УДК 519.67 **DOI: 10.14529/cmse210303**

ПОДХОД К ОЦЕНКЕ ЛОКАЛЬНОСТИ ЗЕРНИСТЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ, ЛОГИЧЕСКИ ОРГАНИЗОВАННЫХ В ДВУМЕРНУЮ СТРУКТУРУ*

© 2021 А.А. Толстиков[1], С.В. Баханович[2], Н.А. Лиходед[1]

[1]*Белорусский государственный университет*
(220030 Республика Беларусь, Минск, пр. Независимости, д. 4),
[2]*Институт математики НАН Беларуси*
(220072 Республика Беларусь, Минск, ул. Сурганова, д. 11)
E-mail: tolstikov@bsu.by, bsv@im.bas-net.by, likhoded@bsu.by

Поступила в редакцию: 31.07.2021

При реализации алгоритмов на многопроцессорных вычислительных устройствах важнейшую роль для достижения высокой производительности играет локальность — вычислительное свойство алгоритма, отражающее степень использования памяти с быстрым доступом. Для суперкомпьютеров с распределенной памятью быстрой считается локальная память вычислительного узла. Параллельные алгоритмы с меньшим объемом и лучшей структурой коммуникационных операций обладают лучшей локальностью. В работе на основе схемы расщепления с весами построен новый параллельный алгоритм численного решения трехмерного линейного уравнения теплопроводности. Алгоритм ориентирован на компьютеры с распределенной памятью, сочетает конвейерный и естественый параллелизм, использует 2D структуру процессов. Схема расщепления обладает естественным параллелизмом. Ранее для случая 1D структуры процессов было показано, что использование конвейерного параллелизма вместо части естественного параллелизма приводит к меньшим объемам и лучшей структуре коммуникационных операций. Построенный 2D алгоритм обобщает известный 1D алгоритм. Использование двумерных структур позволяет уменьшить объем и улучшить структуру коммуникационных операций, уменьшить время разгона и торможения вычислений. Поэтому 2D алгоритм обладает лучшей локальностью по сравнению с использованием 1D структуры процессов. Вычислительные эксперименты на суперкомпьютере показали преимущество нового параллельного алгоритма. По аналогии с представленным алгоритмом можно получить и исследовать параллельные алгоритмы для других схем метода дробных шагов. На примере алгоритма, реализующего схему расщепления, представлен подход к получению асимптотических оценок объема коммуникационных операций зернистых (т.е. уровня макроопераций) параллельных вычислительных процессов, логически организованных в двумерную структуру. Оценки могут быть использованы для сравнения коммуникационных затрат при получении альтернативных вариантов параллельных алгоритмов.

Ключевые слова: параллельные вычисления, многопроцессорная вычислительная система с распределенной памятью, уменьшение обменов данными, метод дробных шагов.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Толстиков А.А., Баханович С.В., Лиходед Н.А. Подход к оценке локальности зернистых вычислительных процессов, логически организованных в двумерную структуру // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 37–55. DOI: 10.14529/cmse210303.

Введение

В качестве целевого компьютера для реализации алгоритмов будем рассматривать многопроцессорную вычислительную систему с распределенной памятью. При многопроцессорной обработке памятью с быстрым доступом считается локальная память процессора, при однопроцессорной — кэш-память. Локальность алгоритма — это вычислительное свойство алгоритма, отражающее степень использования памяти с быстрым доступом. Использова-

* Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2021».

ние локальности играет важнейшую роль для достижения высокой эффективности реализации алгоритмов на многопроцессорных вычислительных устройствах [4].

Параллельные алгоритмы для компьютеров с распределенной памятью должны быть зернистыми: множество операций алгоритма должно быть разбито на множества, называемые зернами вычислений. Можно использовать тайлинг (tiling) — преобразование алгоритма для получения макроопераций-тайлов [18, 20]. Операции одного тайла выполняются атомарно, как одна единица вычислений, а обмен данными происходит массивами.

Локальность параллельного алгоритма, предназначенного для реализации на компьютерах с распределенной памятью, характеризует коммуникационные затраты: чем меньше при заданном числе вычислительных ядер суммарный объем операций обмена данными, тем лучше локальность. Задачей исследования локальности параллельного алгоритма является оценка числа и объема коммуникационных операций. На локальность влияет также структура операций обмена данными: число и топология («близость») процессов, вовлеченных в групповые коммуникации. Хорошей локальности можно достичь путем преобразования алгоритмов и/или удачного распределения операций и данных между процессорами [1, 3, 8, 9, 16, 18, 20].

Целью этой работы является разработка и исследование локальности нового параллельного алгоритма, реализующего схему расщепления с весами численного решения трехмерного линейного уравнения теплопроводности. Трехшаговая схема расщепления [6, 15] является одной из наиболее употребительных схем метода дробных шагов численного решения трехмерных линейных и квазилинейных уравнений теплопроводности. Схема абсолютно устойчива, имеет второй порядок аппроксимации. К преимуществам схемы расщепления относится также простота по сравнению с другими схемами метода дробных шагов, также обладающими свойствами сильной устойчивости и второго порядка точности.

Предлагаемый алгоритм ориентирован на параллельные компьютеры с распределенной памятью, использует 2D структуру процессов и обладает лучшей локальностью по сравнению с известными алгоритмами [14], использующими 1D структуры процессов. Схема расщепления обладает естественным параллелизмом, но при реализации на параллельных компьютерах с распределенной памятью использование этого естественного параллелизма приводит к большим коммуникационным издержкам. Так же, как и 1D алгоритм, предложенный в [14], новый 2D алгоритм использует конвейерный параллелизм вместо части естественного параллелизма, что приводит к меньшим объемам и лучшей структуре коммуникационных операций. Сформулировано и обосновано утверждение, устанавливающее объем и структуру коммуникационных операций алгоритма. По аналогии с представленным алгоритмом можно получить и исследовать 2D параллельные алгоритмы и для других схем метода дробных шагов.

Фактически на примере алгоритма, реализующего схему расщепления, представлен подход к организации параллельных двумерных вычислительных процессов и получению асимптотических оценок объема коммуникационных операций. Подход использует анализ информационных зависимостей, порождающих коммуникационные операции, и основан на сформулированных ранее утверждениях для случая вычислительных процессов, логически организованных в одномерную структуру.

Статья структурирована следующим образом. В разделе 1 дан обзор работ по тематике исследования. В разделе 2 даны разностная схема расщепления численного решения трехмерного уравнения теплопроводности и псевдокод основной части алгоритма, реализующего рассмотренную схему, в котором указаны циклы, итерации которых заведомо можно выполнять независимо. В разделе 3 приведены необходимые для дальнейших исследований функции, задающие информационные связи между операциями рассмотренного алгоритма. В разделе 4 предложен способ получения 2D зернистых (т.е. уровня макроопераций-тайлов) вычислительных процессов, отличительной особенностью которого является предположение независимости первой и второй координат 2D процессов. В разделе 5 задано распределение операций алгоритма, реализующего рассмотренную схему расщепления, между 2D зернистыми процессами; это распределение операций приводит к конвейерному параллелизму вместо части естественного параллелизма. В разделе 6 оценивается объем и структура коммуникационных операций нового параллельного алгоритма, приведен его псевдокод. В разделе 7 обсуждаются результаты вычислительных экспериментов. В заключении суммируются основные результаты, намечаются направления дальнейших исследований.

1. Обзор работ

Как уже отмечалось, при реализации алгоритмов на многопроцессорных вычислительных устройствах для достижения высокой производительности важнейшую роль играет использование локальности. Отметим некоторые подходы к оптимизации использования иерархической памяти, уменьшению числа и объема коммуникационных операций при использовании многоядерных CPU, компьютеров с распределенной памятью, графических ускорителей (GPU).

