ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 2 за 2022 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

О.Ю. Милюкова

Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливанием блочного неполного обратного треугольного разложения IC1

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливанием блочного неполного обратного треугольного разложения IC1 // Препринты ИПМ им. М.В.Келдыша. 2022. № 2. 30 с.
https://doi.org/10.20948/prepr-2022-2
https://library.keldysh.ru/preprint.asp?id=2022-2

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М. В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

О. Ю. Милюкова

Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливанием блочного неполного обратного треугольного разложения IC1

Москва — 2022

Милюкова О.Ю.

Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливанием блочного неполного обратного треугольного разложения IC1

В работе рассматривается предобусловливатель блочного неполного обратного треугольного разложения первого порядка «по значению» BIIC-IC1 для решения систем линейных алгебраических уравнений с симметричной положительно определенной матрицей. Рассматривается способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1, в котором число блоков в предобусловливателе кратно числу используемых процессоров и числу используемых потоков. Предлагается новый способ применения MPI+OpenMP для построения и обращения предобусловливателя BIIC-IC1 с числом блоков, совпадающим с числом процессоров, в котором для применения OpenMP технологии используется специальное упорядочение узлов сетки внутри подобластей, соответствующих расчетам на каждом процессоре. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем BIIC-IC1 с использованием MPI и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse.

Ключевые слова: неявное блочное предобусловливание, неполное треугольное разложение Холецкого, параллельное предобусловливание, метод сопряженных градиентов

Olga Yurievna Milyukova

Approaches MPI+OpenMP implementation of conjugated gradient method with pre-conditioner of block incomplete inverse triangular decomposition of IC1

The paper considers a preconditioner for solving systems of linear algebraic equations with a symmetric positively defined matrix – a preconditioner of the block incomplete inverse triangular decomposition of the first order "by value" BIIC-IC1. The invention considers a method of using MPI+OpenMP technology for the construction and application of the BIIC-IC1 preconditioner, in which the number of blocks in a preconditioner is multiple the number of processor used and the number of threads used. A new method of using MPI+OpenMP technology for constructing and handling a preconditioner BIIC-IC1 with a number of blocks coinciding with the number of processors is proposed, in which a special ordering of grid nodes inside subareas corresponding to calculations on each processor is used to apply the OpenMP technology. Time of solving problems is compared by method of conjugated gradients with preconditioner BIIC-IC1 using MPI and hybrid MPI+OpenMP technology using example of model problem and number of tasks from collection of sparse matrices SuiteSparse.

Keywords: implicit block preconditioning, incomplete Cholesky factorization, parallel preconditioning, conjugate gradient method

1. Введение

Рассмотрим задачу приближенного решения системы линейных алгебраических уравнений (СЛАУ) большого размера

$$Ax = b \qquad (1.1)$$

с симметричной положительно определенной разреженной матрицей A общего вида

$$A = A^T > 0.$$

Проблема построения эффективных численных методов решения СЛАУ (1.1) сохраняет свою актуальность, так как во многих важных прикладных областях продолжают возникать новые постановки таких задач. При этом наблюдается тенденция к росту размера матриц n, усложнению структуры разреженности, а также к ухудшению обусловленности.

В настоящей работе для решения СЛАУ (1.1) большого размера применяется предобусловленный метод сопряженных градиентов (CG), итерации которого осуществляются до выполнения условия

$$\|b - Ax_k\| \le \varepsilon \|b - Ax_0\|, \text{ где } 0 < \varepsilon << 1. \qquad (1.2)$$

Для предобусловливания используется блочное неполное обратное разложение Холецкого (Block Incomplete Inverse Cholesky, BIIC) [1, 2] в сочетании с приближенным треугольным разложением с отсечением по параметру $0<\tau<<1$ первого порядка IC1(τ) – предобусловливание блочного неполного обратного треугольного разложения первого порядка BIIC-IC1(τ) [3]. Один из первых алгоритмов, в котором за основу построения матрицы предобусловливания берется точный алгоритм треугольной факторизации, а на его определенных этапах вносится отсечение возникающих элементов матриц, малых относительно порога, зависящего от τ, опубликован в работе [4]. Предобусловливание IC1(τ) имеет ограниченную область применимости [5] для фиксированного значения τ, в ряде случаев для безотказности требует чрезмерного уменьшения параметра τ.

Основная трудность распараллеливания алгоритма метода сопряженных градиентов с предобусловливанием неполного треугольного разложения связана с рекурсивным характером вычислений при построении и обращении предобусловливателя. Неявный факторизованный предобусловливатель имеет вид: $B = (LL^T)^{-1} \approx A^{-1}$ или $B = (LDL^T)^{-1} \approx A^{-1}$, где L – нижнетреугольная матрица, а D – диагональная матрица. Вместо обращения LL^T выполняется решение двух треугольных систем для нахождения вектора поправки в предобусловленном методе сопряженных градиентов.

В работе [6] предложен подход, который позволяет преодолеть проблему распараллеливания рекурсивных вычислений при построении и обращении предобусловливателя при решении задачи на многопроцессорной вычислительной системе. В этой работе предложен параллелизуемый предобусловливатель, представляющий собой блочную версию предобусловливания неполного обратного треугольного разложения BIIC в

сочетании с неполным треугольным разложением второго порядка IC2(τ) [7] - BIIC-IC2(τ). Для построения предобусловливателя BIIC-IC2(τ) специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение второго порядка IC2(τ).

В работе [8] рассмотрена блочная версия неполного обратного треугольного разложения BIIC-IC2S(τ), в которой, в отличие от работы [6], для построения предобусловливателя внутри блока используется IC2S(τ) разложение [7]. Методы сопряженных градиентов с предобусловливанием BIIC-IC2(τ) и BIIC-IC2S(τ) являются безотказными для любого фиксированного значения τ. Методы сопряженных градиентов с предобусловливанием BIIC-IC2(τ), BIIC-IC2S(τ), BIIC-IC1(τ) были эффективно реализованы на параллельных архитектурах с распределенной памятью [6, 8, 3].

