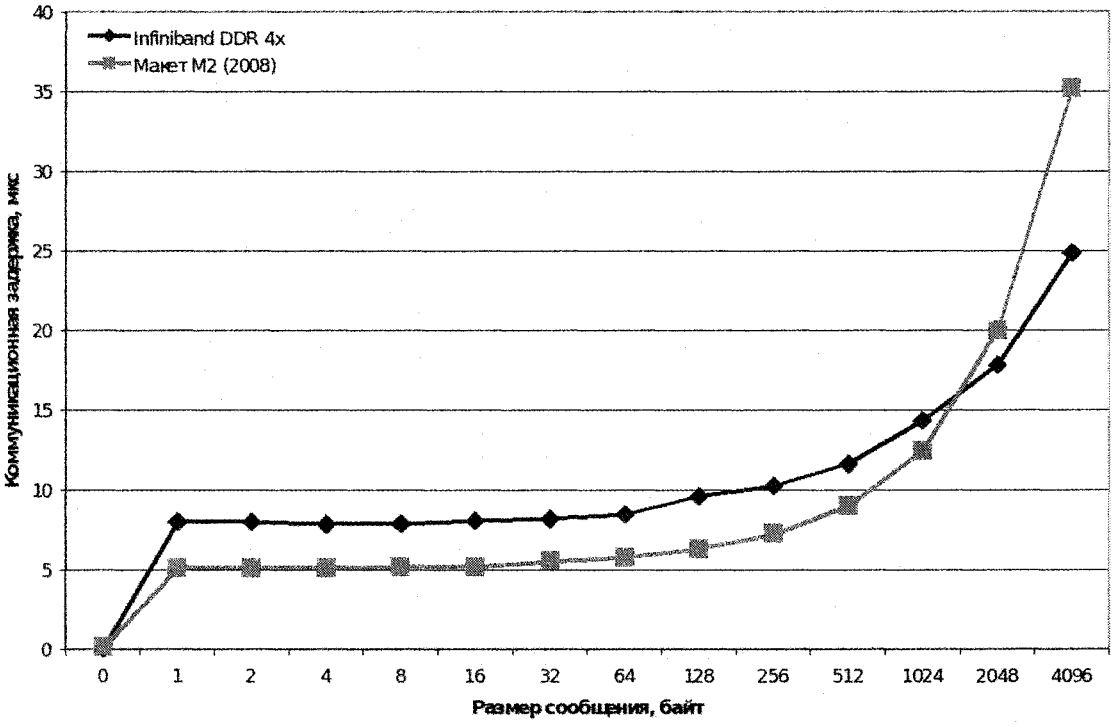


Рис. 3. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте IMB Bcast

3.2. HPCC RandomAccess

RandomAccess — один из семи тестов из набора HPC Challenge Benchmark [7]. Тест характеризует производительность вычислительной системы при наиболее неблагоприятном режиме доступа к памяти.

Тест имеет коммуникационный шаблон все-всем при минимальном размере пакета 8 байт; результатом теста является достигаемое число обновлений ячеек памяти по случайным адресам в секунду.

На макете M2 результаты более чем в 2 раза превышают результаты Infiniband QDR 4x в случае использования API нижнего уровня IB Verbs, и более чем в 10 раз в случае использования MPI (рис. 4).

3.3. Барьерная синхронизация

Барьерная синхронизация — наиболее важная синхронизационная функция, используемая в MPI и SHMEM.

На макетах M2 и M3 до 4-х узлов барьерная синхронизация работает на уровне Infiniband QDR 4x, на большем числе узлов Infiniband QDR 4x значительно проигрывает — выполняется в несколько раз медленнее (рис. 5).

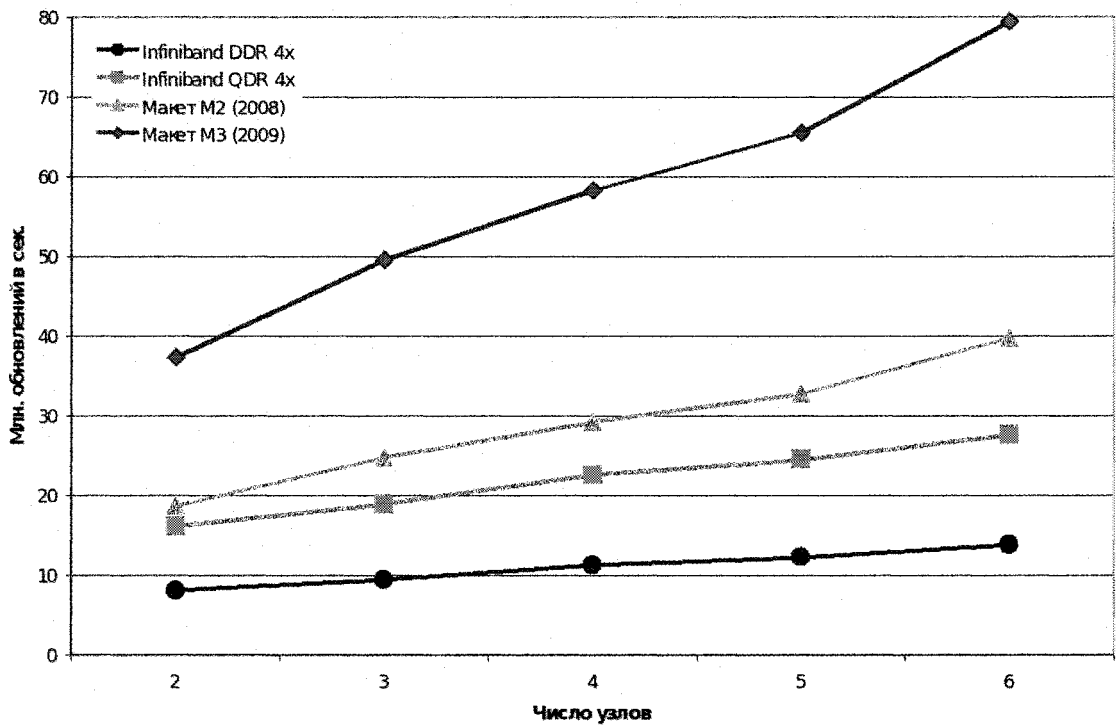


Рис. 4. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте HPCC RandomAccess

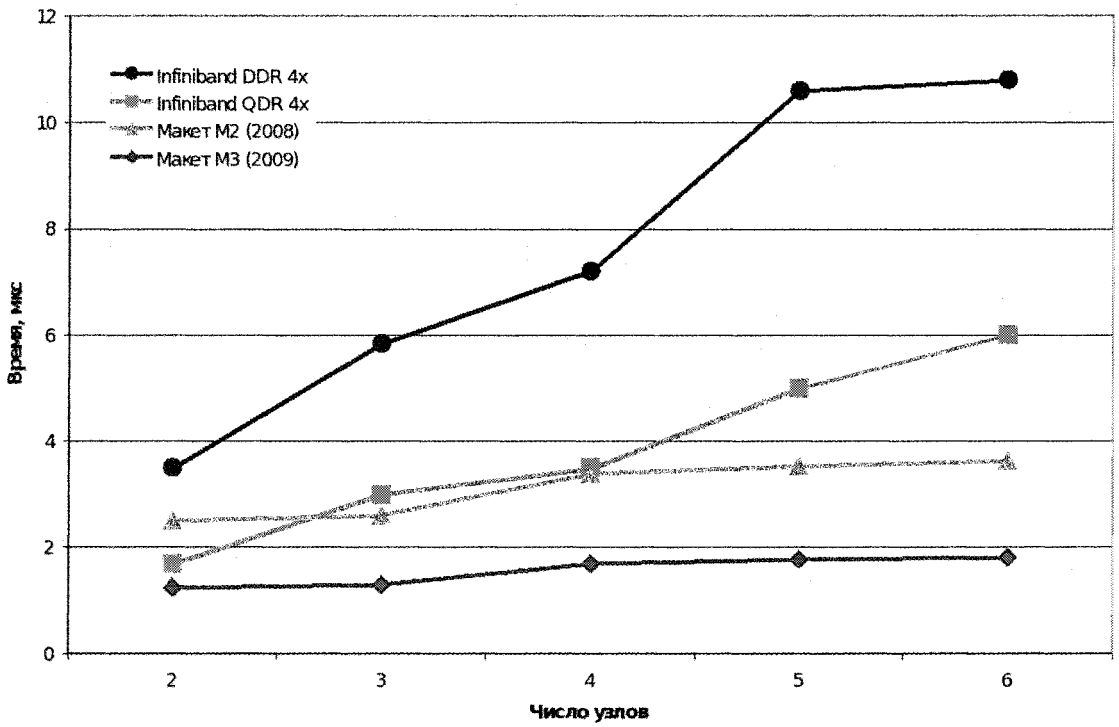


Рис. 5. Сравнение коммуникационной сети, разработанной НИЦЭВТ, с сетью Infiniband на тесте времени выполнения барьерной синхронизации

3.4. NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB) — набор вычислительных задач, предназначенный для измерения производительности суперкомпьютеров в различных актуальных научных приложениях [8]. Набор включает тесты Conjugate Gradient (CG), Fast Fourier Transform (FT), MultiGrid (MG), Integer Sort (IS), Unstructured Adaptive (UA), Data Traffic (DT) и т.д.

Все тесты написаны на языке Fortran 77; существуют параллельные реализации NPB, использующие MPI, OpenMP, HPF, однако версий, использующих односторонние коммуникации в парадигме PGAS — нет.

Авторами статьи были реализованы с использованием SHMEM два теста — CG и UA. Ядром теста CG [8] является итеративное умножение разреженной матрицы на вектор. Поскольку реализованное в NPB блочно-циклическое разбиение матрицы специально нацелено на увеличение доли нерегулярных коммуникаций, на малом числе узлов производительность версии на SHMEM оказалась сравнима с версией на MPI. Оптимизированная версия, в которой используется блочное разбиения матрицы по строкам, позволяет более эффективно реализовать алгоритм на SHMEM, что дачт прирост в среднем на 36%.

В тесте UA [9] решается задача Дирихле уравнения теплопереноса в трехмерной кубической области на нерегулярной декартовой сетке. Источник тепла представляет собой шар, движущийся с постоянной скоростью. Для решения применяется нерегулярная сетка, причём каждые несколько шагов происходит её адаптация: в областях с большим градиентом температуры сетка измельчается, с малым — укрупняется. Таким образом, тест нацелен на измерение производительности при нерегулярном динамически изменяемом шаблоне доступа к памяти.

Тест UA, появившийся в NPB 3.1, до этого распараллеливался лишь с использованием OpenMP, распараллеливание в модели MPI для данной задачи никем успешно не произведено до сих пор (т.е. невозможно на данный момент получить результаты об ускорении на системах с Infiniband), поэтому полученные результаты для SHMEM представляют особый интерес.

Ускорение на макете M2 для версии на SHMEM на 6-ти узлах (ядрах) составляет 5,4 раза (рис. 6); ускорение OpenMP-версии на других системах для 6-ти ядер не превосходит 2,8 раз; для 16-ти ядер — 4,4 раз. Ускорение измерялось для класса C относительно последовательного времени выполнения теста, которое для макета M2 составляло 1527 с, для MVS-Express — 1292 с, для SGI Altix 3700 — 1585 с (OpenMP версия).

4. Заключение

Разработанная коммуникационная сеть 3D-тор с поддержкой глобально адресуемой памяти может быть использована как в кластерах, так и в суперкомпьютерах транспетафлопсного уровня производительности.

Коллективом программистов были адаптированы библиотеки MPI 1.1, SHMEM и MPI-2, а также тесты IMB, NetPIPE. Благодаря аппаратной поддержке односторонних коммуникаций авторам удалось эффективно реализовать тесты NPB CG и UA, используя модель программирования SHMEM.

Модель программирования SHMEM, будучи простейшим представителем семейства языков в парадигме PGAS, является более продуктивной и эффективной, чем модель MPI, при этом она не содержит серьёзных недостатков более сложных языков PGAS, таких как UPC и Co-array Fortran. Данная тема будет подробно рассмотрена в следующей статье. Модель программирования SHMEM обладает достаточной гибкостью и функциональностью, чтобы реализовать все остальные языки PGAS.

В настоящее время успешно используется макетный образец сети второго поколения

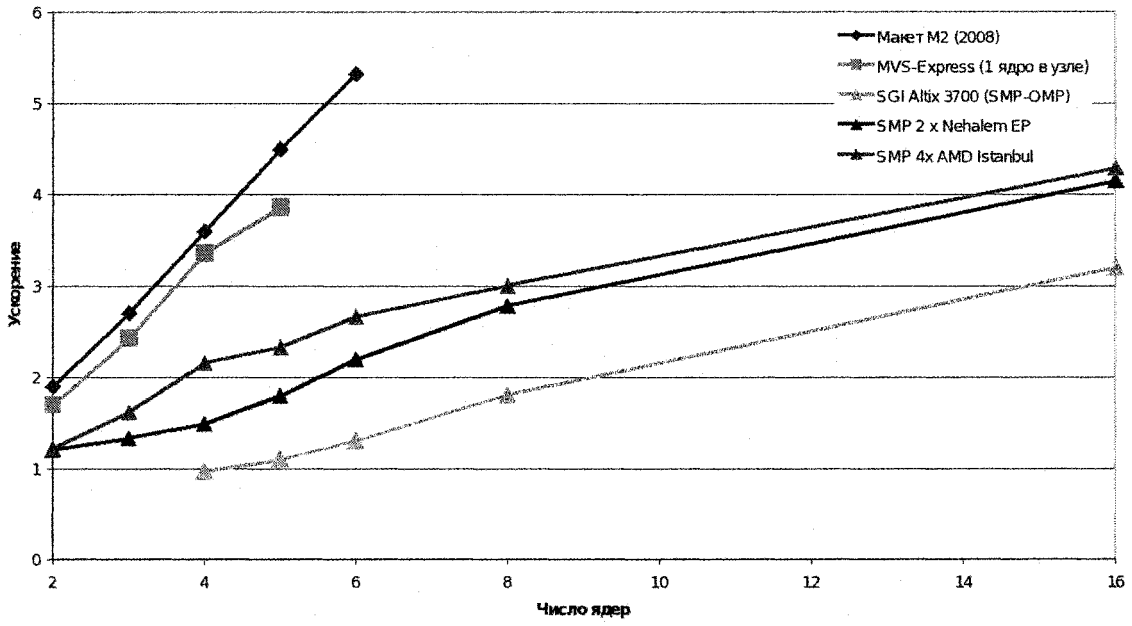


Рис. 6. Ускорение на тесте NPB UA от числа используемых ядер

(М2), заканчивается тестирование макета третьего поколения (М3) и уже ведутся работы по созданию макета четвёртого поколения (М4).

Макет третьего поколения включает в себя эффективную поддержку коллективных операций broadcast и allreduce (по кольцам и измерениям тора), кольцевые буфера в памяти маршрутизатора, поддержку «активных сообщений», RPC и подтверждений.

Макет четвёртого поколения сможет адекватно конкурировать с последними разработками как отечественных фирм, так и зарубежных компаний (в том числе с Infiniband EDR и Cray Gemini). Он будет включать отдельные сети для глобальной синхронизации и коллективных операций (с древовидной топологией), также будет произведена замена интерфейса PCI-Express на более производительный HyperTransport. Параллельно с этим ведутся работы по разработке блейдов под процессоры AMD Opteron, на которые будет интегрирован кристалл коммуникационной сети и мультитредовый ускоритель. Разрабатывается специализированная ОС суперкомпьютера стратегического назначения на базе ядра ОС Linux.

Коллектив разработчиков выражает признательность Л.К. Эйсымонту — за формирование первоначального внимания и интереса к тематике высокоскоростных коммуникационных сетей.

Статья рекомендована к печати программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010» http://agora.guru.ru/pavt.

Работа выполнена при поддержке РФФИ, грант №09-07-13596-офи_ц.

Литература

1. Evaluating NIC hardware requirements to achieve high message rate PGAS support on multicore processors / Keith D. Underwood, Michael J. Levenhagen, Ron Brightwell // Proceedings of the 2007 ACM/IEEE conference on Supercomputing. – 2007. – P. 31 – 40.

2. Scott, S. Synchronization and communication in the T3E multiprocessor / Steven L. Scott //

Proceedings of the seventh international conference on Architectural support for programming languages and operating systems. – 1996. – P. 26 – 36.

3. Duato, J. Interconnection Networks, An Engineering Approach / J. Duato, S. Yalamanchili, L. Ni. – IEEE Computer Society Press, 1997. – 812 p.
4. Джосан, О.В. Организация коммуникационной сети для транспетафлопсных суперкомпьютеров / О.В. Джосан, А.А. Корж // Труды ИСА РАН: динамика неоднородных систем. – 2008. – Т. 32, № 3. – С. 267 – 274.
5. Technology-Driven, Highly-Scalable Dragonfly Topology / J. Kim, W.J. Dally, S. Scott, D. Abts // Computer Architecture. – 2008. – № 08. – С. 77 – 88.
6. Aurora 8B/10B Protocol Specification SP002 (v2.1) [Электронный ресурс]. – Xilinx, Inc, 2009. URL: http://www.xilinx.com/products/design_resources/conn_central /grouping/aurora.htm.
7. Saini, S. Performance evaluation of supercomputers using HPCC and IMB Benchmarks / S. Saini et al. // Journal of Computer and System Sciences. – 2008. – № 74. – P. 965 – 982.
8. Jin, H. The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance / H. Jin, M. Frumkin, J. Yan // NAS Technical Report NAS-99-011 October 1999. – 72 p.
9. Unstructured adaptive (UA) NAS Parallel Benchmark / M. Field, H. Feng, R.F. Van der Wijngaart, R. Biswas // NASA Ames Research Center, CA. – 2004. – 17 p.

Макагон Дмитрий Викторович, научный сотрудник, отдел «Высокоскоростные коммуникационные сети», ОАО «Научно-исследовательский центр электронной вычислительной техники», makagond@mail.ru.

Корж Антон Александрович, начальник отдела, отдел «Высокоскоростные коммуникационные сети», ОАО «Научно-исследовательский центр электронной вычислительной техники».

Поступила в редакцию 7 апреля 2010 г.

УДК 517.977

ОЦЕНКА СКОРОСТИ СХОДИМОСТИ РАЗНОСТНЫХ АППРОКСИМАЦИЙ ПО ФУНКЦИОНАЛУ В ЗАДАЧЕ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ ДЛЯ ЛИНЕЙНОГО УРАВНЕНИЯ ШРЕДИНГЕРА

Н.М. Махмудов

ESTIMATION OF SPEED OF CONVERGENCE DIFFERENCE APPROXIMATIONS ON FUNCTIONAL IN OPTIMAL CONTROL PROBLEM FOR LINEAR SCHRODINGER EQUATION

N.M. Mahmudov

В этой работе устанавливается оценка погрешности аппроксимации и скорости сходимости разностных аппроксимаций по функционалу в задаче оптимального управления для линейного уравнения Шредингера с критерием качества Лионса.

Ключевые слова: разностный метод, уравнение Шредингера, критерии качества Лионса, сходимость по функционалу

In this work the estimation of an error of approximation and speed of convergence of difference approximations on functional in an optimal control problem for a linear Schrodinger equation with Lions' criterion of quality is established.

Keywords: a difference method, a Schrodinger equations, Lions' criterion of quality, a convergence on functional

1. Постановка задачи

В этой работе рассматривается задача оптимального управления для линейного уравнения Шредингера с критерием качества Лионса, и к этой задаче применяется разностный метод. При этом устанавливается оценка скорости сходимости разностных аппроксимаций по функционалу в рассматриваемой задаче оптимального управления. Подобные исследования ранее проведены в работах [1 – 5] для задач оптимального управления для уравнения Шредингера в другой постановке. Отметим, что ранее в работе [6], изучен вопрос сходимости разностного метода решения задачи оптимального управления для нелинейного уравнения Шредингера, где множество допустимых уравнений состоит из квадратично суммируемых функций. При этом, в отличие от настоящей работы, установлена оценка сходимости разностных аппроксимаций по функционалу.

Рассмотрим задачу о минимизации функционала:

$$J(v) = \int\limits_{\Omega} |\psi_1(x, t) - \psi_2(x, t)|^2 \, dxdt \tag{1}$$

на множестве $V \equiv \left\{ v = v(x) : v \in W_2^1(0, l),\ |v(x)| \le b_0,\ \left|\frac{dv(x)}{dx}\right| \le b_1,\ \overset{0}{\forall} x \in (0, l) \right\}$ при условиях

$$i\frac{\partial \psi_p}{\partial t} + a_0\frac{\partial^2 \psi_p}{\partial x^2} - a(x)\psi_p - v(x)\psi_p = f_p(x, t), \quad (x, t) \in \Omega, \tag{2}$$

$$\psi_p(x, 0) = \varphi_p(x), \quad p = 1, 2, \quad x \in (0, l), \tag{3}$$

$$\psi_1(0, t) = \psi_1(l, t) = 0, \quad t \in (0, T), \tag{4}$$

$$\frac{\partial \psi_2(0, t)}{\partial x} = \frac{\partial \psi_2(l, t)}{\partial x} = 0, \quad t \in (0, T), \tag{5}$$

где $i^2 = -1$, $a_0 > 0$, $b_0 > 0$, $b_1 > 0$ – заданные числа, $\Omega = (0, l) \times (0, T)$, $a(x)$ – ограниченная измеримая функция, удовлетворяющая условиям

$$0 < \mu_0 \le a(x) \le \mu_1, \quad \left|\frac{da(x)}{dx}\right| \le \mu_2, \quad \overset{0}{\forall} x \in (0, l), \tag{6}$$

а функции $\varphi_k(x)$, $f_k(x, t)$, $k = 1, 2$ удовлетворяют условиям:

$$\varphi_1 \in W_2^3(0, l), \quad \varphi_1(0) = \varphi_1(1) = \varphi_1''(0) = \varphi_1''(l) = 0,$$

$$\varphi_2 \in W_2^3(0, l), \quad \frac{d\varphi_2(0)}{dx} = \frac{d\varphi_2(l)}{dx} = 0, \tag{7}$$

$$f_1 \in \overset{0}{W}{}_2^{1.1}(\Omega), \quad f_2 \in W_2^{1.1}(\Omega), \quad \frac{\partial^2 f_2}{\partial x \partial t} \in L_2(\Omega), \quad p = 1, 2, \tag{8}$$

$\mu_i > 0$, $i = \overline{0, 2}$ – заданные числа.

Задачу об определении функций $\psi_k = \psi_k(x, t) \equiv \psi_k(x, t; v)$ из условий (2) – (5) при $v \in V$ назовем редуцированной задачей. Под решением этой задачи будем понимать функции $\psi_1 = \psi_1(x, t)$ и $\psi_2 = \psi_2(x, t)$, принадлежащие $B_1 \equiv C^0\left([0, T], W_2^2(0, l)\right) \cap C^1\left([0, T], L_2(0, l)\right)$ и $B_2 \equiv C^0\left([0, T], W_2^2(0, l)\right) \cap C^1\left([0, T], L_2(0, l)\right)$ соответственно и удовлетворяющие условиям (2) – (5) почти всюду. Из (2) – (5) ясно, что для функции $\psi_1 = \psi_1(x, t)$ задача (2) – (5) является первой краевой задачей, а для $\psi_2 = \psi_2(x, t)$ второй краевой задачей для уравнения Шредингера (2).

Исходя из результатов работ [2, 3] можем установить справедливость утверждения:

Теорема 1. *Редуцированная задача* (2) – (5) *имеет единственное решение* $\psi_1 \in B_1$, $\psi_2 \in B_2$, *и верны оценки:*

$$\|\psi_1(\cdot, t)\|_{\overset{0}{W}{}_{2(0,l)}^2} + \left\|\frac{\partial \psi_2(\cdot, t)}{\partial t}\right\|_{L_2(0,l)} \le M_1\left(\|\varphi_1\|_{\overset{0}{W}{}_{2(0,l)}^2} + \|f_1\|_{W_2^{0,1}(\Omega)}\right), \tag{9}$$

$$\|\psi_2(\cdot, t)\|_{W_2^2(0,l)} + \left\|\frac{\partial \psi_2(\cdot, t)}{\partial t}\right\|_{L_2(0,l)} \le M_2\left(\|\varphi_2\|_{\overset{0}{W}{}_{2(0,l)}^2} + \|f_2\|_{W_2^{0,1}(\Omega)}\right) \tag{10}$$

для $\forall t \in [0, T]$, *где* $M_1 > 0$ *и* $M_2 > 0$ – *постоянные не зависят от* φ_k, f_k, $k = 1, 2$.

Из условий (6) – (9) ясно, что функции $a(x)$, $\varphi_k(x)$, $f_k(x, t)$, $k = 1, 2$ являются более гладкими. Поэтому при $v \in V$ с помощью результатов работы [3] можем установить справедливость и следующих оценок:

$$\|\psi_1(\cdot, t)\|_{\overset{0}{W}{}_{2(0,l)}^2} + \left\|\frac{\partial \psi_1(\cdot, t)}{\partial t}\right\|_{L_2(0,l)} + \left\|\frac{\partial^2 \psi_1(\cdot, t)}{\partial x \partial t}\right\|_{L_2(0,l)} \le$$

$$\le M_3\left(\|\varphi_1\|_{\overset{0}{W}{}_{2(0,l)}^2} + \|f_1\|_{\overset{0}{W}{}_2^{1,1}(\Omega)} + \left\|\frac{\partial^2 f_1(\cdot, t)}{\partial x \partial t}\right\|_{L_2(0,l)}\right), \tag{11}$$

$$\|\psi_2(\cdot,\,t)\|_{W_2^3(0,l)}+\left\|\frac{\partial\psi_2(\cdot,\,t)}{\partial t}\right\|_{L_2(0,l)}+\left\|\frac{\partial^2\psi_2(\cdot,\,t)}{\partial x\partial t}\right\|_{L_2(0,l)}\le$$

$$\le M_4\left(\|\varphi_2\|_{W_2^3(0,l)}+\|f_2\|_{W_2^{1,\,1}(\Omega)}+\left\|\frac{\partial^2 f_2(\cdot,\,t)}{\partial x\partial t}\right\|_{L_2(\Omega)}\right) \quad (12)$$

для $\forall t\in[0,\,T]$, $\overset{0}{W}{}_2^3(0,\,l)\equiv W_2^3(0,\,l)\bigcap\overset{0}{W}{}_2^1(0,\,l)$, где $M_3>0$ и $M_4>0$ — постоянные не зависят от φ_k, f_k, $k=1,\,2$.

С использованием результатов работы [2] можем утверждать, что имеет место:

Теорема 2. *Задача оптимального управления* (1) – (5) *имеет хотя бы одно решение.*

Произведя дискретизацию при каждом натуральном $n\ge 1$ рассмотрим задачу о минимизации функции:

$$I_n\left([v]_n\right)=\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|\Phi_{jk}^1-\Phi_{jk}^2\right|^2 \quad (13)$$

на множестве $V_n\equiv\left\{[v]_n:\,[v]_n=(v_1,\,v_2,\,\ldots,\,v_{M-1}),\,v_j\le b_0,\,j=\overline{1,\,M-1},\,|\delta_{\bar x}v_j|\le b_1,\,j=\overline{2,\,M-1}\right\}$ при условиях:

$$i\delta_{\bar t}\Phi_{jk}^p+a_0\delta_{x\bar x}\Phi_{jk}^p-a^j\Phi_{jk}^p-v_j\Phi_{jk}^p=f_{jk}^p,\quad j=\overline{1,\,M-1},\quad k=\overline{1,\,N}, \quad (14)$$

$$\Phi_{j0}^p=\varphi_j^p,\quad j=\overline{0,\,M},\quad p=1,\,2, \quad (15)$$

$$\Phi_{0k}^1=\Phi_{Mk}^1=0,\quad k=\overline{1,\,N}, \quad (16)$$

$$\delta_{\bar x}\Phi_{1k}^2=\delta_{\bar x}\Phi_{Mk}^2=0,\quad k=\overline{1,\,N}, \quad (17)$$

где сеточные функции a^j, φ_j^p, f_{jk}^p, $p=1,\,2$ определены следующими формулами.

$$a_i^j=\frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}a(x)dx,\quad j=\overline{1,\,M-1}, \quad (18)$$

$$\varphi_j^p=\frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\varphi_p(x)dx,\quad j=\overline{1,\,M-1},\quad p=1,\,2,\quad \varphi_0^1=\varphi_M^1=0,\quad \delta_{\bar x}\varphi_1^2=\delta_{\bar x}\varphi_M^2=0, \quad (19)$$

$$f_{jk}^p=\frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}f_p(x,\,t)dxdt,\quad j=\overline{1,\,M-1},\quad k=\overline{1,\,N},\quad p=1,\,2. \quad (20)$$

Эти функции определены на сетке

$$\left\{(x_j,\,t_k)_n\right\},\quad n=0,\,1,\,2,\ldots,\quad x_j=jh-\frac{h}{2},\quad t_k=k\tau,j=\overline{1,\,M_n-1},\quad k=\overline{1,\,N_n},$$

$$h=h_n=\frac{l}{M_n-1},\quad \tau=\tau_n=\frac{T}{N_n},\quad \delta_{\bar t}\Phi_{jk}^p=\frac{\Phi_{jk}^p-\Phi_{j\,k-1}^p}{\tau},\quad \delta_{\bar x}\Phi_{jk}^p=\frac{\Phi_{jk}^p-\Phi_{j-1\,k}^p}{h},$$

$$\delta_x\Phi_{jk}^p=\frac{\Phi_{j+1\,k}^p-\Phi_{j\,k}^p}{h},\quad \delta_{x\bar x}\Phi_{jk}^p=\frac{\Phi_{j+1\,k}^p-2\Phi_{jk}^p+\Phi_{j-1\,k}^p}{h^2},\quad p=1,\,2.$$

Здесь верхний индекс не является степенью. Обозначим $M=M_n$, $N=N_n$.