Очевидным приемом является выявление независимых частей алгоритма [3, 8, 19]. В случае обнаружения таких частей все ядра параллельной вычислительной системы могут работать независимо друг от друга, коммуникационные операции не требуются. На практике, однако, алгоритм не часто допускает декомпозицию на независимые части. Еще один подход заключается в использовании макроопераций-тайлов (получение блочных версий алгоритмов) [8, 16, 17, 20]. Операции одного тайла выполняются атомарно, как одна единица вычислений. При использовании суперкомпьютеров с распределенной памятью обмен данными происходит массивами, тайлинг позволяет увеличить пакет передаваемых данных и уменьшить частоту коммуникаций, минимизировать накладные расходы на обмены данными. Разбиение множества операций на макрооперации-тайлы является очень полезным преобразованием также при реализации алгоритмов на многоядерных персональных компьютерах и на графических ускорителях. В работах [9, 18, 19] предлагаются методы получения аффинных преобразований, которые позволили бы уменьшить число информационных связей между множествами операций разбиваемого на части алгоритма. Операциям алгоритма назначается новый порядок выполнения, учитывающий параллелизм и локальность; такие преобразования приводят к уменьшению коммуникационных операций, так как операции обмена данными порождаются информационными связями.

В работах [2, 7, 21, 22] улучшение локальности значительно повышает скорость выполнения расчета при параллельной реализации сеточных задач.

В работе [11] для случая 1D процессов предложены и доказаны утверждения, позволяющие оценить коммуникационные затраты при реализации алгоритмов на параллельных компьютерах с распределенной памятью; получены выражения, характеризующие число данных, для которых требуются коммуникации, и число процессов, вовлеченных в пересылки этих данных. Использование двумерных структур связано, по сравнению с одномерными структурами, с дополнительными ограничениями при реализации алгоритмов на суперкомпьютерах с распределенной памятью: для возможности организовать полностью загруженные работой параллельные вычислительные процессы требуется произвести тайлинг по трем координатам многомерного итерационного пространства, в то время как для случая одномерной структуры достаточно произвести тайлинг по двум координатам. Но часто, как и в этой работе, случай использования 2D процессов имеет ряд преимуществ: позволяет уменьшить объем коммуникационных операций, уменьшить разгон и торможение вычислений, получить большее число вычислительных процессов (что важно, если число итераций по одной координате сравнительно небольшое) [10].

2. Вычислительный алгоритм, реализующий разностную схему расщепления

Рассмотрим в области $Q_T = G \times [0 < t \le T]$, $G = [0 < x_1 < l_1] \times [0 < x_2 < l_2] \times [0 < x_3 < l_3]$, трехмерное уравнение теплопроводности:

$$\frac{\partial u}{\partial t} = Lu + f(x_1, x_2, x_3, t), \;\; L = \sum_{m=1}^{3} L_m, \;\; L_m = k \frac{\partial^2 u}{\partial x_m^2}, \;\; k{=}\text{const}{>}0, \tag{1}$$

с начальными и краевыми условиями

$$u(x_1, x_2, x_3, 0) = u_0(x_1, x_2, x_3), \; (x_1, x_2, x_3) \in G, \tag{2}$$

$$u(x_1, x_2, x_3, t)\Big|_{S_t} = \mu(x_1, x_2, x_3, t), \quad (x_1, x_2, x_3, t) \in S_t \equiv \partial Q_T. \tag{3}$$

Введем на отрезке $[0{<}t{\le}T]$ сетку узлов $\omega_\tau = \{t_j = j\tau, j = 0, 1, ..., j_0, j_0\tau = T\}$, в области G и на ее границе введем сетку узлов $\overline{\omega}_h = \{x_1^{(i_1)} = i_1 h_1, \;\; i_1 = 0, 1, ..., N_1, \;\; N_1 h_1 = l_1, \;\; x_2^{(i_2)} = i_2 h_2, \;\; i_2 = 0, 1, ..., N_2, \;\; N_2 h_2 = l_2, \;\; x_3^{(i_3)} = i_3 h_3, \;\; i_3 = 0, 1, ..., N_3, \;\; N_3 h_3 = l_3\}$.

Обозначим $\Lambda_m y^j = \frac{y_{i+1}^j - 2y_i^j + y_{i-1}^j}{h_m^2}$, где индекс i соответствует узлам x_m, остальные индексы для простоты опущены; $\Lambda_m y^j$ аппроксимирует вторую производную по направлению x_m в точках временного слоя j ($t{=}j\tau$) со вторым порядком точности. Для численного решения задачи применим схему расщепления с весами [6, 15] (рассмотрен случай $f{=}0$, $k{=}1$, выбраны веса, равные 0.5):

$$\frac{y^{j+1/3} - y^j}{\tau} = \frac{1}{2}\Lambda_1\left(y^{j+1/3} + y^j\right), \;\; \frac{y^{j+2/3} - y^{j+1/3}}{\tau} = \frac{1}{2}\Lambda_2\left(y^{j+2/3} + y^{j+1/3}\right), \;\; \frac{y^{j+1} - y^{j+2/3}}{\tau} = \frac{1}{2}\Lambda_3\left(y^{j+1} + y^{j+2/3}\right).$$

Основную вычислительную часть алгоритма, реализующего рассмотренную трехшаговую разностную схему расщепления с весами, можно представить в виде следующего псевдокода (циклы, итерации которых заведомо можно выполнять независимо, запишем как dopar):

```
do j =1, j0
   dopar i2= 1, N2-1
      dopar i3= 1, N3-1
         do i1= 1, N1-1
S1:         alpha(i1+1)=F1(alpha(i1))
S2:         beta(i1+1)=F2(alpha(i1),beta(i1),y^j(i1,i2,i3),
                          y^j(i1-1,i2,i3),y^j(i1+1,i2,i3))
         do i1= 1, N1-1
S3:         y^{j+1/3}(N1-i1,i2,i3)=alpha(N1-i1+1)y^{j+1/3}(N1-i1+1,i2,i3)+beta(N1-i1+1)
   dopar i1= 1, N1-1
      dopar i3= 1, N3-1
         do i2= 1, N2-1
S4:         alpha(i2+1)=F3(alpha(i2))
S5:         beta(i2+1)=F4(alpha(i2),beta(i2),y^{j+1/3}(i1,i2,i3),
                          y^{j+1/3}(i1,i2-1,i3),y^{j+1/3}(i1,i2+1,i3))
         do i2= 1, N2-1
S6:         y^{j+2/3}(i1,N2-i2,i3)=alpha(N2-i2+1)y^{j+2/3}(i1,N2-i2+1,i3)+beta(N2-i2+1)
   dopar i1= 1, N1-1
      dopar i2= 1, N2-1
         do i3= 1, N3-1
S7:         alpha(i3+1)=F5(alpha(i3))
S8:         beta(i3+1)=F6(alpha(i3),beta(i3),y^{j+2/3}(i1,i2,i3),
                          y^{j+2/3}(i1,i2,i3-1),y^{j+2/3}(i1,i2,i3+1))
         do i3= 1, N3-1
S9:         y^{j+1}(i1,i2,N3-i3)=alpha(N3-i3+1)y^{j+1}(i1,i2,N3-i3+1)+beta(N3-i3+1)
```

Рис. 1. Алгоритм разностной схемы расщепления

Здесь $alpha(i)$ и $beta(i)$ — коэффициенты прогонки, возникающие при решении промежуточных систем линейных алгебраических уравнений. Конечные значения $y^j(i_1,i_2,i_3)$ являются приближенным решением задачи (1)–(3).

3. Информационная структура алгоритма расщепления с весами

Вхождением (a,S_β,q) будем называть q-е вхождение массива a в оператор S_β. Выполнение оператора S_β при конкретных значениях β и вектора параметров цикла J будем называть операцией и обозначать $S_\beta(J)$. Пара вхождений $(a,S_\alpha,1)$ и (a,S_β,q) порождает истинную зависимость $S_\alpha(I)\to S_\beta(J)$, если: $S_\alpha(I)$ выполняется раньше $S_\beta(J)$; $S_\alpha(I)$ переопределяет (изменяет) элемент массива a, а $S_\beta(J)$ использует этот элемент массива в качестве аргумента; между операциями $S_\alpha(I)$ и $S_\beta(J)$ этот элемент не переопределяется. Истинные зависимости и входные данные алгоритма порождают коммуникационные операции.