Другим способом преодоления основной трудности распараллеливания алгоритмов построения и обращения предобусловливателя неполного треугольного разложения, связанной с рекурсивным характером вычислений, является использование переупорядочений узлов сетки и соответствующих перестановок строк и столбцов матрицы, например использование переупорядочений, связанных с разбиением области расчета (DDO - Domain Decompositing Ordering) [9]. Применению такого подхода для крупнозернистого распараллеливания посвящено много работ, например, [10 –15]. В работе [16] предложено использовать упорядочение типа DDO для построения параллельного варианта метода стабилизированного треугольного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG).

Основное внимание в настоящей работе уделяется применению OpenMP технологии для параллельной реализации алгоритма метода сопряженных градиентов с предобусловливателем BIIC-IC1(τ) при проведении расчетов на многопроцессорной вычислительной системе.

Проблеме использования высокоуровневого параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. В работах [17 – 20] было предложено использовать несколько итераций Якоби или блочного Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения, что позволило использовать высокий уровень параллелизма. В работе [21] предложен безытерационный способ применения MPI+OpenMP технологии при обращении неявного факторизованного предобусловливателя, т. е. решении двух треугольных систем (в том числе для случая решения СЛАУ (1.1)). В работе [22] предложен новый итерационный алгоритм вычисления неполного IC(0), IC(1), IC(2) разложений, в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно.

Заметим, что использование явных предобусловливателей позволяет эффективно применять MPI+OpenMP технологии для параллельного решения

системы линейных алгебраических уравнений (1.1) предобусловленным методом сопряженных градиентов, например [23 – 25].

В работах [26, 27, 28] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0) (неполного треугольного разложения Холецкого без заполнения) и IC1(τ). Один из них основан на переупорядочении узлов сетки типа DDO внутри каждой подобласти, другой способ из этих работ основан на уменьшении шаблона разреженности матрицы A при построении предобусловливателя. С помощью расчетов тестовых задач показано, что применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для умеренного числа узлов суперкомпьютерной системы (порядка нескольких десятков).

В работе [3] впервые предложен безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC2S(τ) и BIIC-IC1(τ) на основе использования числа блоков в предобусловливателе, кратного числу используемых процессоров и числу используемых потоков. В работе [3] предложен также безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ), в котором для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO. При этом в работе [3] при построении матрицы предобусловливания для некоторых ее строк осуществлялось отсечение по позициям, при построении и обращении предобусловливателя OpenMP технологии применялись для большинства строк матрицы.

В формуле (1.1) предполагается, что матрица A уже переупорядочена, а вместо A_P стоит A ($A = A_P = P\tilde{A}P^T$), где P – матрица перестановки, а $\tilde{A}$ – матрица коэффициентов исходной задачи. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно, предложенные в работе [29], являющиеся обобщением упорядочения [8]. Подход, предложенный в этих работах, позволяет одновременно произвести разбиение области расчета на подобласти. Будем также предполагать, что матрица A отмасштабирована, т. е. ее диагональные элементы равны единице. Это достигается с использованием формулы: $A_{SP} = D_{A_P}^{-1/2} A_P D_{A_P}^{-1/2}$, где D_{A_P} – диагональная часть матрицы A_P. Далее вместо A_{SP} будем использовать обозначение A, предполагая, что переупорядочение и масштабирование уже выполнены.

В настоящей работе рассматривается предобусловливатель BIIC-IC1(τ) [3] для решения СЛАУ (1.1) методом сопряженных градиентов. Рассматривается безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ) – способ 1 [3], в котором число блоков в предобусловливателе кратно числу используемых

процессоров и числу используемых потоков. В настоящей работе предлагается новый безытерационный способ 2 применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ), в котором в предобусловливателе BIIC используется число блоков, совпадающее с числом используемых процессоров, а для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO. В отличие от способа 2 из работы [3], в способе 2 из настоящей работы при построении матрицы предобусловливания не производится отсечение по позициям для каких-либо строк матрицы. Для реализации способа 2 применения MPI+OpenMP технологии рассматриваются два различных способа разбиения области расчета. При построении и обращении предобусловливателя с использованием способа 2 OpenMP технологии применяются для большинства строк матрицы.

В настоящей работе проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем BIIC-IC1(τ) с использованием MPI и MPI+OpenMP подходов способами 1 и 2 на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [30]. Проводится сравнение времени решения этих тестовых задач методом BIIC-IC1(τ)-CG с использованием двух рассматриваемых в работе способов применения MPI+OpenMP технологии, а также сравнение времени решения тестовых задач методом BIIC-IC1(τ)-CG с использованием способа 2 применения MPI+OpenMP технологии при проведении разбиения области расчета двумя различными способами.

2. Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1.1). Алгоритм предобусловленного метода сопряженных градиентов (см., например, [31]) имеет следующий вид:

Алгоритм 1

$$r_0 = b - Ax_0,\ p_0 = w_0 = Hr_0,\ \gamma_0 = r_0^T p_0,$$

для k=0,… пока $(r_k^T r_k) \le \varepsilon^2 (r_0^T r_0)$ выполнять

$$q_k = Ap_k, \qquad \alpha_k = \gamma_k / (p_k^T q_k),$$

$$x_{k+1} = x_k + \alpha_k p_k, \quad r_{k+1} = r_k - \alpha_k q_k, \quad z_{k+1} = Hr_{k+1},$$

$$\gamma_{k+1} = r_{k+1}^T z_{k+1}, \qquad \beta_k = \gamma_{k+1} / \gamma_k, \qquad p_{k+1} = z_{k+1} + \beta_k p_k,$$

где $0 < \varepsilon << 1$, H – матрица предобусловливания ($H \approx A^{-1}$). Этот алгоритм использует операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений, элементарные векторные операции, а также вычисление $z_{k+1} = Hr_{k+1}$, $p_0 = w_0 = Hr_0$. Принципиальная возможность эффективной параллельной реализации всех операций, кроме вычисления Hr_{k+1}, Hr_0, не вызывает сомнений, даже при использовании большого числа процессоров и (или) применения OpenMP технологии.