Следует отметить, что задача об определении сеточной функции $\Phi^p_{jk} = \Phi^p_{jk}([v]_n)$ при каждом $[v_n] \in V_n$ из условий (14) – (17) является разностной схемой. Подобная разностная схема для нелинейного уравнения Шредингера изучена в работе [2]. Из этой работы в частном случае следует:

Теорема 3. *Для решения разностной схемы* (14) – (17) *при* $[v]_n \in V_n$ *верна оценка:*

$$h\sum_{j=1}^{M-1}\left|\Phi^p_{jm}\right|^2 \le M_5\left(h\sum_{j=1}^{M-1}\left|\varphi^p_j\right|^2 + \tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|\varphi^p_{jk}\right|^2\right), \quad p=1,\,2 \tag{21}$$

для любого $m \in \{1, 2, \ldots, N\}$.

2. Оценка погрешности разностной схемы

Основной целью настоящей работы является установление оценки о скорости сходимости разностных аппроксимаций по функционалу в рассматриваемой задаче. Поэтому сначала оценим погрешность аппроксимации разностной схемы (14) – (17). С этой целью рассмотрим следующие усреднения решения редуцированной задачи (2) – (5), при $v \in V$:

$$[\psi_p(x,t;v)]_n = \left\{\psi^p_{jk}\right\}, \quad \psi^p_{jk} = \frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} \psi_p(x,t_k)\,dx, \quad j=\overline{1,\,M-1}, \quad k=\overline{1,\,N},$$

$$\psi^p_{j0} = \varphi^p_j, \quad j=\overline{0,\,M}, \quad p=1,\,2,$$

$$\psi^1_{0k} = \psi^1_{Mk} = 0, \quad \psi^2_{0k} = \psi^2_{1k}, \quad \psi^2_{Mk} = \psi^2_{M-1k}, \quad k=\overline{1,\,N}. \tag{22}$$

Определим оператор Q_n на множестве V формулой:

$$Q_n(v) = \{w_j\}, \quad w_j = \frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} v(x)dx, \quad j=\overline{1,\,M-1}. \tag{23}$$

Обозначим $[z^p]_n = \left\{z^p_{jk}\right\} = \left\{\Phi^p_{jk}\right\} - \left\{\psi^p_{jk}\right\}$. Ясно, что $\left\{z^p_{jk}\right\}$ будет решением следующей системы:

$$i\delta_{\bar t} z^p_{jk} + a_0\delta_{x\bar x}z^p_{jk} - a_j z^p_{jk} - v_j z^p_{jk} = F^p_{jk}, \quad j=\overline{1,\,M-1}, \quad k=\overline{1,\,N}, \tag{24}$$

$$z^p_{j0} = 0, \quad j=\overline{0,\,M}, \quad p=1,\,2, \tag{25}$$

$$z^1_{0k} = z^1_{Mk} = 0, \quad k=\overline{1,\,N}, \tag{26}$$

$$\delta_{\bar x} z_{1k} = \delta_{\bar x} z_{Mk} = 0, \quad k=\overline{1,\,N}, \tag{27}$$

где

$$F^p_{jk} = \frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left(a_0\frac{\partial^2\psi_p}{\partial x^2} - a(x)\psi_p - v(x)\psi_p\right)dxdt + v_j\psi^p_{jk} - a_0\delta_{x\bar x}\psi^p_{jk} + a^j\psi^p_{jk},$$

$$j=\overline{1,\,M-1}, \quad k=\overline{1,\,N}, \quad p=1,\,2. \tag{28}$$

Теорема 4. *Пусть выполнено условие согласования:* $c_0 \le \frac{\tau}{h^2} \le c_1$, *где* c_0, $c_1 > 0$ – *постоянные, независящие от* h *и* τ. *Тогда верны оценки:*

$$h\sum_{j=1}^{M-1}\left|z_{jm}^{p}\right|^2 \le M_6\left(\tau + h + \|Q_n(v) - [v]_n\|^2\right), \quad p = 1,\ 2, \tag{29}$$

для $\forall m \in \{1,\ 2, \ldots, N\}$, *где*

$$\|Q_n(v) - [v]_n\| = \left(h\sum_{j=1}^{M-1}|w_j - v_j|^2\right)^{\frac{1}{2}}. \tag{30}$$

Доказательство. Аналогично получению оценки (21) можно доказать справедливость оценки:

$$h\sum_{j=1}^{M-1}\left|z_{jm}^{p}\right|^2 \le M_7\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|F_{jk}^{p}\right|, \quad p = 1,\ 2, \tag{31}$$

$\forall m \in \{1,\ 2, \ldots, N\}$.

Используя формулу (28), сеточную функцию F_{jk}^{p} можем представить в виде

$$F_{jk}^{p} = F_{jk}^{p1} + F_{jk}^{p2} + F_{jk}^{p3}, \quad p = 1,\ 2, \tag{32}$$

где F_{jk}^{p1}, F_{jk}^{p3} определены формулами,

$$F_{jk}^{p1} = \frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\frac{a_0\partial^2\psi_p}{\partial x^2}dxdt - a_0\delta_{x\bar{x}}\psi_{jk}^{p}, \tag{33}$$

$$F_{jk}^{p3} = \frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}a(x)\psi_p(x,\ t)\,dxdt - a_j\psi_{jk}^{p}, \quad j = \overline{1,\ M-1}, \quad k = \overline{1,\ N}, \quad p = 1,\ 2, \tag{34}$$

а F_{jk}^{p2} определяется формулой

$$F_{jk}^{p2} = \frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}v(t)\psi_p(x,\ t)\,dxdt - v_k\psi_{jk}^{p}, \quad j = \overline{1,\ M-1}, \quad k = \overline{1,\ N}, \quad p = 1,\ 2. \tag{35}$$

Используя формулу (34) и (18), можем установить справедливость неравенств:

$$\left|F_{jk}^{p3}\right|^2 \le \frac{2\mu_1^2\tau}{h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_p(x,\ t)}{\partial t}\right|^2 dxdt + \frac{2\mu_1^2 h}{\tau}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_p(x,\ t)}{\partial x}\right|^2 dxdt,$$

$$j = \overline{1,\ M-1}, \quad k = \overline{1,\ N}, \quad p = 1,\ 2. \tag{36}$$

С помощью леммы Брэмбла – Гильберта [7] можем получить следующие неравенства:

$$\left|F_{jk}^{11}\right| \le M_8 a_0 h^{\frac{1}{2}} \tau^{-\frac{1}{2}} \left(\int_{t_{k-1}}^{t_k} \int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} \left| \frac{\partial^3 \psi_1}{\partial x^3} \right|^2 dx dt \right)^{\frac{1}{2}} +$$

$$+ a_0 \tau^{\frac{1}{2}} h^{-\frac{1}{2}} \left[\left(\int_{t_{k-1}}^{t_k} \int_{x_{j+1}-\frac{h}{2}}^{x_{j+1}+\frac{h}{2}} \left| \frac{\partial^2 \psi_1(x,\, t)}{\partial x \partial t} - \frac{\partial^2 \psi_1(x-h,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} + \right.$$

$$\left. + \left(\int_{t_{k-1}}^{t_k} \int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} \left| \frac{\partial^2 \psi_1(x,\, t)}{\partial x \partial t} - \frac{\partial^2 \psi_1(x-h,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} \right], \quad (37)$$

$$\left|F_{jk}^{21}\right| \le M_9 a_0 h^{\frac{1}{2}} \tau^{-\frac{1}{2}} \left(\int_{t_{k-1}}^{t_k} \int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} \left| \frac{\partial^3 \psi_2}{\partial x^3} \right|^2 dx dt \right)^{\frac{1}{2}} +$$

$$+ a_0 \tau^{\frac{1}{2}} h^{-\frac{1}{2}} \left[\left(\int_{t_{k-1}}^{t_k} \int_{x_{j+1}-\frac{h}{(}2)}^{x_{j+1}+\frac{h}{2}} \left| \frac{\partial^2 \psi_2(x,\, t)}{\partial x \partial t} - \frac{\partial^2 \psi_2(x-h,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} + \right.$$

$$\left. + \left(\int_{t_{k-1}}^{t_k} \int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}} \left| \frac{\partial^2 \psi_2(x,\, t)}{\partial x \partial t} - \frac{\partial^2 \psi_2(x-h,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} \right], \quad j = \overline{2,\, M-1}, \quad k = \overline{1,\, N}. \quad (38)$$

Далее с помощью формулы для сеточной функции F_{jk}^{p1}, $j = 1,\, M-1$, $k = \overline{1,\, N}$, $p = 1,\, 2$ имеем:

$$\left|F_{1k}^{11}\right| \le 2 a_0 \tau^{\frac{1}{2}} h^{-\frac{3}{2}} \left[\left(\int_{t_{k-1}}^{t_k} \int_{x_2-\frac{h}{2}}^{x_2+\frac{h}{2}} \left| \frac{\partial^2 \psi_1(x,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} + \left(\int_{t_{k-1}}^{t_k} \int_{x_1-\frac{h}{2}}^{x_1+\frac{h}{2}} \left| \frac{\partial^2 \psi_1(x,\, t)}{\partial x \partial t} \right|^2 dx dt \right)^{\frac{1}{2}} + \right.$$

$$+ M_{19} \tau^{-\frac{1}{2}} \left(\int_{t_{k-1}}^{t_k} \left[\left\| \frac{\partial^2 \psi_1(\cdot,\, t)}{\partial x^2} \right\|^2_{L_2(0,l)} + \left\| \frac{\partial^3 \psi_1(\cdot,\, t)}{\partial x^3} \right\|^2_{L_2(0,l)} \right] dt \right)^{\frac{1}{2}} +$$

$$\left. + M_{11} \tau^{-\frac{1}{2}} h^{\frac{1}{2}} \left(\int_{t_{k-1}}^{t_k} \int_{x_1-\frac{h}{2}}^{x_1+\frac{h}{2}} \left| \frac{\partial^3 \psi_1(x,\, t)}{\partial x^3} \right|^2 dx dt \right)^{\frac{1}{2}} \right], \quad k = \overline{1,\, N}, \quad (39)$$

$$\left|F^{11}_{M-1k}\right| \le 2a_0\tau^{\frac{1}{2}}h^{-\frac{1}{2}}\left[\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-1}-\frac{h}{2}}^{x_{M-1}+\frac{h}{2}}\left|\frac{\partial^2\psi_1(x,\,t)}{\partial x\partial t}\right|^2 dxdt\right)^{\frac{1}{2}} + \left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-2}-\frac{h}{2}}^{x_{M-2}+\frac{h}{2}}\left|\frac{\partial^2\psi_1(x,\,t)}{\partial x\partial t}\right|^2_{L_2(0,l)} dxdt\right)^{\frac{1}{2}} + M_{12}\tau^{-\frac{1}{2}}\left(\int\limits_{t_{k-1}}^{t_k}\left[\left\|\frac{\partial^2\psi_1(\cdot,\,t)}{\partial x^2}\right\|^2_{L_2(0,l)} + \left\|\frac{\partial^3\psi_1(\cdot,\,t)}{\partial x^3}\right\|^2_{L_2(0,l)}\right]dt\right)^{\frac{1}{2}} + M_{13}\tau^{-\frac{1}{2}}h^{\frac{1}{2}}\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-2}-\frac{h}{2}}^{x_{M-2}+\frac{h}{2}}\left|\frac{\partial^3\psi_1(x,\,t)}{\partial x^3}\right|^2 dxdt\right)^{\frac{1}{2}}, \quad k=\overline{1,\,N}, \quad (40)$$

$$\left|F^{21}_{1k}\right| \le 2a_0\tau^{\frac{1}{2}}h^{-\frac{3}{2}}\left[\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_2-\frac{h}{2}}^{x_2+\frac{h}{2}}\left|\frac{\partial^2\psi_2(x,\,t)}{\partial x\partial t}\right|^2 dxdt\right)^{\frac{1}{2}} + \left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_1-\frac{h}{2}}^{x_1+\frac{h}{2}}\left|\frac{\partial^2\psi_2(x,\,t)}{\partial x\partial t}\right|^2 dxdt\right)^{\frac{1}{2}}\right] + M_{14}\tau^{-\frac{1}{2}}h^{\frac{1}{2}}\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_1-\frac{h}{2}}^{x_1+\frac{h}{2}}\left|\frac{\partial^3\psi_2(x,\,t)}{\partial x^3}\right|^2 dxdt\right)^{\frac{1}{2}}, \quad k=\overline{1,\,N}, \quad (41)$$

$$\left|F^{21}_{M-1k}\right| \le 2a_0\tau^{\frac{1}{2}}h^{-\frac{3}{2}}\left[\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-1}-\frac{h}{2}}^{x_{M-1}+\frac{h}{2}}\left|\frac{\partial^2\psi_2(x,\,t)}{\partial x\partial t}\right|^2 dxdt\right)^{\frac{1}{2}} + \left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-2}-\frac{h}{2}}^{x_{M-2}+\frac{h}{2}}\left|\frac{\partial^2\psi_2(x,\,t)}{\partial x\partial t}\right|^2 dxdt\right)^{\frac{1}{2}}\right] + M_{15}\tau^{-\frac{1}{2}}h^{\frac{1}{2}}\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_{M-1}-\frac{h}{2}}^{x_{M-1}+\frac{h}{2}}\left|\frac{\partial^3\psi_1(x,\,t)}{\partial x^3}\right|^2 dxdt\right)^{\frac{1}{2}}, \quad k=\overline{1,\,N}. \quad (42)$$

С помощью формулы (35) для F_{jk}^{p2}, $j=\overline{1,\,M-1}$, $k=\overline{1,\,N}$, $p=1,\,2$ получим:

$$F_{jk}^{p2}=\frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left(v(t)\psi_p(x,\,t)-v_k\psi_{jk}^p\right)dxdt=\frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\psi_{jk}^p\left(v(t)-v_k\right)dxdt+$$

$$+\frac{1}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}v(t)\left(\psi_p(x,\,t)-\psi_{jk}^p\right)dxdt. \quad (43)$$

Учитывая, что $[v]_n\in V_n$, $v\in V$, имеем:

$$\left|F_{jk}^{p2}\right|\le\left|\psi_{jk}^p\right|\left|w_k-v_k\right|+\frac{b_0}{\tau h}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\psi_p(x,\,t)-\psi_{jk}^p\right|dxdt,$$

$$j=\overline{1,\,M-1},\quad k=\overline{1,\,N},\quad p=1,2. \quad (44)$$

Теперь рассмотрим разность $\psi_p\left(x,\,t\right)-\psi_{jk}^p$, $p=1,\,2$. С помощью формулы (22), (23) можем написать следующие равенства:

$$\frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left(\psi_p(x,\,t)-\psi_p\left(\xi,\,t_k\right)\right)d\xi=\frac{1}{h}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left[\int\limits_t^{t_k}\frac{\partial\psi_p\left(\xi,\vartheta\right)}{\partial\vartheta}d\vartheta+\int\limits_x^{\xi}\frac{\partial\psi_p\left(\eta,\,t\right)}{\partial\eta}d\eta\right]d\xi,\ p=1,2.$$

Тогда из (44) получим

$$\left|F_{jk}^{p2}\right|\le\left|\psi_{jk}^p\right|\left|w_k-v_k\right|+\frac{b_0}{\tau h^2}\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left[\int\limits_t^{t_k}\left|\frac{\partial\psi_p\left(\xi,\vartheta\right)}{\partial\vartheta}\right|d\vartheta+\right.$$

$$\left.+\int\limits_x^{\xi}\left|\frac{\partial\psi_p\left(\eta,\vartheta\right)}{\partial\eta}\right|d\eta\right]d\xi dxdt\le\left|\psi_{jk}^p\right|\left|w_k-v_k\right|+\frac{b_0\sqrt{\tau}}{\sqrt{h}}\left(\int\limits_{t_{k+1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_p(x,\,t)}{\partial t}\right|^2dxdt\right)^{\frac{1}{2}}+$$

$$+\frac{b_0\sqrt{h}}{\sqrt{\tau}}\left(\int\limits_{t_{k+1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_p(x,\,t)}{\partial x}\right|^2dxdt\right)^{\frac{1}{2}}. \quad (45)$$

В силу формулы для ψ_{jk}^p имеем:

$$\left|\psi_{jk}^p\right|\le\frac{\sqrt{\tau}}{\sqrt{h}}\left(\int\limits_{t_{k-1}}^{t_k}\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_p(x,\,t)}{\partial t}\right|^2dxdt\right)^{\frac{1}{2}}+$$

$$+\frac{1}{\sqrt{h}}\left(\int\limits_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\psi_p(x,\,t)\right|^2dx\right)^{\frac{1}{2}},\quad p=1,2,\quad j=\overline{1,\,M-1}. \quad (46)$$

Используя эти неравенства, имеем:

$$h\sum_{j=1}^{M-1}\left|\psi_{jk}^{p}\right|^{2}\le 2\left(\left\|\frac{\partial\psi_p}{\partial t}\right\|^{2}_{L_\infty(0,T;L_2(0,l))}+\|\psi_p\|^{2}_{L_\infty(0,T;L_2(0,l))}\right),\quad p=1,2,\quad k=\overline{1,\,N}. \tag{47}$$

Отсюда в силу оценки (9), (10) имеем:

$$h\sum_{j=1}^{M-1}\left|\psi_{jk}^{p}\right|^{2}\le M_{16},\quad p=1,2,\quad k=\overline{1,\,N}. \tag{48}$$

Учитывая (48) в (45) и в силу оценок (11), (12) получим

$$\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|F_{jk}^{p2}\right|^{2}\le 3M_{17}\tau\sum_{k=1}^{N}|w_k-v_k|^2+3b_0^2\left(\tau^2\left\|\frac{\partial\psi}{\partial t}\right\|^{2}_{L_2(\Omega)}+h^2\left\|\frac{\partial\psi}{\partial x}\right\|^{2}_{L_2(\Omega)}\right)\le$$
$$\le M_{18}\left(\tau^2-h^2+\|Q_n(v)-[v]_n\|^2\right),\quad p=1,2. \tag{49}$$

С помощью условия согласования шагов сетки и оценок (11), (12) из неравенств (36) – (42) получим справедливость неравенств:

$$\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|F_{jk}^{p1}\right|^{2}\le M_{19}\left(\tau+h\right),\quad p=1,2, \tag{50}$$

$$\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|F_{jk}^{p3}\right|^{2}\le M_{20}\left(\tau^2+h^2\right),\quad p=1,2. \tag{51}$$

Таким образом, используя неравенства (49) – (51) и формулы (32), получим справедливость неравенства:

$$\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|F_{jk}^{p}\right|^{2}\le M_{21}\left(\tau-h+\|Q_n(v)-[v]\|^2\right),\quad p=1,2. \tag{52}$$

Учитывая (52) в неравенстве (32) получим справедливость утверждения теоремы. □

3. Оценка скорости сходимости разностных аппроксимаций по функционалу

Теперь установим оценку о скорости разностных аппроксимаций по функционалу. С этой целью сначала оценим разность исходного функционала и дискретной функции.

Теорема 5. *Пусть выполнены условия теоремы 4. Тогда для любого* $v\in V$ *и* $[v]_n\in V$ *имеет место оценка*

$$|J(v)-I_n([v]_n)|\le M_{22}\left(\sqrt{\tau}+\sqrt{h}+\|Q_n(v)-[v]_n\|\right),\quad n=1,\,2,\ldots, \tag{53}$$

Доказательство. Используя формулу (1) и (13), имеем:

$$|J(v) - I_n([v]_n)| \le M_{23}\left[\left(\sum_{k=1}^{N}\sum_{j=1}^{M-1}\int_{t_{k-1}}^{t_k}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\psi_1(x,\,t)-\Phi_{jk}^1\right|^2 dxdt\right)^{\frac{1}{2}} + \right.$$
$$\left. + \left(\sum_{k=1}^{N}\sum_{j=1}^{M-1}\int_{t_{k-1}}^{t_k}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\psi_2(x,\,t)-\Phi_{jk}^2\right|^2 dxdt\right)^{\frac{1}{2}}\right] = M_{24}\left(J_1+J_2\right). \quad (54)$$

С помощью формулы для J_1 имеем

$$(J_1)^2 \le 2\sum_{k=1}^{N}\sum_{j=1}^{M-1}\int_{t_{k-1}}^{t_k}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\psi_1(x,\,t)-\psi_{jk}^1\right|^2 dxdt + 2\tau h\sum_{k=1}^{N}\sum_{j=1}^{M-1}\left|\psi_{jk}^1-\Phi_{jk}^1\right|^2 = J_{11}+J_{12}. \quad (55)$$

В силу утверждения теоремы 4 имеем:

$$J_{12} \le M_{25}\left(\tau + h + \|Q_n(v)-[v]_n\|^2\right). \quad (56)$$

Рассмотрим разность $\psi_{jk}^1 - \psi_1(x,\,t)$. Тогда с учетом формулы (25) имеем

$$\frac{1}{h}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\psi_1(\xi,\,t_k)\,d\xi - \psi_1(x,\,t) = \frac{1}{h}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left(\psi_1(\xi,\,t_k)-\psi_1(x,\,t)\right)d\xi =$$
$$= \frac{1}{h}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left[\int_{t}^{t_k}\frac{\partial\psi_1(\xi,\,\theta)}{\partial\theta}d\theta + \int_{x}^{\xi}\frac{\partial\psi_1(\eta,\,t)}{\partial\eta}d\eta\right]d\xi. \quad (57)$$

Если это учесть в формуле для слагаемого J_{11}, то имеем

$$J_{11} \le 2\sum_{k=1}^{N}\sum_{j=1}^{M-1}\int_{t_{k-1}}^{t_k}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left(\frac{1}{h}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\int_{t_{k-1}}^{t_k}\left|\frac{\partial\psi_1(\xi,\,\theta)}{\partial\theta}\right|d\theta d\xi + \right.$$
$$\left. + \frac{1}{h}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\int_{x_j-\frac{h}{2}}^{x_j+\frac{h}{2}}\left|\frac{\partial\psi_1(\eta,\,t)}{\partial\eta}\right|d\eta d\xi\right)^2 dxdt \le 4h^2\left\|\frac{\partial\psi_1}{\partial x}\right\|_{L_2(\Omega)}^2 + 4\tau^2\left\|\frac{\partial\psi_1}{\partial t}\right\|_{L_2(\Omega)}^2. \quad (58)$$

В силу оценки (11) отсюда получим

$$J_{11} \le M_{26}\left(\tau^2+h^2\right). \quad (59)$$

Тогда, складывая (56) и (57), получим

$$(J_1)^2 \le M_{27}\left(\tau^2+h^2+\tau+h+\|Q_n(v)-[v]\|^2\right). \quad (60)$$

Аналогично получим следующее:

$$(J_2)^2 \le M_{28}\left(\tau^2 + h^2 + \tau + h + \|Q_n(v) - [v]\|^2\right). \tag{61}$$

Отсюда и из (54) получим утверждение теоремы. □

Теперь приведем две вспомогательные леммы.

Лемма 1. *Пусть выполнены условия теоремы 5. Пусть, кроме того, оператор Q_n определяется формулой (23). Тогда $Q_n(v) \in V_n$ и имеет место оценка*

$$|J(v) - I_n(Q_n(v))| \le M_{29}\left(\sqrt{\tau} + \sqrt{h}\right), \quad n = 1, 2, \ldots. \tag{62}$$

Доказательство этой леммы проводится с использованием утверждением теоремы 5. Пусть оператор P_n определяется формулой

$$P_n([v]_n) = \tilde{v}(x), \tag{63}$$

где

$$\tilde{v}(x) = \begin{cases} v_j + \delta_{\bar{x}} v_j \left(x - x_j - \dfrac{h}{2}\right), & x_j - \dfrac{h}{2} \le x \le x_j + \dfrac{h}{2}, j = \overline{2,\, M-1} \\ v_1, x_1 - h\dfrac{h}{2} \le x \le x_1 + \dfrac{h}{2}. \end{cases} \tag{64}$$

Лемма 2. *Пусть выполнены условия теоремы 5. Пусть, кроме того, оператор P_n определяется формулами* (63), (64). *Тогда $P_n([v]_n) \in V$, и имеет место оценка*

$$|J(P_n([v]_n)) - I_n([v]_n)| \le M_{30}\left(\sqrt{\tau} + \sqrt{h}\right), \quad n = 1, 2, \ldots. \tag{65}$$

Доказательство. С помощью формул (63), (64) и структуры множества V нетрудно установить справедливость соотношения:

$$P_n([v]_n) \in V.$$

Поэтому в теореме 6, вместо v выбирая $\tilde{v}(x) = P_n([v]_n)$ и проведя доказательство, получим справедливость оценки

$$|J(P_n([v]_n)) - I_n([v]_n)| \le M_{31}\left(\sqrt{\tau} + \sqrt{h} + \|Q_n(\tilde{v}) - [v]_n\|\right), \quad n = 1, 2, \ldots. \tag{66}$$

Ясно, что

$$\|Q_n(\tilde{v}) - [v]_n\|^2 = h\sum_{j=1}^{M-1} |\tilde{w}_j - v_j|^2 = h\sum_{j=1}^{M-1} \left| \frac{1}{h} \int\limits_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \tilde{v}(x)dx - v_j \right|^2 =$$

$$= h\sum_{j=2}^{M-1} \left| \frac{\delta_{\bar{x}} v_j}{h} \int\limits_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \left(x - x_j - \frac{h}{2}\right) dx \right|^2 = h\sum_{j=2}^{M-1} \left| \frac{\delta_{\bar{x}} v_j}{2h} \left(x - x_j - \frac{h}{2}\right)^2 \Big|_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \right|^2 \le \frac{l b_1^2 h^2}{4}.$$

Отсюда получим

$$\|Q_n(\tilde{v}) - [v]_n\| \le \frac{\sqrt{l} b_1}{2} h.$$

Учитывая данную оценку в (66), получим утверждение леммы. □

Теперь приведем теорему о скорости сходимости разностных аппроксимаций по функционалу.

Теорема 6. *Пусть выполнены условия леммы 1 и 2. Пусть, кроме того,* $v^* \in V$ *и* $[v]_n^* \in V_n$ *являются решениями задач (1)–(5) и (13)–(17) соответственно, то есть*

$$J_* = \inf_{v \in V} J(v) = J(v^*), \quad I_{n*} = \inf_{[v]_n \in V_n} I_n([v]_n) = I_n([v]_n^*).$$

Тогда последовательность разностных задач (13)–(17) аппроксимирует задачу (1)–(5), то есть

$$\lim_{n \to \infty} I_{n*} = J_* \tag{67}$$

и справедлива оценка о скорости сходимости:

$$|I_{n*} - J_*| \le M_{32} \left(\sqrt{\tau} + \sqrt{h} \right), \quad n = 1, 2, \ldots. \tag{68}$$

Доказательство этой теоремы проводится с использованием леммы 1 и 2 и методики работы [8].

Литература

1. Потапов, М.М. Аппроксимация и регуляризация задачи оптимального управления типа Шредингера / М.М. Потапов, А.В. Разгулин, Т.Ю. Шамеева // Вестн. Московск. ун-та. Вычислительная математика и кибернетика. – 1987. – Сер. 15, №1. – С. 8 – 13.
2. Искендеров, А.Д. Оптимальное управление кванто-механической системой с критерием качества Лионса / А.Д. Искендеров, Н.М. Махмудов // Изв. АНА, Сер. физ.-тех. матем. наук. – 1995. – Т. XVI, №5 – 6. – С. 30 – 35.
3. Ягубов, Г.Я. Оптимальное управление коэффициентом квазилинейного уравнения Шредингера: дис...д-ра...наук / Г.Я. Ягубов. – Киев, 1994. – 318 с.
4. Ягубов, Г.Я. Разностный метод решения задачи оптимального управления коэффициентом квазилинейного уравнения Шредингера с интегральным критерием качества по границе области / Г.Я. Ягубов // Проблемы математического моделирования и оптимального управления. – Баку, 2001. – С. 37 – 48.
5. Ягубов, Г.Я. Сходимость разностного метода решения задачи оптимального управления для нелинейного уравнения Шредингера с интегральным критерием качества / Г.Я. Ягубов // Вестн. Сумгаит. гос. ун-та. – 2001. – №1. – С. 37 – 42.
6. Махмудов, Н.М. Разностный метод решения задачи оптимального управления для уравнения Шредингера с критерием качества Лионса / Н.М. Махмудов // Изв. Челяб. науч. центра. – 2009. – №3(45). – С. 1 – 6.
7. Самарский, А.А. Разностные схемы для дифференциальных уравнений с обобщенными решениями / А.А. Самарский, Р.Д. Лазаров, В.Л. Макаров. – М.: Высш. шк., 1987. – 296 с.
8. Васильев, В.П. Методы решения экстремальных задач / В.П. Васильев. – М.: Наука, 1981. – 400 с.

