Наталия Михайловна ГАВРИЛОВА[1]
Юрий Анатольевич ПЛОТОНЕНКО[2]
Андрей Анатольевич СТУПНИКОВ[3]

УДК 519.63

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ИССЛЕДОВАНИЯ РАСПАРАЛЛЕЛИВАНИЯ ВЫЧИСЛЕНИЙ

[1] кандидат физико-математических наук,
доцент кафедры программного обеспечения,
Тюменский государственный университет
n.m.gavrilova@utmn.ru; ORCID: 0000-0002-8697-5639

[2] кандидат педагогических наук,
доцент кафедры программного обеспечения,
Тюменский государственный университет
y.a.plotonenko@utmn.ru

[3] кандидат физико-математических наук,
доцент кафедры программного обеспечения,
Тюменский государственный университет
a.a.stupnikov@utmn.ru; ORCID: 0000-0001-5201-1260

Аннотация

Использование многопроцессорных вычислительных систем является одним из важнейших путей повышения скорости решения сложных задач.

В данной работе представлено описание опыта разработки программного обеспечения для организации научных исследований и решения учебных задач с применением технологий распараллеливания вычислений.

Цитирование: Гаврилова Н. М. Разработка интеллектуального программного обеспечения для исследования распараллеливания вычислений / Н. М. Гаврилова, Ю. А. Плотоненко, А. А. Ступников // Вестник Тюменского государственного университета. Физико-математическое моделирование. Нефть, газ, энергетика. 2021. Том 7. № 3 (27). С. 152-169.
DOI: 10.21684/2411-7978-2021-7-3-152-169

Рассматриваются подходы к организации распараллеливания вычислений с применением многопроцессорной системы с общей памятью для задачи определения численного решения системы линейных уравнений с трехдиагональной матрицей коэффициентов, возникающей при решении краевой задачи для дифференциального уравнения в частных производных параболического типа, уравнения теплопроводности. Для численного решения уравнения теплопроводности в двумерном случае использована конечно-разностная схема переменных направлений.

Для реализации метода трехдиагональной прогонки применяются последовательный и параллельные алгоритмы (двухпоточный алгоритм встречной прогонки, многопоточный вариант горизонтально блочной прогонки), ориентированные на выполнение в вычислительных системах с общей памятью.

Для распараллеливания вычислений использованы две технологии организации параллельных вычислений для вычислительных устройств с общей памятью: на основе технологии OpenMP, в рамках которой обслуживание процесса распараллеливания и балансировки нагрузки выполнялось средствами среды компиляции программы, а также средствами из библиотеки .NET, позволяющими использовать ручное управление процессами распараллеливания потоков.

В качестве оценки эффективности описываемого подхода приведено время расчета с применением последовательного и параллельных алгоритмов в зависимости от размера задачи и количества используемых потоков.

Сравнение рассматриваемых алгоритмов распараллеливания и технологий реализации выполняется на основе анализа получаемого ускорения. Показано, что общее время вычислений при распараллеливании на Thread-потоках оказалась в несколько раз больше, чем на OpenMP-потоках, а ускорение расчетов меньше, соответственно.

Разработано приложение, позволяющее в реальном времени с использование технологий параллельных расчетов получать визуальный результат моделирования процесса распространения температуры в исследуемой области.

Ключевые слова

Система линейных алгебраических уравнений, трехдиагональная матрица, метод прогонки, технологии распараллеливания расчетов, алгоритмы распараллеливания, блочный алгоритм, конечно-разностная схема.

DOI: 10.21684/2411-7978-2021-7-3-152-169

Введение

С развитием многопроцессорных компьютеров и вычислительных систем выросла необходимость в использовании алгоритмов, использующих распараллеливание вычислений. Исследователь, с одной стороны, должен определить подходящий метод решения и способ его реализации, с другой — уделить внимание визуализации численного решения, внедрения результатов в другие проекты решения. При этом ему приходится затрачивать время на вспомога-

тельные операции, связанные с дальнейшей обработкой полученных результатов. В их число входит представление результатов различных форматов с целью последующего использования в специализированных пакетах и визуализации для систематизации итогов и получения наглядного представления параметров рассматриваемого процесса.

Вычислительные приложения, реализующие параллельные алгоритмы, дают возможность повысить эффективность программных решений и использовать преимущества многоядерных процессоров и многопроцессорных систем.

Предлагается комплексное решение для исследователя, позволяющее в рамках одного программного продукта получать результаты численного моделирования с визуализацией результатов для динамического отслеживания процесса. Для ускорения вычислений используются параллельные вычисления в рамках реализации численных алгоритмов.

При организации схем распараллеливания вычислений достаточно заметный рост производительности можно обеспечить средствами готовых имеющихся средств.

Исследователь получает инструмент, удобство которого определяется, в частности, тем, что можно использовать различные алгоритмы для распараллеливания вычислений для конкретной вычислительной задачи. С другой стороны, результаты можно помещать в хранилище данных и демонстрировать на экране различные варианты расчетов: другие входные данные, параметры и т. д., интерактивно получать и анализировать результаты.

Программный комплекс позволяет свести многоэтапный процесс получения результатов (последовательное решение отдельных задач) к единому процессу параллельных вычислений с одновременным получением результатов, в том числе и визуальном представлении. Это дает возможность делать анализ, строить непрерывные карты, моделировать последствия тех или иных решений. Для большей наглядности можно представлять изолинии, тепловые карты. Вся математическая обработка получения и визуализации расчетов собрана вместе. Современные процессоры, фактически состоящие из нескольких ядер, вполне позволяют именно так строить решение.