3. Предобусловливатель блочного неполного обратного треугольного разложения первого порядка

Приведем краткое описание блочной версии алгоритма неполного обратного разложения Холецкого ВИС [1, 2]. Пусть матрица *A* переупорядочена и разбита на блоки, причем на блочной диагонали расположены *p* квадратных блоков размера $n_s \times n_s$, $1 \le s \le p$. Для каждого *s*-го диагонального блока определим базисное множество индексов как $\{k_{s-1}+1,...,k_s\}$, где $k_s = n_1 + ... + n_s$ ($k_0 = 0, k_p = n$), и введем «перекрывающиеся» множества индексов: $\{j_s(1), ..., j_s(m_s - n_s)\}$, $j_s(l) \le k_{s-1}$ $(l = 1, ..., m_s - n_s)$. Для каждого *s* такое индексное множество включает индексы, не превосходящие k_{s-1}, которые оказываются наиболее существенно связаны с *s*-м базисным множеством индексов, например, в смысле графа смежности разреженной матрицы *A*. Здесь $m_s \ge n_s$, m_s – размеры расширенных блоков, причем $m_1 = n_1$. Для каждого *s*-го диагонального блока определим прямоугольные матрицы

$$V_s = \left[e_{j_s(1)} \middle| ... \middle| e_{j_s(m_s - n_s)} \middle| e_{k_{s-1}+1} \middle| ... \middle| e_{k_s} \right], \qquad (3.1)$$

столбцы которых являются единичными *n*-векторами. Пусть $\overline{U}_s$ – правый множитель в точном треугольном разложении Холецкого «расширенной» диагональной ($m_s \times m_s$) подматрицы

$$V_s^T A V_s = \overline{U}_s^T \overline{U}_s, \quad s=1,\ldots, p, \qquad (3.2)$$

где V_s определены в (3.1). Предобусловливатель блочного неполного обратного разложения Холецкого (ВИС) имеет вид [1, 2]

$$H = \sum_{s=1}^{p} V_s \overline{U}_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} \overline{U}_s^{-T} V_s^T . \qquad (3.3)$$

В работах [1, 2] показано, что ВИС предобусловливатель обладает свойством *K*-оптимальности. Это означает, что предобусловливатель (3.3) обеспечивает минимизацию *K(HA)* – *K*-числа обусловленности [1, 2] матрицы *HA* на множестве разреженных нижнетреугольных матриц *L*, которые имеют ненулевые элементы в позициях $j \in \{j_s(1), ..., j_s(m_s - n_s), k_{s-1}+1, ..., i\}$, $k_{s-1} + 1 \le i \le k_s$. Целесообразность минимизации *K(HA)* объясняется оценкой числа итераций [1] метода сопряженных градиентов

$$i_K(\varepsilon) = \left\lceil \log_2 K(HA) + \log_2(\varepsilon^{-1}) \right\rceil .$$

В формуле (3.3) для каждого *s*=1,…,*p* для аппроксимации $m_s \times m_s$ подматриц $A_s = V_s^T A V_s$ заменим точное разложение Холецкого (3.2) соответствующим приближенным треугольным разложением первого порядка «по значению» IC1(τ)

$$A_s = \hat{U}_s^T \hat{U}_s - E_s ,$$

где $\hat{U}_s$ – верхнетреугольные матрицы, $\|E_s\| = O(\tau)$ и $0 < \tau << 1$ порог отсечения. Определим предобусловливатель ВПС-IC1(τ) формулой

$$\hat{H} = \sum_{s=1}^{p} V_s \hat{U}_s^{-1} \begin{bmatrix} 0\,0 \\ 0\,I_{n_s} \end{bmatrix} \hat{U}_s^{-T} V_s^T. \qquad (3.4)$$

В алгоритме построения матрицы $\hat{U}_s$ в предобусловливателе IC1(τ) для матрицы A_s размера $m_s \times m_s$ для каждого i выбор элементов e_{ij} матрицы погрешности E_s осуществляется так, чтобы «убрать» все «малые» значения $v_{ij} = u_{ii}u_{ij} - e_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si}u_{sj}$, $j=i,\ldots,n$. Здесь и далее a_{ij} – элементы матрицы A_s, u_{ij} – элементы матрицы $\hat{U}_s$. Перед построением матрицы предобусловливания IC1(τ) для матрицы A_s необходимо обеспечить, чтобы матрицы A_s были отмасштабированы.

В настоящей работе при написании программы использовался следующий алгоритм [27]. Напомним, что после масштабирования $a_{ii} = 1$.

Алгоритм 2

1. Инициализация вспомогательной диагональной матрицы:
   for $i=1,\ldots,n$
   $d_i := 1$
   end for
   for $i=1,\ldots,n$ (*цикл по строкам* A *для вычисления строк* U)
2. Инициализация вектора v при помощи i-ой строки A:
   for $j=i+1,\ldots,n$
   $v_j := a_{ij}$
   end for
3. Сделать поправку к вектору v:
   for $s=1,\ldots,i-1$
   for $j=i+1,\ldots,n$
   $v_j = v_j - u_{si}u_{sj}$
   end for
   end for
4. Вычисление элементов u_{ii} и нормализация вектора v:
   $u_{ii} := \sqrt{d_i}$
   for $j=i+1,\ldots,n$
   $v_j := v_j / u_{ii}$
   end for
5. Вычисление элементов u_{ij} при $j>i$:
   for $j=i+1,\ldots,n$
   if $|v_j| \geq \tau$ then

$u_{ij} := v_j$

else

$u_{ij} := 0$

endif

end for

6. Выполнение поправки к вспомогательной диагональной матрице:

for $j=i+1,\ldots,n$

$d_j := d_j - u_{ij}^2$

end for

end for (*конец цикла по* i)

Заметим, что для вычисления поправки к вектору v в пункте 3 алгоритма 2 в программе следует обеспечить доступ к элементам матрицы $\hat{U}_s$ по строкам и по столбцам этой матрицы.

При вычислении u_{ii} – диагональных элементов матрицы $\hat{U}_s$ в п. 4 необходимо извлекать квадратный корень из числа, которое определяется в процессе вычисления элементов строк этих матриц с номерами, не превосходящими i. Это выражение может оказаться отрицательным. В этом случае следует уменьшить значение τ и использовать метод BIIC-IC1(τ)-CG с уменьшенным значением τ.

4. Алгоритмы параллельной реализации

Пусть матрица A переупорядочена, отмасштабирована и разбита на p блоков. В настоящей работе, не ограничивая общности, в предобусловливателе BIIC-IC1(τ) будем использовать налегание [1, 2, 6] с шириной q=1. Перед вычислением матрицы предобусловливания в каждом процессоре с номером s ($1 \le s \le p$) строятся матрицы $A_s = V_s^T A V_s$ размера $m_s \times m_s$, где V_s определены в (3.1). Для этого осуществляются необходимые пересылки строк матрицы A.