Зависимости между операциями можно задать функциями вида

$$\overline{\Phi}_{\alpha,\beta}(J)=\Phi_{\alpha,\beta}J+\Psi_{\alpha,\beta}N-\varphi^{\alpha,\beta},$$

$$J\in V_{\alpha,\beta},\ N\in\mathbb{Z}^s,\ \Phi_{\alpha,\beta}\in\mathbb{Z}^{n_\alpha\times n_\beta},\ \Psi_{\alpha,\beta}\in\mathbb{Z}^{n_\alpha\times s},\ \varphi^{\alpha,\beta}\in\mathbb{Z}^{n_\alpha},$$

где $N\in\mathbb{Z}^s$ — вектор внешних переменных алгоритма, s — число внешних переменных, n_α и n_β — глубина вложенности циклов, окружающих операторы S_α и S_β. Функция зависимостей $\overline{\Phi}_{\alpha,\beta}(J)$ позволяет для операции $S_\beta(J)$ найти операцию $S_\alpha(I)$, от которой $S_\beta(J)$ зависит. Функции зависимостей, называемые также покрывающими функциями графа ал-

горитма, являются удобным математическим аппаратом для описания информационных связей между операциями алгоритма [3].

Истинные зависимости представленного алгоритма, реализующего схему расщепления с весами, определяются следующими функциями зависимостей:

$\overline{\Phi}_{1,1}(j,\ i_2,\ i_3,\ i_1)=\overline{\Phi}_{1,2}(j,\ i_2,\ i_3,\ i_1)=\overline{\Phi}_{2,2}(j,\ i_2,\ i_3,\ i_1)=\overline{\Phi}_{3,3}(j,\ i_2,\ i_3,\ i_1)=(j,\ i_2,\ i_3,\ i_1{-}1)$,
$\overline{\Phi}_{1,3}(j,\ i_2,\ i_3,\ i_1)=\overline{\Phi}_{2,3}(j,\ i_2,\ i_3,\ i_1)=(j,\ i_2,\ i_3,\ N_1{-}i_1)$,
$\overline{\Phi}_{9,2}(j,\ i_2,\ i_3,\ i_1)=(j{-}1,\ i_1{-}\varphi,\ i_2,\ N_3{-}i_3)$, где φ равно 0, 1, –1,
$\overline{\Phi}_{4,4}(j,\ i_1,\ i_3,\ i_2)=\overline{\Phi}_{4,5}(j,\ i_1,\ i_3,\ i_2)=\overline{\Phi}_{5,5}(j,\ i_1,\ i_3,\ i_2)=\overline{\Phi}_{6,6}(j,\ i_1,\ i_3,\ i_2)=(j,\ i_1,\ i_3,\ i_2{-}1)$,
$\overline{\Phi}_{4,6}(j,\ i_1,\ i_3,\ i_2)=\overline{\Phi}_{5,6}(j,\ i_1,\ i_3,\ i_2)=(j,\ i_1,\ i_3,\ N_2{-}i_2)$,
$\overline{\Phi}_{3,5}(j,\ i_1,\ i_3,\ i_2)=(j,\ i_2{-}\varphi,\ i_3,\ N_1{-}i_1)$, где φ равно 0, 1, –1,
$\overline{\Phi}_{7,7}(j,\ i_1,\ i_2,\ i_3)=\overline{\Phi}_{7,8}(j,\ i_1,\ i_2,\ i_3)=\overline{\Phi}_{8,8}(j,\ i_1\ i_2,\ i_3)=\overline{\Phi}_{9,9}(j,\ i_1,\ i_2,\ i_3)=(j,\ i_1,\ i_2,\ i_3{-}1)$,
$\overline{\Phi}_{7,9}(j,\ i_1,\ i_2,\ i_3)=\overline{\Phi}_{8,9}(j,\ i_1\ i_2,\ i_3)=(j,\ i_1,\ i_2,\ N_3{-}i_3)$,
$\overline{\Phi}_{6,8}(j,\ i_1,\ i_2,\ i_3)=(j,\ i_1,\ i_3{+}\varphi,\ N_2{-}i_2)$, где φ равно 0, 1, –1.

Матрично-векторное представление некоторых из этих функций:

$$\overline{\Phi}_{1,1}(j,\ i_2,\ i_3,\ i_1)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}j\\i_2\\i_3\\i_1\end{pmatrix}-\begin{pmatrix}0\\0\\0\\1\end{pmatrix},$$

$$\overline{\Phi}_{9,2}(j,\ i_2,\ i_3,\ i_1)=\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&-1&0\end{pmatrix}\begin{pmatrix}j\\i_2\\i_3\\i_1\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}j_0\\N_1\\N_2\\N_3\end{pmatrix}-\begin{pmatrix}1\\\varphi\\0\\0\end{pmatrix},$$

$$\overline{\Phi}_{4,6}(j,\ i_1,\ i_3,\ i_2)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{pmatrix}\begin{pmatrix}j\\i_1\\i_3\\i_2\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}j_0\\N_1\\N_2\\N_3\end{pmatrix},$$

$$\overline{\Phi}_{3,5}(j,\ i_1,\ i_3,\ i_2)=\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&-1&0&0\end{pmatrix}\begin{pmatrix}j\\i_1\\i_3\\i_2\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}j_0\\N_1\\N_2\\N_3\end{pmatrix}-\begin{pmatrix}0\\\varphi\\0\\0\end{pmatrix},$$

$$\overline{\Phi}_{6,8}(j,\ i_1,\ i_2,\ i_3)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&-1&0\end{pmatrix}\begin{pmatrix}j\\i_1\\i_2\\i_3\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}j_0\\N_1\\N_2\\N_3\end{pmatrix}-\begin{pmatrix}0\\0\\\varphi\\0\end{pmatrix}.$$

Функции зависимостей можно найти исходя из определения зависимостей или методом работы [3].

4. Организация и подход к оценке локальности 2D зернистых вычислительных процессов

Будем считать, что алгоритм задан последовательной программой линейного класса [3]. Основную вычислительную часть такой программы составляют циклы произвольной структуры вложенности; границы изменения параметров циклов задаются неоднородными формами, линейными по совокупности параметров циклов и внешних переменных; шаги изменения параметров циклов равны 1. Пусть в гнезде циклов имеется

Θ наборов выполняемых операторов. Под набором операторов обычно понимают один или несколько выполняемых операторов, окруженных одним и тем же множеством циклов. В этой работе, для простоты изложения и не ограничивая общности, будем считать, что оператор S_β составляет «набор» операторов с номером β. Выполняемые операторы линейно упорядочены расположением их в записи алгоритма.

Как уже отмечалось, тайлинг — это преобразование алгоритма для получения макроопераций, называемых также зерном вычислений или тайлами. При тайлинге каждый цикл разбивается на два цикла: глобальный, параметр которого определяет на данном уровне вложенности порядок вычисления тайлов, и локальный, в котором параметр исходного цикла изменяется в границах одного тайла. Если разбиение не производить и все итерации считать принадлежащими глобальному циклу, то получим так называемый глобальный не разбиваемый цикл; если все итерации отнести к локальному циклу, то получим локальный не разбиваемый цикл. Перестановкой и распределением циклов алгоритм преобразуется таким образом, чтобы глобальные циклы были внешними по отношению к локальным.

Размеры тайлов задаются натуральными числами $r_1^\beta,...,r_{n_\beta}^\beta$. Параметр r_ζ^β обозначает число значений параметра цикла j_ζ, приходящихся на один тайл β-го оператора. Если операторы S_α и S_β имеют общий цикл с параметром j_ζ, то $r_\zeta^\beta = r_\zeta^\beta$. Число частей, на которые при формировании тайлов разбивается область значений параметра j_ζ цикла, окружающего β-й оператор, обозначим Q_ζ^β Тайлы нумеруются по каждой координате в пределах от 0 до Q_ζ^β –1, $1 \le \zeta \le n_\beta$. Каждый тайл можно обозначить некоторым вектором J^{gl} или, подробнее, вектором $J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})$.