Применение механизма распараллеливания позволяет не разделять расчеты и визуализацию, температурную карту можно получать в виде изолиний, для отрисовки которых можно использовать интерполяцию.

Применение представленного подхода представляет интерес и для ряда других случаев, в частности, для экологических задач, рассматривающих распространение загрязнений, или для человеко-машинных систем, обеспечивающих мониторинг температурного режима здания, и т. п. Рассмотренные приемы распараллеливания могут быть задействованы в рамках реализации программных комплексов, решающих задачи как автоматического, так и визуального обнаружения критических ситуаций и последующего оповещения.

Ранее авторы выполнили реализацию первого части этого комплекса, в котором было реализовано распараллеливание вычислительного алгоритма на примере краевой задачи для одномерного уравнения теплопроводности [5]. Для рас-

параллеливания были использованы две технологии организации параллельных вычислений для вычислительных устройств с общей памятью: на основе технологии OpenMP, в рамках которой обслуживание процесса распараллеливания и балансировки нагрузки выполнялось средствами среды компиляции программы, а также средствами из библиотеки .NET, позволяющими использовать ручное управление процессами распараллеливания потоков.

Полученные результаты продемонстрировали эффективность применения для распараллеливания расчетов различных подходов: на основе потоков и с использованием библиотек технологии OpenMP.

В данной работе представлены расчеты краевой задачи для двумерного уравнения теплопроводности. Численное решение задачи реализовано с применением неявной конечно-разностной схемы переменных направлений [4], которая сводит исходную постановку к известной задаче решения системы линейных алгебраических уравнений (СЛАУ) с трехдиагональной основной матрицей.

Для двумерного случая распараллеливание алгоритма расчета можно проводить по двум пространственным переменным.

Методы и принципы исследования

Принципы организации параллельных вычислений при использовании многопроцессорных вычислительных систем подробно рассмотрены в работах [1-3, 6].

Традиционно для распараллеливания вычислений СЛАУ с ленточной матрицей используются следующие известные в вычислительной практике методы: встречной прогонки, параллельно-циклической редукции, параметрической прогонки Яненко [7], параллельно-конвейерный метод для решения множества трехдиагональных систем [6], горизонтально-блочный параллельный алгоритм [3].

Методы распараллеливания трехдиагональной прогонки рассматриваются в работах [2, 3, 6, 7]. Параллельные расчеты организуются как с использованием общей памяти средствами OpenMP технологии, так и, в случае распределенной памяти, средствами стандарта параллельного программирования MPI.

Построение разностной схемы

Дифференциальное уравнение, описывающее процесс распространения температуры на плоской пластине со сторонами l_1 и l_2 и граничными условиями I-го рода и заданными начальными условиями, записывается в виде [4]:

$$\frac{\partial u}{\partial t}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f(x,y,t),\ x\in(0,l_1),\ y\in(0,l_2),\ t>0;$$

$$u(x,0,t)=\varphi_1(x,t),\ x\in[0,l_1],\ y=0,\ t>0,$$

$$u(x,l_2,t)=\varphi_2(x,t),\ x\in[0,l_1],\ y=l_2,\ t>0,$$

$$u(0,y,t)=\varphi_3(y,t),\ x=0,\ y\in[0,l_2],\ t>0,$$

$$u(l_1,y,t)=\varphi_4(y,t),\ x=l_1,\ y\in[0,l_2],\ t>0,$$

$$u(x,y,0)=\psi(x,y),\ x\in[0,l_1],\ y\in[0,l_2],\ t=0.$$

Разностная схема метода переменных направлений в двумерном случае имеет вид:

Подсхема 1

$$\frac{u_{ij}^{k+1/2}-u_{ij}^{k}}{\tau/2}=\frac{a^2}{h_1^2}\left(u_{i+1j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1j}^{k+1/2}\right)+\frac{a^2}{h_2^2}\left(u_{ij+1}^{k}-2u_{ij}^{k}+u_{ij-1}^{k}\right)+$$

$$+f_{ij}^{k+1/2}$$

Подсхема 2

$$\frac{u_{ij}^{k+1}-u_{ij}^{k+1/2}}{\tau/2}=\frac{a^2}{h_1^2}\left(u_{i+1j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1j}^{k+1/2}\right)+\frac{a^2}{h_2^2}\left(u_{ij+1}^{k+1}-2u_{ij}^{k+1}+u_{ij-1}^{k+1}\right)+$$

$$+f_{ij}^{k+1/2}$$

$$i=1..I-1;\ j=1..J-1;\ k=0..N-1.$$

Здесь h_1, h_2 и τ — соответственно шаги по осям x, y, t и $x_i = ih_1$, $i = 0..I$; $y_j = jh_2$, $j = 0..J$; $t_k = k\tau$, $k = 0..N$.

Значения искомой температуры в узлах равномерной сетки обозначены u_{ij}^{k}, где верхний индекс соответствует временному слою, а нижние — текущим пространственным координатам.

Схема МПН абсолютно устойчива, имеет второй порядок точности по времени и по пространственным переменным [2].