Махмудов Нурмали Мехрали оглы, кандидат физико-математических наук, доцент, кафедра «Информатики», Нахичеванский государственный университет (Азербайджан), nuralimaxmudov@rambler.ru.

Поступила в редакцию 24 июня 2009 г.

УДК 004.45

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СУПЕРКОМПЬЮТЕРОВ СЕМЕЙСТВА «СКИФ АВРОРА» НА ИНДУСТРИАЛЬНЫХ ЗАДАЧАХ

А.А. Московский, М.П. Перминов, Л.Б. Соколинский, В.В. Черепенников, А.В. Шамакина

RESEARCH PERFORMANCE FAMILY SUPERCOMPUTERS «SKIF AURORA» ON INDUSTRIAL PROBLEMS

A.A. Moskovsky, M.P. Perminov, L.B. Sokolinsky, V.V. Cherepennikov, A.V. Shamakina

В работе проведено сравнительное исследование производительности ряда приложений численного моделирования на суперЭВМ «СКИФ»: «СКИФ Аврора» и «СКИФ Урал», установленных в Южно-Уральском государственном университете (Челябинск), а также на кластере «Endeavor» компании Intel (DuPont, США). В качестве приложений были выбраны задача газовой динамики, задачи конечно-элементного анализа и задача конденсации наночастиц. В результате анализа результатов показано, что в большинстве случаев суперЭВМ «СКИФ Аврора» демонстрирует наилучшую производительность, в особенности в задачах, требовательных к пропускной способности подсистемы памяти.

Ключевые слова: суперкомпьютер, производительность, индустриальные задачи, масштабируемость приложений

In a comparative study of the performance of applications of numerical simulation on the supercomputer SKIF: «SKIF Aurora» and «SKIF Ural», set in the South Ural State University (Chelyabinsk), as well as on a cluster of «Endeavor» Company Intel (DuPont, USA). As applications, we chose the problem of gas dynamics, the problem of finite-element analysis and the problem of condensation of nanoparticles. The analysis results show that in most cases, the supercomputer «SKIF Aurora» demonstrates the best performance, especially in tasks demanding memory bandwidth.

Keywords: supercomputer, performance, industrial problems, scalability of applications

Введение

Сравнительное исследование производительности и масштабируемости различных приложений крайне важно для суперкомпьютерных центров, как с точки зрения оптимизации нагрузки на существующие машины, так и с точки зрения политики закупки новых платформ. В Южно-Уральском государственном университете установлены две машины семейства «СКИФ»: «СКИФ Урал» (2008 г.) и «СКИФ Аврора» (2010 г.) В качестве задач были выбраны не стандартные наборы тестов производительности, а несколько приложений пользователей суперкомпьютерного центра ЮУрГУ. Такой выбор позволяет получить более адекватную оценку возможностей вычислительных систем. Дополнительно, при помощи специализированных инструментальных средств, нами проведен анализ особенностей приложений, обуславливающих характеристики производительности приложений.

1. Архитектура суперкомпьютера «СКИФ Аврора»

Платформа «СКИФ Аврора» изначально разрабатывалась как основа для высокопроизводительных систем большого масштаба. Целый ряд технических решений, использованных в «СКИФ» ряда 4, сдвигает баланс свойств в сторону специализации для применения именно в суперкомпьютерах. Подробно характеристики решения рассмотрены в работе [1]. Установка «СКИФ Аврора» в Южно-Уральском государственном университете является первым пилотным проектом по развертыванию системы такого класса. В данном разделе кратко описываются особенности «СКИФ Аврора» с учетом ее конфигурации в ЮУрГУ.

Проект системы, включая ресурсы систем охлаждения и бесперебойного электропитания, позволяет установить до 8 вычислительных шасси «СКИФ Аврора». Каждое шасси включает 64 двухпроцессорных узла с четырехядерными процессорами Intel Xeon X5570 (Nehalem), с рабочей частотой 2,93 ГГц. Таким образом, в рамках одного монтажного шкафа удалось собрать 2048 процессорных ядер. Максимальная теоретическая производительность системы, состоящей из одного шкафа, составляет 24 ТФлопс.

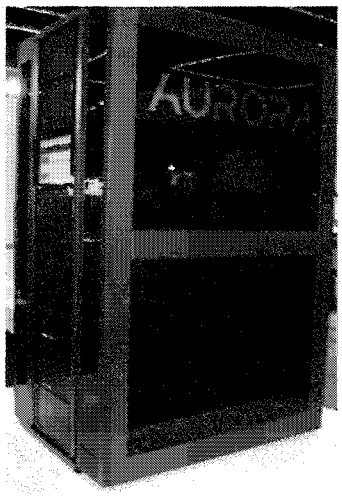


Рис. 1. Шкаф вычислителя «СКИФ Аврора»

1.1. Вычислительная часть

Высокая плотность упаковки процессоров в вычислителе диктует необходимость использования жидкостной системы охлаждения. Вычислительные узлы выполнены в виде печатных плат, с интегрированными на материнской плате коммуникационными, сервисными микросхемами, модулями памяти. Тестирование плат проводится на заводе-изготовителе, что уменьшает число отказов компонент при инсталляции и первичной настройке системы. Каждый узел-плата накрыт плотно прилегающей пластиной охлаждения. Пластины охлаждения оснащены быстроразъемными муфтами, что позволяет демонтировать отдельный вычислительный узел без демонтажа системы охлаждения корзины (шасси) в целом.

Каждый узел оснащен твердотельным накопителем объемом 80 Гбайт. Использование твердотельных накопителей также направлено на повышение надежности вычислителя - отказы шпиндельных дисковых накопителей составляют львиную долю причин отказов узлов в кластерных установках и вычислительных фермах.

1.2. Коммуникационные сети

Ключевым компонентом любого суперкомпьютера является его коммуникационная среда. Узлы «СКИФ Аврора» обладают суммарным каналом пропускания до 100 Гбит/с, учитывая как системную и вспомогательную коммуникационные сети. Если во вспомогательной сети используются стандартные решения Infiniband QDR, то системная сеть является оригинальной разработкой.

Системная сеть имеет топологию трехмерного тора, маршрутизаторы сети реализованы на уровне адаптеров. Суммарная пропускная способность сети в пересчете на один узел составляет 60 Гбит/с. Сеть позволяет обойтись без использования дополнительного оборудования (маршрутизаторов) и задействовать при монтаже кабели одинаковой длины, вне зависимости от размера системы. Соединения на уровне половины шасси (корзины) выполнены на соединительной плате. Трехмерная организация сети позволяет легче распределить задачи между узлами кластера при моделировании объектов реального мира (трехмерных) и распараллеливании методом декомпозиции области. Для системной сети создана реализация MPI на основе MPICH2, удовлетворяющая спецификации версии MPI 2.0.

Вспомогательная сеть – сеть Infiniband QDR (40 Гбит/с) с полной бисекционной пропускной способностью. Адаптеры сети интегрированы на платах-узлах. Маршрутизаторы первого уровня интегрированы на уровне корзин (шасси) на так называемых «корневых платах». Соединения между узлами и маршрутизатором первого уровня выполнены на соединительной плате (backplane), что существенно уменьшает количество кабелей Infiniband, подключаемых вручную при установке системы. Поскольку маршутизаторы первого уровня уже присутствуют в системе, на втором уровне сети можно использовать относительно недорогие 36-портовые маршрутизаторы - количество Infiniband кабелей и их длина от этого не меняется.

1.3. Подсистема мониторинга и управления

Подсистема мониторинга и управления обеспечивает надежное выполнение всех функций по удаленному обслуживанию установки, за исключением функций, требующих физических манипуляций. Подсистема использует как возможности стандартных IPMI средств мониторинга, так и оригинальную разработку – сеть Servnet. Компоненты Servnet присутствуют во всех основных модулях «СКИФ Аврора»:

1) на уровне узлов интегрированы контроллер и датчики температуры и влажности;
2) на уровне «корневой» платы интегрированы датчики и контроллер управления;
3) на плате блока питания интегрированы датчики и контроллер управления питанием;
4) соединительная плата обеспечивает связь сети Servnet на уровне половины шасси.

Отличительной особенностью Servnet является возможность осуществления мониторинга даже в случае полного отключения электропитания всех основных систем – питание Servnet осуществляется независимо.

«Корневые» платы играют важную роль в системе управления установкой. Именно программное обеспечение, работающее на корневой плате, позволяет отключать и включать электропитание отдельных узлов, осуществлять мониторинг характеристик системы во время работы. Программное обеспечение «корневой платы» осуществляет вывод информации на сенсорные дисплеи, установленные в торцах шасси.

Программное обеспечение мониторинга интегрирует информацию из различных источников, включая подсистемы электропитания, охлаждения, хранения данных, отображает и хранит архив данных. Поскольку установка «СКИФ Аврора» носит экспериментальный характер, под нужды управления и мониторинга выделен отдельный сервер.

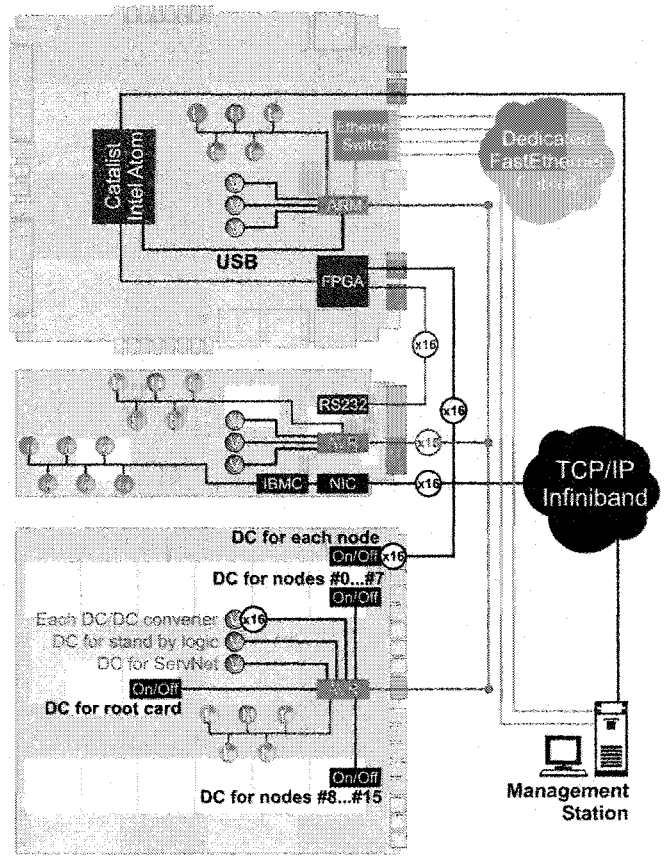


Рис. 2. Сети системы управления и мониторинга

1.4. Подсистема электропитания

Подсистема электропитания вычислителя «СКИФ Аврора» осуществляется постоянным током с напряжением 48В. За счет использования постоянного тока подсистема бесперебойного электроснабжения оказывается проще – содержит лишь выпрямитель и аккумуляторные батареи. Преобразователь постоянного тока в переменный оказывается не нужен.

Бесперебойное питание сервера мониторинга дополнительно резервировано – для обеспечения автономной работы в течение полутора часов. Таким образом, система мониторинга вполне в состоянии исполнять роль «черного ящика» вычислительной системы.

Подсистема хранения данных реализована на основе параллельной файловой системы Lustre. Общий объем подсистемы – более 50 терабайт. Теоретически подсистема должна обеспечивать производительность более 4000 операций ввода-вывода (IOPS) и пропускную способность более 500 Мбайт/с. Узлы вычислителя имеют доступ к хранилищу данных по вспомогательной сети – Infiniband QDR.

2. Описание задач

Для исследования эффективности выполнения приложений на вычислителе «СКИФ Аврора» группой сотрудников ЮУрГУ были отобраны несколько задач. Данные задачи анализировались с точки зрения их масштабирования и оптимизации группой специалистов компании Интел (Нижний Новгород), работающей с HPC проектами. Среди приложений можно выделить задачи инженерного проектирования и анализа, решаемые с использова-

нием стандартных инженерных пакетов, а также программный комплекс, реализованный на языке Фортран с использованием библиотеки MPI для решения задачи моделирования процессов формирования металлических наночастиц методом газофазной конденсации. Ниже представлены описания приложений.

2.1. Задача вычислительной гидродинамики тонких турбулентных слоев в щелевых уплотнениях питательных насосов электрических станций

Надежность питательного насоса определяется его вибрационным состоянием. Основным источником вибрации является неуравновешенный ротор, динамика которого в значительной мере зависит от упругих, демпфирующих и инерционных свойств турбулентной жидкости, дросселируемой в щелевых уплотнениях [2].

Щелевые уплотнения характеризуются малым зазором (0,1 – 0,5 мм) по сравнению с линейными размерами (для цилиндрических уплотнений – длина $\sim$ 200 мм, диаметр $\sim$ 200 мм, для радиальных – внутренний радиус $\sim$ 140 мм, длина $\sim$ 40 мм), а также наличием перекоса и эксцентриситета.

Традиционно при расчетах гидродинамики тонких турбулентных слоев в щелевых уплотнениях используются укороченные уравнения Навье – Стокса (уравнения тонкого слоя), которые принципиально не позволяют определить падение давления на входном участке тонкой щели.

Использование численных методов расчета полных уравнений Навье – Стокса с Рейнольдсовым осреднением позволяет в общем виде решить задачу формирования тонкого слоя на начальном участке и течения жидкости в щели при нестационарном движении твердой стенки. Определение гидродинамических сил в тонких слоях щелевых уплотнений мощных питательных насосов требует решения системы уравнений с числом неизвестных 50 – 100 млн. Решение подобных задач возможно только с использованием высокопроизводительных вычислительных систем и мощных пакетов CFD.

2.2. Деформирование и разрушение тканевых бронежилетов при локальных ударах

Основной задачей при проектировании бронежилетов является минимизация их массы при сохранении заданного уровня защиты. Проверка качества бронежилета, не находящегося в контакте с защищаемым объектом, проводится с определением баллистического предела. А если бронежилет контактирует с защищаемым объектом (тело человека), то в этом случае существует критерий определения тупой травмы, который применяется для сравнения бронежилетов различных классов [3].

В экспериментах в качестве тела человека используют либо технический пластилин (при этом довольно сложно оценить степень травмирования тела человека), либо дорогостоящие экспериментальные модели грудной клетки. Экспериментально-аналитический путь оптимизации конструкции многослойных тканевых преград позволяет достаточно быстро определить оптимальное соотношение параметров для фиксированного воздействия (конкретных формы индентора и скорости нагружения), однако этот метод весьма затратный.

Чисто аналитических моделей, точно описывающих процесс динамического взаимодействия пули и бронежилета с учетом разрушения, на данный момент не существует и, очевидно, их получение невозможно из-за сложности физических явлений, происходящих в этом процессе: большие перемещения, скольжение, фрикционные контакты, повышение температуры. Для того, чтобы учесть эти сложные физические явления, необходимо учитывать структуру баллистической ткани.

Вычислительные возможности кластеров позволяют решать сложные контактные за-

дачи, в которых нельзя использовать механику сплошной среды. Полученные результаты и методы исследования сопротивления тканевых преград ударам огнестрельного оружия используются при разработке новых средств защиты тела человека, значительно сокращая этап предварительной оценки служебных свойств такого рода изделий.

2.3. Моделирование механического поведения грудной клетки человека при локальных ударах

При разработке персональной защитной брони минимальной массы необходимо иметь представление о механизме повреждений, которые возникают в теле человека при локальных ударных воздействиях. Поэтому учеными разных стран ведется работа по созданию теоретических и экспериментальных моделей тела человека, которые в точности повторяют форму человеческого тела и обладают такими же свойствами. Учеными Университета имени Джона Хопкинса в США (Вашингтон) были созданы конечно-элементная и экспериментальная модели грудной клетки человека, были построены ребра, грудина, хрящи, позвоночник, сердце, легкие, печень, желудок, мышцы и кожа. Если значения ускорений, полученные экспериментально и с помощью расчета, близки, то отличие давлений существенно. Таким образом, разработки теоретических и экспериментальных моделей грудной клетки человека активно продолжаются, однако какие-либо достоверные данные пока получены не были. К тому же были созданы только модели деформирования тела человека без учета степени травмирования [3].

Для того, чтобы использовать численную модель грудной клетки человека для проектирования бронежилетов, необходимо знать механические свойства всех ее элементов. Идентификацию параметров грудной клетки можно провести, сопоставив экспериментальные и расчетные перемещения при статическом нагружении, и ускорений, спектра собственных частот колебаний при динамическом нагружении. При этом динамическое нагружение грудной клетки реального человека должно быть низкоскоростным, чтобы не нанести травм человеку.

Экспериментальные модели грудной клетки человека имеют высокую стоимость, поэтому численное решение данной задачи является актуальной проблемой. При численном исследовании задачи возможно оценить степень травмирования грудной клетки человека.

2.4. Деформационные изменения структуры трикотажных полотен на различных участках фигуры человека

Сегодня изделия из трикотажного полотна имеют широкое распространение, поэтому актуальным является вопрос быстрого и качественного проектирования новых моделей. Трикотажные изделия значительно растягиваются при эксплуатации, причем не одинаково на разных участках тела человека, к тому же в изделии присутствуют различные виды швов (стачные, окантовочные, в подгибку с открытым срезом), которые имеют другие механические свойства. Поэтому при разработке трикотажных изделий использование геометрического метода является некорректным [4].

В настоящее время используют параллельные алгоритмы для изучения поведения тканей. Проектирование с использованием суперкомпьютеров позволяет значительно сократить материальные затраты и время на разработку нового изделия. В виртуальной модели можно легко менять различные параметры: механические свойства ткани и швов, геометрию тела человека и изделия.

2.5. Моделирование процессов газофазной конденсации металлических наночастиц

В производстве микро- и наночастиц различных веществ часто используется метод «самосборки» частиц при их конденсации в пересыщенном паре в атмосфере инертного газа. При этом для дальнейшего использования полученного наноразмерного порошка необходимо соблюдение определенных требований к размеру частиц. Для этого необходимо задать определенный температурный режим в рабочей камере реактора, давление и вид инертного газа, геометрию установки, а также длительность производственного цикла. В настоящее время все эти параметры подбираются экспериментально, методом «проб и ошибок». В таких условиях очень сложно осуществлять управление технологическим процессом и прогнозировать выходное распределение частиц по размерам. Разрабатываемые математическая модель и программный комплекс направлены на решения данных задач [5].

Обычная схема формирования металлических наночастиц конденсацией из газовой фазы выглядит следующим образом: в камеру с охлаждаемыми стенками накачивается инертный газ, вблизи дна камеры помещается нагреваемый сосуд с кипящим жидким металлом, который служит источником атомов металла - мономеров. Испаряясь с поверхности жидкости, металлический пар распространяется в объеме камеры, где, по мере его охлаждения, могут возникнуть высокие степени пересыщения, являющиеся необходимым условием нуклеации.

Задача математического моделирования заключается в численном анализе процессов образования металлических наночастиц с целью прогнозирования размеров этих частиц, полученных при различных условиях: перепадах температуры в камере, давлении инертного газа, вида инертного газа, геометрических размеров рабочей камеры и испарителя и, возможно, расположения в камере охлаждаемых поверхностей, на которых будет происходить инерционное осаждение наночастиц. Решение задачи состоит из двух частей: первая заключается в моделировании конвективных течений в камере и расчете распределения температуры, плотности и концентрации мономеров в газовой смеси, вторая состоит в численной симуляции образования кластеров при их объемной конденсации в атмосфере инертного газа.

3. Анализ вычислительных экспериментов

Исследования эффективности выполнения приложений проводились на трех вычислительных системах: «СКИФ Урал», «СКИФ Аврора» и «Endeavor»(DuPont, США). Характеристики данных систем:

1. «СКИФ Урал»: Intel Xeon E5472 (Harpertown) 4 ядра на сокет, 2 процессора на узел, тактовая частота 3.00 ГГц.

2. «СКИФ Аврора»: Intel Xeon X5570 (Nehalem) 4 ядра на сокет, 2 процессора на узел, тактовая частота 2.93 ГГц

3. «Endeavor»: Intel Xeon X5670 (Westmere) 6 ядер на сокет, 2 процессора на узел, тактовая частота 2.93 ГГц.

Рассмотрим более подробно результаты запусков задач на данных вычислителях.

3.1. Задача вычислительной гидродинамики тонких турбулентных слоев в щелевых уплотнениях питательных насосов электрических станций

Для задачи вычислительной гидродинамики тонких турбулентных слоев в щелевых уплотнениях питательных насосов электрических станций приведены графики ускорений на рис. 3, 4. Исходный файл для решателя инженерного пакета ANSYS CFX содержит 29 тыс. элементов сетки.

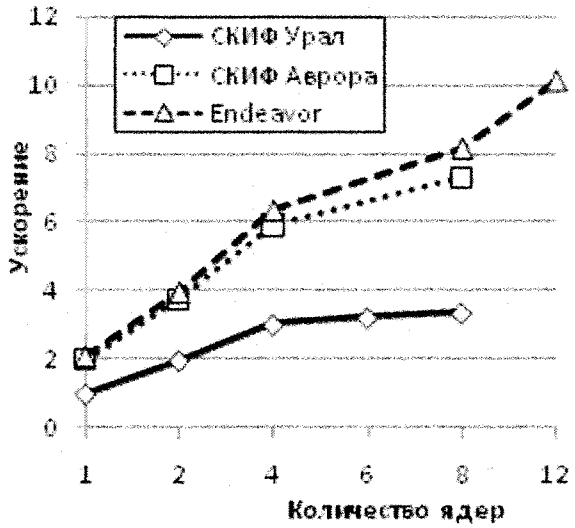


Рис. 3. График ускорений задачи гидродинамики тонких турбулентных слоев в пределах одного узла

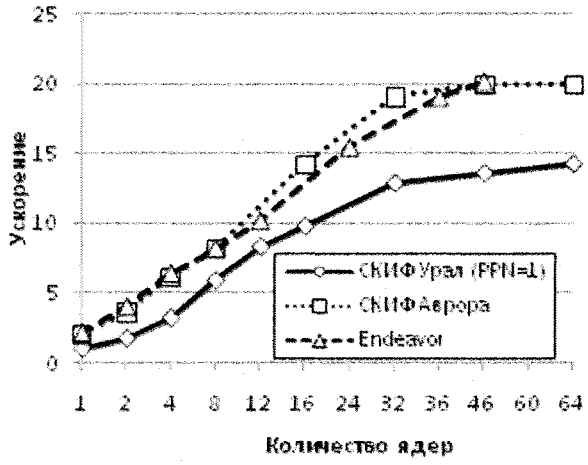


Рис. 4. График ускорений задачи гидродинамики тонких турбулентных слоев в пределах вычислительной системы

Отметим, что исследуемое приложение имеет небольшие размеры, и насыщение шкалируемости на вычислительных системах происходит достаточно быстро (рис. 4).

Основным достоинством процессоров поколения Nehalem и Westmere является многократно увеличенная пропускная способность подсистемы памяти. Если проводить сравнение с помощью теста Stream, то заметим, что пропускная способность увеличилась с типичных 10.5 Гбайт/с для систем Harpertown до 38 Гбайт/с для системы Nehalem (с памятью DDR3-1333). Соответственно, в предельных случаях на некоторых задачах вполне возможен прирост производительности в 3,5 раза. В свою очередь, увеличение пропускной способности в Nehalem вызвано несколькими причинами:

1) отказом от шинной архитектуры в пользу решения типа NUMA;

2) интегрированием контроллера памяти в сам процессор;

3) увеличением числа поддерживаемых каналов обмена с памятью с 4 до 6 (на систему из 2-х процессоров);

4) переходом на технологию памяти DDR3.

Первые два пункта увеличили реальную эффективность (40% – 60%), а последние два – ее теоретическую пропускную способность с 25,6 Гбайт/с до 64 Гбайт/с. В процессоре Westmere эффективность была еще улучшена, это дало пропускную способность около 42,5 Гбит/с на тесте Stream (эффективность уже 66% от теоретического значения).

Принимая во внимание то обстоятельство, что средний поток данных в этой задаче ~ 20 Гбайт/с легко объяснить полученные результаты производительности. Для всех систем мы наблюдаем неидеальную шкалируемость внутри одного узла, которая объясняется тем, что некоторые ядра простаивают в ожидании необходимых данных из памяти. В то же время если для «СКИФ Урал» насыщение масштабируемости происходит довольно быстро (10,5 Гбайт/с $<$ 20 Гбайт/с), то на «СКИФ Аврора» (38 Гбайт/с $>$ 20 Гбайт/с) и «Endeavor» (42,5 Гбайт/с $>$ 20 Гбайт/с) производительность продолжает расти с ростом числа задействованных ядер.

3.2. Деформирование и разрушение тканевых бронежилетов при локальных ударах

Задача деформирования и разрушения тканевых бронежилетов при локальных ударах рассчитывалась для пакета размером 5x5 см из 5 слоев баллистических тканей.

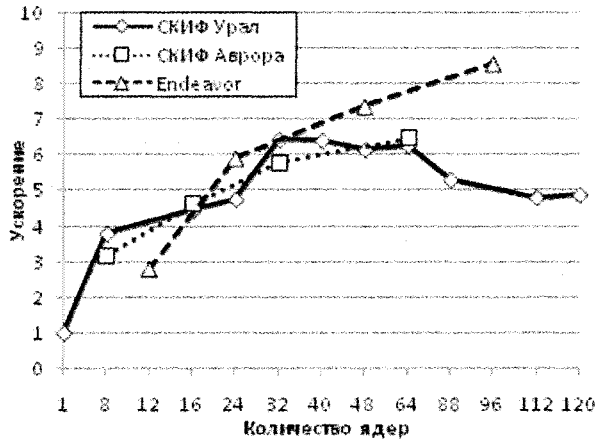


Рис. 5. График ускорений задачи деформирования и разрушения тканевых бронежилетов при локальных ударах

Данная задача относится к задачам другого типа - в ней превалируют вычисления, а взаимодействие с памятью не столь заметно. В силу этих обстоятельств, «СКИФ Урал» показывает результаты сравнимые со «СКИФ Аврора» за счет более высокой тактовой частоты. Однако «Endeavor» показывает несколько более высокие результаты за счет большего количества ядер на сокет (6 против 4). Стоит отметить, что в этом случае имеют место вычисления с одинарной точностью. При использовании двойной точности ситуация выглядела бы иначе, за счет возросшего в два раза объема взаимодействий с памятью. Обратим внимание на еще один момент, связанный с насыщением масштабируемости этой задачи. Дело в том, что исходный файл для решателя инженерного пакета LS-Dyna имеет небольшие размеры, и время, затрачиваемое на коммуникации между узлами, быстро становится сравнимым со временем вычислений.

3.3. Задачи моделирования механического поведения грудной клетки человека при локальных ударах и деформационных изменений структуры трикотажных полотен на различных участках фигуры человека

Графики ускорений для обеих задач приведены на рис. 6, 7.

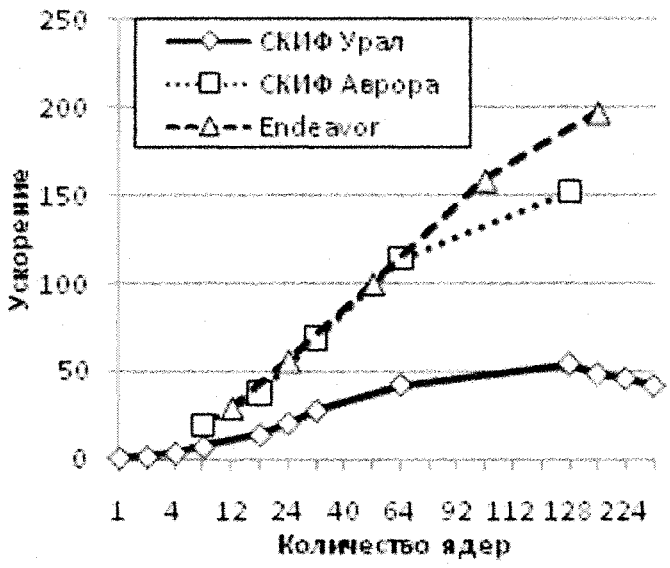


Рис. 6. График ускорений для задачи моделирования механического поведения грудной клетки человека

Задача моделирования механического поведения грудной клетки человека при локальных ударах и задача деформационных изменений структуры демонстрируют сходное поведение, в целом типичное для HPC приложений. Производительность систем определяется правильным балансом между их вычислительной способностью и скоростью взаимодействия с памятью. За счет улучшения этого баланса системы нового поколения показывают лучшую производительность, нежели «СКИФ Урал». Стоит также отметить, что при почти одинаковой производительности в расчете на одно ядро, «Endeavor» показывает более высокие результаты в расчете на один узел, так как имеет большее количество ядер – 6.