Параллельная реализация вычисления предобусловливателя BIIC-IC1(τ) с применением только MPI [3] не представляет труда. Для вычисления элементов матриц $\hat{U}_s$ в формуле (3.4) используется алгоритм 2, приведенный выше, пересылок не требуется. Матрицы $\hat{U}_s^T$ находятся с помощью транспонирования матриц $\hat{U}_s$ в каждом процессоре с номером s, что не требует межпроцессорного обмена.

Обращение предобусловливателя: $z = \hat{H} r$, где $\hat{H}$ определено в (3.4), происходит следующим образом [3]. Перед началом вычислений элементы вектора r_s с индексами $\{j_s(1), \ldots, j_s(m_s - n_s)\}$ должны быть пересланы в процессор с номером s. Затем в каждом процессоре с номером s вычисляются

$$z_s = \hat{U}_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} \hat{U}_s^{-T} r_s . \qquad (4.1)$$

После вычисления z_s выполняются необходимые пересылки элементов вектора z_s с индексами $\{j_s(1), ..., j_s(m_s - n_s)\}$ в другие процессоры и завершается вычисление $z = \hat{H}r$, где $\hat{H}$ определено в (3.4).

Заметим, что можно не вычислять элементы матриц $\hat{U}_s^T$, а при обращении матриц $\hat{U}_s^T$ использовать способ обращения транспонированных матриц, первоначально предложенный в работе [8].

Рассмотрим способ 1 параллельной реализации с применением MPI+OpenMP технологии этапов построения и обращения предобусловливателя, предложенный ранее в работе [3]. Разбиение всей области расчета при использовании способа 1 производится сразу на *pm* подобластей, где *p* – число используемых процессоров, *m* – число используемых потоков в каждом процессоре. Процессор с номером *s* будет производить вычисления в «большой» подобласти с номером *s*, состоящей из подобластей с номерами $t=(s-1)m+1, ..., sm$, полученными при разбиении. При использовании MPI+OpenMP технологии способом 1 число блоков в предобусловливателе BIIC-IC1(τ) равно *pm*. В программе необходимо задать число «внутренних» блоков с налеганием, равное *m*, и инициализировать число нитей, совпадающее с числом «внутренних» блоков.

Для построения предобусловливателя BIIC-IC1(τ) в каждом процессоре с номером *s* сначала создаются *m* матриц $A_t = V_t^T A V_t$, где $t=(s-1)m+1, \ldots, sm$, для этого совершаются необходимые пересылки строк матрицы *A*. Затем по матрицам A_t в процессоре с номером *s* ($s=1, \ldots, p$) строятся матрицы $\hat{U}_t$ ($t=(s-1)m+1, \ldots, sm$). При этом для каждого *t* вычисления элементов матриц $\hat{U}_t$ происходят в своем потоке с использованием алгоритма 2, в котором вместо матрицы A_s используются отмасштабированные матрицы A_t. Все рекурсивные вычисления при построении матриц $\hat{U}_t$ происходят внутри потоков. В настоящей работе, так же как в работе [3], на этапе построения матриц $\hat{U}_t$ для циклов по $t1=1, \ldots, m$ использовалась директива **do** с опцией **schedule static.** При построении матриц A_t ($t=(s-1)m+1, \ldots, sm$) в каждом процессоре с номером *s* ($s=1, \ldots, p$) тоже используются OpenMP технологии с числом нитей *m*, в настоящей работе использовалась директива **do** с опцией **schedule static**.

Обращение предобусловливателя BIIC-IC1(τ) с использованием формулы

$$z_t = \hat{U}_t^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_t} \end{bmatrix} \hat{U}_t^{-T} r_t, \quad t=(s-1)m+1, \ldots, sm, \quad s=1, \ldots, p \qquad (4.2)$$

происходит следующим образом [3]. Перед началом вычислений элементы векторов r_t для $t=(s-1)m+1, \ldots, sm$, необходимые для расчетов в процессоре с номером *s* и хранящиеся в памяти других процессоров, должны быть пересланы в процессор с номером *s*. Для каждого $t=(s-1)m+1, \ldots, sm$ вычисления элементов векторов z_t по формуле (4.2) происходят в своем потоке. При вычислении

элементов векторов z_t для выполнения циклов по $t1$=1, …, m в настоящей работе использовалась директива **do** с опцией **schedule static**. Используется способ обращения матриц $\hat{U}_t^T$, аналогичный способу обращения транспонированной матрицы из работы [8]. При этом на этапе построения матрицы предобусловливания не нужно вычислять в явном виде матрицы $\hat{U}_t^T$. После вычисления z_t для всех t=(s-1)m+1, …, sm, s=1, …, p следует выполнить необходимые пересылки и завершить вычисление вектора z.

Заметим, что в случае p=1 выше описан способ параллельной реализации с применением OpenMP технологии этапов построения и обращения предобусловливателя BIIC-IC1(τ) для m блоков с налеганием.

Рассмотрим новый способ применения MPI+OpenMP технологии построения и обращения предобусловливателя BIIC-IC1(τ) - способ 2, аналогичный способу 1 из работ [26,27,28]. В способе 2 в предобусловливателе BIIC-IC1(τ) используется p блоков с налеганием, где p – число процессоров ($p \neq 1$), а внутри каждого блока используется упорядочение узлов сетки расширенных подобластей типа DDO [16]. Заметим, что в способе 2 из работы [3] использовалось упорядочение узлов сетки нерасширенных подобластей типа DDO [16].

Будем использовать два способа разбиения области расчета на подобласти.

При первом способе произведем разбиение всей области расчета сразу на pm подобластей, где p – число используемых процессоров, m – число используемых потоков. Так же, как в способе 1 применения MPI+OpenMP технологии, процессор с номером s будет производить вычисления в «большой» подобласти с номером s, состоящей из подобластей с номерами t=(s-1)m+1, …, sm, полученными при разбиении. При этом каждая нерасширенная подобласть с номером s, соответствующая расчетам на своем процессоре, уже получается разбита на m «внутренних» подобластей, а при применении OpenMP технологии следует использовать m нитей. При разбиении расширенной подобласти будем использовать это разбиение узлов нерасширенной подобласти, а узлы налегания добавим в первую «внутреннюю» подобласть и расположим вначале первой «внутренней» подобласти. Такой способ разбиения при использовании второго способа применения MPI+OpenMP технологии будем называть способом разбиения 1 или просто разбиением 1.