Уравнение Подсхемы 1 можно переписать в виде:

$$A_i u_{i-1j}^{k+1/2}+B_i u_{ij}^{k+1/2}+C_i u_{i+1j}^{k+1/2}=F_{ij}^{k},$$

$$A_i=-\frac{a^2\tau}{2h_1^2};\ B_i=1+\frac{a^2\tau}{h_1^2};\ C_i=-\frac{a^2\tau}{2h_1^2},$$

$$F_{ij}^{k}=A_j u_{ij-1}^{k}+B_j u_{ij}^{k}+C_j u_{ij+1}^{k}+\frac{\tau}{2}f_{ij}^{k+1/2}.$$

Получаемая СЛАУ имеет трехдиагональный вид и может быть решена методом прогонки по оси x. Прогонка выполняется по индексу i.

Уравнение Подсхемы 2 можно переписать в виде:

$$A_j u_{ij-1}^{k+1}+B_j u_{ij}^{k+1}+C_j u_{ij+1}^{k+1}=F_{ij}^{k+1/2},$$

$$A_i=-\frac{a^2\tau}{2h_2^2};\ B_i=1+\frac{a^2\tau}{h_2^2};\ C_i=-\frac{a^2\tau}{2h_2^2},$$

$$F_{ij}^{k+1/2}=A_i u_{i-1j}^{k+1/2}+B_i u_{ij}^{k+1/2}+C_i u_{i+1j}^{k+1/2}+\frac{\tau}{2}f_{ij}^{k+1/2}.$$

Аналогично вышесказанному СЛАУ имеет трехдиагональный вид и может быть решена методом прогонки по оси y. Прогонка выполняется по индексу *j*.

Описание алгоритма встречной прогонки

Классический последовательный алгоритм трехдиагональной прогонки следует видоизменить, чтобы организовать вычисления отдельных частей алгоритма параллельно. В данной работе рассматриваются два подхода к распараллеливанию алгоритма прогонки для систем с общей памятью. Это двухпотоковый алгоритм встречной прогонки и многопотоковый блочный алгоритм.

Для системы линейных алгебраических уравнений вида

$$\begin{gathered} a_i x_{i-1} - b_i x_i + c_i x_{i+1} = f_i \quad i = 0, \dots, n, \\ a_0 = 0, \quad c_n = 0 \end{gathered} \tag{1}$$

последовательный алгоритм (правой) прогонки реализует прямой ход для вычисления прогоночных коэффициентов:

$$\begin{gathered} \alpha_1 = \frac{-b_0}{c_0}, \quad \beta_1 = \frac{-f_0}{c_0}, \\ \alpha_{i+1} = \frac{-b_i}{a_i \alpha_i + c_i}, \qquad \beta_{i+1} = \frac{f_i - a_i b_i}{a_i \alpha_i + c_i}, \quad i = 1, \dots, n-1. \end{gathered} \tag{2}$$

Обратный ход вычисляет неизвестные x_i.

$$x_n = \frac{f_n - a_n b_n}{a_n \alpha_n + c_n}, \quad x_i = \alpha_{i+1} x_{i+1} + \beta_{i+1}, \qquad i = n-1, \dots, 0. \tag{3}$$

Алгоритм левой прогонки реализуется по аналогичным формулам. Прогоночные коэффициенты прямого хода:

$$\xi_n = \frac{-a_n}{c_n}, \ \eta_n = \frac{f_n}{c_n}, \xi_i = \frac{-a_i}{b_i - c_i \xi_{i+1}}, \eta_i = \frac{f_i - c_i \eta_{i+1}}{b_i - c_i \xi_{i+1}}, \qquad i = n-1, \dots, 1. \tag{4}$$

Обратный ход вычисляет неизвестные x_i.

$$x_0 = \frac{f_0 - c_0 \eta_1}{b_0 - c_1 \xi_1}, x_{i+1} = \xi_{i+1} x_i + \eta_{i+1}, \qquad i = 0, \dots, n-1. \tag{5}$$

Параллельный алгоритм встречной прогонки организован на основе одновременного выполнения правой и левой прогонок в двух независимых потоках раздельно для верхний и нижней половины системы (1) с внепоточным расчетом поворотной точки x_p из системы уравнений (2):

$$\begin{cases} x_p = \alpha_{p+1} x_{p+1} + \beta_{p+1} \\ x_{p+1} = \xi_{p+1} x_p + \eta_{p+1} \end{cases} \qquad x_p = \frac{f_p - a_p \beta_p - b_p \eta_{p+1}}{c_p + a_p \alpha_p + b_p \xi_{p+1}} \tag{6}$$

В случае последовательной реализации метода трехдиагональной прогонки условное общее время работы алгоритма оценивается как $T_1 = 10n\tau$, где за τ взято время выполнения одной операции [6]. Таким образом, время расчета линейно зависит от величины размерности системы.

В случае применения метода параллельной встречной прогонки общее время работы алгоритма оценивается как $T_2 = 5n\tau + \delta$, где δ — время, необходимое на организацию и закрытие параллельной секции. Предполагаемое теоретическое ускорение ожидаемо приближается к двум.

Описание блочного алгоритма прогонки

Подавляющее число современных компьютеров, обладая более чем двумя встроенными процессорами, позволяют получить соответственно большее ускорение, чем дает алгоритм встречной прогонки. Для этого процедуру расчета представленных выше схем следует разделить между всеми имеющимися процессорами. При этом ожидаемое теоретическое ускорение в идеале приближается к числу задействованных процессоров. Подобная возможность возникает при использовании специального блочного алгоритма [3, с. 32-36].