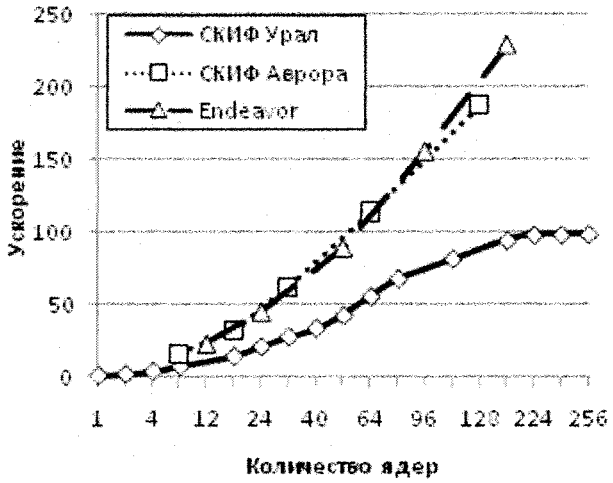


Рис. 7. График ускорений для задачи деформационных изменений структуры

3.4. Моделирование процессов газофазной конденсации металлических наночастиц

График ускорений для задачи моделирования процессов газофазной конденсации металлических наночастиц приведен на рис. 8.

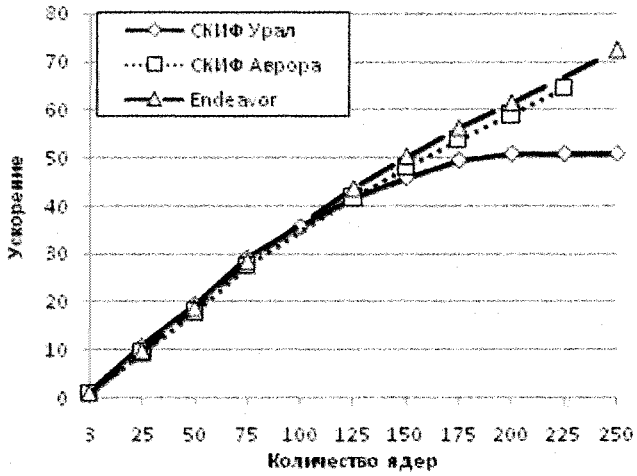


Рис. 8. График ускорений для задачи моделирования процессов газофазной конденсации металлических наночастиц

Данная задача в целом очень сходна с задачей деформирования и разрушения тканевых бронежилетов при локальных ударах, поскольку в ней зависимость от скорости общения с памятью не так велика. И все же она сказывается, хотя и достаточно неочевидным способом. На кластере «СКИФ Урал» производился запуск на меньшем количестве процессов на один узел с целью исключить конфликты обращения в память, исходящих от различных ядер внутри одного узла. Это приводит к росту числа задействованных узлов кластера, а соответственно и увеличению объема коммуникаций между ними. Увеличение объема ком-

муникаций в свою очередь приводит к более быстрому насыщению шкалируемости, которое наблюдается на «СКИФ Урал». Таким образом, производительность подсистемы процессор-память внутри одного узла ограничивает кластерную масштабируемость задачи.

Заключение

В данной работе рассмотрены несколько HPC приложений, обладающих различными свойствами: первая задача главным образом зависит от пропускной способности системы процессор-память, вторая задача – от вычислительной мощности системы, пятая задача чувствительна к объемам коммуникаций, а остальные сочетают в себе все вышеперечисленные свойства. Подводя общий итог, можно сделать следующий вывод. При работе с вышеперечисленными приложениями сталкиваемся с достаточно типичной ситуацией для HPC вычислений – производительность зависит не только от частоты процессоров и количества ядер на чипе, не менее важными факторами являются производительность системы процессор-память, коммуникации в распределенной системе и иногда скорость файлового ввода-вывода. Для обеспечения максимальной производительности требуется нахождение оптимального баланса этих факторов. Системы типа «СКИФ Аврора», построенная на процессорах с архитектурой Nehalem, делает значительный шаг вперед по сравнению с системой «СКИФ Урал» (архитектура процессора Harpertown), обеспечивая улучшение баланса между вычислительной мощностью процессора и пропускной способностью подсистемы процессор-память. Система «Endeavor» (архитектура процессора Westmere) является следующим шагом на этом пути развития, улучшая как вычислительную способность (6 ядер на чипе вместо 4), так и скорость взаимодействия с памятью (42.5 Гбайт/с против 38 Гбайт/с). Все эти технологические новшества позволяют исследователям расширять «область поиска»и производить более глубокий анализ интересующих явлений за счет увеличения уровня детализации, принятия во внимание эффектов, которые прежде игнорировались.

Авторы выражают благодарность сотруднику корпорации Intel Николаю Местеру за организационную и методическую помощь при выполнении исследований, представленных в данной работе.

Работа выполнена при финансовой поддержке Программы СКИФ-ГРИД (контракт с 2009-СГ-03), ФЦП «Научные и научно-педагогические кадры инновационной России»(контракт с П2036) и РФФИ (проекты 10-07-96001-р_урал_а и 10-07-96007-р_урал_а).

Статья рекомендована к печати программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010» http://agora.guru.ru/pavt.

Литература

1. Абрамов, С.М. СуперЭВМ Ряда 4 семейства СКИФ: штурм вершины суперкомпьютерных технологий / С.М. Абрамов // Параллельные вычислительные технологии (ПаВТ'2009): тр. Междунар. науч. конф. (Нижний Новгород, 30 марта – 3 апр. 2009 г.). – Челябинск, 2009. – С. 5 – 16.

2. Васильев, В.А. Сравнительный анализ области применения тестовых задач оценки вычислительной мощности HPC систем / В.А. Васильев, А.Ю. Ницкий // Параллельные вычислительные технологии (ПаВТ'2010): тр. Междунар. науч. конф. (Уфа, 29 март. – 2 апр. 2010 г.). – Челябинск, 2010. – С. 431 – 441.

3. Долганина, Н.Ю. Моделирование ударных процессов в тканевых бронежилетах и теле человека на вычислительном кластере СКИФ Урал / Н.Ю. Долганина, С.Б. Сапожников

// Параллельные вычислительные технологии (ПаВТ'2010): тр. Междунар. науч. конф. (Уфа, 29 марта – 2 апр. 2010 г.). – Челябинск, 2010. – С. 141 – 152.

4. Долганина, Н.Ю. Суперкомпьютерное моделирование деформационных изменений трикотажных полотен на фигуре человека / Н.Ю. Долганина, А.Ю. Персидская, И.Н. Усенко // Параллельные вычислительные технологии (ПаВТ'2010): тр. Междунар. науч. конф. (Уфа, 29 марта – 2 апр. 2010 г.). – Челябинск, 2010. – С. 606 – 610.
5. Терзи, Д.В. Моделирование процессов газофазной конденсации металлических наночастиц на вычислительном кластере «СКИФ Урал»/ Д.В. Терзи // Параллельные вычислительные технологии (ПаВТ'2010): тр. Междунар. науч. конф. (Уфа, 29 марта – 2 апр. 2010 г.). – Челябинск, 2010. – С. 600 – 605.

Александр Александрович Московский, кандидат химических наук, с.н.с., Институт программных систем РАН (Переяславль-Залесский), moskov@rsc-skif.ru.

Максим Павлович Перминов, ведущий инженер программного обеспечения, Интел Текнолоджис, Инк. (Нижний Новгород), maxim.perminov@intel.com.

Леонид Борисович Соколинский, доктор физико-математических наук, профессор, директор Суперкомпьютерного центра ЮУрГУ (Челябинск), sokolinsky@acm.org.

Валерий Владимирович Черепенников, кандидат физико-математических наук, руководитель HPC-проектов, Интел Текнолоджис, Инк. (Нижний Новгород), Valery.Cherepennikov@intel.com.

Анастасия Валерьевна Шамакина, аспирант, начальник отдела прикладных задач Суперкомпьютерного центра ЮУрГУ (Челябинск), sham2004@bk.ru.

Поступила в редакцию 20 июля 2010 г.

УДК 519.687

СИМУЛЯТОР ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА И ЕГО УПРАВЛЯЮЩЕЙ СИСТЕМЫ, ИСПОЛЬЗУЕМЫЙ ДЛЯ ИССЛЕДОВАНИЯ АЛГОРИТМОВ ПЛАНИРОВАНИЯ ЗАДАЧ

П.Н. Полежаев

SIMULATOR OF COMPUTER CLUSTER AND ITS MANAGEMENT SYSTEM USED FOR RESEARCH OF JOB SCHEDULING ALGORITHMS

P.N. Polezhaev

В данной работе описывается симулятор вычислительного кластера и его управляющей системы, учитывающий его топологию, коммуникационные задержки при передаче данных, а также многопроцессорность вычислительных узлов. Он применяется для исследования алгоритмов планирования параллельных задач на вычислительном кластере. Разработана имитационная схема и модель кластера, модель его вычислительной загрузки, приводятся исследуемые алгоритмы планирования, а также описывается система критериев и метрик их сравнения.

Ключевые слова: симулятор вычислительного кластера и его управляющей системы, планирование задач, высокопроизводительные вычисления, имитационное моделирование

In this work the simulator of computer cluster and its management system which considers its topology, communication latency in data transmition and multiprocessor nodes is described. It is used for investigation of parallel jobs scheduling algorithms on computer cluster. The simulating scheme and model of cluster, its computational load model were developed, job scheduling algorithms were presented, and also the system of criteria and metrics for their comparison is described.

Keywords: a simulator of computer cluster and its management system, jobs scheduling, high performance computings, a simulation modeling

Введение

В настоящее время весьма актуальной является проблема эффективного планирования параллельных задач, поступающих в очередь управляющей системы вычислительного кластера, т.к. современные алгоритмы планирования не обеспечивают должного качества составляемых расписаний запуска задач. Весьма часто возникает ситуация, при которой в расписании остается значительное число окон, в то время, когда в очереди имеется большое количество ожидающих своего исполнения задач. Это приводит к снижению производительности всей вычислительной системы.

Другим существенным недостатком современных управляющих систем вычислительного кластера является использование алгоритмов планирования, не учитывающих топологию при размещении процессов задач на вычислительных узлах, а также их многопроцессорность. Учет данных факторов позволит увеличить эффективность работы алгоритмов планирования.

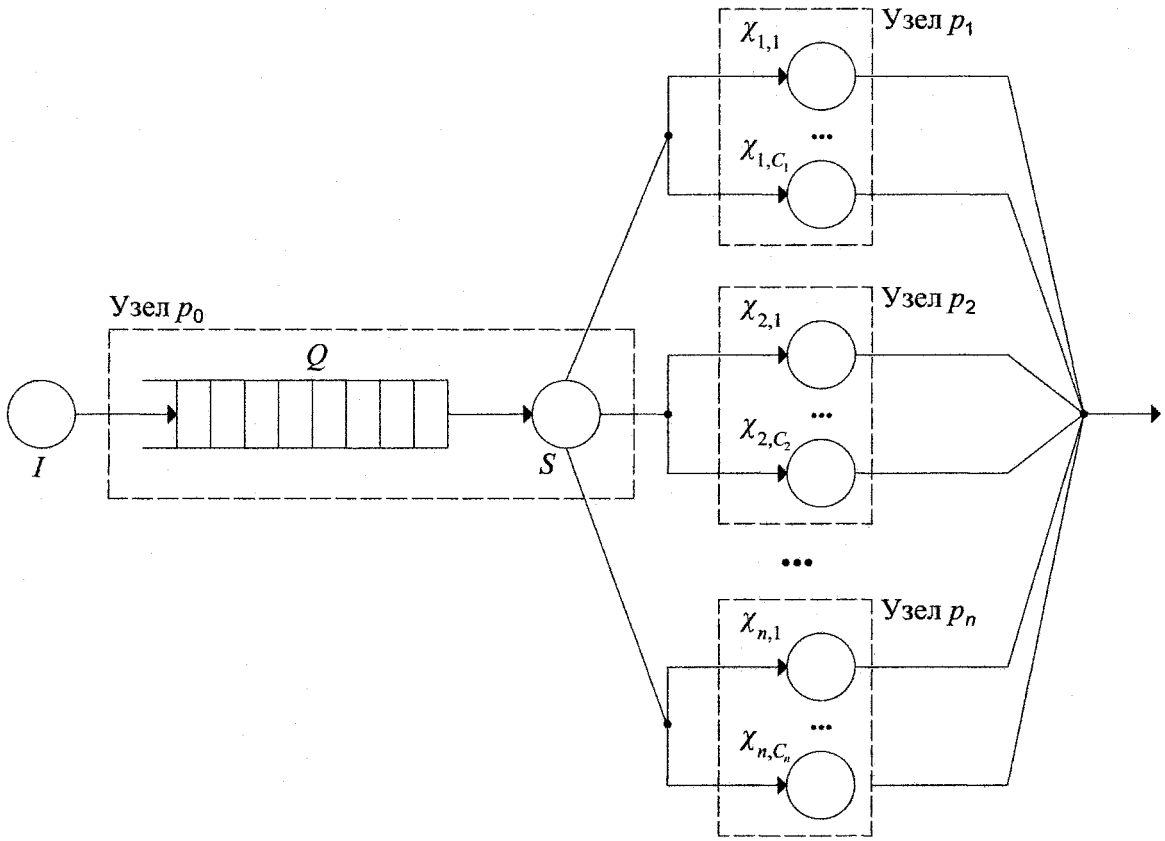


Рис. 1. Имитационная схема управляющей системы вычислительного кластера

В связи с этим возникает необходимость исследования существующих алгоритмов планирования и разработка их новых более эффективных вариантов. Для этой цели в настоящей работе создается симулятор вычислительного кластера и его управляющей системы, учитывающий ее топологию, коммуникационные задержки при передаче данных, а также многопроцессорность вычислительных узлов.

Использование симулятора для исследования алгоритмов планирования перед применением реальной управляющей системы, исполняющейся на настоящем кластере, имеет ряд преимуществ. Во-первых, имеется возможность обеспечения контролируемой вычислительной среды – не надо учитывать возможные проблемы, связанные с необходимостью обеспечения отказоустойчивости или безопасности. Во-вторых, при использовании симулятора обеспечивается гибкость и простота модификации аппаратной конфигурации и режимов работы вычислительного кластера. И, в-третьих, в случае применения реального кластера и управляющей системы имела бы место проблема формирования потоков реальных параллельных задач, отвечающих заданным характеристикам, что, напротив, делается очень просто при использовании имитационного моделирования.

1. Имитационная схема и модель вычислительного кластера

Имитационная схема управляющей системы вычислительного кластера приведена на рис. 1. Источник I генерирует поток параллельных задач, отправляемых пользователями в очередь Q управляющего узла p_0. Канал S представляет собой планировщик, который в

соответствии с заложенным в него алгоритмом осуществляет извлечение задач из очереди Q и назначение их на свободные вычислительные ядра подходящих по конфигурации узлов.

Множество $P = \{p_1, p_2, \ldots, p_n\}$ содержит все вычислительные узлы кластера, связанные с выделенным узлом p_0 с помощью отдельной управляющей сети. Пусть $\mathrm{X}_i = \{\chi_{i,1}, \chi_{i,2}, \ldots, \chi_{i,C_i}\}$ – множество вычислительных ядер узла p_i. Они могут относиться как к отдельным процессорам узла (по одному ядру на каждом), так и к нескольким многоядерным процессорам. С целью упрощения модели они рассматриваются, как единое множество X_i. Обозначим $\mathrm{X} = \bigcup_{i=1}^{n} \mathrm{X}_i$ – множество вычислительных ядер всех узлов кластера.

Конфигурация узлов вычислительного кластера и соединяющей их высокопроизводительной сети может быть описана с помощью следующего взвешенного ориентированного графа:

$$G_T = (P, K, E, b, c, m, d, s), \tag{1}$$

где $K = \{k_1, k_2, \ldots, k_z\}$ – множество коммутаторов, E – множество направленных сетевых связей между ними, $b\colon E \to \mathbb{Z}_+ \bigcup \{0\}$ – отображение, характеризующее пропускную способность каждой сетевой связи в байтах в секунду. Функции $c, m, d\colon P \to \mathbb{Z}_+$ определяют для каждого вычислительного узла p_i соответственно количество его вычислительных ядер $c(p_i) = C_i$, объемы оперативной $m(p_i) = M_i$ и дисковой памяти $d(p_i) = D_i$ в килобайтах. Отображение $s\colon P \to \mathbb{R}_+$ для узла p_i задает относительную производительность $s(p_i) = S_i$ каждого его вычислительного ядра, которая определяет, во сколько раз ядра данного узла работают быстрее вычислительных ядер самого непроизводительного узла кластера.

Заметим, что орграф (1) позволяет моделировать статические и динамические сети с полнодуплексными и полудуплексными связями. Настоящая модель также позволяет рассматривать однородные и неоднородные вычислительные системы, как по составу вычислительных узлов, так и по конфигурации высокопроизводительной сети. В рамках проводимого исследования будут рассматриваться только однородные вычислительные сети с полнодуплексными связями.

Статические параметры C_i, M_i, D_i и S_i описывают конфигурацию вычислительного узла p_i, а динамические параметры характеризуют его состояние в произвольный момент времени $t \in [0; +\infty)$: $u_i(t)$ – загруженность вычислительных ядер, $m_i(t)$ и $d_i(t)$ – объемы доступной оперативной и дисковой памяти.

Имитационная схема вычислительного узла p_i приведена на рис. 2. p_i соединен с другими узлами и коммутаторами вычислительной сети с помощью r_i дуплексных связей. Все входящие пакеты, а также пакеты сообщений, генерируемых процессами, выполняющимися на вычислительных ядрах, сначала поступают в очередь $Q_{inp.i}$, а затем маршрутизируются каналом обслуживания R_i. Если соответствующий пакет предназначен для локального вычислительного ядра, то он передается ему непосредственно, иначе – помещается в одну из очередей $Q_{out.i,1}, Q_{out.i,2}, \ldots, Q_{out.i,r_i}$, соответствующую выбранной алгоритмом маршрутизации исходящей связи. Предполагается использование алгоритмов статической и динамической маршрутизации, характерных для конкретных топологий вычислительной системы, в том числе алгоритмы кратчайших коммуникационных путей и покоординатной маршрутизации.

При выполнении маршрутизации каналом R_i моделируется временная задержка величиной $l_{routing}$. Заметим, если два процесса одной задачи выполняются на соседних ядрах узла, то обмен пакетами между ними также осуществляется через очередь $Q_{inp.i}$ и канал R_i.

Имитационная схема коммутатора, приведенная на рис. 3, представляет собой урезанный вариант имитационной схемы узла. Отличие только в том, что коммутатор не может выполнять вычисления, и поэтому у него отсутствуют вычислительные ядра.

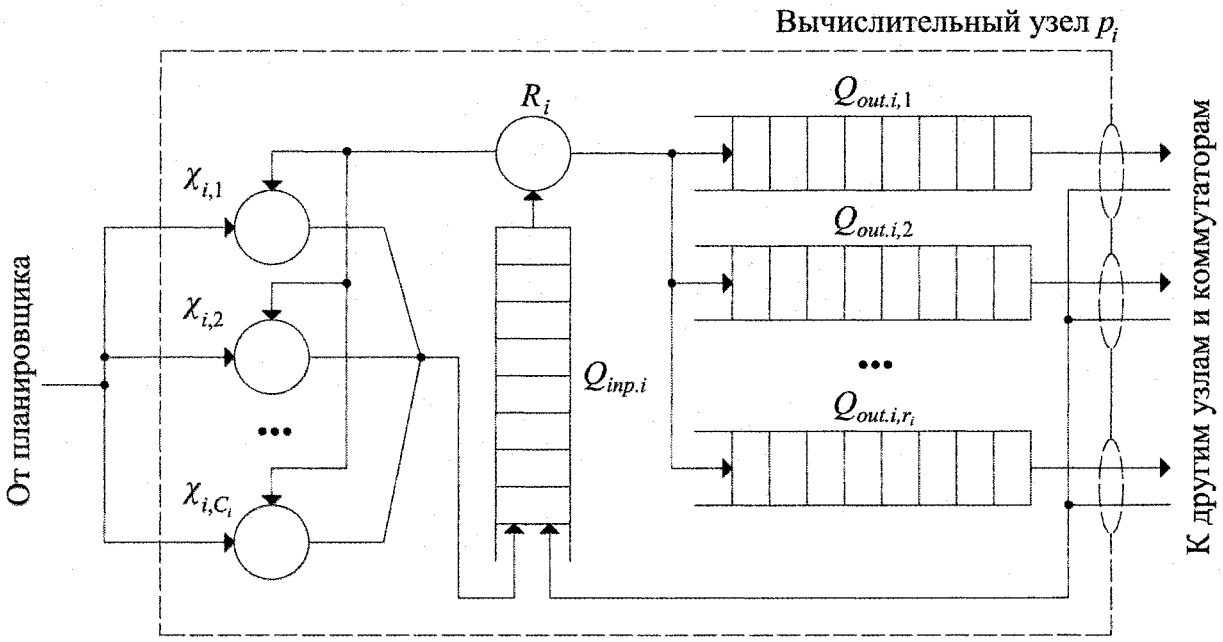


Рис. 2. Имитационная схема вычислительного узла p_i

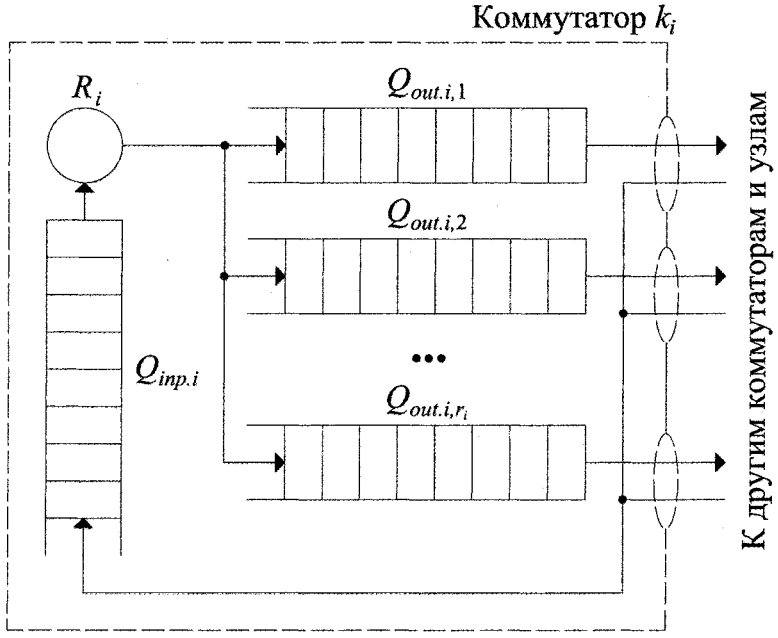


Рис. 3. Имитационная схема коммутатора k_i

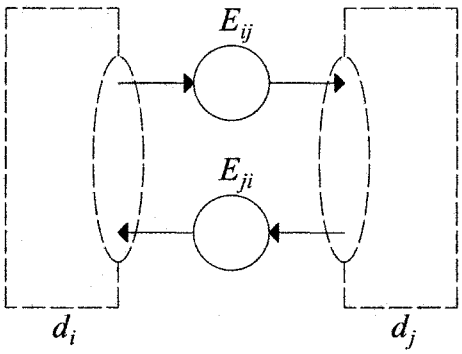


Рис. 4. Схема работы дуплексной связи L_{ij}

Обозначим в качестве $D = P \bigcup K$ множество всех сетевых устройств системы. На рис. 4 приведена схема работы дуплексной связи $L_{ij} = \{E_{ij}, E_{ji}\}$, соединяющей сетевые устройства $d_i \in D$ и $d_j \in D$. Каналы обслуживания E_{ij} и E_{ji} добавляют к времени передачи пакета задержку величиной $b(E_{ij}) = b(E_{ji})$.

Настоящая модель вычислительной системы предполагает использование принципа коммутации пакетов при передаче данных. Пусть g задает стандартный фиксированный размер одного пакета, а $h < g$ – размер его заголовочной части, тогда для передачи сообщения размера q потребуется $p = \left\lceil \frac{q}{g-h} \right\rceil$ пакетов.

Время передачи сообщения от сетевого устройства d_{i_0} к d_{i_R} вдоль пути $\alpha = (d_{i_0}, d_{i_1}, \ldots, d_{i_R})$ может быть вычислено по формуле:

$$T_{trans}(\alpha, p) = \sum_{k=0}^{R-1} \frac{g}{b(E_{d_{i_k} d_{i_{k+1}}})} + \frac{g(p-1)}{\min\limits_{k=\overline{0,R-1}} b(E_{d_{i_k} d_{i_{k+1}}})} + l_{routing} \cdot (R+1)p. \qquad (2)$$

Заметим, данная формула не учитывает возможные задержки, связанные с сетевой конкуренцией пакетов.

Настоящая работа предполагает проведение исследования алгоритмов планирования параллельных задач для случаев однородных и неоднородных вычислительных узлов, связанных гомогенной сетью. Рассматриваются следующие топологии современных кластерных вычислительных систем: двухуровневые толстые деревья, $2D$ и $3D$ торы, k-арные n-кубы, k-арные n-деревья и звезды с единственным коммутатором в центре. Все они могут быть описаны с помощью ориентированного графа (1).

2. Модель вычислительной загрузки кластера

Вычислительная загрузка кластера формируется потоком задач $J = (J_1, ..., J_m)$, помещаемых пользователями в очередь его управляющей системы. Каждая задача представляет собой параллельную неинтерактивную программу, способную работать в пакетном режиме. Ее процессы запускаются планировщиком одновременно на всех выделенных вычислительных ядрах, во время работы они обмениваются сообщениями между собой. Ресурсы, выделенные задаче, освобождаются при завершении всех ее процессов.

Каждая задача J_j характеризуется следующим набором параметров:

$$J_j = (n_j, mr_j, dr_j, \tilde{t}_j, a_j, \tau_j, CP_j), \qquad (3)$$

где n_j – количество необходимых вычислительных ядер, mr_j и dr_j – соответственно объемы оперативной и дисковой памяти в килобайтах, запрашиваемые для исполнения каждого

процесса задачи. Величина $\tilde{t}_j$ задает оценку пользователя в секундах времени выполнения программы на узлах с единичной относительной производительностью. Параметр a_j определяет время поступления задачи в очередь, а $\tau_j \in (0; \tilde{t}_j]$ – время, затрачиваемое задачей только на вычисления (без сетевых коммуникаций) при условии единичной относительной производительности всех назначенных ей вычислительных узлов. Множество CP_j задает набор коммуникационных паттернов задачи J_j.

Следующая формула позволяет вычислить t_j – оценку времени выполнения задачи с учетом производительности выделенных ей планировщиком вычислительных узлов:

$$t_j = \frac{\tilde{t}_j}{\min\limits_{i \in I_j} S_i}, \tag{4}$$

где I_j – множество номеров узлов, ядра которых были отданы задаче. Аналогичная формула применяется для корректировки значения параметра τ_j.

В рамках данной модели предполагается, что пользователь, делая оценку времени выполнения задачи, возможно, допускает ошибку, переоценивая ее по сравнению с реальным временем выполнения. Противоположный случай ошибок не рассматривается, т.к. управляющая система будет принудительно завершать задачи, превысившие лимит выделенного им времени.

Структура программы каждой параллельной задачи J_j может быть представлена следующей SPMD-моделью:

Process u**:**
for $i = 1$ to q_j do
{
    Communication(i, j);
    Computation(i, j);
}

Выполнение каждого процесса параллельной задачи представляет собой чередование фаз коммуникации и вычислений.

Коммуникационная часть каждой итерации может быть описана с помощью коммуникационного паттерна – ориентированного взвешенного графа следующего вида:

$$CP_{ij} = (\Pi_j, p_{ij}), \tag{5}$$

где $\Pi_j = \{\pi_1, \pi_2, \ldots, \pi_{n_j}\}$ – множество процессов задачи, $p_{ij} \colon \Pi_j \times \Pi_j \to \mathbb{Z}_+ \bigcup \{0\}$ – функция, определяющая вес каждой дуги, равный размеру в пакетах передаваемого сообщения. $p_{ij}(u, v)$ определяет количество пакетов сообщения, передаваемого от процесса u к процессу v на i-й итерации SPMD.

Тогда $CP_j = \{CP_{1j}, CP_{2j}, \ldots, CP_{q_j j}\}$ – множество всех коммуникационных паттернов задачи J_j.

Пусть $pred_{ij}(u) = \{v \in \Pi_j : p_{ij}(v, u) > 0\}$ – множество предшественников процесса u в орграфе (5), $succ_{ij}(u) = \{v \in \Pi_j : p_{ij}(u, v) > 0\}$ – множество его последователей. Выполнение фазы Communication(i, j) для каждого процесса u заключается в том, что сначала происходит неблокируемая рассылка сообщений всем процессам множества $succ_{ij}(u)$, а затем выполняется блокируемый прием всех сообщений от процессов $pred_{ij}(u)$.