Рассмотрим следующий способ получения 2D зернистых (т.е. уровня макроопераций-тайлов) вычислительных процессов. Пусть оператор β исходного алгоритма окружен циклами с параметрами j_η и j_ξ. Произведем тайлинг и зафиксируем два глобальных цикла с уровнями вложенности η, ξ. К одному 2D процессу с координатами $(j_\eta^{gl}, j_\xi^{gl})$ отнесем вычисления всех тайлов с одинаковыми значениями функций (для каждого β — одна из функций).

$$\Pr^\beta\left(J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})\right) = (j_\eta^{gl}, j_\xi^{gl}), \tag{4}$$

$$\Pr^\beta\left(J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})\right) = (j_\eta^{gl}, Q_\xi^\beta - j_\xi^{gl} - 1), \tag{5}$$

$$\Pr^\beta\left(J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})\right) = (Q_\eta^\beta - j_\eta^{gl} - 1, j_\xi^{gl}). \tag{6}$$

Обозначим $(p_\eta,p_\xi) = (j_\eta^{gl}, j_\xi^{gl})$. Вычислительный процесс с координатами (p_η,p_ξ) будем обозначать $\Pr_{p_\eta,p_\xi}$.

Будем рассматривать распределение операций между 2D зернистыми вычислительными процессами как два независимых распределения операций между 1D процессами, т.е. будем независимо рассматривать первую и вторую координаты 2D процессов. При таком подходе для оценки коммуникационных затрат можно воспользоваться результатами для 1D процессов [11]. Утверждения, сформулированные в работе [11], позволяют получить асимптотические оценки объема коммуникационных операций для случая 2D процессов, задаваемых функциями вида (4)–(6). Для конкретных параллельных алгоритмов асимптотические оценки можно уточнять (заменить на точные выражения). За-

метим, что если есть «некоординатные» зависимости, то некоторая часть объема коммуникационных операций может учитываться дважды (в этой работе такая ситуация не возникает).

Параметр цикла j_ζ будет обозначать параметр j_η или j_ξ, т.е. под циклом уровня вложенности ζ понимается цикл уровня вложенности η или уровня вложенности ξ.

К одному 1D вычислительному процессу отнесем вычисления тайлов с одинаковыми значениями функций $\Pr^\beta\left(J^{gl}\right)$,$1\leq\beta\leq\Theta$, отображающих тайлы на процессы [5, 12, 13]. Будем полагать

$$\Pr^\beta\left(J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})\right)=j_\zeta^{gl} \tag{7}$$

или

$$\Pr^\beta\left(J^{gl}(j_1^{gl},...,j_{n_\beta}^{gl})\right)=Q_\zeta^\beta-j_\zeta^{gl}-1. \tag{8}$$

Функция $\Pr^\beta$ вида (7) или (8) задает вычислительный процесс $\Pr^\beta\left(J^{gl}\right)$ выполнения операций тайлов J^{gl} с одинаковыми значениями ζ-й координаты j_ζ^{gl} векторов J^{gl}.

Для возможности организовать полностью загруженные работой двумерные параллельные вычислительные процессы вида (4) достаточно, при определенных ограничениях на структуру и длину циклов, произвести тайлинг по трем координатам многомерного итерационного пространства [10]; для случая одномерной структуры вида (7) или (8) достаточно произвести тайлинг по двум координатам [12, 13].

Проиллюстрируем предлагаемый подход в следующем разделе. Рассматриваемый пример имеет и самостоятельное значение.

5. Распределение операций параллельного алгоритма, реализующего схему расщепления на 2D структуре процессов

Пусть $P^\eta\times P^\xi$ — число процессов, предназначенных для реализации алгоритма. Для простоты изложения будем считать числа $r_1=(N_1-1)/P^\eta$ и $r_2=(N_2-1)/P^\xi$ целыми.

Зафиксируем параметр j самого внешнего цикла. Пусть числа Q_3 и r_3 связаны соотношениями $r_3=\left\lceil\left(N_3-1\right)/Q_3\right\rceil$.

Зададим распределение операций между 2D зернистыми (т.е. уровня макроопераций-тайлов) вычислительными процессами. Будем считать, что по первой координате распределение операций, порождаемых операторами S_1, S_2, S_3, определяется циклами уровня вложенности 4, а распределение операций, порождаемых операторами S_4, S_5, S_6 и S_7, S_8, S_9, определяется параллельными циклами уровня вложенности 2; все эти циклы имеют параметр i_1, изменяющийся от 1 до N_1-1. Будем считать, что по второй координате распределение операций, порождаемых операторами S_1, S_2, S_3, определяется параллельным циклом уровня вложенности 2, распределение операций, порождаемых операторами S_4, S_5, S_6, определяется циклами уровня вложенности 4, а распределение операций, порождаемых операторами S_7, S_8, S_9, определяется параллельным циклом уровня вложенности 3; все эти циклы имеют параметр i_2, изменяющийся от 1 до N_2-1.

Опишем подробнее распределение операций между 2D зернистыми вычислительными процессами.

Для операций, порождаемых операторами S_1 и S_2, каждому процессу по первой координате назначим выполнить некоторое число r_1 итераций цикла с параметром i_1: первые r_1 итераций — процессу с номером (первой координаты) 0, следующие r_1 итераций — про-

цессу с номером 1, и так далее. Такой порядок выполнения итераций назовем прямым. Для операций, порождаемых операторами S_1 и S_2, каждому процессу по второй координате также назначим прямой порядок выполнения итераций: r_2 итераций цикла с параметром i_2. Операции этих итераций циклов с параметрами i_1 и i_2, а также r_3 итераций цикла с параметром i_3 составляют зерно вычислений, которое назовем тайлом $1a$ типа. Всего, при фиксированном j, одному процессу $\mathrm{Pr}_{p_\eta,p_\xi}$, $0 \le p_\eta \le P^\eta - 1$, $0 \le p_\xi \le P^\xi - 1$, назначается Q_3 тайлов $1a$ типа:

$$\mathrm{Pr}^{\beta}\left(J^{gl}(j,p_\xi,q_3,p_\eta)\right) = (p_\eta,p_\xi),\ \beta=1,\ \beta=2,\ q_3=0,1,\ldots,Q_3-1. \tag{9}$$

Для операций, порождаемых оператором S_3, каждому процессу по первой координате назначим так называемый обратный порядок выполнения r_1 итераций цикла с параметром i_1: первые r_1 итераций — процессу с номером $P^\eta - 1$, следующие r_1 итераций — процессу с номером $P^\eta - 2$, и так далее, процессу с номером 0 — последние r_1 итераций цикла. Для операций оператора S_3 по второй координате назначим, как и для операторов S_1 и S_2, прямой порядок выполнения итераций цикла с параметром i_2. Операции этих итераций циклов с параметрами i_1 и i_2, а также r_3 итераций цикла с параметром i_3 составляют зерно вычислений, которое назовем тайлом $1b$ типа. Всего, при фиксированном j, одному процессу $\mathrm{Pr}_{p_\eta,p_\xi}$, $0 \le p_\eta \le P^\eta - 1$, $0 \le p_\xi \le P^\xi - 1$, назначается Q_3 тайлов $1b$ типа:

$$\mathrm{Pr}^{\beta}\left(J^{gl}(j,p_\xi,q_3,p_\eta)\right) = (Q_\eta^\beta - p_\eta - 1,\ p_\xi),\ \beta=3,\ q_3=0,1,\ldots,Q_3-1. \tag{10}$$

Цикл с параметром i_1 является параллельным, но вместо этого естественного параллелизма по первой координате процессов используется конвейерный параллелизм. Это приводит к разгону и торможению вычислений, зато значительно улучшается локальность. Под разгоном (торможением) вычислений понимается число единиц времени, когда еще не все процессы начали (закончили) выполнение.