Будем использовать также другой способ разбиения области расчета на подобласти при применении MPI+OpenMP технологии – способ разбиения 2 или сокращенно разбиение 2. При этом вся область расчета разбивается на p подобластей так же, как при использовании только MPI. Затем каждая нерасширенная подобласть с номером s (s=1, …, p) разбивается на m «внутренних» подобластей, где m – число используемых нитей (потоков) при применении OpenMP технологии, с приблизительно равным числом узлов

сетки. В настоящей работе разбиение нерасширенных подобластей осуществлялось в порядке следования узлов нерасширенных подобластей, установленном ранее, следующим образом [27]. Пусть $\hat{l}_s = [n_s / m]$, первые m-1 «внутренних» подобластей содержат $\hat{l}_s$ узлов, последняя внутренняя подобласть содержит $n_s - (m-1)\hat{l}_s$ узлов. При разбиении расширенной подобласти способом 2 использовалось это разбиение узлов нерасширенной подобласти, а узлы налегания добавлялись в первую «внутреннюю» подобласть и располагались вначале первой «внутренней» подобласти.

После проведения разбиения 1 или разбиения 2 расширенной подобласти производится переупорядочение узлов расширенной подобласти в два этапа. На первом этапе используется упорядочение типа DDO, предложенное в работе [16], а затем использованное для применения OpenMP технологии в работах [26, 27, 28]. Введем множество узлов разделителей – множество узлов сетки во «внутренних» подобластях, у которых имеются соседи из «внутренних» подобластей с бóльшими номерами. Остальные узлы сетки во «внутренних» подобластях будем называть «внутренними». Множество узлов разделителей разобьем на 3 части. Узел разделителя назовем узлом разделителя первого уровня, если в шаблоне этого узла нет узлов разделителей из других подобластей с номерами, бóльшими, чем номер рассматриваемой подобласти. Узел разделителя назовем узлом разделителя второго уровня, если в шаблоне этого узла нет узлов разделителей более высокого, чем первый, уровня, расположенных в подобластях с бóльшими номерами. Остальные узлы разделителей назовем узлами разделителей третьего уровня. Заметим, что в случае 7-точечной разностной задачи Дирихле для уравнения Пуассона на равномерной ортогональной сетке и разбиения кубической области расчета на подобласти плоскостями параллельными граням куба при равномерном разбиении на одинаковое число в каждом направлении (когда получившиеся подобласти являются кубическими) получим: разделители первого уровня – некоторые грани маленьких кубиков (получившихся после разбиения), второго уровня – некоторые ребра, а третьего уровня – некоторые вершины маленьких кубиков. Установим следующий порядок следования узлов сетки в расширенной подобласти. Сначала идут все «внутренние» узлы «внутренних» подобластей в порядке следования номеров «внутренних» подобластей, причем сохраняется порядок следования узлов внутри каждой «внутренней» подобласти, введенный ранее. Затем идут узлы разделителей первого уровня, затем второго уровня, а затем третьего уровня. При этом для каждого уровня разделителей узлы следуют с сохранением порядка следования «внутренних» подобластей и порядка следования узлов внутри подобласти, введенного ранее. Заметим, что можно использовать другое упорядочение узлов сетки типа DDO. Как показали расчеты тестовых задач из работы [27] методом сопряженных градиентов с предобусловливателем блочного Якоби в сочетании IC1(τ)

(BJIC1(τ)), использование другого упорядочение узлов сетки типа DDO [9] не приводит к существенному изменению времени счета задачи.

Определим $\bar{l}_s$ – минимальное значение количества «внутренних» узлов при разбиении расширенной подобласти с номером *s* на «внутренние» подобласти способом 1 или 2. Предполагается, что $\bar{l}_s \neq 0$. Обозначим $M1 = m\bar{l}_s$. На втором этапе переупорядочения узлов сетки расширенной подобласти с номером *s* (*s*= 1, …, *p*) установим следующий порядок следования узлов. Сначала идут $\bar{l}_s$ «внутренних» узлов первой «внутренней» подобласти, затем $\bar{l}_s$ «внутренних» узлов второй «внутренней» подобласти и т. д, наконец, $\bar{l}_s$ «внутренних» узлов *m*-ой «внутренней» подобласти. Затем идут оставшиеся «внутренние» узлы «внутренних» подобластей в порядке следования номеров «внутренних» подобластей и в порядке следования узлов внутри «внутренних» подобластей. Затем следуют узлы разделителей в установленном ранее на первом этапе переупорядочения порядке.

Заметим, что узлы налегания в принципе могут попасть в множество узлов разделителей, в отличие от способа применения MPI+OpenMP технологии на основе применения переупорядочения типа DDO в работе [3], когда использовалось отсечение по позициям при построении матрицы предобусловливателя в строках, соответствующих узлам из налегания. В предложенном в настоящей работе способе 2 использования MPI+OpenMp технологии никакого отсечения по позициям при построении предобусловливателя не производится.

Заметим, что, как показали расчеты тестовых задач методом сопряженных градиентов с предобусловливанием BJIC1(τ), применение OpenMP технологии для вычисления верхнетреугольного множителя предобусловливателя и обращения матрицы предобусловливателя BJIC1(τ) для всех «внутренних» узлов «внутренних» подобластей в подавляющем большинстве случаев нецелесообразно.

В способе 2 применения OpenMP технологии в каждом процессоре с номером *s* (*s*=1,…,*p*) при построении первых *M*1 строк верхнетреугольного множителя матрицы предобусловливания IC1(τ) при новом окончательном упорядочении узлов сетки используется цикл по *k*2=1, …, *m*, внутри которого осуществляется вычисление элементов строк искомой матрицы с номерами 1, …, *M*1. При этом все рекурсивные вычисления происходят внутри потоков. Для выполнения цикла по *k*2 в настоящей работе использовалась директива **do** с опцией **schedule static**. Затем без использования OpenMP технологии производится вычисление элементов оставшихся строк этой матрицы. Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства строк верхнетреугольного множителя матрицы предобусловливания IC1(τ) вычисления его элементов происходят с использованием OpenMP технологии. При вычислении элементов строк этой

матрицы используется алгоритм 2. Затем с помощью транспонирования производится определение элементов нижнетреугольного множителя матрицы предобусловливания IC1(τ) при новом окончательном упорядочении. На этом этапе OpenMP технологии не применяются, так как это оказалось неэффективно.