Ниже приведены формулы алгоритма блочной прогонки для СЛАУ вида (1).

Матрица коэффициентов трехдиагональной системы уравнений размерности $n \times n$ разбивается на число полос, соответствующее числу потоков $p > 2$. Число строк каждой полосы равно $m = [n/p]$. В каждом потоке с номером k содержатся строки с номерами i, где $1 + (k - 1)\, m \le i \le km$, $k = 1, \ldots, p$.

Таким образом, расчеты для системы с матрицей размерности $n \times n$ и числе потоков p фактически сводятся к независимому решению p систем размерности $m \times n$.

Блочный алгоритм выполняется в несколько этапов.

1. Прямой ход: исключение элементов, расположенных ниже диагональных (поддиагональных) элементов. Каждая (кроме первой) полоса основной матрицы содержит по три неизвестных в каждой строке и обрабатывается в соответствующем потоке с номером k. Реализация расчетов прямого хода приведет к образованию в каждой из этих полос дополнительного столбца новых трех коэффициентов.

   Вид СЛАУ после прямого хода:

$$x_i = \alpha_{i+1} x_{i+1} + \beta_{i+1} + \gamma_{i+1} x_{pm-1}, i = (p-1)m, \ldots, pm - 1.$$

   По итогам прямого хода алгоритма для каждой полосы размера $m = [n/p]$ (т. е. в каждом потоке) будут получены прогоночные коэффициенты α, β и γ по формулам:

$$\alpha_1 = \frac{-b_0}{c_0}, \quad \beta_1 = \frac{f_0}{c_0}, \quad \gamma_1 = \frac{-a_0}{c_0},$$
$$\alpha_{i+1} = \frac{-b_i}{a_i\alpha_i + c_i}, \quad \beta_{i+1} = \frac{f_i - a_i\beta_i}{a_i\alpha_i + c_i}, \quad \gamma_{i+1} = \frac{-a_i\gamma_i}{a_i\alpha_i + c_i}, \quad i = 1,..,m-1. \tag{7}$$

В случае системы уравнений для первой полосы (p = 1) верны формулы правой прогонки (2).

2. Обратный ход: исключение наддиагональных элементов в каждой полосе.

Вид СЛАУ после обратного хода:

$$x_i = m_{i+1}x_{pm-1} + l_{i+1} + k_{i+1}x_{m-1}, i = pm-1,..,m(p-1).$$

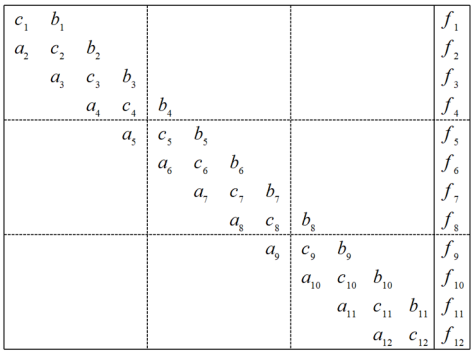
Формулы для прогоночных коэффициенты m, l, k после обратного хода:

$$m_{m-1} = \alpha_{m-i}, \; l_{m-1} = \beta_{m-i}, \; k_{m-1} = \gamma_{m-i},$$
$$m_i = \alpha_i m_{i+1}, \; l_i = \alpha_i l_{i+1} + \beta_i, \; k_i = \alpha_i k_{i+1} + \gamma_i, \; i = m-2, \ldots, 1. \tag{8}$$

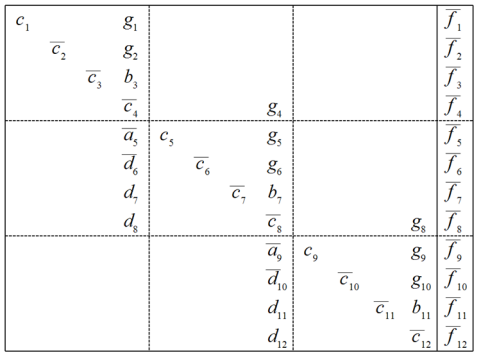
В результате матрица исходной системы уравнений (рис. 1, n = 12, p = 3) принимает вместо трехдиагонального блочный вид, где уравнения каждой полосы матрицы (кроме первой) содержат по 3 неизвестных.

3. Формирование для каждого блока отдельной СЛАУ размера (p×p), решение которой находится классическим последовательным методом прогонки. Каждая СЛАУ имеет трехдиагональную матрицу, элементами которой являются коэффициенты уравнений для нижних границ каждой полосы:

$$A_i x_{i-1} - B_i x_i + C_i x_{i+1} = F_i, i = 1, \ldots, p-1,$$



а)



б)

Рис. 1. Вид матрицы СЛАУ:
а) исходный, б) после обратного хода метода блочной прогонки

Fig. 1. SLAE (system of linear algebraic equations) matrix: a) initial,
b) after the reverse run of the block sweep method