Для операций, порождаемых операторами S_4 и S_5, каждому процессу $\mathrm{Pr}_{p_\eta,p_\xi}$ как по первой, так и по второй координате назначим прямой порядок выполнения итераций (Q_3 тайлов $2a$ типа):

$$\mathrm{Pr}^{\beta}\left(J^{gl}(j,p_\eta,q_3,p_\xi)\right) = (p_\eta,p_\xi),\ \beta=4,\ \beta=5,\ \ q_3=0,1,\ldots,Q_3-1. \tag{11}$$

Для операций, порождаемых оператором S_6, каждому процессу назначим по первой координате прямой порядок выполнения итераций, а по второй координате назначим обратный порядок выполнения итераций (Q_3 тайлов $2b$ типа):

$$\mathrm{Pr}^{\beta}\left(J^{gl}(j,p_\eta,q_3,p_\xi)\right) = (p_\eta,Q_\xi^\beta - p_\xi - 1),\ \beta=6,\ q_3=0,1,\ldots,Q_3-1. \tag{12}$$

Параллельность цикла с параметром i_2 не используется (используется конвейерный параллелизм).

Для операций, порождаемых операторами S_7, S_8, S_9, каждому процессу $\mathrm{Pr}_{p_\eta,p_\xi}$ как по первой, так и по второй координате назначим прямой порядок выполнения итераций (тайл 3-го типа):

$$\mathrm{Pr}^{\beta}\left(J^{gl}(j,p_\eta,p_\xi,0)\right) = (p_\eta,p_\xi),\ \beta=7,\ \beta=8,\ \beta=9. \tag{13}$$

6. Исследование локальности параллельного алгоритма. Псевдокод алгоритма

Уровень вложенности, определяющий распределение операций $S_\beta(J)$ между процессами по первой координате, будем обозначать η^β. Уровень вложенности, определяющий распределение операций $S_\beta(J)$ между процессами по второй координате, будем обозначать ξ^β.

Имеем: η^β=4 для операторов S_β, $1\le\beta\le3$, η^β=2 для операторов S_β, $4\le\beta\le9$; ξ^β=2 для операторов S_β, $1\le\beta\le3$, ξ^β=4 для операторов S_β, $4\le\beta\le6$, ξ^β=3 для операторов S_β, $7\le\beta\le9$.

Утверждение 1. Пусть $N_1-1=N_2-1=N_3-1=M$. Если распределение операций между 2D-процессами задается формулами (9)–(13), то требуются коммуникационные операции только между процессами, номера которых отличаются по первой или по второй координате на 1. Суммарный объем коммуникационных операций, требуемых на одном слое, равен $5(P^\eta+P^\xi-2)M^2$.

Доказательство. Воспользуемся результатами работы [11] , при этом асимптотические оценки для рассматриваемого примера заменим на точные объемы данных.

Пусть вхождение (a,S_β,q) порождает истинную зависимость, задаваемую функцией $\overline{\mathbf{\Phi}}_{\alpha,\beta}(J)$. Обозначим через $\left(\mathbf{\Phi}_{\alpha,\beta}\right)_\zeta$ и $\left(\mathbf{\Psi}_{\alpha,\beta}\right)_\zeta$ строки матриц с номером ζ, а через $e^{(4)}_\zeta$ вектор-строку размера 4, у которой координата с номером ζ равна 1, а остальные координаты нулевые.

Приведем выкладки для второй координаты. Для первой координаты выкладки фактически приведены в работе [14].

Если $\left(\mathbf{\Phi}_{\alpha,\beta}\right)_{\zeta^\alpha}=e^{(4)}_{\zeta^\beta}$, $\left(\mathbf{\Psi}_{\alpha,\beta}\right)_{\zeta^\alpha}$=0, $\varphi^{\alpha,\beta}_{\zeta^\alpha}$=0, то коммуникационных операций не требуется. Данные условия выполняются для функций $\overline{\mathbf{\Phi}}_{1,1}$, $\overline{\mathbf{\Phi}}_{1,2}$, $\overline{\mathbf{\Phi}}_{2,2}$, $\overline{\mathbf{\Phi}}_{3,3}$, $\overline{\mathbf{\Phi}}_{1,3}$, $\overline{\mathbf{\Phi}}_{2,3}$, $\overline{\mathbf{\Phi}}_{7,7}$, $\overline{\mathbf{\Phi}}_{7,8}$, $\overline{\mathbf{\Phi}}_{8,8}$, $\overline{\mathbf{\Phi}}_{9,9}$, $\overline{\mathbf{\Phi}}_{7,9}$, $\overline{\mathbf{\Phi}}_{8,9}$, $\overline{\mathbf{\Phi}}_{9,2}$, поэтому эти функции коммуникационных операций не порождают.

Если $\left(\mathbf{\Phi}_{\alpha,\beta}\right)_{\zeta^\alpha}=e^{(4)}_{\zeta^\beta}$, $\left(\mathbf{\Psi}_{\alpha,\beta}\right)_{\zeta^\alpha}$=0, $\varphi^{\alpha,\beta}_{\zeta^\alpha}$=1 или $\varphi^{\alpha,\beta}_{\zeta^\alpha}$=–1, то коммуникационные операции происходят между процессами, номера которых отличаются на 1. Функции $\overline{\mathbf{\Phi}}_{4,4}$, $\overline{\mathbf{\Phi}}_{4,5}$, $\overline{\mathbf{\Phi}}_{5,5}$, $\overline{\mathbf{\Phi}}_{6,6}$ порождают коммуникационные операции суммарного объема $3(P^\xi-1)M^2$ (функции $\overline{\mathbf{\Phi}}_{4,4}$ и $\overline{\mathbf{\Phi}}_{4,5}$ порождают одну коммуникационную операцию). Функция $\overline{\mathbf{\Phi}}_{3,5}$ порождает коммуникационные операции объема $2(P^\xi-1)M^2$.

При выполнении операций оператора S_6 процессам назначается обратный порядок выполнения итераций. Поэтому, при α=6 или β=6 (но не $\alpha=\beta$=6) следует воспользоваться следующим результатом: если $\left(\mathbf{\Phi}_{\alpha,\beta}\right)_{\zeta^\alpha}=-e^{(4)}_{\zeta^\beta}$, $\left(\mathbf{\Psi}_{\alpha,\beta}\right)_{\zeta^\alpha}$=(0 0 1 0), $\varphi^{\alpha,\beta}_{\zeta^\alpha}$=0, то коммуникационных операций не требуется. Функции $\overline{\mathbf{\Phi}}_{4,6}$, $\overline{\mathbf{\Phi}}_{5,6}$, $\overline{\mathbf{\Phi}}_{6,8}$ коммуникационных операций не порождают.

Для первой координаты имеем [14]: функции $\overline{\mathbf{\Phi}}_{1,1}$, $\overline{\mathbf{\Phi}}_{1,2}$, $\overline{\mathbf{\Phi}}_{2,2}$, $\overline{\mathbf{\Phi}}_{3,3}$ порождают коммуникационные операции суммарного объема $3(P^\eta-1)M^2$ (функции $\overline{\mathbf{\Phi}}_{1,1}$ и $\overline{\mathbf{\Phi}}_{1,2}$ порождают одну коммуникационную операцию); функция $\overline{\mathbf{\Phi}}_{9,2}$ порождает коммуникационные операции объема $2(P^\eta-1)M^2$.

Суммарно процессы получают $5(P^{\eta}-1)M^2$ данных по первой координате и $5(P^{\xi}-1)M^2$ данных по второй координате. Итого — $5(P^{\eta}+P^{\xi}-2)M^2$ данных, коммуникационные операции происходят между процессами, номера которых отличаются только на 1 по первой или по второй координате.