При обращении предобусловливателя перед началом вычислений элементы вектора r_s, необходимые для вычисления z_s в процессоре с номером s и хранящиеся в памяти других процессоров, должны быть пересланы в процессор с номером s. Затем производится переупорядочение элементов вектора r_s. Первый этап обращения предобусловливателя (обращение нижнетреугольных матриц) с использованием OpenMP технологии происходит аналогично вычислению верхнетреугольного множителя матрицы предобусловливания IC1(τ) (при новом упорядочении). С использованием OpenMP технологии вычисляются $M1$ элементов искомого вектора с номерами 1, … $M1 = m\bar{l}_s$ (при новом упорядочении), в настоящей работе использовалась директива **do** с опцией **schedule static**. На этапе обращении верхнетреугольных матриц вычисления происходят в обратном порядке, элементы искомого вектора с номерами $M1 = m\bar{l}_s$, …,1 (при новом упорядочении) вычисляются с использованием OpenMP технологии, в настоящей работе использовалась директива **do** с опцией **schedule static**. После вычисления z_s следует вернуться к первоначальному упорядочению, выполнить необходимые пересылки и завершить вычисление вектора z.

Вычисление элементов вектора $q_k = Ap_k$, где k – номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, когда матрица A хранится в памяти в распределенном CRS-формате, содержит верхний и нижний треугольники, с использованием MPI и MPI+OpenMP технологии достаточно хорошо изучено, описано, например, в [24, 28]. При применении MPI + OpenMP подхода для умножения матрицы на вектор в настоящей работе использовалась директива **do** с опцией **schedule static.** MPI+ OpenMP реализация вычислений векторных операций и скалярных произведений в алгоритме 1 тоже хорошо известна. При применении MPI + OpenMP подхода для вычислений векторных операций и частичных сумм в скалярных произведениях в настоящей работе использовалась директива **do** с опцией **schedule static.**

5. Результаты расчетов

Программы, реализующие применение метода BIIC- IC1-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии. Здесь и далее параметр τ в названии предобусловливателя может быть опущен. Расчеты проводились на многопроцессорном вычислительном кластере К60, установленном в ЦКП

ИПМ им. М.В. Келдыша РАН. Тестирование и сравнение методов производилось с помощью решения модельной задачи – разностной задачи Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке, причем n=1048576. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы **5_1048576**). Для тестирования рассматриваемых параллельных методов использовались также некоторые матрицы из коллекции разреженных матриц SuiteSparse [30]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

apache2 – трехмерная конечно-разностная схема;
parabolic_fem – уравнение диффузии-конвекции с постоянным переносом;
ecology2 – приложение теории электрических цепей к задаче передачи генов;
boneS01 – модель трубчатой кости.

В таблице 1 приведены некоторые свойства этих матриц, причем значения $Cond(A_0)$, где $A_0 = (D_A)^{-1/2} A (D_A)^{-1/2}$ – матрица системы уравнений после масштабирования, взяты из работы [32], Id – количество строк без диагонального преобладания, Ip – количество положительных внедиагональных элементов, NZA – число ненулевых элементов матрицы A, $nz_{\min}$, $nz_{\max}$ – минимальное и максимальное числа ненулевых элементов в строках матрицы A, N – число неизвестных в системе уравнений (1.1).

Таблица 1

Свойства некоторых матриц из коллекции разреженных матриц SuiteSparse

Матрица	N	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_S)$
apache2	715176	4817870	2	0	4	8	0.12+7
parabolic_fem	525825	3674625	0	1048576	3	7	0.20+6
ecology2	999999	4995991	1124	0	3	5	0.63+8
boneS01	127224	5516602	127222	2064830	12	67	0.13+8

Модельную задачу далее будем называть задачей 1. Задачи с матрицами **parabolic_fem, apache2, ecology2** будем называть соответственно задачей 2, задачей 3 и задачей 4. Задачу с более заполненной матрицей **boneS01** будем называть задачей 5.

Решалось уравнение $Ax = b$, где $A = A_0$, с единичной правой частью ($b_i \equiv 1$), начальное приближение $x_0 \equiv 0$, счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета на p подобластей, а также на pm подобластей (при использовании способа 1 применения MPI+OpenMP технологии и способа 2 применения MPI+OpenMP технологии с разбиением 1) использовался алгоритм [29]. При решении задач 1 – 4 методом BIIC-IC1(τ)-CG использовалось значение $\tau = 0.01$, а задачи 5 с матрицей **boneS01** – $\tau = 0.005$, что было продиктовано требованием безотказности метода BIIC-IC1-CG при решении задачи 5.

В таблицах 2 – 6 приведены числа итераций и времена счета методом BIIC-IC1(τ)-CG тестовых задач 1 – 5 при использовании для параллельной реализации MPI и MPI+OpenMP технологии способами 1 и 2. Под временем вычислений в таблицах 2 – 6 подразумевается время счета в секундах итерационного процесса в сумме с временем вычисления предобусловливателя. При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 12, 16 нитей. В таблицах 2 – 6 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей, которые приведены в таблицах в скобках. Результаты расчетов с применением MPI+OpenMP технологии способом 2 с использованием разбиения 1 обозначены var2.1, а с применением MPI+OpenMP технологии способом 2 с использованием разбиения 2 - var 2.2.

Для способов 1, 2 в таблицах 2 – 6 сверху приводятся результаты расчетов с применением только MPI с разбиения всей области расчета на p подобластей с помощью алгоритма [29]. В следующей строке приводятся результаты расчетов с применением только MPI с разбиением всей области расчета на pm подобластей с помощью алгоритма [29], при этом каждый процессор с номером s (s=1, …, p) производил расчеты в подобласти, являющейся объединением m подобластей с номерами t=(s-1)m+1, …, sm. Для каждого p в этих строках в таблицах 2-6 использовалось значение m, при котором было получено минимальное время счета задачи при применении MPI+OpenMP технологии соответственно способом 1 или способом 2 при разбиении 1.