с коэффициентами:

$$A_1 = 0, \qquad B_1 = -\alpha_m \cdot m_{m+1} + \beta_m, \qquad C_1 = 1 - \alpha_m \cdot k_{m+1},$$

$$F_1 = \alpha_m \cdot l_{m+1} + \beta_m, \quad A_p = -\gamma_{p \cdot m}, \quad B_p = 1, \quad C_p = 0, \quad F_p = \beta_{p \cdot m},$$

$$A_i = -\gamma_{m(i+1)}, \qquad B_i = -\alpha_{m(i+1)} \cdot m_{m(i+1)+1}, \tag{9}$$

$$C_i = 1 - \alpha_{m(i+1)} \cdot k_{m(i+1)+1}, \quad F_i = \alpha_{m(i+1)} \cdot l_{m(i+1)+1} + \beta_{m(i+1)}.$$

4. На основе вычисленных граничных значений неизвестных для каждой полосы находятся значения внутренних переменных в каждом блоке по формулам:

$$x_i = m_{i+1} \cdot x_{j \cdot m-1} + k_{i+1} \cdot x_{(j-1)m-1} + l_{i+1}, \quad j = 1, \dots p, \qquad i = (j-1)m, \dots, jm - 1. \tag{10}$$

Очевидно, что в силу взаимной независимости эти значения возможно вычислять в разных потоках.

Итоговую общую трудоемкость метода горизонтально-блочной прогонки можно оценить как $T_p = 20m + 10p$, где m — число полос в матрице, p — число процессоров.

Показатели ускорения S_p и эффективности E_p параллельных вычислений в соответствии с [6] оцениваются по формулам:

$$S_p = \frac{T_1}{T_p} = \frac{10n}{20\frac{n}{p}+10p} = p\,\frac{10n}{20n+p^2}, \qquad E_p = \frac{S_p}{p} = \frac{10n}{20n+p^2}.$$

T_1 — время работы последовательного алгоритма, T_p — время работы параллельного алгоритма.

Применение алгоритмов прогонки

Описанные выше алгоритмы распараллеливания метода прогонки для СЛАУ с трехдиагональной матрицей легли в основу разработанного программного комплекса, реализующего идеи распараллеливания вычислительных процессов. Получаемые результаты расчетов одновременно в параллельном режиме обрабатываются с помощью встроенных библиотек и представляются в виде изолиний и тепловых карт.

Для оценки реальной эффективности используемых параллельных аналогов метода прогонки для решения СЛАУ с трехдиагональной основной матрицей были проведены вычислительные эксперименты в среде разработки MS Visual Studio 2017 на компьютере, имеющим четырехъядерный процессор Intel Xeon E5520 (2.27 GHz) Intel®Core™ *i*5-825OU и объем памяти 8 Gb.

При выполнении численного эксперимента для исследования разных технологий параллельных вычислений были разработаны две программы на языках программирования C# и C++ в среде MS Visual Studio с использованием технологий Thread и OpenMP соответственно.

Переход к точечным оценкам

Для тестирования алгоритмов распараллеливания решения СЛАУ с трехдиагональной матрицей расчеты выполнялись для систем, матрица коэффициентов которых соответствовала т. н. диагональному преобладанию, в случае которого элементы главной диагонали равны удвоенной сумме элементов соответствующей строки. Выбор правой части СЛАУ был обусловлен необходимостью существования точного решения.

Для обеспечения равномерной нагрузки на процессоры при задействовании разного количества вычислительных нитей рассматривались задачи размера n, кратного 12. Поскольку при каждом запуске алгоритма количество квантов процессорного времени, выделяемых отдельным потокам, не является детерминированным, то для каждого алгоритма, каждого размера матрицы и каждого числа используемых потоков выполнялось по 10 запусков однотипных вычислений, по которым определялось среднее время работы алгоритма.

В ситуациях неоднородной плотности процессорного времени, выделяемого разным потокам, важной является оценка разброса получаемых значений времени, затраченного на выполнение алгоритма. Соответственно, в ходе экспериментов вычислялся доверительный интервал для полученного среднего времени расчета.

Статистические показатели рассчитывались по следующим формулам:

$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$ — среднее арифметическое значение времени расчета (n — размер выборки).

Дисперсия — $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(T_i - \bar{T})^2$.

Стандартное отклонение по выборке $S_0 = \sqrt{\frac{\sum_{i=1}^{n}(T_i - \bar{T})^2}{n-1}}$.

Соответственно нижняя и верхняя граница доверительного интервала $T_{1,2}$ = $= \bar{T} \pm \frac{S_0}{\sqrt{n}} \cdot c_\gamma$, где γ — доверительная вероятность (обычно равна 0,9, 0,95 или 0,99), $c_\gamma = \Phi^{-1}((1+\gamma)/2)$ — обратное значение функции нормального распределения (функции Лапласа).

В результате возникающий разброс значений времени работы алгоритма оказался незначительным. Например, для 10 испытаний (n = 10) при среднем значении $\bar{T}$ =248,8 (мсек) получен доверительный интервал [224,2-273,4].

Таким образом, влияние внутренних процессов на компьютере на время расчета по рассматриваемым моделям составляет порядка 10%. Учитывая это, в дальнейшем будет использоваться не интервальная, а точечная оценка времени.

Результаты

Приведем результаты вычислительного эксперимента на основе различных подходов к распараллеливанию вычислений в виде следующих характеристик расчетов: время работы параллельных алгоритмов T_p (мсек) и ускорение работы программ S_p относительно последовательного алгоритма.

Размерность задачи N варьировалась от 1.0E+02 до 4.2E+03 по двум пространственным переменным.