Структуру кода выполнения алгоритма 2D процессами можно для каждого процесса $\mathrm{Pr}_{p_\eta,p_\xi}$, $0\le p_\eta\le P^{\eta}-1$, $0\le p_\xi\le P^{\xi}-1$, представить в следующем виде:

```
do j =1, j0
   do q3= 0, Q3-1
      sending and receiving caused by Φ9,2
      receiving caused by Φ1,1, Φ1,2 and caused by Φ2,2
      dopar i2= 1 + pξr2, min((pξ+1)r2, N2-1) // Tile 1a
         dopar i3= 1 + q3r3, min((q3+1)r3, N3-1)
            do i1= 1 + pηr1, (pη +1)r1
S1:            alpha(i2,i3,i1+1)=F1(alpha(i2,i3,i1))
S2:            beta(i2,i3,i1+1)=F2(alpha(i2,i3,i1),beta(i2,i3,i1),
                        y^j(i1,i2,i3),y^j(i1-1,i2,i3),y^j(i1+1,i2,i3))
      sending caused by Φ1,1, Φ1,2 and caused by Φ2,2
   do q3= 0, Q3 -1
      receiving caused by Φ3,3
      dopar i2= 1 + pξr2, min((pξ+1)r2, N2-1) // Tile 1b
         dopar i3= 1 + q3r3, min((q3+1)r3, N3-1)
            do i1= 1 + (Pη-pη-1)r1, (Pη-pη)r1
S3:            y^{j+1/3}(N1-i1,i2,i3)=
                        alpha(i2,i3,N1-i1+1)y^{j+1/3}(N1-i1+1,i2,i3)+
                        beta(i2,i3,N1-i1+1)
      sending caused by Φ3,3
   do q3= 0, Q3 -1
      sending and receiving caused by Φ3,5
      receiving caused by Φ4,4, Φ4,5 and caused by Φ5,5
      dopar i1= 1 + pηr1, (pη +1)r1 // Tile 2a
         dopar i3= 1 + q3r3, min((q3+1)r3, N3-1)
            do i2= 1 + pξr2, min((pξ+1)r2, N2-1)
S4:            alpha(i1,i3,i2+1)=F3(alpha(i1,i3,i2))
S5:            beta(i1,i3,i2+1)=F4(alpha(i1,i3,i2),beta(i1,i3,i2),
              y^{j+1/3}(i1,i2,i3),y^{j+1/3}(i1,i2-1,i3),y^{j+1/3}(i1,i2+1,i3))
      sending caused by Φ4,4, Φ4,5 and caused by Φ5,5
   dopar q3= 0, Q3 -1
      receiving caused by Φ6,6
      dopar i1= 1 + pη r1, (pη +1)r1 // Tile 2b
         dopar i3= 1 + q3r3, min((q3+1)r3, N3 -1)
            do i2= 1 + (Pξ-pξ-1)r2, min((Pξ-pξ)r2, N2 -1)
S6:            y^{j+2/3}(i1,N2-i2,i3)=
                        alpha(i1,i3,N2-i2+1) y^{j+2/3}(i1,N2-i2+1,i3)+
                        beta(i1,i3,N2-i2+1)
      sending caused by Φ6,6
   dopar i1= 1 + pηr1, (pη +1)r1 // Tile 3
      dopar i2= 1 + pξr2, min((pξ+1)r2, N2 -1)
         do i3= 1, N3-1
S7:         alpha(i3+1)=F5(alpha(i3))
S8:         beta(i3+1)=F6(alpha(i3),beta(i3),y^{j+2/3}(i1,i2,i3),
                        y^{j+2/3}(i1,i2,i3-1),y^{j+2/3}(i1,i2,i3+1))
         do i3= 1, N3-1
S9:         y^{j+1}(i1,i2,N3-i3)=
                  alpha(N3-i3+1) y^{j+1}(i1,i2,N3-i3+1)+beta(N3-i3+1)
```

Рис. 2. Структура MPI-кода (2D процессы), реализующего схему расщепления

Пусть $P^{\eta} \times P^{\xi} = P \times P = P^2$. Тогда объем коммуникационных операций, требуемых на одном слое, равен $10(P-1)M^2$. Если P^2 процессов организовать в одномерную структуру, то объем коммуникационных операций, требуемых на одном слое, равен $5(P^2-1)M^2$, что при большом числе процессов существенно больше $10(P-1)M^2$.

Кроме того, использование 2D структуры процессов позволяет, по сравнению со случаем 1D структуры (также использующей конвейерный параллелизм), существенно сократить разгон и торможение вычислений. Примем, что множество операций любого тайла выполняется за одну единицу времени. Тогда на каждом временном слое время разгона вычислений равно $2(P-1)$ при использовании 2D структуры и P^2-1 при использовании 1D структуры; такие же временные затраты и на торможение вычислений.

Отметим, что трехшаговая схема расщепления обладает естественным параллелизмом: при реализации схемы на параллельных компьютерах с распределенной памятью вычислительные процессы на каждом из трех шагов одного временного слоя могут выполняться независимо. Если P^2 процессов организовать в одномерную структуру, то объем коммуникационных операций, требуемых на одном слое, равен [14] $2\left(1-\frac{1}{P^2}\right)M^3$; кроме того, такая реализация требует обращение каждого процесса к локальной памяти всех других процессов. С ростом M и числа используемых процессов это приводит к большим накладным расходам на коммуникационные операции. Если P^2 процессов организовать в двумерную структуру (с использованием только естественного параллелизма), то не удается уменьшить объем коммуникационных операций.

7. Вычислительные эксперименты

Для вычислительного эксперимента был использован суперкомпьютер ЦОД МФТИ. Для сравнения эффективности двух реализаций: алгоритма с улучшенной локальностью, использующего 1D структуру процессов [14], и алгоритма, использующего 2D структуру процессов, будем использовать задачу (1)–(3) с заданным решением $u(x,t) = e^{3t+x_1+x_2+x_3}$.

Дополнительно в численном эксперименте алгоритмы сравниваются с реализацией алгоритма с естественным параллелизмом.

В табл. 1 и табл. 2 представлены результаты вычислительных экспериментов при $N_1=N_2=N_3=300$, $j_0=100$. Необходимо отметить, что время выполнения алгоритмов с улучшенной локальностью при фиксированном количестве процессов зависит от размера тайла $r_1 \times r_2 \times r_3$. Если размер тайла по координате отображения на процессы определяется автоматически, то размер тайла по свободным координатам можно определять при запуске алгоритма. За счет сокращения коммуникации между узлами суперкомпьютера проявляются преимущества разработанной версии алгоритма. В случае меньшей пересылки по сети (например, несколько процессов на одном узле) эффект менее заметен.

Для сравнения результатов при увеличении количества процессов были выполнены эксперименты с следующими параметрами: $N_1=N_2=N_3=400$ и $j_0=100$; количество процессов совпадает с количеством вычислительных узлов; параметры $r_2=r_3$ принимают значения 20, 24 и 28 (для 1D структуры); параметр r_3 принимает значения 20, 24 и 28 (для 2D структуры).

Таблица 1

Время работы алгоритма (N=300, 16 процессов на 16 узлах)

Алгоритм	r_2	r_3	**Время работы, с**
Естественный параллелизм	—	—	48.03
1D структура процессов	16	16	48.78
	20	20	47.50
	24	24	46.97
	28	28	46.61
	32	32	47.53
2D структура процессов	—	2	46.31
	—	6	44.19
	—	10	44.01
	—	18	44.07

Таблица 2

Время работы алгоритма (N=300, 16 процессов на 4 узлах)

Алгоритм	r_2	r_3	**Время работы, с**
Естественный параллелизм	—	—	42.62
1D структура процессов	16	16	36.40
	20	20	35.28
	24	24	36.26
	28	28	37.77
	32	32	43.88
2D структура процессов	—	2	37.10
	—	6	39.48
	—	10	40.57
	—	18	44.04

В табл. 3 и на рис. 3 представлены обобщенные результаты: для фиксированного числа процессов (узлов) выбрано наименьшее время работы алгоритма. Таким образом полученные данные показывают суммарный эффект применения тайлинга и предложенного подхода по уменьшению объема пересылаемых данных между узлами.

Экспериментальные результаты согласуются с оценкой эффективности разработанных алгоритмов в оптимизации пересылаемых данных, однако при увеличении количества используемых узлов полезный эффект становится менее заметным.

Приведенная параллельная версия алгоритма реализована на языке C++ с использованием MPI.