Можно выделить два основных способа задания множества CP_j: случайная генерация и моделирование реальных программ, например, эталонных тестов, реализации быстрого

преобразования Фурье, алгоритма параллельного умножения матриц и др. Типичные коммуникационные паттерны, встречаемые в научной литературе [1, 2]: Random, Pairs, Ring, One-to-all и All-to-all.

Заметим, что в рамках настоящей модели рассматриваются только коммуникации вида «точка-точка». Моделировать коллективные операции достаточно сложно, т.к. способ их выполнения в виде совокупности точечных операций зависит от нескольких факторов: используемых алгоритмов, характеристик вычислительной системы и передаваемых сообщений. В частности, коллективные операции MPI по-разному реализованы в библиотеках разных производителей, а также в разных версиях библиотеки одного производителя. В будущем планируется расширение данной модели за счет учета коллективных операций.

Пусть функция $\gamma_j\colon \Pi_j \to P$ позволяет определить для каждого процесса задачи J_j вычислительный узел, на ядро которого он был назначен планировщиком. Для простоты положим, что каждый процесс тратит одинаковое время на выполнение вычислительной части каждой итерации SPMD-модели задачи. Тогда время выполнения вычислительной фазы Computation(i,j) процессом u можно вычислить по формуле:

$$T_{comp.ij}(u) = \frac{\tau_j}{q_j s(\gamma(u))}. \tag{6}$$

Пусть $T_{comm.ij}(u)$ – время выполнения коммуникационной фазы Communication(i,j) процессом u задачи J_j, тогда реальное время выполнения данной задачи T_j может быть вычислено по формуле:

$$T_j = \max_{u\in\Pi_j} \sum_{i=1}^{q_j} (T_{comm.ij}(u) + T_{comp.ij}(u)) = \max_{u\in\Pi_j} \{\sum_{i=1}^{q_j} T_{comm.ij}(u) + \frac{\tau_j}{s(\gamma(u))}\}. \tag{7}$$

Величина $T_{comm.ij}(u)$ зависит от сетевой конкуренции и от физического расположения процессов на вычислительных узлах, ее значение может быть вычислено в процессе симуляции работы вычислительного кластера и его управляющей системы. Заметим, что в рамках данной модели мы, в силу незначительности, пренебрегаем временем, которое тратится на разбиение сообщения на пакеты и его сборку из них.

Симуляция работы вычислительного кластера и его управляющей системы предполагает формирование потока задач. Все описанные выше количественные параметры генерируются на основе определенных законов распределений случайных величин, подобранных в результате анализа реальных трасс, собираемых на кластерных вычислительных системах (см. таблицу). Более подробную информацию можно найти в статье [3].

Таблица

Законы распределений параметров модели

Параметр	Используемый закон распределения
$\log n_j$	двухэтапное равномерное распределение с выделением классов последовательных и 2^k-задач
$\log \tau_j$	гипер-гамма распределение, параметр p которого линейно зависит от n_j
$\tilde{t}_j$	равномерное распределение на отрезке $[\tilde{\tau}_j; 2\tilde{\tau}_j]$
$a_{j+1} - a_j$	экспоненциальное распределение
mr_j, dr_j	равномерное распределение
$p_{ij}(u,v)$	экспоненциальное распределение для всех дуг, существующих в CP_{ij}

3. Алгоритм работы симулятора

В основе симулятора вычислительного кластера и его управляющей системы лежит алгоритм моделирования по принципу особых состояний.

В рамках рассматриваемой модели выделены следующие особые состояния (события):

1. Поступление новой задачи в очередь.
2. Окончание передачи пакета по сетевой связи.
3. Окончание маршрутизации пакета на сетевом устройстве.
4. Окончание вычислительной фазы процесса задачи.
5. Запуск спланированной задачи.

Для работы алгоритма моделирования по принципу особых состояний используется очередь Q_{event}, содержащая события перечисленных выше типов. Они сортируются по неубыванию времени наступления, а при равных значениях времени – в порядке приоритетности событий согласно данному списку. Это необходимо во избежание возможных конфликтов.

Обобщенная блоксхема работы алгоритма моделирования приведена на рис. 5. После

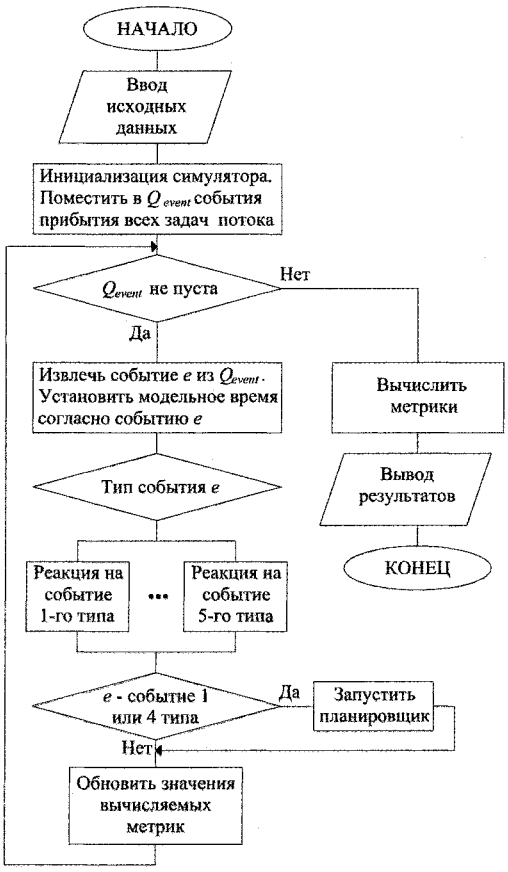


Рис. 5. Обобщенный алгоритм работы симулятора

ввода параметров модели выполняется инициализация симулятора. Затем, пока очередь

Q_{event} не пуста, происходит извлечение очередного события и его обработка, в процессе которой в Q_{event} могут добавляться новые события. Например, при обработке события окончания передачи пакета по сетевой связи может создаваться событие окончания маршрутизации этого пакета на сетевом устройстве-получателе.

При наступлении событий поступления новой задачи или окончания вычислительной фазы процесса задачи происходит запуск алгоритма планирования. Значения количественных метрик обновляются в конце каждой итерации цикла моделирования и окончательно вычисляются по его завершению.

4. Система критериев и метрик сравнения алгоритмов планирования

С целью учета всех аспектов эффективности работы алгоритмов планирования задач для кластерных вычислительных систем была построена система критериев и метрик их сравнения. Она включает следующие критерии:

1. Производительность расписаний. Содержит метрики: средняя загруженность вычислительных ядер узлов кластера U_{cores}, потеря производительности кластера CL, максимальное время завершения задачи C_{max}, среднее время ожидания задач в очереди $\overline{t}_{wait}$ и среднее ограниченное замедление задач $\overline{s}_{lim}$. Метрики U_{cores}, CL и C_{max} характеризуют непосредственно производительность расписания, поэтому они более важны для исследования, чем $\overline{t}_{wait}$ и $\overline{s}_{lim}$, которые имеют косвенный характер.

2. Используемость оперативной и дисковой памяти. Включает метрики $\overline{m}$ и $\overline{d}$, определяющие соответственно средние загруженности оперативной и дисковой памяти вычислительных узлов. Позволяет выявить процент неиспользованных ресурсов.

3. Сбалансированность загрузки узлов. Определяет метрики Du, Dm и Dd, обозначающие соответственно дисперсии загруженности ядер, оперативной и дисковой памяти вычислительных узлов. Демонстрирует честность алгоритма планирования по отношению к узлам.

4. Гарантированность обслуживания задач. Включает метрики максимального времени ожидания задач в очереди $t_{wait.max}$ и максимального ограниченного замедления задач $s_{lim.max}$. Данный критерий имеет особую важность в случае, если кластер используется в рамках системы реального времени.

5. Честность по отношению к задачам. Содержит метрику Dt_{wait} – дисперсию времени ожидания задач в очереди. Позволяет оценить степень равноправности задач с точки зрения алгоритма планирования.

6. Коммуникационный критерий. Включает следующие метрики: среднее суммарное расстояние $\overline{SD}$ между вычислительными ядрами, назначенными процессам параллельных задач, средняя дисперсность задач $\overline{JD}$ и среднее время блокировки сообщений $\overline{MB}$. Данный критерий позволяет оценить сетевую конкуренцию и степень фрагментации параллельных задач.

При исследовании различных алгоритмов планирования нас, прежде всего, интересует производительность составляемых ими расписаний. Поэтому при анализе алгоритмов, в первую очередь, будет рассматриваться первый критерий системы, остальные – имеют вторичный характер.

Анализ алгоритмов планирования по определенному критерию представляет собой сравнение для различных алгоритмов графиков зависимостей метрик этого критерия от величины системной загрузки L и выбор лучшего варианта для того или иного случая. При этом используются усредненные значения метрик на 100-1000 различных сгенерированных потоках параллельных задач.

Системная загрузка определяется как отношение суммарного объема вычислительной работы всех задач к объему работы, который кластер может потенциально выполнить (при полной загруженности) за время до появления в управляющей системе последней задачи:

$$L = \frac{\sum\limits_{j=1}^{m} n_j \tilde{\tau}_j}{\sum\limits_{i=1}^{n} S_i C_i \cdot \max\limits_{j=\overline{1,m}} \{a_j\}} \cdot 100\%. \tag{8}$$

Данная величина позволяет характеризовать вычислительную насыщенность потока работ, поступающих в управляющую систему вычислительного кластера.

Приведем формулы для вычисления основных метрик критерия производительности. Обозначим в качестве l количество моментов времени, когда в процессе моделирования наступают определенные события, $t_1, t_2, ..., t_l$ – сами моменты времени. При этом гарантируется, что никакие параметры модели и метрики не меняются на интервале времени (t_k, t_{k+1}) $\forall k = \overline{1, l-1}$. Будем также считать, что если мы вычисляем параметр модели или метрику в момент времени t_k, то это выполняется после того, как все события модели, назначенные на это время, были обработаны и выполнен цикл планирования задач. Обозначим $Q(t_k)$ – множество задач, находящихся в очереди планировщика в момент времени t_k.

Средняя загруженность вычислительных ядер узлов кластера может быть вычислена по формуле:

$$U_{cores} = \frac{\sum\limits_{i=1}^{n} \overline{u}_i}{n} \cdot 100\%, \tag{9}$$

в свою очередь средняя загруженность каждого отдельного узла определяется выражением

$$\overline{u}_i = \frac{\sum\limits_{k=1}^{l-1} u_i(t_k)(t_{k+1} - t_k)\delta_k}{\sum\limits_{k=1}^{l-1} (t_{k+1} - t_k)\delta_k}, \tag{10}$$

где

$$\delta_k = \begin{cases} 1, & |Q(t_k)| \neq 0, \\ 0, & \text{иначе.} \end{cases} \tag{11}$$

Величина потери производительности вычислительного кластера может быть вычислена следующим образом:

$$CL = \frac{\sum\limits_{k=1}^{l-1} \sum\limits_{i=1}^{n} S_i F_{ik} (t_{k+1} - t_k)\sigma_k}{\sum\limits_{k=1}^{l-1} (t_{k+1} - t_k)\sigma_k \cdot \sum\limits_{i=1}^{n} S_i C_i} \cdot 100\%, \tag{12}$$

где F_{ik} – количество свободных ядер на узле p_i в момент времени t_k, а σ_k определяется по формуле:

$$\sigma_k = \begin{cases} 1, & |Q(t_k)| \neq 0 \wedge \sum\limits_{i=1}^{n} F_{ik} < |\mathrm{X}|, \\ 0, & \text{иначе.} \end{cases} \tag{13}$$

Заметим, что формулы (9) и (12) соответственно не учитывают отсутствие загруженности ресурсов вычислительных узлов и потерю производительности в случае пустой очереди задач планировщика. Это позволяет более объективно судить о состоянии вычислительной системы.

5. Исследуемые алгоритмы планирования

К числу алгоритмов планирования, которые будут экспериментально исследованы с помощью симулятора вычислительного кластера и его управляющей системы, относятся сочетания алгоритмов выбора задачи из очереди Most Processors First Served Scan и Backfill со следующими методами назначения, учитывающими топологию вычислительной системы [1, 2, 4, 5, 6]:

1. Paging, Multiple Buddy System, Adaptive None-Contiguous Allocation, Greedy Available Busy List, Minimizing Contention, Minimizing Contention Incremental, Manhattan Median, NEP (для топологий торов);
2. Contiguous Allocation, Quasi-Contiguous Allocation (k-арные n-деревья);
3. алгоритмы сортировки узлов (толстые деревья, звезды);
4. модифицированный вариант алгоритма Minimizing Contention (для произвольных топологий).

А также со следующими методами назначения, не принимающими ее во внимание [3]: First Fit, Best Fit, Fastest Node Fist, Random First.

Заключение

В данной работе описывается симулятор вычислительного кластера, учитывающий его топологию, коммуникационные задержки при передаче данных, а также многопроцессорность вычислительных узлов. Он будет использован для исследования алгоритмов планирования задач.

Приведена имитационная схема и модель кластера, описывается модель его вычислительной загрузки, сформирована система критериев и метрик сравнения алгоритмов планирования. Также определен набор алгоритмов планирования, которые будут исследованы с помощью данного симулятора.

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010».

Работа проводилась при финансовой поддержке Федерального агентства по образованию в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг. (государственный контракт № П2039).

Литература

1. Moore, S.Q. The Effects of Network Contention on Processor Allocation Strategies / S.Q. Moore, M.N. Lionel // Proceedings of the 10th International Parallel Processing Symposium. – Washington, DC: EEE Computer Society, 1996. – P. 268 – 273.
2. Bani-Mohammad, S. An efficient processor allocation strategy that maintains a high degree of contiguity among processors in 2D mesh connected multicomputers / S. Bani-Mohammad,

M. Ould-Khaoua, I. Abaneh // Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, AICCSA. – 2007. – P. 934 – 941.

3. Полежаев, П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора / Полежаев, П.Н. // Параллельные вычислительные технологии (ПАВТ'2010): тр. междунар. конф. – Челябинск, 2010. – С. 287 – 298.
4. Bender, M.A. Communication-Aware Processor Allocation for Supercomputers / M.A. Bender, D.P. Bunde, E.D. Demaine // Lecture Notes in Computer Science. – 2005. – V. 3608/2005. – P. 169 – 181.
5. Cheng, C. Improving Performance of Mesh-connected Multicomputers by Reducing Fragmentation / C. Cheng, P. Mohapatra // J. of Parallel and Distributed Computing. – 1998. – V. 52(1). – P. 40 – 68.
6. Pascual, J.A. Effects of Topology-Aware Allocation Policies on Scheduling Performance / J.A. Pascual, J. Navaridas, J. Miguel-Alonso // Lecture Notes in Computer Science. – 2009. – V. 5798/2009. – P. 138 – 156.

Полежаев Петр Николаевич, аспирант, кафедра математического обеспечения информационных систем, Оренбургский государственный университет, peter.polezhaev@mail.ru.

Поступила в редакцию 20 мая 2010 г.

УДК 004.451

О ВОССТАНОВЛЕНИИ ПРОГРАММ ИЗ КОНТРОЛЬНЫХ ТОЧЕК

А.Ю. Поляков

ON PROGRAM RESTORATION FROM CHECKPOINTS SET

A.Y. Polyakov

В работе описаны два подхода к проблеме восстановления распределенных программ из контрольных точек. Предложен алгоритм восстановления взаимосвязей типа «родитель-потомок» и алгоритм принадлежности к группам и сеансам для набора процессов в рамках элементарной машины распределенной вычислительной системы. Предложен алгоритм координированного восстановления набора связанных процессов, перезапускаемых раздельно (на различных элементарных машинах или терминалах). Описанные подходы реализованы в системе создания контрольных точек *DMTCP* (*Distributed MultiThreaded CheckPointing*).

Ключевые слова: распределенные вычислительные системы, контрольные точки восстановления, отказоустойчивость

In paper two approaches to distributed programs restore problem from checkpoints set are described. Computation node wide algorithm of parent-child relationships and group/session assignement recreation at restore time is proposed. Also coordinated algorithm for process set restoration from several nodes/terminals is designed. Described algorightms are implemented in checkpointing package called *DMTCP* (*Distributed MultiThreaded CheckPointing*).

Keywords: HPC, rollback-recovery, checkpointing, fault tolerance

Введение

Распределенные вычислительные системы (ВС) – это важнейший вычислительный инструмент, который используется для проведения научных, инженерных и экономических расчетов [1]. Такие ВС являются большемасштабными, они состоят из сотен тысяч процессорных ядер и имеют производительность порядка *PetaFLOPS* [2]. Однако даже на таких высокопроизводительных системах многие современные задачи требуют для своего решения дни, недели и месяцы. Несмотря на высокий уровень развития элементной базы и схемотехники, аппаратные ресурсы распределенных ВС не являются абсолютно надежными. В связи с их большемасштабностью вероятность выхода из строя одной или нескольких составляющих становится достаточно высокой. Отказы процессоров, жестких дисков, сетевых адаптеров, кабелей и шин передачи данных могут повлечь за собой потерю значительного количества промежуточных вычислений, что приведет к снижению технико-экономической эффективности ВС. Таким образом, актуальной задачей является обеспечение отказоустойчивого выполнения программ на распределенных ВС.

Наиболее распространенным подходом к решению данной проблемы является создание контрольных точек (КТ) [3]. В процессе выполнения программы происходит периодическое сохранение ее состояния на надежный носитель данных. В случае отказа производится

«откат» к ближайшей доступной контрольной точке, и работа возобновляется. При этом теряется незначительное количество промежуточных вычислений.

В данной работе описаны два подхода к проблеме восстановления распределенных программ из контрольных точек. Предложен алгоритм координированного восстановления набора связанных процессов, перезапускаемых раздельно (на различных элементарных машинах или терминалах). Разработан алгоритм восстановления взаимосвязей типа «родитель-потомок» и алгоритм принадлежности к группам и сеансам для набора процессов в рамках элементарной машины (ЭМ) распределенной ВС. Данные алгоритмы реализованы в программном пакете создания КТ *DMTCP* (*Distributed MultiThreaded CheckPointing*) [4], который позволяет формировать КТ для последовательных, параллельных и распределенных программ в ОС *GNU\Linux*.

1. Классификация средств создания контрольных точек

Существует достаточно много средств создания КТ (ССКТ) [4 – 7], каждое из них имеет свои преимущества и недостатки. Рассмотрим несколько подходов к классификации ССКТ.

Существует две основные схемы взаимодействия ССКТ с защищаемой программой: *явная* и *прозрачная* (неявная). ССКТ, построенные на основе явной схемы, позволяют задать ограниченный набор информации, которую необходимо сохранить в КТ. Это позволяет снизить объем дискового ввода/вывода, т.е. значительно уменьшает накладные расходы таких ССКТ. Недостатком явной схемы является необходимость модификации исходного кода, что не позволяет применять ее к программам, доступным только в бинарном виде. Кроме того, КТ могут создаваться только в моменты времени, определяемые программой и связанные с завершенностью определенного периода вычислений.

ССКТ, построенные на основе прозрачной схемы, выполняют сохранение КТ незаметно для программы, что обеспечивает простоту и универсальность их использования. Недостатком этой схемы является больший объем дискового ввода/вывода, так как сохраняется все пространство памяти программы.

По классам поддерживаемых программ ССКТ можно разделить на *сосредоточенные* и *распределенные*. Сосредоточенные ССКТ обеспечивают отказоустойчивость выполнения одного или нескольких процессов в рамках вычислительного узла. Распределенные ССКТ обычно строятся на базе сосредоточенных и позволяют выполнять создание КТ для распределенных и параллельных программ, что делает их важным инструментом организации функционирования ВС. Для создания распределенной КТ (РКТ) необходимо:

1) создать сосредоточенные КТ для всех процессов, входящих в состав распределенной программы (РП);

2) сохранить граф связей между процессами РП;

3) сохранить сообщения, которые были отправлены, но не доставлены на момент создания РКТ (такие сообщения также называют *in-transit*).

Для распределенных ССКТ различают *координированный* и *некоординированный* подходы. При создании РКТ каждый процесс РП сохраняет свое состояние в КТ. Целостной РКТ [3] называется набор из N локальных КТ, формирующих допустимое состояние программы. Такая РКТ может быть использована для восстановления программы после сбоя. При координированном подходе создание КТ происходит синхронно, что гарантирует целостность РКТ. При некоординированном подходе каждый процесс создает КТ независимо от других. Следовательно, при восстановлении необходимо выполнять поиск целостного состояния программы на основе набора независимых КТ, что вносит дополнительные накладные расходы. Для некоординированного подхода существует опасность возникновения «эффекта домино»,

когда в процессе поиска целостного состояния происходит откат к начальному состоянию программы.

Распределенные ССКТ также можно разделить на *универсальные* и *MPI-ориентированные*. Первые позволяют создавать РКТ для любых распределенных и параллельных программ, в том числе для различных реализаций модели передачи сообщений (*PVM, MPI*). Что касается вторых, то существует несколько ССКТ, построенных на базе конкретных реализаций *MPI*. Например, *OpenMPI* [8], *MVAPICH2* [9], *LAM-MPI* [10]. Все они используют ССКТ *BLCR* [5] для создания сосредоточенных КТ и реализуют собственные механизмы сохранения графа связей и транзитных сообщений.

Для создания КТ сосредоточенного процесса необходимо сохранить информацию о его состоянии. Это может быть реализовано на различных программных уровнях:

1. **Уровень операционной системы (ОС)**. Предусматривает сохранение содержимого пространства ядра и пространства пользователя для всех процессов ОС. Такой подход подразумевает использование систем виртуализации, например, *VMWare*.
2. **Уровень ядра ОС**. Предусматривает внедрение дополнительных компонентов, позволяющих сохранить необходимую информацию: содержимое памяти конкретного процесса и состояние ядра, относящееся к нему.
3. **Уровень системных библиотек**. Предусматривает сохранение содержимого памяти и состояния ядра с использованием средств, предоставляемых ОС для управления процессами.
4. **Прикладной уровень**. Предусматривает сохранение минимального объема информации, необходимого для восстановления каждой конкретной программы.

Средства создания КТ уровней ядра ОС, ядра и системных библиотек реализуются в рамках прозрачной схемы. Кроме того, некоторые ССКТ уровней ядра и системных библиотек предоставляют программе возможность влиять на процесс обеспечения отказоустойчивости, например, выбирать наиболее удобные моменты для создания КТ. Прикладной уровень предусматривает только явную схему.

Преимуществом первого уровня является простота реализации, а недостатком – значительный объем дискового ввода/вывода и отсутствие гибкости. Второй уровень позволяет получать прямой доступ к внутренним структурам ядра и памяти процесса и выполнять сохранение необходимой для восстановления информации при меньшем объеме ввода/вывода. Недостатком данного подхода является зависимость от изменений в ядре ОС (новые версии ядра ОС *GNU\Linux* выходят в среднем с частотой раз в 3 – 4 месяца). Также данный подход требует привилегий суперпользователя для установки и управления, а ошибки, допущенные в программном обеспечении уровня ядра, приводят к нарушению работы всей ОС.

Третий уровень позволяет обеспечить создание КТ, не требуя при этом привилегий суперпользователя и не подвергая угрозе функционирование всей ОС. Однако при данном подходе невозможно осуществить прямой доступ к внутренним структурам ядра, которые описывают защищаемый процесс. Для этого требуется перехват и обработка системных вызовов.

На четвертом уровне сохраняется лишь содержимое буферов, которые явно указываются в программе.

2. Distributed MultiThreaded Checkpointing – DMTCP

Программный пакет *DMTCP* реализован на уровне системных библиотек и является универсальной координированной распределенной ССКТ. DMTCP разработан в Северо-

восточном университете (*Northeastern University*) США под руководством профессора Дж. Купермана.

Наиболее распространенной сосредоточенной ССКТ на данный момент является пакет *BLCR*. Кроме того, как было отмечено ранее, он используется во многих распределенных *MPI*-ориентированных ССКТ. Таблица отражает сравнение ССКТ *DMTCP* и *BLCR* по поддерживаемым функциям ОС. *BLCR* используется для создания сосредоточенных КТ в нескольких *MPI*-ориентированных распределенных ССКТ.

Таблица

Функции, поддерживаемые ССКТ

Поддерживаемые компоненты ОС	DMTCP		BLCR	
	Полностью	Частично	Полностью	Частично
Обработка сигналов	X		X	
Сокеты	X		–	–
Многопоточные приложения	X		X	
Идентификаторы ресурсов ОС (процессы, группы, сессии)		**X**	**X**	
Именованные и неименованные каналы	X		X	
Открытые файлы	X		X	
Отображенные (mapped) файлы	X		X	
/proc файлы	X		X	
Статически скомпилированные программы	–	–		**X**
Отлаживаемые программы	X		–	–

Из таблицы видно, что *DMTCP* уступает *BLCR* по двум параметрам. Во-первых, нет поддержки статически скомпилированных программ, т.к. для перехвата системных вызовов используется «предзагрузка» служебной динамической библиотеки *dmtcphijack.so*. Однако данный пункт не полностью поддерживается и в *BLCR*. Во-вторых, отсутствует восстановление идентификаторов ресурсов ОС, таких как идентификаторы групп и сессий. В пространстве ядра в связи с прямым доступом к его внутренним структурам данная задача является более простой. В *DMTCP* (на уровне системных библиотек) была реализована частичная виртуализация идентификаторов процессов. В данной работе предложен алгоритм, позволяющий более полно восстанавливать идентификационную информацию. Он был интегрирован и используется в *DMTCP* в настоящее время.

На рис. 1 показан запуск программы с применением *DMTCP*. В процессе ее работы автоматически осуществляется контроль над созданием новых процессов с использованием системного вызова *fork()*.

```
host1$ dmtcp_checkpoint ./program1
```

Рис. 1. Запуск программы *program1* под управлением *DMTCP* на узле *host1*

Как было сказано ранее, *DMTCP* реализует координированное создание КТ. На каждую вычислительную группу создается один координатор (*dmtcp_coordinator*). Он может быть запущен явно, как показано на рис. 2а. Если при запуске программы (рис. 1) процесс координатор не обнаружен, то он будет запущен автоматически.

```
host1$ dmtcp_checkpoint ./program2
```
а)

```
host1$ dmtcp_checkpoint ./program2
```
б)

```
host2$ DMTCP_HOST=host1 dmtcp_checkpoint ./program3
```
в)

Рис. 2. Использование DMTCP. а) Запуск процесса-координатора на узле *host1*, б) Запуск программы *program2* под управлением *DMTCP* на узле *host1*, в) Запуск программы *program3* под управлением *DMTCP* на узле *host2*

Возможно создание РКТ для нескольких взаимодействующих программ, запускаемых с разных терминалов. Например, как показано на рис. 1 и 2б. В этом случае вспомогательный модуль *DMTCP*, интегрированный в каждую из программ, выполнит соединение с координатором.

Также возможно создание РКТ для процессов, работающих на разных узлах сети. Для этого необходимо указать через переменную окружения *DMTCP_HOST* адрес узла, на котором выполняется координатор. Так, на рис. 2в показан запуск программы *program3*, которая подключается к вычислительному процессу, уже содержащему программы *program1* и *program2*.

Создание РКТ происходит следующим образом: каждый процесс сохраняет свое состояние в отдельном файле, а координатор формирует *shell*-скрипт, содержащий последовательность действий, необходимых для запуска вычислений из данной РКТ.

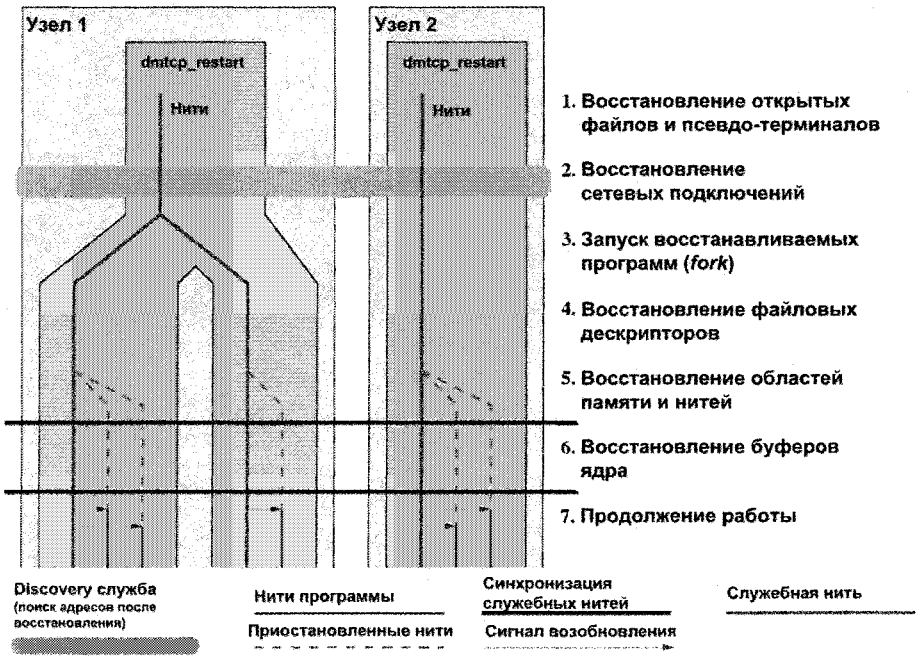


Рис. 3. Восстановление вычислений из РКТ DMTCP

Как показано на рис. 3, на этапе восстановления на каждом узле запускается один служебный процесс, использующий РКТ для воссоздания компонентов программы (этап 3). Для синхронизации узлов используется этап восстановления сокетов (этап 2).