При сравнении разных технологий организации распараллеливания расчеты проводились для разного количества thread-потоков и OpenMP-потоков в разумном диапазоне. Ниже представлены результаты, полученные при использовании алгоритмов последовательной правой прогонки и встречной прогонки (параллельный вариант предусматривает использование двух потоков $p = 2$).

При использовании технологии Thread при малой размерности массива ($N < 2.4\times10^3$) параллельный алгоритм оказался более затратен по времени. При значениях $N > 2.4\cdot10^3$ встречная прогонка в распараллеливании на два потока реализуется за меньшее время. Величина ускорения $S_p > 1$ начинается со значений $N > 2.4\times10^3$ и становится равной ~1,65 для $N = 3.0\times10^3$. Данное значение становится неизменным при дальнейшем увеличении размерности массива.

При использовании технологии OpenMP результаты расчетов соответствует данным, полученным с помощью технологии Thread: встречная прогонка в распараллеливании на двух потоках реализуется за меньшее время, чем последовательный вариант (уменьшение времени расчета начинается со значений $N > 8.0\times10^2$, что на порядок меньше, чем потоковый вариант). Величина ускорения $S_p > 1$ начинается со значений $N > 8.0\times10^2$ и становится равной ~1,765 для $N > 9.0\times10^3$.

Более подробно рассмотрим результаты реализации параллельного блочного метода прогонки с числом потоков $p > 2$.

Для выявления временных затрат для каждого этапа алгоритма горизонтально-блочной прогонки и определения наиболее нагруженного из них после завершения каждого этапа выполнялись отсчеты временных точек.

Во всех испытаниях (расчет для $N = 3.0\times10^3$ и $p = 4$) время счета в параллельном режиме более чем в 1,5 раза меньше последовательного варианта. При этом наиболее затратным по времени является первый этап метода блочной прогонки, в котором выполняется прямой и обратный ход алгоритма блочной прогонки для каждой полосы (потока).

В таблице 1 приведены результаты измерения времени расчета T_p в зависимости от размера задачи N и числа потоков p, полученные при использовании технологии thread.

Результаты оценки времени работы алгоритма горизонтально-блочной прогонки показывают, что при $N < 3.6\times10^4$ увеличение числа блоков приводит к увеличе-

нию общего времени работы. Ожидаемое сокращение временных затрат (ускорение расчетов $S_p > 1$) достигается при $N > 3.6 \times 10^4$ и числа использованных потоков $p > 6$.

Приведем результаты вычислительного эксперимента приведенных характеристик с использованием технология OpenMP-потоков.

В таблице 2 приведены результаты измерения времени расчета T_p (мсек) в зависимости от размера задачи N и числа потоков p.

Таблица 1

Время работы алгоритма горизонтально-блочной прогонки T_p (мсек) в зависимости от размера задачи N и числа потоков p (технологии Thread)

Table 1

The running time of the horizontal block matrix algorithm T_p (ms) depending on the size of the task N and the number of threads p (Thread technology)

N-число элементов массива ($\times 10^3$)	Число потоков p				
	1	4	6	8	10
7,2	331	364	417	510	594
12,0	594	683	691	763	788
24,0	907	1 111	1 145	1 383	1 286
36,0	1 843	2 563	2 296	1 829	1 717
42,0	2 912	3 436	2 947	2 185	2 000

Таблица 2

Время работы алгоритма горизонтально-блочной прогонки T_p (мсек) в зависимости от размера задачи N и числа потоков p (технология OpenMP)

Table 2

The running time of the horizontal block matrix algorithm T_p (ms) depending on the size of the task N and the number of threads p (OpenMP technology)

N-число элементов массива ($\times 10^3$)	T_p в зависимости от числа потоков p						
	1	2	3	4	5	6	8
2,4	154,2	137,2	82,4	84,9	86,2	90,2	89,7
3,6	249,5	207,7	126,2	125,8	127,7	135	137,9
4,8	302,2	277,5	178,3	171,2	183,1	160,2	179,0
6,0	356,2	335,1	221,9	217,9	215,8	183,8	222,3
7,2	415,2	404,5	257,1	252,6	259,7	259,6	264,5
8,4	480,9	468,1	296,2	294,5	306,2	302,2	313,1
9,6	575,3	533,9	341,8	343,8	349,1	359,6	353,9

Результаты проведенных расчетов показывают, что при использовании технологии OpenMP ожидаемое сокращение времени расчета при увеличении количества блоков, на которые разбивается исходная система (каждый блок обрабатывается в отдельном потоке, число которых p варьируется от 2 до 6) достигается для значений размерности $N > 4.8\times10^3$.

Наибольшее ускорение (фактически двукратное) достигается при использовании числа потоков p от 2 до 4. При изменении числа потоков p от 4 до 8 расчетное время практически стабилизируется. При использовании числа потоков больше количества ядер процессора происходит очевидный рост временных затрат.

Как видно из таблицы 2, величина ускорения S_p больше 1 во всех испытаниях. Так, при разбиении расчетной зоны на 4 потока в случае $N = 3{,}6\times10^3$ измеренное время расчета составляло $T_p = 125{,}8$ мсек, в то время как при последовательном варианте потребовалось $T_1 = 249{,}5$ мсек, что соответствует достигнутому ускорению $S_p \geq 1{,}98$, т. е. по сравнению с применением последовательного алгоритма достигается двукратный выигрыш во времени.