Таблица 2

Числа итераций и времена счета методом BIIC-IC1(τ)-CG задачи с матрицей **5_1048576** на p процессорах без применения и с применением OpenMP технологии

p	2	4	8	10	16
способ1	401, 19.55	401, 11.02	435, 6.67	427, 5.04	444, 3.86
MPI	411, 20.19	429, 12.12	457, 6.39	451, 5.23	482, 3.7
MPI+OpenMP	469, 6.13	444, 6.33	469, 4.05	493, 4.17	493, 4.16
μ	(12) *3.29*	(4) *1.91*	(3) *1.58*	(3) *1.25*	(3) *0.89*
способ2 MPI	401, 18.94	401, 10.76	435, 6.4	427, 4.8	444, 3.77
	411, 19.39	421,10,19	457, 6.17	464, 5.18	482,3.62
MPI+OpenMP var2.1	449, 5.99	474, 4.51	530, 3.9	530, 4.18	553, 3.57
μ	(12) *3.23*	(6) *2.26*	(3) *1.58*	(4) *1.24*	(3) *1.01*
var2.2	483, 6.02	465, 4.17	493, 3.76	515, 3.75	531, 3.88
μ	(12) *3.14*	(6) *2.58*	(3) *1.7*	(4) *1.28*	(3) *0.97*

Таблица 3

Числа итераций и времена счета методом BIIC-IC1(τ)-CG задачи с матрицей **parabolic_fem** на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ1 MPI	415, 0.98 453, 12.15	418, 5.79 454, 6.3	453, 3.54 440, 3.27	463, 3.09 476,3.27	353, 1.80 488, 2.21
MPI+OpenMP μ	454, 3.19 (12) *3.8*	453, 3.14 (4) *2.0*	454, 2.08 (3) *1.7*	497, 2.66 (4) *1.23*	509, 2.28 (3) *0.96*
способ2 MPI	415, 10.34 453,11.81	418, 5.7 465, 6.23	440, 3.25 453, 3.46	463, 3.00 476, 3.17	453, 1.78 488, 2.17
MPI+OpenMP var2.1 μ	526, 4.1 (12) *2.88*	533, 2.79 (6) *2.23*	521, 2.20 (3) *1.57*	586, 2.72 (4) *1.16*	574, 2.29 (3) *0.95*
var2.2 μ	483, 3.24 (12) 3.19	493, 2.31 (6) *2.46*	530, 2.06 (3) *1.57*	521, 2.4 (4) *1.25*	571, 2.12 (3) *0.84*

Таблица 4

Числа итераций и времена счета методом BIIC-IC1(τ)-CG задачи с матрицей **apache2** на *p* процессорах без применения и с применением OpenMP технологии

p	2	4	8	10	16
способ1 MPI	489, 16.69 499, 17.9	501,11.21 530, 9.91	529, 5.36 559, 5.87	613, 5.15 540, 4.60	542, 2.97 619, 3.31
MPI+OpenMP μ	542, 6.78 (8) *2.64*	611, 6.89 (8) *1.43*	577, 4.3 (3) *1.36*	618, 5.29 (3) *0.97*	645,4.5 (3) *0.73*
способ2 MPI	489, 16.48 499, 17.23	501, 10.87 513, 9.41	529, 5.20 559, 5.88	540, 4.38 559, 5.88	542, 2.99 559, 5.88
MPI+OpenMP var2.1 μ	538, 5.7 (12) *3.02*	630, 4.58 (6) *2.05*	720, 4.25 (3) *1.38*	776, 4.28 (4) *1.15*	850, 3.94 (3) *0.83*
var2.2 μ	568, 7.96 (8) *2.07*	587, 4.65 (6) *2.33*	675, 3.72 (3) *1.4*	745, 4.19 (4) *1.04*	750, 3.52 (3) *0.84*

При оценке ускорения от применения OpenMP технологии способом 1 и способом 2 при разбиении 1 имеет смысл производить также сравнение времен счета при применении MPI и MPI+OpenMP при использовании одинаковых разбиений области расчета, так как в этом случае число узлов и первоначальное упорядочение узлов сетки (до применения переупорядочения типа DDO) в

нерасширенных p подобластях совпадают. В таблицах 2 – 6 приведены курсивом коэффициенты μ ускорения счета благодаря использованию OpenMP

Таблица 5

Числа итераций и времена счета методом BIIC-IC1(τ)-CG задачи с матрицей **ecology2** на p процессорах без применения и с применением OpenMP технологии

p	2	4	8	10	16
способ1 MPI	697, 31.84 729,35.16	721,17.34 729,35.16	740, 9.87 796, 11.06	751, 8.25 836, 9.26	790, 6.27 836, 9.26
MPI+OpenMP μ	809,10.19 (12) *3.45*	790, 9.98 (4) *1.91*	809,6.68 (3)*1.66*	846,8.2 (4) *1.13*	854,7.07 (3) *0.82*
способ2 MPI	697, 30.87 729, 33.83	721,16.64 769,18.14	740, 9.32 769, 8.14	751, 7.96 836, 9.10	790, 6.03 836, 9.10
MPI+OpenMP var2.1 μ	768, 9.29 (12) *3.64*	843,6.65 (6) *2.73*	878,5.68 (3) *1.9*	952, 6.7 (4) *1.36*	852, 5.49 (3) *1.07*
var2.2 μ	821, 8.17 (12) *3.77*	842, 5.89 (6) *2.82*	838, 5.03 (3) *1.85*	870, 5.79 (4) *1.37*	926, 5.52 (3)*1.09*

Таблица 6

Числа итераций и времена счета методом BIIC-IC1(τ)-CG задачи с матрицей **boneS01**на p процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ1 MPI	301,10.09 331,11.99	348, 6.19 351, 6.76	335, 3.28 349, 3.66	376, 3.02 362, 3.08	380,2.04 396,2.31
MPI+OpenMP μ	365, 3.4 (12) *3.52*	402, 3.2 (12) *2.11*	365, 2.44 (3) *1.5*	378, 2.99 (3) *1.03*	404,2.32 (3) *0.99*
способ2MPI	301,10.24 310,10.89	348, 6.26 322, 6.34	335, 3.49 349, 3.7	376, 3.16 362, 3.14	380, 2.1 396,2.32
MPI+OpenMP var2.1 μ	387, 7.89 (4) *1.38*	440, 4.95 (6) *1.28*	471, 3.48 (3) *1.06*	512,3.76 (3) *0.83*	577,3.55 (3) *0.65*
var2.2 μ	395, 9.36 (4) *1.09*	419,6.76 (6) *0.93*	461,3.85 (3) *0.9*	509,4.24 (4) *0.74*	530, 3.39 (3) *0.61*

технологии на том же числе процессоров, которые вычислялись с помощью сравнения времен счета с использованием MPI и MPI+OpenMP при

использовании одинакового способа разбиения области расчета на p подобластей.