Недостаток данной схемы заключается в том, что процессы, выполнявшиеся на разных терминалах, будут перезапущены уже на одном. Например, *DMTCP* используется в качестве основы для универсального реверсивного отладчика *URDB* [11]. Типичным сценарием

применения URDB является подключение (*attach*) к уже выполняющейся программе и ее отладка. При восстановлении такой отладочной сессии необходимо сохранить принадлежность к разным терминалам, однако отсутствие средств синхронизации не позволяет этого сделать.

Рассмотрим другой пример: восстановление из контрольной точки группы процессов, распределенных по разным узлам сети. Процессы разбиты на подгруппы, не связанные между собой постоянными сетевыми соединениями. В этом случае барьер, образованный этапом 2 (*Recreate and reconnect sockets*), не является достаточным для синхронизации. Если одна подгруппа была запущена значительно раньше остальных, ее выполнение будет продолжено, а остальные подгруппы не будут иметь возможности возобновить работу.

Для устранения указанных недостатков был предложен дополнительный компонент схемы синхронизации, который представлен в данной работе.

3. Дополнительные компоненты схемы синхронизации

В *DMTCP* предусмотрен барьер, позволяющий синхронизировать восстановление из РКТ только для процессов, связанных постоянными сетевыми соединениями. Как было показано в разделе 3, существуют программы, для которых это условие не выполняется. Для устранения этого недостатка было предложено расширение существующей схемы синхронизации, которое будет рассмотрено далее.

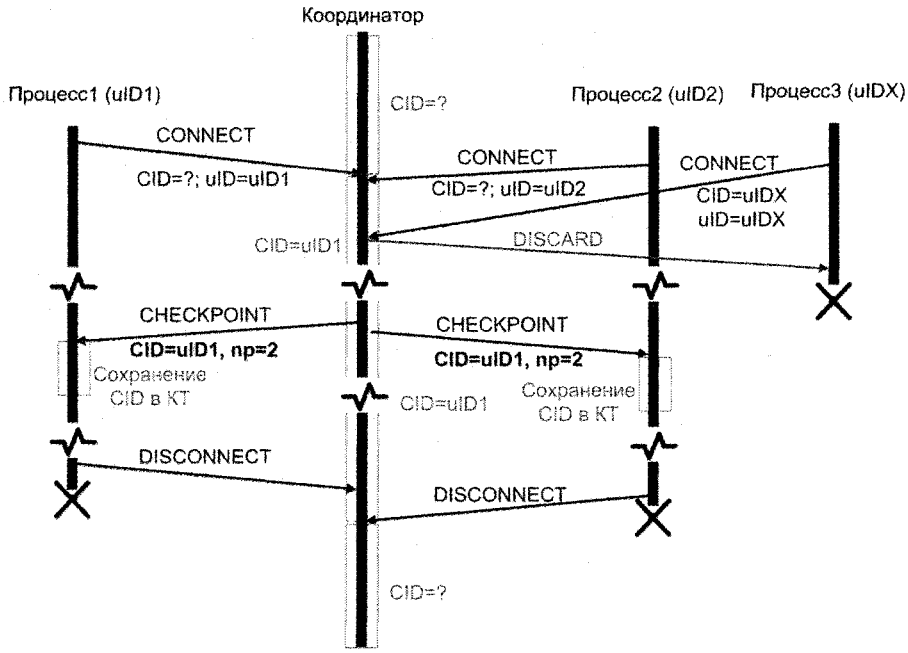


Рис. 4. Создание РКТ

Каждый процесс в DMTCP имеет уникальный идентификатор *uID (unique ID)*, который формируется из трех компонент: *<хеш-код имени сетевого узла>-<PID>-<временная метка>*. Координатор играет роль службы, предоставляющей сервис синхронизации. Его состояние подстраивается под выполняемые задачи и не сохраняется в РКТ. В качестве синхронизационного условия выбрано число процессов, принадлежащих вычислительной группе (ВГ) на момент создания контрольной точки. Как показано на рис. 4, для идентификации ВГ (*CID – computational group ID*) используется *uID* процесса, который выполнил подключение первым. Если приходит запрос на подключение от другой

ВГ (процесс 3 на рис. 4), то оно отклоняется. На этапе создания РКТ координатор рассылает *CID* текущей ВГ и число ее участников (*np – number of process*). Эта информация сохраняется в каждой локальной КТ. При отключении последнего процесса из текущей ВГ координатор переходит в состояние *CID=?* и готов принимать новые запросы на услуги синхронизации от других ВГ. На этапе восстановления (рис. 5) процесс считывает *CID* и *np* из КТ и отправляет координатору при подключении. Если координатор не занят обслуживанием других заявок, он устанавливает параметр *CID* в значение, которое содержится в сообщений. Также запоминается количество клиентов, которое должно выполнить подключение до того, как можно будет продолжить вычислительный процесс. Если подключение выполняет клиент, не имеющий *CID* или имеющий *CID*, который отличается от текущего, то такое соединение отклоняется.

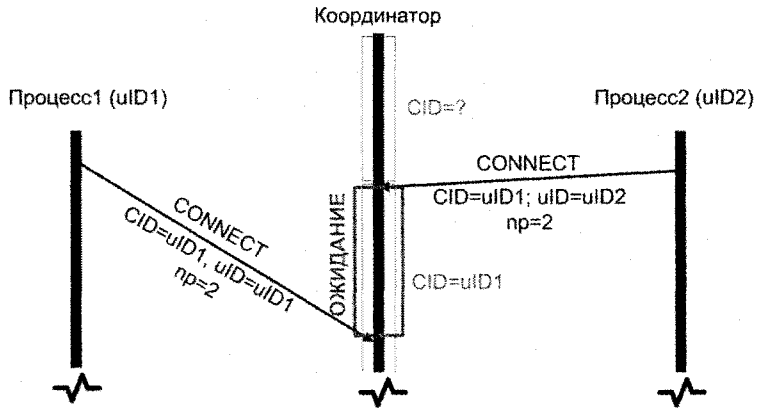


Рис. 5. Восстановление из РКТ

4. Алгоритм восстановления идентификационной информации

Как было сказано ранее, восстановление идентификационной информации на уровне системных библиотек затруднено отсутствием прямого доступа к внутренним структурам ядра ОС GNU\Linux. Для решения данной проблемы автором предложен алгоритм восстановления идентификационной информации, который имитирует процесс первоначального запуска набора процессов.

4.1. Идентификационные ресурсы

В ОС *GNU\Linux* процесс описывается набором идентификаторов. Первый из них – идентификатор процесса *PID (process ID)*. *PID* назначается при создании системными вызовами *fork()* или *vfork()* и используется для того, чтобы указать на процесс в ряде важных системных вызовов, таких как *kill(), ptrace(), setpriority(), waitpid()*. Отношение родитель-потомок строится на основе *PID*. Процесс, выполнивший системный вызов fork, становится родителем созданного процесса. Для доступа к информации об идентификаторе родителя (*parent PID – PPID*) используется системный вызов *getppid()*. Если процесс завершается, а потомки продолжают существование, их родителем становится системный процесс *init*, имеющий *PID=1*. Каждый процесс принадлежит к одной и только одной сессии, для создания новой используется системный вызов *setsid()*. Идентификатор сессии *SID* (*session ID*) равен идентификатору процесса-создателя (или лидера). Принадлежность к сессии наследуется потомком от родителя. Каждый процесс принадлежит к одной и только одной группе. Если его идентификатор совпадает с идентификатором группы, то он называется ее

лидером. Все процессы группы принадлежат одной и только одной сессии. Данные механизмы используются командными интерпретаторами при организации конвейеров, некоторыми отладчиками для управления отлаживаемыми программами и т.д.

4.2. Постановка задачи

Пусть имеется множество контрольных точек $C = \{c_i\}, i = 1...N$, каждая из которых однозначно соответствует восстанавливаемому процессу $p_i \in P$. КТ описывается четырьмя параметрами $c_i = (pid_i, ppid_i, sid_i, img_i)$, где: pid_i – уникальный идентификатор $(\forall i, j = 1...N, i \neq j, pid_i \neq pid_j)$ в рамках ЭМ ВС; $ppid_i$ – идентификатор процесса, создавшего p_i через системный вызов *fork()*; sid_i – идентификатор сессии, если $pid_i = sid_i$, то КТ c_i содержит процесс-лидер сессии sid_i; img_i – сохраненное состояние процесса, необходимое для его перезапуска. Обозначим через $s = \bigcup_{i=1}^{N} sid_i$ множество уникальных идентификаторов сеансов, к которым принадлежат процессы из P. Пусть $S = \{S_k\}, k = 1...|s|$ – множество сеансов, где $S_k = \{c_i | c_i \in C, sid_i = s_k\}$ – подмножество КТ, содержащих процессы одного сеанса. Очевидно, что $C = \bigcup_{S_k \in S} S_k$ и $\forall k_1, k_2 = 1...|s|, k_1 \neq k_2, S_{k_1} \cap S_{k_2} = \emptyset$. Пусть также определено множество $R = \{c_i | \forall j = 1...N, j \neq i, ppid_i \neq pid_j\}$ независимых КТ.

Требуется, используя программный интерфейс ОС *GNU\Linux*, выполнить запуск процессов из контрольных точек так, чтобы восстановить их исходную иерархию и принадлежность к сеансам. При этом для изменения сеанса имеется только системный вызов *setsid()*, который позволяет процессу создать собственный сеанс, в котором он становится лидером.

4.3. Алгоритм восстановления иерархии процессов и сеансов

На вход алгоритма подается множество контрольных точек C. Алгоритм состоит из следующих шагов:

4.3.1. Формирование отношений типа родитель-потомок

Строится лес деревьев $T = \{T_i\}, l = 1...|R|$ (рис. 6), для которого определена функция однозначного соответствия f между КТ и узлами деревьев леса: $f = \{(t, c) | \exists l = 1...|R|, t \in T_l, c \in C, t \Leftrightarrow c\}$. Справедливы следующие утверждения:

1. $\forall T_l \in T$, если t-корень T_l, то $f(t) \in R$
2. $\forall T_l \in T, \forall t_q, t_2 \in T_l$, t_1 – непосредственный потомок $t_2 \Leftrightarrow \exists i, j : c_i = f(t_1), c_j = f(t_2)$ и $pid_i = ppid_j$

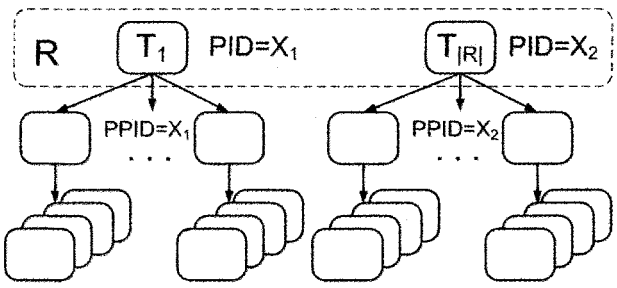


Рис. 6. Лес деревьев T, соответствующий восстанавливаемым КТ

4.3.2. Построение метаинформации

Для того, чтобы восстанавливать принадлежность к одной сессии процессов, которым соответствуют узлы различных деревьев, выполняется построение метаинформации. Для каждого дерева $T_l \in T$ выполняется его обход в глубину. Для каждого обрабатываемого узла $t \in T_l$ определяется номер соответствующей ему КТ $i : c_i = f(t)$. Строится метаинформация, которая представляется в виде пары (x_k^i, y_k^i), где $x_k^i \in x^i$,

$$x^i = \{x_k^i | x_k^i = sid_j, j = 1...|N|, f^{-1}(c_j) \in SubTree(t)\},$$

$$y_k^i = \begin{cases} 1, & \exists c_j \in C : f_{-1}(c_j) \in SubTree(t), \\ 0, & \text{иначе.} \end{cases}$$

Второй компонент пары (y_k^i) указывает на наличие или отсутствие лидера сессии с идентификатором x_k^i. На рис. 7 показан пример сбора метаинформации. Рассмотрим узел X, которому соответствует метаинформация, состоящая из одной пары $(SID3, 0)$. X является листовым и не является лидером $SID3$. Корень дерева (узел Y) содержит метаинформацию из четырех пар: $(SID1, 1), (SID2, 1), (SID3, 1), (SID4, 1)$. Это означает, что на текущем и нижележащих уровнях дерева имеется четыре сессии с идентификаторами $SID1, SID2, SID3, SID4$ для каждой из них были найдены лидеры.

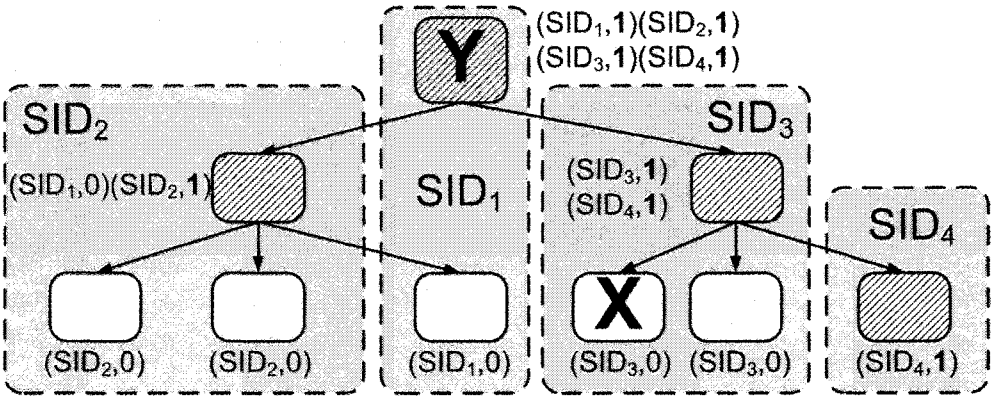


Рис. 7. Построение метаинформации

4.3.3. Построение зависимостей между элементами T

Каждому узлу $t \in T$ ставится в соответствие множество $D(t) = \emptyset$. Далее выполняется обработка метаинформации, соответствующей корням деревьев из леса T. Пусть $T_{M_1}, T_{M_2} \in T$, как показано на рис. 8. Пусть t_1 – корень T_{M_1}, t_2 – корень T_{M_2}, i, j – соответствующие индексы КТ: $c_i = f(t_1), c_j = f(t_2)$. Тогда T_{M_2} зависит от T_{M_1}, если $\exists k_1, k_2 : x_{k_1}^i = x_{k_2}^j \vee y_{k_2}^j = 1 \vee y_{k_1}^i = 0$. В этом случае выполняется поиск лидера сессии $x_{k_1}^i$ – $\tilde{t} = f^{-1}(c_l) : pid_l = sid_l = x_{k_1}^i$ и модифицируется соответствующее множество $D(\tilde{t}) = D(\tilde{t}) \cup \{t_2\}$.

На рис. 8 показано, что метаинформация корня дерева T_{M_2} указывает на отсутствие лидера сессии SID_k в T_{M_2}. Пара $(SID_k, 1)$, соответствующая корню дерева T_{M_1}, указывает на наличие узла $\tilde{t}$ и контрольной точки $\tilde{c} = f(\tilde{t})$, содержащей лидера SID_k. Тогда считаем, что дерево T_{M_2} зависит от узла $\tilde{t}$. То есть при запуске процессов из КТ сначала должна быть восстановлена КТ, соответствующая $\tilde{c}$, и создана новая сессия SID_k, а после этого восстановлено дерево T_{M_2}.

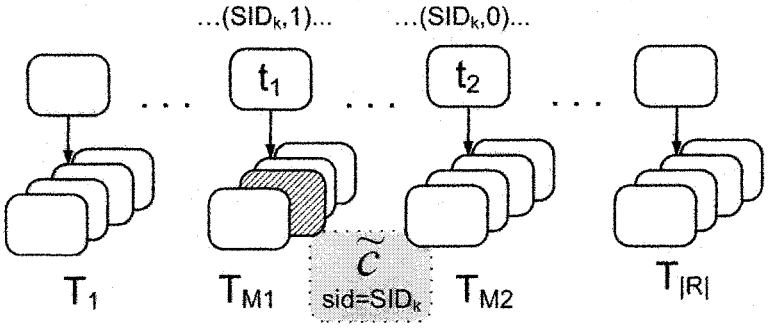


Рис. 8. Построение зависимостей между деревьями леса T

4.3.4. Запуск процессов из КТ

Для всех независимых деревьев ($\forall T_l \in T : \tilde{t} = root(T_l) \vee D(\tilde{t}) = \emptyset$) выполняется восстановление исходной структуры процессов с использованием *fork()* и *setsid()* по алгоритму, приведенному ниже.

procedure $restore(t, psid)$

1: $i \leftarrow k : (c_k == f(t))$
2: **if** $pid_i \neq sid_i$ **then**
3: **for** $t_1 \leftarrow childs(t)$ **do**
4: $fork() \vee restore(f(t1), psid)$
5: **end for**
6: $start(c_i)$
7: **else**
8: **for** $t_1 \leftarrow childs(t)$ **do**
9: $j \leftarrow k : (c_k == f(t_1))$
10: **if** $sid_j \neq pid_i$ **then**
11: **if** $fork() = 0$ **then**
12: $restore(f(t1), psid)$
13: **end if**
14: **end if**
15: **end for**
16: $psid = setsid()$
17: **for** $t \leftarrow D(t)$ **do**
18: **if** $fork() = 0$ **then**
19: **if** $fork() = 0$ **then**
20: $restore(f(t1), psid)$
21: **else**
22: $exit(0)$
23: **end if**
24: **end if**
25: **end for**
26: **for** $t_1 \leftarrow childs(t)$ **do**
27: $j \leftarrow k : (c_k == f(t_1))$
28: **if** $sid_j = pid_i$ **then**
29: **if** fork() $= 0$ **then**
30: $restore(f(t_1), psid)$
31: **end if**

32: **end if**
33: **end for**
34: $start(f(t))$
35: **end if**

5. Экспериментальные данные

На рис. 9 показаны структуры двух программ, для которых были созданы КТ с применением ССКТ DMTCP.

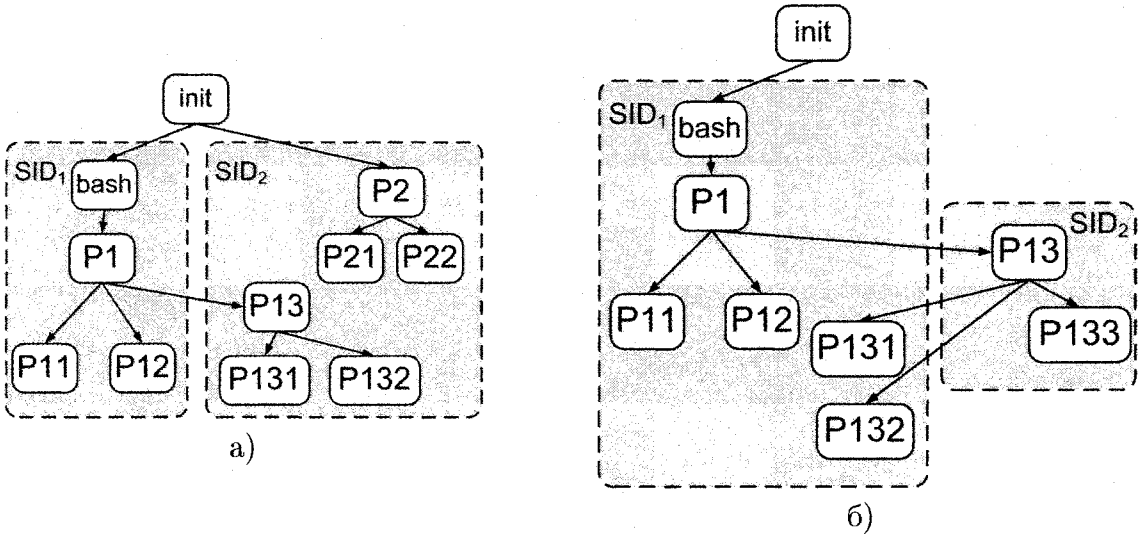


Рис. 9. Структуры тестовых программ

На рис. 10 показаны отношения между восстановленными процессами без применения разработанного алгоритма. Как видно из рис. 10а, все процессы принадлежат одной сессии, а процесс *p2*, который должен быть потомком *init(PID=1)*, является потомком *p1*. На рис. 10б мы также видим, что принадлежность к сессиям не восстанавливается.

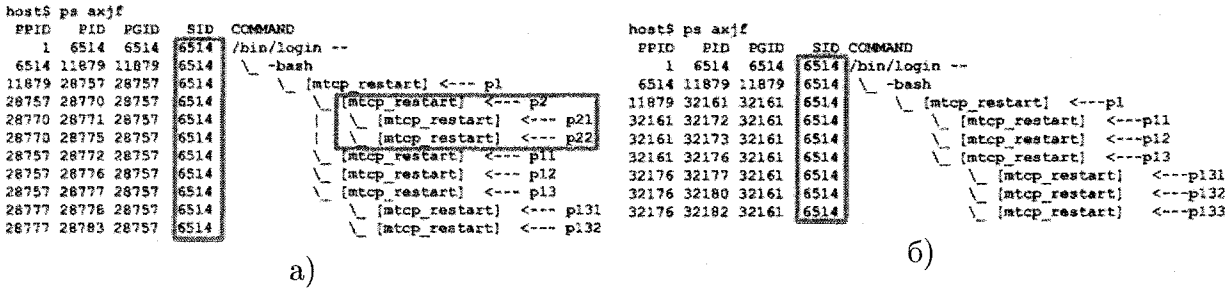
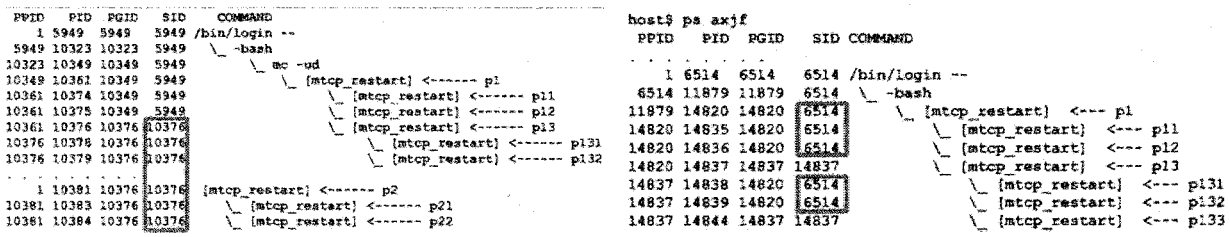


Рис. 10. Восстановление с неполным учетом идентификационной информации

На рис. 11а, 11б приведены результаты восстановления из КТ с применением предложенного алгоритма для соответствующих программ рис. 9. Рассмотрим более подробно рис. 11а. Очевидно, что процессы *p13, p131, p132, p2, p21, p22* принадлежат одной сессии, лидером которой является *p13*, как это и должно быть. Остальные процессы принадлежат сессии командного интерпретатора. Элемент p2 восстановлен, как наследник *init (PID=1)*. На рис. 11б процессы *p13, p133* находятся в новой сессии, в которой *p13* является лидером. При этом потомки *p13 – p131* и *p132* остались в сессии командного интерпретатора, как это было в исходной программе.



а) б)

Рис. 11. Полное восстановление идентификационной информации

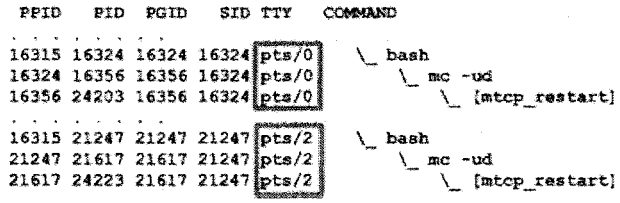


Рис. 12. Восстановление процессов на разных терминалах

На рис. 12 показано восстановление тестовой программы. Она состоит из двух процессов, взаимодействующих через механизмы *IPC*. Запуск компонентов выполнен с двух различных терминалов: *pts/0* и *pts/2*. До реализации предложенного расширения схемы синхронизации подобное восстановление исходными средствами *DMTCP* было невозможно.

Заключение

В работе были рассмотрены подходы к восстановлению программ из распределенных контрольных точек, реализованные в ССКТ *DMTCP*. Предложен алгоритм восстановления отношений родитель-потомок и алгоритм принадлежности к сессиям и группам с использованием стандартного механизма системных вызовов ОС *GNU\Linux*. Так как *DMTCP* реализован на уровне системных библиотек, он не имеет прямого доступа к внутренним структурам ядра. Следовательно, для восстановления указанных отношений между процессами требуется имитация основных шагов их запуска. Для этого выполняется построение древовидной структуры, отражающей родственные отношения между узлами. Далее происходит сбор метаинформации, содержащей описание принадлежности к сессиям. Разработанный алгоритм позволяет расширить диапазон программ, поддерживаемых *DMTCP*. Расширена схема синхронизации, используемая в *DMTCP*: добавлен новый барьер, позволяющий выполнять восстановление процессов из РКТ на различных терминалах, расположенных на одном или разных узлах сети. Это позволяет использовать *DMTCP* для восстановления из РКТ программ, которые не связаны постоянными сетевыми соединениями. Предложенная доработка также необходима для организации реверсивной отладки, построенной на базе *DMTCP*[11].

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010».

Работа проводилась при финансовой поддержке РФФИ (гранты 08-07-00018, 08-07-00022, 08-08-00300, 09-07-00185, 09-07-12016, 09-07-13534, 09-07-90403)

Литература

1. Хорошевский, В.Г. Архитектура вычислительных систем / В.Г. Хорошевский. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. TOP500 supercomputer site [Электронный ресурс].- Режим доступа: http://www.top500.org/ . – Загл. с экрана. – яз. англ.
3. A survey of rollback-recovery protocols in message-passing systems / E.N. Elnozahy, L. Alvisi, Y.M. Wang, D.B. Johnson // ACM Computing Surveys. – 2002. – V. 34, № 3. – P. 375 – 408.
4. Ansel, J. DMTCP: Transparent Checkpointing for Cluster Computations and the Desktop / J. Ansel, K. Arya, G. Cooperman // Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS'09). – Rome, 2009. – P. 1 – 12. – ISBN: 978-1-4244-3751-1.
5. Hargrove, P.H. Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters / P.H. Hargrove, J.C. Duell // In Proceedings of SCIENTIFIC DISCOVERY THROUGH ADVANCED COMPUTING (SciDAC 2006). – Denver, 2006. – V. 46. – P. 494 – 499. – ISSN 1742-6588.
6. Checkpoint and migration of UNIX processes in the Condor distributed processing system / M. Litzkow, T. Tannenbaum, J. Basney, M. Livny // Technical report 1346, University of Wisconsin, Madison. – Wisconsin, 1997. – P. 8.
7. Libckpt: Transparent checkpointing under Unix / J.S. Plank, M. Beck, G. Kingsley, K. Li // In Proc. of the USENIX Winter 1995 Technical Conference. – New Orleans, 1995. – P. 213 – 323.
8. The design and implementation of checkpoint/restart process fault tolerance for Open MPI / J. Hursey, J. M. Squyres, T. I. Mattox, A. Lumsdaine // In Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE Computer Society. – Long Beach, 2007. – P. 1 – 8. – ISBN: 1-4244-0910-1.
9. Application-transparent checkpoint/restart for MPI programs over InfiniBand / Q. Gao, W. Yu, W. Huang, D. K. Panda // Proceedings of the 2006 International Conference on Parallel Processing / IEEE Computer Society. – Washington, 2006. – P. 471 – 478.
10. FT-MPI, Fault-Tolerant Metacomputing and Generic Name Services: A Case Study / D. Dewolfs, J. Broeckhove, V. Sunderam, G. Fagg // Lecture Notes in Computer Science, Springer Berlin. – Heidelberg, 2006. – P. 133 – 140.
11. Temporal Debugging using URDB / A.M. Visan, A. Polyakov, P.S. Solanki, K. Arya, T. Denniston, G. Cooperman // 2009. – Режим доступа: http://arxiv.org/abs/0910.5046v1.

Поляков Артем Юрьевич, лаборатория вычислительных систем, институт физики полупроводников им. А.В. Ржанова СО РАН, artpol84@gmail.com.

Поступила в редакцию 16 апреля 2010 г.