Оценка эффективности распараллеливания для разных ситуаций

Расчеты, проведенные в рамках использования технологий Thread и OpenMP для организации многопоточного решения матричных уравнений указанного типа, показали, что при малых размерностях основной матрицы применение параллельных методов прогонки (встречная прогонка и горизонтально-блочная прогонка) не дают ожидаемого уменьшения времени расчета. Реальное ускорение достигается при заметно большей размерности матрицы, причем для технологии Thread такое ускорение начинается для $N > 3.6\times10^4$, а при использовании OpenMP критическое значение размерности на порядок меньше. При этом при дальнейшем увеличении размерности основной матрицы реальная эффективность параллельного подхода становится более заметной и стремится к теоретическому максимуму.

При распараллеливании на Thread-потоках получено ускорение в 1,75 раза, на OpenMP-потоках — в 1,9 раза. Увеличение числа используемых потоков до величины большей, чем число ядер процессора, не приводит к ускорению расчетов, но при этом обслуживание дополнительных потоков требует дополнительных временных затрат.

Применение технологий распараллеливания позволяет, не прерывая расчеты по приведенным алгоритмам, визуализировать полученные результаты на каждом шаге расчетного процесса (рис. 2).

В качестве результата визуализации была выбрана карта изолиний. Данная карта строится на основе результатов распределения температуры в тех слоях двумерной области, для которых расчеты уже завершены. Базовое значение температуры, шаг изолиний и градиент цвета задаются заранее. Алгоритм построения изолиний предусматривает отслеживание наличия прохождения некоторого витка семейства изолиний в дискретной близости каждого пикселя битовой карты изображений, соответствующего точке расчетной двумерной области, для которой вычисления температуры на данный момент завершены.

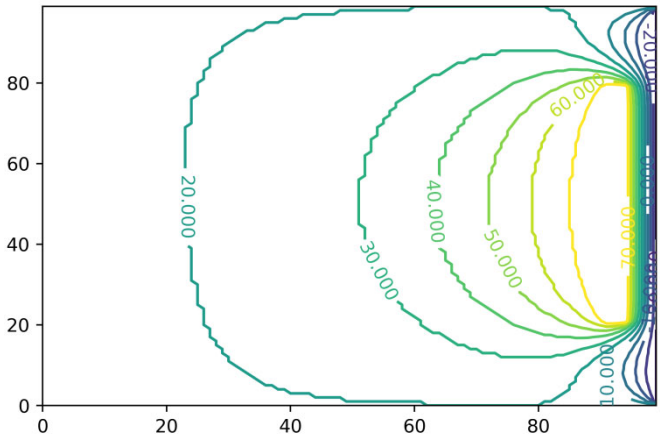


Рис. 2. Карта изолиния для задачи распределения температуры

Fig. 2. Isoline map for the problem of temperature distribution

Хотя данный алгоритм построения карты изолиний, последовательно перебирающий пиксели изображения, достаточно хорошо распараллеливается на несколько потоков, такое распараллеливание было признано необязательным в силу того, что алгоритм расчета распределения температуры для базовой задачи движется послойно вдоль оси *y*, тем самым достаточно рассмотрения только тех пикселей, которые соответствуют последнему рассчитанному слою. При обнаружении замыкания отдельного витка изолинии на себя или на границы области в отдельном потоке формируется надпись с указанием температуры данного витка.

Выводы

В работе рассмотрен подход к повышению эффективности выполнения расчетных алгоритмов в задачах, приводимых к решению систем линейных алгебраических уравнений с трехдиагональной матрицей, за счет распределения вычислительной нагрузки на каждый из процессоров многопроцессорных компьютеров. Рассмотрены схемы распараллеливания решения СЛАУ и выполнен анализ их вычислительной эффективности.

Развитие представленного подхода представляется целесообразным и позволит исследователю в рамках одного программного продукта получать результаты численного моделирования с использованием различных алгоритмов параллельных расчетов и визуализацией результатов для динамического отслеживания исследуемого процесса.

СПИСОК ЛИТЕРАТУРЫ

1. Гергель В. П. Современные языки и технологии параллельного программирования / В. П. Гергель. М.: Издательство Московского университета. 2012. 408 с.

2. Заручевская Г. В. Реализация решения разностной схемы расщепления двумерного уравнения теплопроводности в мелкозернистом локально- параллельном стиле программирования / Г. В. Заручевская // Известия Волгоградского государственного технического университета. 2008. № 2 (40). С. 16-19.
3. Образовательный комплекс «Параллельные численные методы». Лекционные материалы / К. А. Баркалов // Нижегородский государственный университет им. Н. И. Лобачевского. Факультет вычислительной математики и кибернетики. 2011. URL: http://www.hpcc.unn.ru/?doc=491 (дата обращения: 18.01.2021).
4. Самарский А. А. Теория разностных схем / А. А. Самарский. Наука. М.: 1989. 616 с.
5. Ступников А. А. Разработка интеллектуального программного обеспечения для обучения студентов приемам распараллеливания вычислений / А. А. Ступников, Н. М. Гаврилова // Международный научно-исследовательский журнал. 2020. № 5 (95). Часть 3. С. 92-96. DOI: 10.23670/IRJ.2020.95.5.100
6. Федоров А. А. Метод двухуровневого распараллеливания прогонки для решения трехдиагональных линейных систем на гибридных ЭВМ с многоядерными сопроцессорами / А. А. Федоров, А. Н. Быков // Вычислительные методы и программирование. 2016. Том 17. Вып. 3. С. 234-244.
7. Яненко Н. Н. Об организации параллельных вычислений и «распараллеливании» прогонки / Н. Н. Яненко, А. Н. Коновалов, А. Н. Бугров, Г. В. Шустов // Численные методы механики сплошной среды. 1978. Том 9. № 7. С. 139-146.