Таблица 3

Время работы алгоритма (N=400)

Алгоритм	**Время работы, с (количество процессов)**			
	4	**9**	**16**	**25**
Естественный параллелизм	170.97	111.02	48.70	33.28
1D структура процессов	135.97	82.01	50.07	36.00
2D структура процессов	135.11	78.74	46.31	32.64

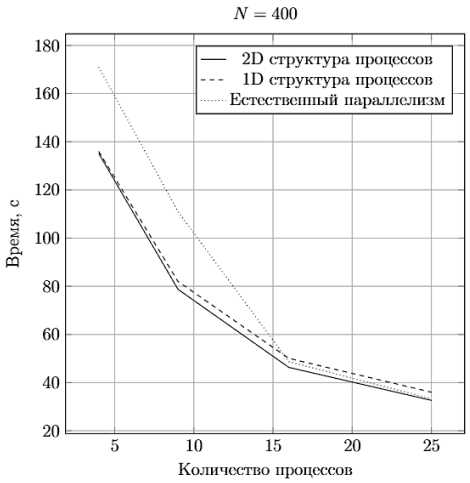

Рис. 3. Время выполнения алгоритмов на суперкомпьютере

Заключение

Время решения задачи на современном компьютере во многом определяется тем, насколько эффективно используется память с быстрым доступом. Особенно важное значение она имеет при организации параллельной обработки данных. Малая степень локализации данных оборачивается низкой эффективностью и масштабируемостью параллельных приложений [4]. В этой работе в качестве компьютера, на котором требуется реализовать алгоритм, рассмотрена многопроцессорная вычислительная система с распределенной памятью. Для этого класса компьютеров быстрой считается локальная память вычислительного узла. В работе исследована локальность нового параллельного алгоритма, реализующего схему расщепления численного решения трехмерного линейного уравнения теплопроводности. Алгоритм использует параллельные вычислительные процессы, логически организованные в двумерную структуру, обладает лучшей локальностью по сравнению с известными алгоритмами, использующими 1D структуры процессов.

На примере алгоритма, реализующего схему расщепления, представлен подход к организации параллельных 2D зернистых вычислительных процессов: распределение операций между процессами рассматривается как два независимых распределения операций между 1D процессами, анализируются информационные зависимостей, коммуникационные затраты оцениваются с помощью утверждений, доказанных для 1D процессов [11]. Отметим пригодность к автоматизации предлагаемого подхода: необходимая формальная информация об исходном алгоритме может быть получена с помощью свободно доступных библиотек и систем, проверка условий для получения объема и структуры коммуникационных операций 1D и 2D процессов допускает программную реализацию, конкретные значения заданных параметрически величин не требуются до компиляции.

Направления дальнейших исследований: автоматизация исследования локальности зернистых вычислительных процессов; разработка и программная реализация алгоритмов метода дробных шагов численного решения линейных и квазилинейных трехмерных параболических уравнений на суперкомпьютерах с распределенной памятью; разработка и программная реализация параллельных алгоритмов для решения прикладных задач с использованием предлагаемого подхода.

Работа выполнена в рамках государственной программы научных исследований Республики Беларусь «Конвергенция-2025», подпрограмма «Математические модели и методы».

Литература

1. Адуцкевич Е.В., Лиходед Н.А. Согласованное получение конвейерного параллелизма и распределения операций и данных между процессорами // Программирование. 2006. Т. 32, № 3. С. 54–65.
2. Баханович С.В., Лиходед Н.А., Мандрик П.А. Улучшение локальности параллельных алгоритмов численного решения двумерных квазилинейных параболических уравнений // Вестник Российского университета дружбы народов. Серия: Математика, информатика, физика. 2014. № 2. С. 211–215.
3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Санкт-Петербург: БХВ-Петербург, 2002. 608 с.
4. Воеводин Вл.В., Воеводин Вад.В. Спасительная локальность суперкомпьютеров // Открытые системы. 2013. № 9. С. 12–15.
5. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Нижний Новгород: ННГУ им. Н.И. Лобачевского, 2003. 184 с.
6. Колешко С.Б., Попов Ф.Д. Механика жидкости и газа. Разностные схемы. Учеб. пособие. СПб.: Изд-во СПбГТУ, 2001. 72 с.
7. Корнеев Б.А., Левченко В. Д. Эффективное решение трехмерных задач газовой динамики Рунге—Кутты разрывным методом Галеркина// Журнал вычислительной математики и математической физики. 2016. № 3. С. 465–475. DOI: 10.7868/S0044466916030121.
8. Лиходед Н.А. Достаточные условия определения и использования данных в одном параллельном зернистом вычислительном процессе // Журнал вычислительной математики и математической физики. 2014. № 8. С. 122–133. DOI: 10.7868/s0044466914080092.
9. Лиходед Н.А., Баханович С.В., Жерело А.В. Получение аффинных преобразований для улучшения локальности гнезд циклов // Программирование. 2005. № 5. С. 52–65. DOI: 10.1007/s11086-005-0036-2.
10. Лиходед Н.А., Полещук М.А. Построение двумерных зернистых параллельных вычислительных процессов // Известия НАН Беларуси. Сер. физ.-мат. наук 2018. Т. 54, № 4. С. 417–426. DOI: 10.29235/1561-2430-2018-54-4-417-426.
11. Лиходед Н.А., Толстиков А.А. Метод оценки локальности параллельных алгоритмов, ориентированных на компьютеры с распределенной памятью // Доклады НАН Беларуси. 2020. Т. 64, № 6. С. 647–656. DOI: 10.29235/1561-8323-2020-64-6-647-656.
12. Лиходед Н.А., Толстиков А.А. Параллельные последовательности зернистых вычислений // Доклады НАН Беларуси. 2010. Т. 54, № 4. С. 36–41.
13. Толстиков А.А., Лиходед Н.А. Корректность разбиений алгоритмов при организации зернистых параллельных вычислительных процессов // Международный конгресс по информатике: информационные системы и технологии (CSIST'2011) (Минск, 31 октября – 3 ноября 2011 г.). Минск: Белорусский государственный университет, 2011. Т. 2. С. 122–126.
14. Толстиков А.А., Лиходед Н.А. Параллельный алгоритм метода расщепления для реализации на суперкомпьютерах с распределенной памятью // Международная научная конференция «Параллельные вычислительные технологии» (ПаВТ'2020) (Пермь, 31 марта – 2 апреля 2020 г.). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2020. С. 287–297.

15. Яненко Н.Н. Метод дробных шагов решения многомерных задач математической физики. Новосибирск: Изд-во «Наука». Сибирское отделение, 1967. 197 с.
16. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model // Lecture Notes in Computer Science. Vol. 4959. Springer, 2008. P. 132–146. DOI: 10.1007/978-3-540-78791-4_9.
17. Buluc A., Gilberta J.R., Budak C. Solving path problems on the GPU // Parallel Computing. 2010. Vol. 36, no. 5–6. P. 241–253. DOI: 10.1016/j.parco.2009.12.002.
18. Dathathri R., Mullapudi R.T., Bondhugula U. Compiling affine loop nests for a dynamic scheduling runtime on shared and distributed memory // ACM Transactions on Parallel Computing (TOPC). 2016. Vol. 3, no. 2. DOI: 10.1145/2948975.
19. Lim A.W., Lam M.S. Maximizing parallelism and minimizing synchronization with affine partitions // Parallel Computing. 1998. Vol. 24, no. 3–4. P. 445–475. DOI: 10.1016/S0167-8191(98)00021-0.
20. Xue J. Loop tiling for parallelism. Springer Science & Business Media, 2000. 256 p. DOI: 10.1007/978-1-4615-4337-4.
21. Zhao J, Cohen A. Flextended tiles: a flexible extension of overlapped tiles for polyhedral compilation // ACM Transactions on Architecture and Code Optimization. 2019. Vol. 16, no. 4, article 47. DOI: 10.1145/3369382.
22. Zhao J, Di P. Optimizing the Memory Hierarchy by Compositing Automatic Transformations on Computations and Data. 53rd IEEE/ACM International Symposium on Microarchitecture (MICRO 2020) (Athens, Greece, October 17–21, 2020). IEEE, 2020. P. 427–441. DOI: 10.1109/micro50266.2020.00044.