УДК 519.6, 517.9

ПОЛУЛОКАЛЬНЫЕ СГЛАЖИВАЮЩИЕ СПЛАЙНЫ СЕДЬМОЙ СТЕПЕНИ

Д.А. Силаев, Ж.Г. Ингтем

SEMILOCAL SMOOTHIHG SPLINES OF SEVENTH DEGREE

D.A. Silaev, J.G. Ingtem

Полулокальные сглаживающие сплайны или S-сплайны были введены Д.А. Силаевым. Ранее рассматривались и применялись сплайны 3-й и 5-й степени. Настоящая работа посвящена построению сплайнов 7-й степени, доказаны теоремы существования и единственности, установлены условия устойчивости таких сплайнов.

Ключевые слова: аппроксимация, сплайн, численные методы

Semilocal smoothing splines or S-splines were initiated by D.A. Silaev. Earlier there were considers and applied the splines of third and fifth degree. This work is devoted to seventh degree splines construction. Uniqueness and existence theorems are proved. Stability and convergence conditions for these splines are established.

Keywords: an approximation, a spline, numerical methods

Введение

Рассматривается задача восстановления функции с помощью полулокального сглаживающего сплайна или S-сплайна, состоящего из полиномов седьмой степени. Здесь будут рассмотрены S-сплайны разных порядков гладкости: классов C^0, C^1, C^2, C^3 и C^4.

Заданные значения приближаемой функции разбиваются на несколько групп, каждая группа содержит определенное количество последовательных значений функции. Условия гладкой склейки определяют коэффициенты при младших степенях полинома в каждой группе. Коэффициенты полинома при старших степенях определяются методом наименьших квадратов по соответствующей группе. Начальные условия задаются значениями функции и её производных в начальной точке в непериодическом случае, либо условием периодичности сплайна на отрезке определения.

1. Построение S-сплайна седьмой степени

Рассмотрим на отрезке $[a,b]$ равномерную сетку $\{x_k\}_{k=0}^{k=K}$ с узлами $x_k = a + kh$, $k = 0,1,\ldots,K$, где $h = \dfrac{b-a}{K}$ —шаг сетки. Разобьем отрезок $[a,b]$ на группы. Для этого введем еще одну равномерную сетку $\{\xi_l\}_{l=0}^{l=L}$ с узлами $\xi_l = a+lH$, где $H = mh$, $m \in \mathbb{N}$, $l = 0,1,\ldots,L$. Таким образом, при переходе из одной группы в другую, будем осуществлять сдвиг системы координат и рассматривать каждый l–й полином на отрезке $[0,H]$.

Пусть $y = (y_0, y_1, \ldots, y_K) \in \mathbb{R}^{K+1}$ значения приближаемой функции на сетке $\{x_k\}_{k=0}^{k=K}$.

Рассмотрим случаи I – V, когда S-сплайн принадлежит классу C^0, C^1, C^2, C^3,C^4.

В дальнейшем будем рассматривать случай III, а именно, будем строить дважды непрерывно дифференцируемый полулокальный S-сплайн (построение остальных сплайнов будут производиться аналогичным образом).

Обозначим через

$$P_S^7 = \{u : u(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7\}$$

множество полиномов 7–й степени с фиксированными коэффициентами a_0, a_1, a_2 .(Здесь зафиксированы только три коэффициента, так как наш сплайн класса C^2. В случаях I, II, IV и V фиксированными будут соответственно коэффициенты a_0; a_0 и a_1; a_0, a_1, a_2 и a_3; a_0, a_1, a_2, a_3 и a_4.)

Рассмотрим функционал $\Phi^l(u) = \sum\limits_{k=0}^{M} \big(u(\xi_l + kh) - y_{ml+k}\big)^2$. Будем искать в P_S^7 такой полином g_l, который минимизирует функционал Φ^l

$$\Phi^l(u) = \sum_{k=0}^{M}(u(\xi_l + kh) - y_{ml+k})^2 \to \min(a_3, a_4, a_5, a_6, a_7)$$

и удовлетворяет условиям гладкой склейки:

$$g_l(0) = a_0^l = g_{l-1}(H), \qquad g_l'(0) = a_1^l = g_{l-1}'(H), \qquad \frac{g_l''(0)}{2} = a_2^l = \frac{g_{l-1}''(H)}{2}, \tag{1}$$

причем $g_0(0) = a_0^0 = g_{L-1}(H)$, $g_0'(0) = a_1^0 = g_{L-1}'(H)$, $\dfrac{g_0''(0)}{2} = a_2^l = \dfrac{g_{L-1}''(H)}{2}$ в периодическом случае.

В непериодическом случае задаются начальные условия [1] : $a_0^0 = y_0$, $a_1^0 = y_0'$, $a_2^0 = \dfrac{y_0''}{2}$.

Здесь $M + 1$ – количество точек осреднения, т.е. необходимые точки для построения полинома g_l, а $m+1$ – количество точек, входящих в область определения полинома g_l. Будем предполагать, что значения заданной функции y_k, $k = 0, 1, \ldots, K$ известны с некоторой погрешностью и с уменьшением шага точность измерения будет увеличиваться, то есть если функция $f \in C^8[a, b]$ задана на сетке $\{x_k\}_{k=0}^{k=K}$ значениями y_k, то $|y_k - f(x_k)| \leqslant Ch^8$.

Определение 1. *S-сплайном седьмой степени будем называть функцию $S_{m,M}^7$, которая совпадает с полиномом седьмой степени g_l на отрезке $[\xi_l, \xi_{l+1}]$.*

[1] Если функция задана таблицей, то $y_0', y_0'', \ldots, y_0^{(p)}$ можно вычислить с помощью формул численного дифференцирования высокого порядка аппроксимации, например

$$y_0^{(r)} = \left.\frac{d^{(r)}N_n(x)}{dx^r}\right|_{x=0} + O(h^{n+1-r}) \text{ при } r = 1, \ldots, p,$$

где $N_n(x)$– интерполяционный полином степени n, построенный по значениям $y_0, y_1, \ldots, y_n$. В форме Ньютона этот полином имеет вид: $N_n(x) = y_0 + \sum_{s=1}^{n} P_s(y_0, y_1, \ldots, y_s)x(x-h)\ldots(x-(s-1)h)$, где $P_s(y_0, y_1, \ldots, y_s) = \sum_{j=0}^{s} C_s^j y_{s-j}/(s!h^s)$ – s-я разделенная разность.

При $n = 8$ мы получаем формулы вида:

$$y_0' = -\frac{1}{840h}[2283y_0 - 6720y_1 + 11760y_2 - 15680y_3 + 14700y_4 - \\ -9408y_5 + 3920y_6 - 960y_7 + 105y_8] + O(h^8),$$

$$y_0'' = \frac{1}{5040h^2}[29531y_0 - 138528y_1 + 312984y_2 - 448672y_3 + \\ +435330y_4 - 284256y_5 + 120008y_6 - 29664y_7 + 3267y_8] + O(h^7).$$

Будем минимизировать функционал Φ^l по коэффициентам a_3, a_4, a_5, a_6, a_7 (в случаях I, II, IV и V минимизируем по коэффициентам $a_1,\dots,a_7$; $a_2,\dots,a_7$; $a_4,\dots,a_7$ и $a_5,\dots,a_7$ соответственно).

Для этого продифференцируем Φ^l_g по этим коэффициентам и приравняем результат к нулю. Получим:

$$\begin{cases} a_3^l h^3 S_6 + a_4^l h^4 S_7 + a_5^l h^5 S_8 + a_6^l h^6 S_9 + a_7^l h^7 S_{10} = c_1^l \\ a_3^l h^3 S_7 + a_4^l h^4 S_8 + a_5^l h^5 S_9 + a_6^l h^6 S_{10} + a_7^l h^7 S_{11} = c_2^l \\ a_3^l h^3 S_8 + a_4^l h^4 S_9 + a_5^l h^5 S_{10} + a_6^l h^6 S_{11} + a_7^l h^7 S_{12} = c_3^l \\ a_3^l h^3 S_9 + a_4^l h^4 S_{10} + a_5^l h^5 S_{11} + a_6^l h^6 S_{12} + a_7^l h^7 S_{13} = c_4^l \\ a_3^l h^3 S_{10} + a_4^l h^4 S_{11} + a_5^l h^5 S_{12} + a_6^l h^6 S_{13} + a_7^l h^7 S_{14} = c_5^l \end{cases}, \tag{2}$$

где

$$S_j = \sum_{k=0}^{M} k^j, \quad c_j^l = \sum_{k=0}^{M} \left[\left(y_{ml+k} - a_0^l - a_1^l hk - a_2^l (hk)^2 \right) k^{2+j} \right]. \tag{3}$$

Сделаем следующую замену переменных $\tilde{a}_i = a_i h^i$, $i = 0, 1, \dots, 7$. Из (2) и (3) получаем систему уравнений для определения коэффициентов при старших степенях:

$$\begin{cases} \tilde{a}_0^l S_3 + \tilde{a}_2^l S_4 + \tilde{a}_2^l S_5 + \tilde{a}_3^l S_6 + \tilde{a}_4^l S_7 + \tilde{a}_5^l S_8 + \tilde{a}_6^l S_9 + \tilde{a}_7^l S_{10} = P_1^l \\ \tilde{a}_0^l S_4 + \tilde{a}_2^l S_5 + \tilde{a}_2^l S_6 + \tilde{a}_3^l S_7 + \tilde{a}_4^l S_8 + \tilde{a}_5^l S_9 + \tilde{a}_6^l S_{10} + \tilde{a}_7^l S_{11} = P_2^l \\ \tilde{a}_0^l S_5 + \tilde{a}_2^l S_6 + \tilde{a}_2^l S_7 + \tilde{a}_3^l S_8 + \tilde{a}_4^l S_9 + \tilde{a}_5^l S_{10} + \tilde{a}_6^l S_{11} + \tilde{a}_7^l S_{12} = P_3^l \\ \tilde{a}_0^l S_6 + \tilde{a}_2^l S_7 + \tilde{a}_2^l S_8 + \tilde{a}_3^l S_9 + \tilde{a}_4^l S_{10} + \tilde{a}_5^l S_{11} + \tilde{a}_6^l S_{12} + \tilde{a}_7^l S_{13} = P_4^l \\ \tilde{a}_0^l S_7 + \tilde{a}_2^l S_8 + \tilde{a}_2^l S_9 + \tilde{a}_3^l S_{10} + \tilde{a}_4^l S_{11} + \tilde{a}_5^l S_{12} + \tilde{a}_6^l S_{13} + \tilde{a}_7^l S_{14} = P_5^l \end{cases}, \tag{4}$$

где $P_j^l = \sum\limits_{k=0}^{M} y_{ml+k} k^{2+j}$, $\quad j = 1, \dots, 5$.

Уравнения гладкой склейки (1) дают нам следующую систему:

$$\begin{cases} \tilde{a}_0^{l-1} + m\tilde{a}_1^{l-1} + m^2\tilde{a}_2^{l-1} + m^3\tilde{a}_3^{l-1} + m^4\tilde{a}_4^{l-1} + m^5\tilde{a}_5^{l-1} + m^6\tilde{a}_6^{l-1} + m^7\tilde{a}_7^{l-1} = \tilde{a}_0^l \\ \tilde{a}_1^{l-1} + 2m\tilde{a}_2^{l-1} + 3m^2\tilde{a}_3^{l-1} + 4m^3\tilde{a}_4^{l-1} + 5m^4\tilde{a}_5^{l-1} + 6m^5\tilde{a}_5^{l-1} + 7m^6\tilde{a}_7^{l-1} = \tilde{a}_1^l \\ \tilde{a}_2^{l-1} + 3m\tilde{a}_3^{l-1} + 6m^2\tilde{a}_4^{l-1} + 10m^3\tilde{a}_5^{l-1} + 15m^4\tilde{a}_6^{l-1} + 21m^5\tilde{a}_7^{l-1} = \tilde{a}_2^l \end{cases}. \tag{5}$$

В дальнейшем волну над переменной a_i^l, $i = 0, 1, \dots, 7$ будем опускать.

Системы уравнений (4) и (5) составляют систему линейных алгебраических уравнений для определения всех коэффициентов полиномов S-сплайна. Запишем эту систему в матричном виде. Введем следующие обозначения:

$$A_0 = \begin{Vmatrix} S_3 & \dots & S_5 \\ \vdots & \ddots & \vdots \\ S_7 & \dots & S_9 \end{Vmatrix}, \quad A_1 = \begin{Vmatrix} S_6 & \dots & S_{10} \\ \vdots & \ddots & \vdots \\ S_{10} & \dots & S_{14} \end{Vmatrix},$$

$$B_0 = \begin{Vmatrix} 1 & m & m^2 \\ 0 & 1 & 2m \\ 0 & 0 & 1 \end{Vmatrix}, \quad B_1 = \begin{Vmatrix} m^3 & m^4 & m^5 & m^6 & m^7 \\ 3m^2 & 4m^3 & 5m^4 & 6m^5 & 7m^6 \\ 3m & 6m^2 & 10m^3 & 15m^4 & 21m^5 \end{Vmatrix}.$$

$$P^l = \begin{pmatrix} P_1^l \\ \vdots \\ P_5^l \end{pmatrix} \text{ и } X^{2l} = \begin{pmatrix} a_0^l \\ a_1^l \\ a_2^l \end{pmatrix}, X^{2l+1} = \begin{pmatrix} a_3^l \\ a_4^l \\ \vdots \\ a_7^l \end{pmatrix}, \text{ где } l = 0, 1, \dots, L-1. \tag{6}$$

E и 0 – единичная и нулевая матрицы.

Систему для определения коэффициентов S-сплайна записываем в виде следующей блочной матрицы, клеточные строки которой состоят по очереди из трех и пяти строк (аналогично столбцы):

$$\begin{pmatrix} -E & 0 & 0 & 0 & \dots & B_0 & B_1 \\ A_0 & A_1 & 0 & 0 & \dots & 0 & 0 \\ B_0 & B_1 & -E & 0 & \dots & 0 & 0 \\ 0 & 0 & A_0 & A_1 & \dots & 0 & 0 \\ 0 & 0 & B_0 & B_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & A_0 & A_1 \end{pmatrix} \begin{pmatrix} X^0 \\ X^1 \\ X^2 \\ X^3 \\ X^4 \\ \vdots \\ X^{2L-1} \end{pmatrix} = \begin{pmatrix} 0 \\ P^0 \\ 0 \\ P^1 \\ 0 \\ \vdots \\ P^{L-1} \end{pmatrix}. \tag{7}$$

Матрицу этой системы обозначим через G [2].

[2]В случае I клетки матрицы G имеют вид:

$$A_0 = \begin{Vmatrix} S_1 \\ \vdots \\ S_7 \end{Vmatrix}, \quad A_1 = \begin{Vmatrix} S_2 & \dots & S_8 \\ \vdots & \ddots & \vdots \\ S_8 & \dots & S_{14} \end{Vmatrix}, \quad B_0 = \begin{Vmatrix} 1 \end{Vmatrix}, \quad B_1 = \begin{Vmatrix} m & \dots & m^7 \end{Vmatrix}.$$

$$P^l = \begin{pmatrix} P_1^l \\ \vdots \\ P_7^l \end{pmatrix} \text{ и } X^{2l} = \begin{pmatrix} a_0^l \end{pmatrix}, X^{2l+1} = \begin{pmatrix} a_1^l \\ \vdots \\ a_7^l \end{pmatrix}, \text{ где } l = 0, 1, \dots, L-1.$$

В случае II иммеем:

$$A_0 = \begin{Vmatrix} S_2 & S_3 \\ \vdots & \vdots \\ S_7 & S_8 \end{Vmatrix}, \quad A_1 = \begin{Vmatrix} S_4 & \dots & S_9 \\ \vdots & \ddots & \vdots \\ S_9 & \dots & S_{14} \end{Vmatrix}, \quad B_0 = \begin{Vmatrix} 1 & m \\ 0 & 1 \end{Vmatrix}, \quad B_1 = \begin{Vmatrix} m^2 & \dots & m^7 \\ 2m & \dots & 7m^6 \end{Vmatrix}.$$

$$P^l = \begin{pmatrix} P_1^l \\ \vdots \\ P_6^l \end{pmatrix} \text{ и } X^{2l} = \begin{pmatrix} a_0^l \\ a_1^l \end{pmatrix}, X^{2l+1} = \begin{pmatrix} a_2^l \\ \vdots \\ a_7^l \end{pmatrix}, \text{ где } l = 0, 1, \dots, L-1.$$

В случае IV имеем:

$$A_0 = \begin{Vmatrix} S_4 & \dots & S_7 \\ \vdots & \ddots & \vdots \\ S_7 & \dots & S_{10} \end{Vmatrix}, \quad A_1 = \begin{Vmatrix} S_8 & \dots & S_{11} \\ \vdots & \ddots & \vdots \\ S_{11} & \dots & S_{14} \end{Vmatrix},$$

$$B_0 = \begin{Vmatrix} 1 & m & m^2 & m^3 \\ 0 & 1 & 2m & 3m^2 \\ 0 & 0 & 1 & 3m \\ 0 & 0 & 0 & 1 \end{Vmatrix}, \quad B_1 = \begin{Vmatrix} m^4 & m^5 & m^6 & m^7 \\ 4m^3 & 5m^4 & 6m^5 & 7m^6 \\ 6m^2 & 10m^3 & 15m^4 & 21m^5 \\ 4m & 10m^2 & 20m^3 & 35m^4 \end{Vmatrix}.$$

$$P^l = \begin{pmatrix} P_1^l \\ \vdots \\ P_4^l \end{pmatrix} \text{ и } X^{2l} = \begin{pmatrix} a_0^l \\ \vdots \\ a_3^l \end{pmatrix}, X^{2l+1} = \begin{pmatrix} a_4^l \\ \vdots \\ a_7^l \end{pmatrix}, \text{ где } l = 0, 1, \dots, L-1.$$

В случае V:

$$A_0 = \begin{Vmatrix} S_5 & \dots & S_9 \\ \vdots & \ddots & \vdots \\ S_7 & \dots & S_{11} \end{Vmatrix}, \quad A_1 = \begin{Vmatrix} S_{10} & \dots & S_{12} \\ \vdots & \ddots & \vdots \\ S_{12} & \dots & S_{14} \end{Vmatrix},$$

Для непериодического сплайна изменения в системе (7) происходят только в первой строке, которая имеет вид:

$$\begin{pmatrix} E & 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} X^0 \\ \vdots \\ X^{2L-1} \end{pmatrix} = Y^0 \quad \text{где} \quad Y^0 = \begin{pmatrix} y_0 \\ hy_0' \\ \frac{1}{2}h^2 y_0'' \end{pmatrix}. \tag{8}$$

Преобразуем нечетные строки матрицы G таким образом, чтобы избавиться от зависимости от нечетных неизвестных $X^1, X^3, \dots, X^{2L-1}$. Для этого необходимо, чтобы определитель матрицы A_1 был отличен от нуля.

$$\begin{aligned}\det(A_1) = &\frac{1}{428072077674867225600000} M^5(M-4)(2M+1)(M+5)(M+4)^2(M-3)^2(M+3)^3 \times \\ &\times (M-2)^3(M+2)^4(M-1)^4(M+1)^5 \Big(-62747360064M + 34407503328M^2 + \\ &+ 191507803664M^3 + 55638343352M^4 - 121599386484M^5 - \\ &- 109726189632M^6 + 33394853149M^7 + 94469535622M^8 + \\ &+ 14160714141M^9 - 34983098157M^{10} - 13778627821M^{11} + \\ &+ 3270586487M^{12} + 2372934151M^{13} + 146954773M^{14} - 114362976M^{15} - \\ &- 26623653M^{16} + 3671136M^{17} + 3749928M^{18} + 276444M^{19} - 323568M^{20} - \\ &- 74844M^{21} + 4788M^{22} + 3024M^{23} + 252M^{24} + 14649189120 \Big).\end{aligned}$$

Уравнение $\det(A_1) = 0$ имеет 6 положительных вещественных корней: $\{1;\ 1.474;\ 2;\ 2.471;\ 3;\ 4\}$. Так как M – целое положительное число, то при $M \geqslant 5$, существует обратная матрица[3] A_1^{-1}.

Чтобы избавиться от зависимости от нечетных неизвестных $X^1, X^3, \dots, X^{2L-1}$, из третьей строки вычтем вторую, умноженную на матрицу $B_1 A_1^{-1}$. Выпишем для наглядности первые три строки:

$$\begin{pmatrix} -E & 0 & 0 & 0 & \dots & B_0 & B_1 \\ A_0 & A_1 & 0 & 0 & \dots & 0 & 0 \\ B_0 - B_1 A_1^{-1} & 0 & -E & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} X^0 \\ \vdots \\ X^{2L-1} \end{pmatrix} = \begin{pmatrix} 0 \\ P^0 \\ -B_1 A_1^{-1} P^0 \end{pmatrix} \tag{9}$$

Произведем аналогичные преобразования для остальных пар соседних строк с номерами $2l+1$ и $2l$, где $l = 1, \dots, L-1$, а так же для первой и последней строки. Обозначим через

$$B_0 = \begin{Vmatrix} 1 & m & m^2 & \dots & m^4 \\ 0 & 1 & 2m & \dots & 4m^3 \\ \vdots & & \ddots & & \vdots \\ 0 & & \dots & & 1 \end{Vmatrix}, \quad B_1 = \begin{Vmatrix} m^5 & m^6 & m^7 \\ 5m^4 & 6m^5 & 7m^6 \\ 10m^3 & 15m^4 & 21m^5 \\ 10m^2 & 20m^3 & 35m^4 \\ 5m & 15m^2 & 35m^3 \end{Vmatrix}.$$

$$P^l = \begin{pmatrix} P_1^l \\ P_2^l \\ P_3^l \end{pmatrix} \text{ и } X^{2l} = \begin{pmatrix} a_0^l \\ \vdots \\ a_4^l \end{pmatrix}, X^{2l+1} = \begin{pmatrix} a_5^l \\ a_6^l \\ a_7^l \end{pmatrix}, \text{ где } l = 0, 1, \dots, L-1.$$

[3]В случаях I, II, IV и V обратная матрица существует при $M \geqslant 7$; $M \geqslant 6$; $M \geqslant 4$ и $M \geqslant 3$ соответственно.

U матрицу $B_0 - B_1 A_1^{-1}$. Преобразованная система (7) расщепляется на системы для нахождения X^{2l} и для нахождения X^{2l+1}, $l = 0, 1, \ldots, L-1$. Система для нахождения X^{2l} имеет вид:

$$\begin{pmatrix} -E & 0 & 0 & \ldots & 0 & U \\ U & -E & 0 & \ldots & 0 & 0 \\ 0 & U & -E & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & U & -E \end{pmatrix} \begin{pmatrix} X^0 \\ X^2 \\ X^4 \\ \vdots \\ X^{2L-2} \end{pmatrix} = \begin{pmatrix} -B_2 A_2^{-1} P^{L-1} \\ -B_2 A_2^{-1} P^0 \\ -B_2 A_2^{-1} P^1 \\ \vdots \\ -B_2 A_2^{-1} P^{L-2} \end{pmatrix},$$

то есть

$$UX^{2l} - X^{2(l+1)} = -B_1 A_1^{-1} P^l, \quad l = 0, 1, \ldots, L-1, \; X^{2L} = X^0. \tag{10}$$

Система для нахождения X^{2l+1}, $l = 0, 1, \ldots, L-1$:

$$\begin{pmatrix} A_0 & A_1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & A_0 & A_1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & A_0 & A_1 \end{pmatrix} \begin{pmatrix} X^0 \\ X^1 \\ \vdots \\ X^{2L-1} \end{pmatrix} = \begin{pmatrix} P^0 \\ P^1 \\ \vdots \\ P^{L-1} \end{pmatrix},$$

то есть

$$A_1 X^{2l+1} = P^l - A_0 X^{2l}, \quad l = 0, 1, \ldots, L-1. \tag{11}$$

Для непериодического сплайна система уравнений для определения коэффициентов имеет следующий вид:

$$\begin{aligned} &X^0 = Y_0, \quad UX^{2l} - X^{2(l+1)} = -B_1 A_1^{-1} P^l, \quad l = 0, 1, \ldots, L-2, \\ &A_1 X^{2l+1} = P^l - A_0 X^{2l}, \; l = 0, 1, \ldots, L-1. \end{aligned} \tag{12}$$

2. Существование и единственность S-сплайна седьмой степени

Имеют место следующие теоремы.

Теорема 1. *Пусть $M \geqslant 5$, тогда, для любой функции $f(x)$, заданной на отрезке $[a, b]$ своими значениями y_k в точках $x_k = a + kh$, $h = \dfrac{b-a}{K}$, существует единственный непериодический сплайн седьмой степени класса C^2.*

Доказательство. По формулам (12) последовательно находим $X^0, X^1, \ldots, X^{L-1}$. Тем самым все коэффициенты полиномов, составляющих сплайн, однозначно определены. □

Теорема 2. *Пусть числа m и $M \geqslant 5$ таковы, что собственные числа матрицы U не равны корню степени L из единицы (L–число полиномов, составляющих сплайн). Тогда, для любой периодической функции $f(x)$, заданной на отрезке $[a, b]$ своими значениями y_k в точках $x_k = a + kh$, $h = \dfrac{b-a}{K}$, существует единственный периодический сплайн седьмой степени класса C^2.*

Доказательство. Рассмотрим систему (10). Умножим первую строку системы на матрицу U и сложим со второй, полученную вторую строку умножаем на U и складываем с третьей и т.д. Поменяем знаки и получим следующую систему:

$$\begin{pmatrix} E & 0 & \dots & 0 & \dots & -U \\ 0 & E & \dots & 0 & \dots & -U^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & E & \dots & -U^l \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & E-U^L \end{pmatrix} \begin{pmatrix} X^0 \\ X^2 \\ \vdots \\ X^{2(l-1)} \\ \vdots \\ X^{2(L-1)} \end{pmatrix} = \begin{pmatrix} D_0 \\ D_1 \\ \vdots \\ D_{(l-1)} \\ \vdots \\ D_{(L-1)} \end{pmatrix},$$

где $D_0 = B_2A_2^{-1}P^{L-1}$, $D_l = \sum_{j=0}^{l-1} U^jB_2A_2^{-1}P^{l-1-j} - U^lB_2A_2^{-1}P^{L-1}$, $l = 0,1,\dots,L-1$.

По условию теоремы $\det(E-U^L) \neq 0$, следовательно

$$X^{2(L-1)} = (E-U^L)^{-1}D_{L-1}, \quad X^{2l} = D_l + U^{l+1}X^{2(L-1)},$$

а из системы (11) следует, что $X^{2l+1} = A_2^{-1}(P_l - A_1X^{2l})$. Таким образом, все коэффициенты периодического сплайна найдены. □

Аналогичные утверждения справедливы для случаев I, II,IV и V при соответствующих M.

Теорема 3. *Пусть периодическая функция $f(x) \in C^8[a,b]$, и пусть выполнено условие $|f(x_k)-y_k| \leqslant Ch^{8+\varepsilon}$, $\varepsilon \geqslant 0$. Пусть, кроме того, числа m, M, p, n таковы, что $\det(A_1) \neq 0$ и собственные значения матрицы U по модулю меньше единицы. Тогда периодический сплайн седьмой степени $S^7_{m,M} \in C^2[a,b]$ с узлами на равномерной сетке имеет дефект 5 и для $x \in [a,b]$ справедливы следующие оценки:*

$$|f^r(x) - \frac{d^r}{dx^r}S^7_{m,M}(x)| \leqslant C^r h^{8-r},$$

для $r = 0,1,\dots,7$; при $r = 3,\dots,7$, $x \neq \xi_l$; в этом случае $\varphi^{(r)}(\xi_l) \equiv \varphi^{(r)}(\xi_l+0)$, где $\varphi(x) = f(x) - S^7_{m,M}(x)$.

Доказательство производится аналогично доказательству теоремы о сходимости S-сплайна в работах [1, 2, 3].

Аналогичные теоремы справедливы и для непериодического варианта, а также для случаев I, II, IV и V.

3. Устойчивость S-сплайна седьмой степени

Для устойчивости S-сплайна необходимо, чтобы собственные числа матрицы U по модулю были меньше единицы (а если они еще и различны, то и достаточно). Собственные числа матрицы U определяются из уравнения

$$det(U - \lambda E) = 0. \tag{13}$$

Для случая малых значений M (при $3 \leq M \leq 10$) в результате расчета были получены значения собственных чисел матрицы U. В случае $p = 0$ матрица U состоит из одного числа. Показано, что при $M = 7$ и $m = 1,\dots,7$ матрица $U = 0$. Некоторые наиболее интересные

полученные значения m и M, при которых достигаются наименьшие значения $max|\lambda_i|$ и аппроксимация S-сплайнами устойчива, представлены в таблице.

Как показано в случаях $n = 3$ и $n = 5$, для обеспечения этого условия устойчивости необходимо перекрывание. Это означает, что имеются такие элементы исходной таблицы значений функции, которые участвуют в определении коэффициентов не менее двух соседних полиномов, составляющих сплайн. Если перекрывание достаточно большое, то это в ряде случаев является и достаточным условием [1], [3]. На практике наиболее употребительными являются те сплайны, для построения которых используется небольшое число точек осреднения M.