Natalia M. GAVRILOVA[1]
Yuri A. PLOTONENKO[2]
Andrey A. STUPNIKOV[3]

UDC 519.63

DEVELOPING INTELLIGENT SOFTWARE FOR COMPUTING PARALLELIZATION RESEARCH

[1] Cand. Sci. (Phys.-Math.), Associate Professor,
Department of Software, University of Tyumen
n.m.gavrilova@utmn.ru; ORCID: 0000-0002-8697-5639

[2] Cand. Sci. (Ped.), Associate Professor,
Department of Software, University of Tyumen
y.a.plotonenko@utmn.ru

[3] Cand. Sci. (Phys.-Math.), Associate Professor,
Department of Software, University of Tyumen
a.a.stupnikov@utmn.ru; ORCID: 0000-0001-5201-1260

Abstract

One of the most important ways of improving the speed of complex task solving is employing a multiprocessor computational system.

This paper describes the experience of software development for research management and solving educational problems using parallel computing technologies.

The authors describe approaches to computation parallelization using a multiprocessor system with shared memory within a task of finding a numerical root of a system of linear equations with a tridiagonal coefficient matrix that appears when solving a boundary problem for a partial differential equation of parabolic type, the heat equation.

Additionally, the approaches to parallelization implementation of the tridiagonal matrix method for the heat equation in the two-dimensional case within a numerical root-finding algorithm using the alternating-direction implicit method for a multiprocessor system with shared memory are described.

Citation: Gavrilova N. M., Plotonenko Yu. A., Stupnikov A. A. 2021. "Developing intelligent software for computing parallelization research". Tyumen State University Herald. Physical and Mathematical Modeling. Oil, Gas, Energy, vol. 7, no. 3 (27), pp. 152-169.
DOI: 10.21684/2411-7978-2021-7-3-152-169

A finite-difference method of variable directions is used to find a numerical root of a heat equation in the two-dimensional case.

Sequential and parallel algorithms (two-sided Thomas algorithm and multithread horizontal block Thomas algorithm) that fit an execution on computational systems with shared memory have been used to implement a tridiagonal matrix method.

Two parallel computation organization technologies for computational systems with shared memory have been used for computation parallelization: one based on OpenMP technology and one using .NET framework facilities. The parallelization process and load balance serving have been performed by means of the environment in the first case as manual operation of threads parallelization process is allowed in the latter one.

As an assessment of the described approach performance, the calculation time for sequential and parallel algorithms is given depending on the task's size and the number of threads used.

Comparison of the considered parallelization algorithms and implementation technologies is performed based on the analysis of the resulting acceleration. This paper shows that total computation time is several times smaller and calculation acceleration is several times bigger when using Thread instead of OpenMP.

An application has been developed that allows obtaining a visual result of modelling of process of temperature propagation in the study area using parallel calculation technologies in real time.

Keywords

System of linear equations, tridiagonal matrix, tridiagonal matrix algorithm (TDMA), parallelization algorithms, parallelization algorithms, block algorithm, finite-difference scheme.

DOI: 10.21684/2411-7978-2021-7-3-152-169

REFERENCES

1. Gergel V. P. 2012. Modern Languages and Technologies of Parallel Programming. Moscow: MSU Publishing. 408 pp. [In Russian]
2. Zaruchevskaya G. V. 2008. "Implementation of the solution of difference scheme of a two-dimensional heat conduction equation splitting with fine-grained local parallel programming style". Izvestia Volgogradskogo gosudarstvennogo tekhnicheskogo universiteta, no. 2 (40), pp. 16-19. [In Russian]
3. Barkalov K. 2011. "Educational complex 'Parallel Numerical Methods'. Lecture materials". Lobachevsky State University of Nizhny Novgorod. Accessed 18 January 2021. http://www.hpcc.unn.ru/?doc=491 [In Russian]
4. Samarsky A. A. 1989. Theory of Difference Schemes. Moscow: Nauka. 616 pp. [In Russian]
5. Stupnikov A. A., Gavrilova N. M. 2020. "Development of intellectual software to train students to computate parallelly". International Research Journal, no. 5 (95), part 3, pp. 92-96. DOI: 10.23670/IRJ.2020.95.5.100 [In Russian]

6. Fedorov A. A, Bykov A. N. 2016. “A method of two-level parallelization of the Thomas algorithm for solving tridiagonal linear systems on hybrid computers with multicore coprocessors”. Numerical Methods and Programming, vol. 17, no. 3, pp. 234-244. [In Russian]
7. Yanenko N. N., Konovalov A. N., Bugrov A. N., Shustov G. V. 1978. “On the organization of parallel computations and “parallelization” of the run”. Chislennye metody mekhaniki sploshnoy sredy, vol. 9, no. 7, pp. 139-146. [In Russian]