Толстиков Алексей Александрович, старший преподаватель кафедры вычислительной математики факультета прикладной математики и информатики Белорусского государственного университета (Минск, Республика Беларусь)

Баханович Сергей Викторович, к.ф.-м.н., заместитель директора по научной и инновационной работе ГНУ «Институт математики НАН Беларуси» (Минск, Республика Беларусь)

Лиходед Николай Александрович, д.ф.-м.н., профессор кафедры вычислительной математики факультета прикладной математики и информатики Белорусского государственного университета (Минск, Республика Беларусь)

DOI: 10.14529/cmse210303

AN APPROACH TO ESTIMATE THE LOCALITY OF GRAINED COMPUTATION PROCESSES ORGANIZED TO A TWO-DIMENSIONAL STRUCTURE

© 2021 A.A. Tolstsikau[1], S.V. Bakhanovich[2], N.A. Likhoded[1]
[1]*Belarusian State University (Nezavisimosti Avenue 4, Minsk, 220030 Belarus),*
[2]*Institute of Mathematics NAS of Belarus (str. Surganova 11, Minsk, 220072 Belarus)*
E-mail: tolstikov@bsu.by, bsv@im.bas-net.by, likhoded@bsu.by
Received: 31.07.2021

Implementing algorithms for distributed memory computers, one should pay attention to locality, a computational property of the algorithm that reflects the usage of fast access memory, that plays an important role in achieving high performance. For computing systems with distributed memory the local memory of each node is considered fast. Usually the parallel algorithms have better locality if it has a smaller total size of communication data and a better structure of communication operations. In this work, a new parallel algorithm for the numerical solution of a three-dimensional linear heat equation is constructed on the basis of a splitting method with weights. The algorithm is specified for distributed memory computers, combines pipeline and natural parallelism, and uses a 2D process structure. For the case of 1D process structure it was shown that usage of a pipeline parallelism instead of part of natural parallelism reduces the total size of communication operations. The proposed 2D algorithm generalizes the known 1D algorithm. The usage of two-dimensional structures allows to reduce the total volume of communication operations and to reduce idle time of a pipeline. Therefore, the 2D algorithm has better locality compared to the 1D process structure. The computational experiments on a distributed computing sytem have shown the advantage of a new parallel algorithm. Proposed method can be used to obtain and to examine parallel algorithms for other schemes of the fractional step method. Additionally, the implemented splitting scheme algorithm shows an approach to obtain asymptotic estimation of the communication volume, presented method operates with granular (the macro-operation level) parallel computing processes organized logically into a two-dimensional structure. The estimations can be used to compare communication cost for alternative versions of parallel algorithms. The approach is developed on the basis of early result that allows to obtain asymptotic estimations of the communication operations volume of computational processes organized in a one-dimensional structure.

Keywords: parallel computing, distributed memory parallel computer, data communication reduction, fractional step method.

FOR CITATION

Tolstsikau A.A., Bakhanovich S.V., Likhoded N.A. An Approach to Extimate the Locality of Grained Computation Processes Organized to a Two-dimensional Structure. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 3. P. 37–55. (in Russian) DOI: 10.14529/cmse210303.

References

1. Adutskevich E.V., Likhoded N.A. A consistent generation of pipeline parallelism and distribution of operations and data among processors. Programming and Computer Software. 2006. Vol. 32, no. 3. P. 166–176.
2. Bakhanovich S.V., Likhoded N.A., Mandrik P.A. Improvement of locality of parallel algorithms of the numerical solutions of the two-dimensional quasilinear parabolic equation. RUDN Journal of Mathematics, Information Sciences and Physics. 2014. No. 2. P. 211–215. (in Russian)

3. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St. Petersburg, BKhV-Petersburg Publ., 2002. 608 p. (in Russian)
4. Voevodin Vl.V., Voevodin Vad.V. The fortunate locality of supercomputers. Open Systems. 2013. No. 9. P. 12–15. (in Russian)
5. Gergel V.P., Strongin R.G. Introduction to parallel computing for multiprocessor systems. Nizhny Novgorod, Lobachevsky State University of Nizhny Novgorod, 2003. 184 p. (in Russian)
6. Koleshko S.B., Popov F.D. Mechanics of Liquids and Gases. Difference schemes. Tutorial. St. Petersburg, SPbGTU Publ., 2001. 72 p. (in Russian)
7. Korneev B.A., Levchenko V.D. Effective solving of three-dimensional gas dynamics problems with the Runge-Kutta discontinuous Galerkin method. Comput. Math. and Math. Phys. 2016. Vol. 56. P. 460–469. DOI: 10.1134/S0965542516030118.
8. Likhoded N.A. Sufficient conditions for the determination and use of data in the same granular parallel computation process. Computational Mathematics and Mathematical Physics. 2014. Vol. 54, no. 8. P. 1316–1326. (in Russian) DOI: 10.1134/s0965542514080077.
9. Likhoded N.A., Bakhanovich S.V., Zherelo A.V. Obtaining affine transformations to improve the locality of loop nests. Programming and Computer Software. 2005. Vol. 31, no. 5. P. 270–281. (in Russian) DOI: 10.1007/s11086-005-0036-2.
10. Likhoded N.A., Poleshchuk M.A. Tiled parallel 2D computational processes. Proceedings of the National Academy of Sciences of Belarus. Physics and Mathematics Series. 2018. Vol. 54, no. 4. P. 417–426. (in Russian) DOI: 10.29235/1561-2430-2018-54-4-417-426.
11. Likhoded N.A., Tolstsikau A.A. Locality estimation of parallel algorithm for distributed memory computers. Proceedings of the National Academy of Sciences of Belarus. 2020. Vol. 64, no. 6. P. 647–656. (in Russian) DOI: 10.29235/1561-8323-2020-64-6-647-656.
12. Likhoded N.A., Tolstsikau A.A. Parallel sequences of grain computations. Proceedings of the National Academy of Sciences of Belarus. 2010. Vol. 54, no. 4. P. 36–41. (in Russian)
13. Tolstsikau A.A., Likhoded N.A. Algorithm partition correctness while organizing grained parallel computing processes. International Congress on Computer Science: Information Systems and Technologies (CSIST'2011) (Minsk, Belarus, October 31 – November 3, 2011). Minsk, Belarusian State University, 2011. Vol. 2. P. 122–126. (in Russian)
14. Tolstsikau A.A., Likhoded N.A. Parallel algorithm implementing split method for a distributed memory computer. Parallel Computational Technologies (PCT 2020) (Perm, Russia, March 31 – April 2, 2020). Short papers and posters. Chelyabinsk, Publishing of the South Ural State University, 2020. P. 287–297. (in Russian)
15. Yanenko N.N. The method of fractional steps for solving multi-dimensional problems of mathematical physics. Novosibirsk, Science, 1967. 196 p. (in Russian)
16. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model. Lecture Notes in Computer Science. Vol. 4959. Springer, 2008. P. 132–146. DOI: 10.1007/978-3-540-78791-4_9.
17. Buluc A., Gilberta J.R., Budak C. Solving path problems on the GPU. Parallel Computing. 2010. Vol. 36, no. 5–6. P. 241–253. DOI: 10.1016/j.parco.2009.12.002.
18. Dathathri R., Mullapudi R.T., Bondhugula U. Compiling affine loop nests for a dynamic scheduling runtime on shared and distributed memory. ACM Transactions on Parallel Computing (TOPC). 2016. Vol. 3, no. 2. DOI: 10.1145/2948975.

19. Lim A.W., Lam M.S. Maximizing parallelism and minimizing synchronization with affine partitions. Parallel Computing. 1998. Vol. 24, no. 3–4. P. 445–475. DOI: 10.1016/S0167-8191(98)00021-0.
20. Xue J. Loop tiling for parallelism. Springer Science & Business Media, 2000. 256 p. DOI: 10.1007/978-1-4615-4337-4.
21. Zhao J., Cohen A. Flextended tiles: a flexible extension of overlapped tiles for polyhedral compilation. ACM Transactions on Architecture and Code Optimization. 2019. Vol. 16, no. 4, article 47. DOI: 10.1145/3369382.
22. Zhao J., Di P. Optimizing the Memory Hierarchy by Compositing Automatic Transformations on Computations and Data. 53rd IEEE/ACM International Symposium on Microarchitecture (MICRO 2020) (Athens, Greece, October 17–21, 2020). IEEE, 2020. P. 427–441. DOI: 10.1109/micro50266.2020.00044.