Таблица

Собственные числа матрицы U

p	M	m	$\max\|\lambda_i\|$	p	M	m	$\max\|\lambda_i\|$	p	M	m	$\max\|\lambda_i\|$
0	7	1...7	0	1	7	5	0,107	3	5	3	0,371
0	8	1	0,0006	1	7	6	0,173	3	6	1	0,499
0	8	2	0,0022	1	8	1	0,0931	3	6	2	0,495
0	8	3	0,0044	1	8	2	0,452	3	6	3	0,577
0	8	4	0,005	1	8	3	0,129	3	6	4	0,305
0	8	5	0,00435	1	8	4	0,0341	3	7	1	0,555
0	8	6	0,00218	1	8	5	0,0695	3	7	2	0,623
0	8	7	0,000622	2	5	1	0,570	3	7	3	0,305
0	9	1	0,00263	2	5	2	0,573	3	7	4	0,763
0	9	2	0,00774	2	6	1	0,272	3	7	5	0,655
0	9	3	0,0115	2	6	2	0,141	3	7	6	0,568
0	9	4	0,00749	2	6	3	0,233	4	4	1	0,690
0	9	5	0,00230	2	6	4	0,501	4	4	2	0,881
0	9	6	0,00807	2	7	1	0,324	4	5	1	0,715
0	9	7	0,00625	2	7	2	0,242	4	5	2	0,824
0	9	8	0,00226	2	7	3	0,248	4	6	1	0,756
1	6	1	0,167	2	7	4	0,0908	4	6	3	0,770
1	6	2	0,0667	2	7	5	0,321	4	7	1	0,787
1	6	3	0,0500	2	8	1	0,373	4	7	2	0,693
1	6	4	0,0667	2	8	2	0,387	4	7	3	0,790
1	6	5	0,167	2	8	3	0,131	4	7	4	0,817
1	7	1	0,165	2	8	4	0,271	4	8	1	0,812
1	7	2	0,0253	2	8	5	0,148	4	8	2	0,698
1	7	3	0,0570	3	5	1	0,428	4	8	4	0,765
1	7	4	0,0943	3	5	2	0,306	4	9	2	0,714

Авторы благодарят студента ВМК Кочнева Ю.К., который выполнял вычисление собственных чисел матрицы устойчивости U.

Литература

1. Силаев, Д.А. Приближение S-сплайнами гладких функций / Д.А. Силаев, Г.И. Якушина // Труды семинара имени И.Г. Петровского. – М., 1984. – Вып. 10. – С. 197.

2. Полулокальные сглаживающие сплайны класса C^1 / Д.А. Силаев, А.В. Амилющенко, А.И. Лукьянов, Д.О. Коротаев // Труды семинара имени И.Г. Петровского. – М., 2007. – Вып. 26. – С. 347 – 367.

3. Силаев, Д.А. Дважды непрерывно дифференцируемый полулокальный сглаживающий сплайн / Д.А. Силаев // Вестн. Моск. ун-та. Сер. 1, Математика, механика, 2009. – №5. – С. 11 – 19.

Силаев Дмитрий Алексеевич, кандидат физико-математических наук, доцент, кафедра «Общие проблемы управления», Механико-математический факультет, Московский государственный университет, dasilaev@mail.ru.

Ингтем Женни Гастоновна, кафедра математической физики, факультет «Вычислительная математика и кибернетика», Московский государственный университет, nmail2002@yandex.ru.

Поступила в редакцию 24 мая 2010 г.

УДК 004.75

ИЕРАРХИЧЕСКИЕ МЕТОДЫ УЛУЧШЕНИЯ МАСШТАБИРУЕМОСТИ И ЭФФЕКТИВНОСТИ РАСПРЕДЕЛЕННЫХ РАСЧЕТОВ В СИСТЕМЕ МЕТАКОМПЬЮТИНГА X-COM

С.И. Соболев

HIERARCHICAL METHODS OF PERFORMANCE AND EFFICIENCY IMPROVING FOR DISTRIBUTED COMPUTATIONS WITH X-COM METACOMPUTING SYSTEM

S.I. Sobolev

Система метакомпьютинга X-Com – инструментарий для организации распределенных неоднородных вычислительных сред и проведения расчетов в таких средах. В статье обсуждаются архитектурные решения, примененные в системе X-Com последнего поколения, направленные на улучшение масштабируемости и повышение эффективности распределенных расчетов.

Ключевые слова: X-Com, метакомпьютинг, распределенные вычисления

An X-Com metacomputing system is a software toolkit for an organization of distributed heterogeneous computational environments and performing computations in such environments. In this paper the architectural solutions applied in the X-Com system of last generation and aimed to improving of scalability and efficiency of distributed computations are discussed.

Keywords: X-Com, metacomputing, distributed computations

Введение

Применение технологий распределенных вычислений для решения больших вычислительно сложных задач стало привычным фактом. Однако уже сейчас на этом пути встают проблемы, связанные с координацией огромного числа взаимодействующих компьютеров и потоков данных между ними. Технологически несложно объединить для работы над единой задачей, скажем, пять суперкомпьютерных комплексов, находящихся в различных регионах России, однако эффективность такого решения будет под вопросом. Предположим, обработка одной независимой порции задачи занимает в среднем 5 минут, а всего над задачей работают 6 тысяч процессоров распределенной среды. Организуя расчеты по обычной клиент-серверной схеме, получаем, что сервер должен корректно обрабатывать до 40 клиентских запросов в секунду. Такая нагрузка на сервер распределенных вычислений имеет тот же порядок, что и суммарная нагрузка на веб-сайты компании Google, обслуживаемые большими фермами серверов. При этом мы не принимаем в расчет затраты ресурсов сервера на генерацию вычислительных порций и обработку результатов. При использовании шифрования данных нагрузка на сервер увеличится еще на порядок. Кроме того, приведенные

в качестве примера 6 тысяч процессоров – это реалии сегодняшнего дня, однако уже сейчас в России строятся суперкомпьютеры с десятками тысяч процессорных ядер. Мы естественным образом приходим к необходимости использования распределенных и иерархических технологий при организации самих распределенных вычислений.

В НИВЦ МГУ имени М.В. Ломоносова разрабатывается система метакомпьютинга X-Com [1, 4] – программный инструментарий, предназначенный для организации распределенных неоднородных вычислительных сред и проведения вычислений в таких средах. Базовыми компонентами системы является сервер задачи и клиент X-Com. Сервер задачи отвечает за разбиение конкретной прикладной задачи на независимые вычислительные порции, распределение их на вычислительные узлы и объединение получаемых результатов. Клиенты X-Com, устанавливаемые на узлах, принимают от сервера данные, запускают вычислительные модули прикладной задачи и отправляют результаты обработки порций обратно на сервер.

В последние два года [6, 7] основная работа над системой была сосредоточена на улучшении ее архитектуры с целью повышения масштабируемости и эффективности проводимых с помощью нее расчетов. Наиболее важные реализованные и планируемые архитектурные решения обсуждаются в настоящей статье.

1. Иерархия «сервер задач – сервисы запросов»

Требование обработки сервером задач крайне интенсивного потока запросов от вычислительных узлов приводит к необходимости распределения его работы на отдельные процессы, способные выполняться на разных физических машинах. Естественным представляется разбиение функциональности сервера задач на процесс-координатор, обладающий полной информацией о ходе задачи, и сервисы обработки запросов, взаимодействующие непосредственно с вычислительными узлами. Такая архитектура будет особенно эффективна при включенном шифровании данных, требующем дополнительных затрат процессорного времени. В этом случае шифрование и дешифрование может проводиться на выделенных компьютерах, нагрузка на компьютер с процессом-координатором при этом существенно снижается.

В серверной части X-Com выделяются следующие типы запросов и соответствующих им сервисов (рис. 1). Сервис TSR отвечает за выдачу первоначальной информации о задаче. Файловый сервис – это фактически файловый сервер, у которого клиент запрашивает файлы прикладной задачи и вспомогательные файлы, если они необходимы (отметим, что в качестве альтернативного файлового сервиса может использоваться любой httpd-сервер). Через сервис REQ клиент запрашивает очередную порцию данных у сервера задачи. Результаты выполнения прикладной задачи над данными очередной порции клиент пересылает сервису ASW. Сервис STAT предназначен для предоставления статистической информации о ходе расчета в форматах HTML и XML.

Обмен данными между клиентами X-Com и сервисами запросов производится по специальному протоколу на основе XML поверх стандартного протокола HTTP в сети TCP/IP. Обмен данными между сервисами запросов и процессом-координатором ведется по более простому и экономичному протоколу, причем осуществляться он может как поверх TCP/IP (сервер задач и процессы запросов могут работать на различных компьютерах; поддерживается в любой ОС), так и посредством локальных сокетов UNIX (сервер задач и процессы запросов должны работать на одном компьютере; поддерживается только для ОС семейства UNIX/Linux). Первый способ более универсален, второй обеспечивает более высокую производительность при работе всех серверных процессов на одной физической машине.

Сервер задачи может быть запущен в режиме любого из сервисов вручную (в доку-

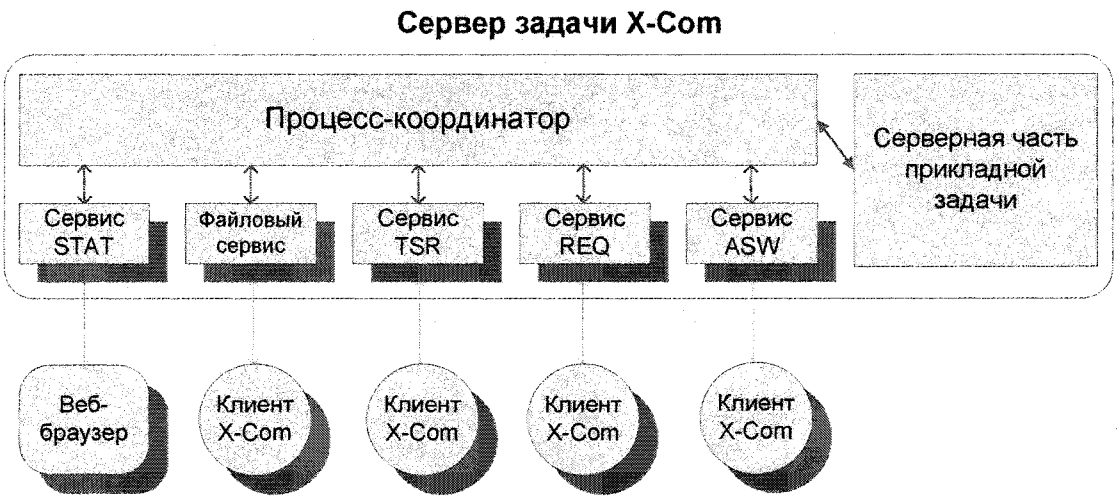


Рис. 1. Структура сервера задачи X-Com

ментации [2] данный режим обозначен как Subserver). При этом в настройках указываются точка доступа (пара хост-порт или имя сокета) процесса-координатора.

2. Иерархия «подсистема управления заданиями – серверы задач»

Работа в масштабной распределенной среде – это, в том числе, огромное число задач и пользователей. Запуск и контроль прохождения заданий в таких средах вручную попросту невозможен – слишком много факторов нужно учесть. Необходимы простые и понятные средства взаимодействия с пользователями, а также механизмы централизованного управления потоками заданий. В системе X-Com эти функции обеспечиваются подсистемой управления заданиям XQSERV. Прототип подсистемы XQSERV был описан в работе [3], настоящая статья описывает актуальное состояние подсистемы.

Подсистема управления заданиями XQSERV состоит из серверной части, отвечающей за распределение задач в вычислительной среде, и клиентской, реализующей пользовательские интерфейсы (рис. 2). Эти интерфейсы позволяют пользователям работать с вычислительной средой с использованием привычных метафор традиционных высокопроизводительных комплексов – поставить задание или набор заданий в очередь, проконтролировать ход их выполнения, удалить задание из очереди. Базовый метод работы с клиентом подсистемы управления заданиями – вызов клиента из командной строки с необходимыми опциями. Общение между клиентом и сервером XQSERV может осуществляться двумя способами – через сокеты UNIX либо поверх TCP/IP. Первый способ работоспособен только в операционной среде UNIX/Linux и только в том случае, когда клиент и сервер XQSERV запущены на одной физической машине. Этот способ средствами операционной системы обеспечивает получение сервером корректной информации об имени пользователя, вызвавшем клиента. Второй способ обмена данными (TCP/IP) такой информации, вообще говоря, не предоставляет, однако он более универсален и может использоваться при работе клиента на удаленной машине с отличной от серверной операционной системой.

Еще один пользовательский интерфейс, реализуемый подсистемой XQSERV – веб-интерфейс, позволяющий отслеживать общее состояние очереди заданий и ход выполнения каждой задачи из очереди.

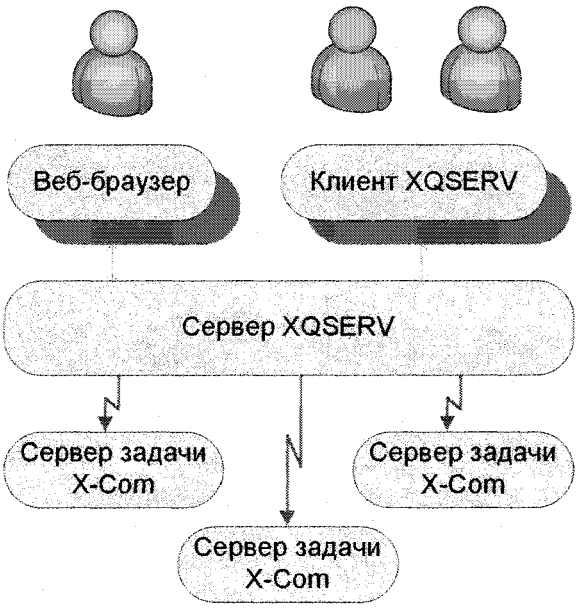


Рис. 2. Подсистема управления заданиями XQSERV

Сервер подсистемы управления заданиями XQSERV (управляющий сервер) реализует логику распределения заданий в вычислительной среде. За каждое конкретное задание отвечает свой сервер задачи; с точки зрения управляющего сервера запуск задания означает запуск соответствующего сервера задачи. В простейшем случае управляющий сервер организует линейную очередь, запуская все поступающие от пользователей задания последовательно на всех доступных ресурсах. Такой метод распределения заданий реализует основную идею метакомпьютерных вычислений, а именно использование максимального объема ресурсов для решения задачи, и позволяет достичь максимальной суммарной производительности подключенных ресурсов. Этот метод, однако, не гарантирует максимальной эффективности их использования, в частности, не учитывая требования прикладных задач к тем ресурсам, на которых они будут выполняться. С другой стороны, в ряде случаев может возникнуть необходимость в одновременной работе более одной задачи. Очевидно, необходимы методы динамического разбиения вычислительной среды на сегменты с заданными свойствами, каждый из которых будет выделяться одной задаче, при этом при изменении состава заданий состав сегментов также будет меняться.

3. Иерархическая сегментация среды

Задачи в распределенной среде могут предъявлять различные требования к ресурсам, на которых они должны выполняться. Рассмотрим типичный случай [5]: предположим, что время обработки каждой вычислительной порции достаточно велико, при этом оно существенно зависит от тактовой частоты процессора, а среда объединяет узлы как с высокой, так и низкой частотой CPU. В этом случае вполне возможен вариант, при котором слабые узлы, получив свои порции в самом начале обработки задания, не закончат их обработку до момента завершения всего расчета. Подключение таких узлов для данного расчета окажется нецелесообразным; в то же время, их вполне можно было бы использовать, например, для решения относительно небольших задач либо для тестовых запусков приложений.

При постановке задания в очередь XQSERV можно указать следующие требования к ре-

сурсам: минимальную и максимальную тактовую частоту CPU и/или производительность, минимальный и максимальный объем оперативной памяти, минимальное и максимальное число процессорных ядер, тип операционной системы, процессорную архитектуру. Можно также указать список кластеров, на которых разрешается расчет (принадлежность к кластерам определяется по идентификаторам вычислительных узлов).

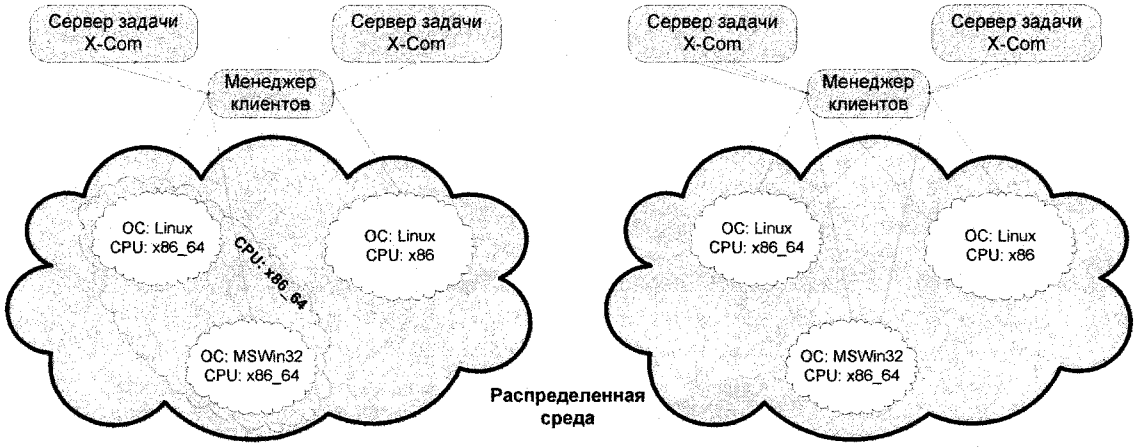


Рис. 3. Примеры сегментации распределенной среды

Чтобы реализовать динамическое перераспределение вычислительных узлов между заданиями, в состав подсистемы управления заданиями XQSERV включен менеджер клиентов. Менеджер клиентов представляет собой процесс-сервер, к которому обращаются при первом запуске все клиенты на вычислительных узлах. Менеджер перенаправляет вычислительные узлы к тем серверам задач, требованиям которых они соответствуют, либо (при отсутствии явно указанных требований) распределяет клиентов поровну на каждую запущенную задачу. Если имеется несколько одновременно работающих задач, и подключающийся узел удовлетворяет требованиям каждой из них, он будет перенаправлен к задаче, запущенной раньше всех. После завершения задачи клиенты на узлах вновь обращаются к менеджеру за новым назначением. Примеры возможной сегментации распределенной среды приведены на рис. 3.

4. Иерархия промежуточных серверов

Рассмотренные выше механизмы иерархической сегментации среды реализуют логическую группировку вычислительных ресурсов. Однако зачастую ресурсы распределенной среды уже имеют явно выраженную физическую группировку, представляя собой, например, узлы вычислительных кластеров или машины компьютерных классов. Как правило, такие узлы размещены в рамкой закрытой локальной сети, из которой прямой доступ к серверу задач через Интернет может отсутствовать по соображениям безопасности. В этом случае в распределенную среду может быть введен еще один компонент – промежуточный (буферизирующий) сервер. С точки зрения нижележащих узлов промежуточный сервер представляется единственным сервером, доступным данным узлам, с точки же зрения центрального сервера промежуточный сервер сам представляется вычислительным узлом. Помимо организации доступа в закрытую сеть, промежуточный сервер несет еще одну важную функцию – буферизацию обмена данными между «своими» узлами и центральным сервером, снижая тем самым нагрузку на него за счет минимизации числа сетевых соединений (это достигается группировкой нескольких порций в едином запросе) и оптимизации

сетевого трафика (клиент на вычислительном узле скачивает необходимые файлы не с центрального, а с ближайшего к нему промежуточного сервера).

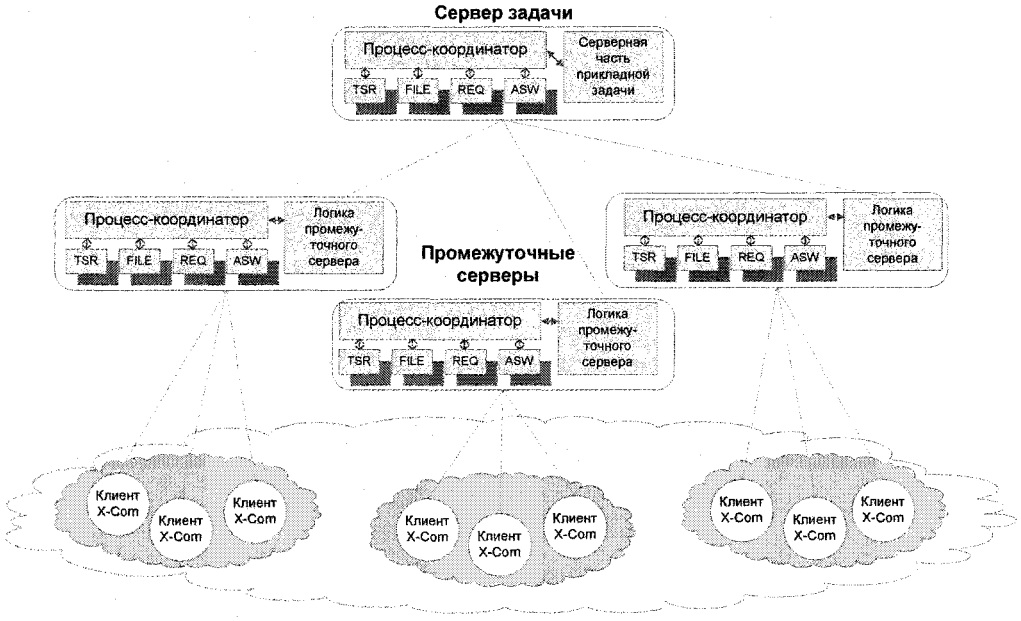


Рис. 4. Организация вычислений с использованием промежуточных серверов

Механизм промежуточных серверов достаточно подробно описан в предыдущих публикациях по системе X-Com, в частности, в [7]. С точки зрения реализации промежуточный сервер – это один из режимов работы сервера задачи (в документации [2] обозначен как Proxy), при котором в качестве «задачи» подключается модуль, реализующий общение с вышестоящим сервером (полностью аналогично клиентской части X-Com). Промежуточные серверы позволяют формировать распределенную среду в виде дерева с произвольным числом ярусов, в корне которого будет центральный сервер X-Com, в узлах – промежуточные серверы, а листьями будут являться клиенты X-Com (рис. 4).

Хотелось бы отметить, что анализ модели промежуточных серверов и их реализация существенно способствовали внедрению иерархических методов и в другие компоненты системы X-Com.

5. Иерархия «вычислительный клиент – рабочие процессы»

Клиентская часть X-Com (вычислительный клиент), устанавливающаяся на всех узлах распределенной среды и взаимодействующая с сервером задачи, отвечает за запуск вычислительной части прикладной задачи. Архитектура современных вычислительных узлов позволяет запускать сразу несколько вычислительных процессов на одном узле (обычно по числу процессорных ядер). В настоящее время для реализации этой возможности на каждом узле запускается по несколько клиентов X-Com, которые взаимодействуют с сервером задач и запускают вычислительные процессы независимо друг от друга. Логичным представляется переход к версии клиента X-Com, поддерживающей многопроцессорные и многоядерные конфигурации. Такой клиент будет однократно скачивать все рабочие файлы прикладной задачи на узел (это позволит уменьшить сетевой трафик и нагрузку на сервер задач) и запускать необходимое число параллельных вычислительных процессов (рис. 5).

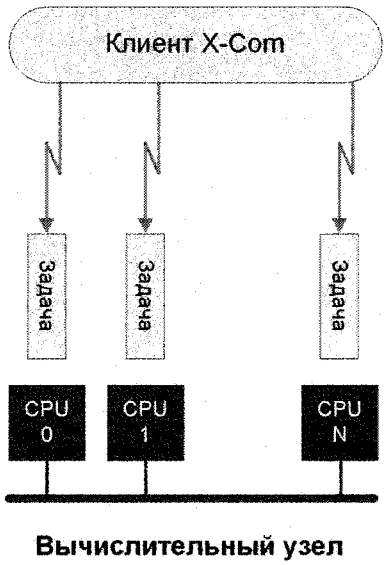


Рис. 5. Перспективная архитектура клиента X-Com

Архитектура такого клиента слегка напоминает архитектуру промежуточного сервера. Автоматически появляется возможность запускать на узлах как чисто последовательные программы, так и программы, написанные с использованием OpenMP. Число запускаемых процессов на узле может указываться при установке клиента, а может быть параметром при запуске каждой задачи.

6. Заключение

Внедрение иерархических и распределенных методов обработки данных на всех уровнях архитектуры системы X-Com способствует снижению накладных расходов системы метакомпьютинга и, как следствие, увеличивает эффективность ее работы. Распределение функциональности сервера задач на независимые процессы, способные работать на отдельных компьютерах, уменьшает загрузку процессора и каналов связи каждой из машин. Подсистема управления заданиями позволяет разбить все множество имеющихся вычислительных ресурсов на подмножества, эффективно решающие имеющиеся прикладные задачи, а кроме того, обеспечивает высокоуровневые пользовательские интерфейсы для работы в распределенной среде. Промежуточные серверы оптимизируют передачу больших объемов данных и снижают загрузку центрального сервера распределенных вычислений. Иерархическая организация клиентской части также позволит экономить сетевой трафик и оптимизировать загрузку вычислительных узлов.

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2010».

Работа выполняется при поддержке гранта Президента РФ для молодых ученых МК-3040.2009.9 и частичной поддержке научно-технической программы Союзного государства СКИФ-ГРИД.

Литература

1. Семейство программ X-Com (официальный сайт) [Электронный ресурс]. – Режим доступа: http://x-com.parallel.ru/ (дата обращения: 12.04.2010).

2. Руководство пользователя системы X-Com2 [Электронный ресурс]. – Режим доступа: http://x-com.parallel.ru/documentation.html (дата обращения: 12.04.2010).
3. Соболев, С.И. Управление заданиями в Виртуальном метакомпьютерном центре на основе технологий X-Com / С.И. Соболев // Распределенные вычисления и Грид-технологии в науке и образовании: тр. второй междунар. конф. (Дубна, 26 – 30 июня 2006 г.). – Дубна, 2007. – С. 401 – 404.
4. Соболев, С.И. Использование распределенных компьютерных ресурсов для решения вычислительно сложных задач / С.И. Соболев // Системы управления и информационные технологии. – 2007. – №1.3 (27). – С. 391 – 395.
5. Соболев, С.И. Эффективная работа в распределенных вычислительных средах // С.И. Соболев // Численные методы, параллельные вычисления и информационные технологии. – 2008. – С. 249 – 258.
6. Соболев, С.И. Архитектура нового поколения системы метакомпьютинга X-Com // С.И. Соболев// Распределенные вычисления и Грид-технологии в науке и образовании: тр. третьей междунар. конф. (Дубна, 30 июня – 4 июля 2008 г.). – Дубна, 2008. – С. 123 – 127.
7. Эволюция системы метакомпьютинга X-Com / Вл.В. Воеводин, Ю.А. Жолудев, С.И. Соболев, К.С. Стефанов // Вестн. Нижегород. гос. ун-та им. Н.И. Лобачевского. – 2009. – №4. – С. 157 – 164.

Соболев Сергей Игоревич, кандидат физико-математических наук, научный сотрудник, Научно-исследовательский вычислительный центр МГУ имени М.В. Ломоносова, sergeys@parallel.ru.

Поступила в редакцию 13 апреля 2010 г.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Серия «Математическое моделирование и программирование» основана в 2008 году.

Основной целью издания является пропаганда союза качественных и количественных исследований математических моделей. Предпочтение при публикации будут иметь статьи, посвященные либо изучению новых математических моделей, либо обсуждению теоретических основ программирования с приложениями к математическим моделям.

Свидетельство о регистрации ПИ №ФС77-26455 выдано 13 декабря 2006 г. Федеральной службой по надзору законодательства в сфере массовых коммуникаций и охране культурного наследия.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Zentralblatt MATH», «Mathematical Reviews».

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта ЮУрГУ (http://www.susu.ac.ru) следуя ссылкам: «Научная деятельность»⇒ «Издательская деятельность»⇒ «Вестник ЮУрГУ»⇒ «Серии». **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются и назад авторам не высылаются.**

2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Математическое моделирование и программирование»

Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, Южно-Уральский государственный университет, механико-математический факультет, кафедра УМФ, ответственному секретарю доценту Манаковой Наталье Александровне.

3. Адрес электронной почты редакции: vestnikmmp@gmail.com

4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

5. Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Математическое моделирование и программирование»: 29126, каталог «Пресса России». Периодичность выхода – 2 номера в год (май и ноябрь).

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

№ 35 (211) 2010

Серия
«МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ
И ПРОГРАММИРОВАНИЕ»

Выпуск 6

Издательский центр Южно-Уральского государственного университета

Подписано в печать 02.11.2010. Формат 60×84 1/8. Печать трафаретная.
Усл. печ. л. 14,18. Уч.-изд. л. 11. Тираж 500 экз. Заказ 387/673.

Отпечатано в типографии Издательского центра ЮУрГУ. 454080, г. Челябинск, пр. им. В.И. Ленина, 76.