Как видно из таблиц 2 – 6, даже при использовании только MPI для фиксированного числа процессоров p числа итераций и времена счета различны при использовании разных способов разбиения. При использовании только MPI и одинакового разбиения области расчета времена счета в таблицах 2 – 6 отличаются в строках, соответствующих способу 1 и способу 2 из-за использования в способе 2 явного представления матриц $\hat{U}_s^T$.

На рисунках 1 – 5 приведены графики зависимости времени счета тестовых задач методом BIIC-IC1-CG от числа процессоров p в логарифмическом масштабе с использованием только MPI и разбиения на p подобластей с помощью алгоритма [29] (сплошные линии), а также с использованием MPI+OpenMP технологии (штриховые и штрих-пунктирные линии). Обозначения var1 и var2 на рис. 1 – 5 соответствуют расчетам с использованием только MPI, причем обозначение var1 соответствует расчетам без вычисления элементов матриц $\hat{U}_s^T$, а var2 – с вычислением элементов матриц $\hat{U}_s^T$ с помощью транспонирования. Обозначение var1,th на рис. 1 – 5 соответствует результатам расчетов с использованием MPI+OpenMP технологии способом 1. Обозначение var2.1,th на рисунках соответствует

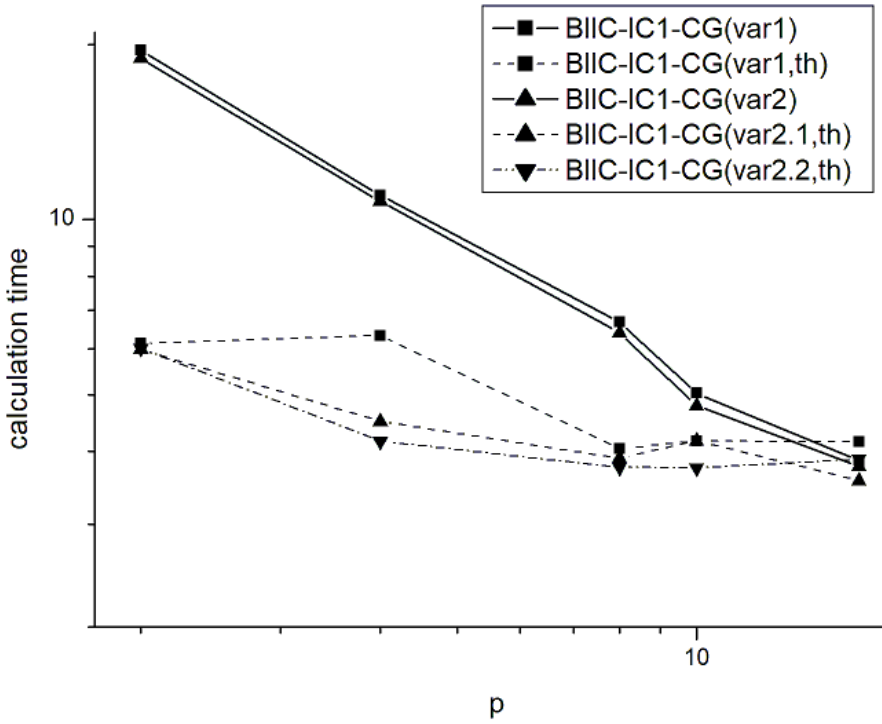

Рис. 1. Времена счета задачи с матрицей **5_1048576** методом BIIC-IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

результатам расчетов с использованием способа 2 применения MPI+OpenMP технологии и разбиения 1, а var2.2,th - результатам расчетов с использованием способа 2 применения MPI+OpenMP технологии и разбиения 2.

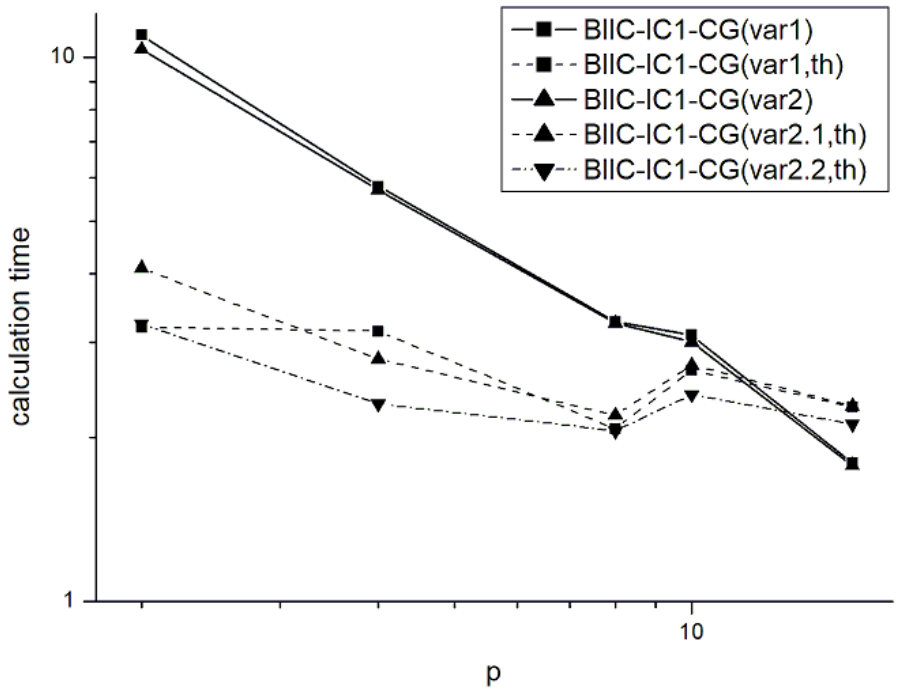

Рис. 2. Времена счета задачи с матрицей **parabolic_fem** методом BIIC-IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

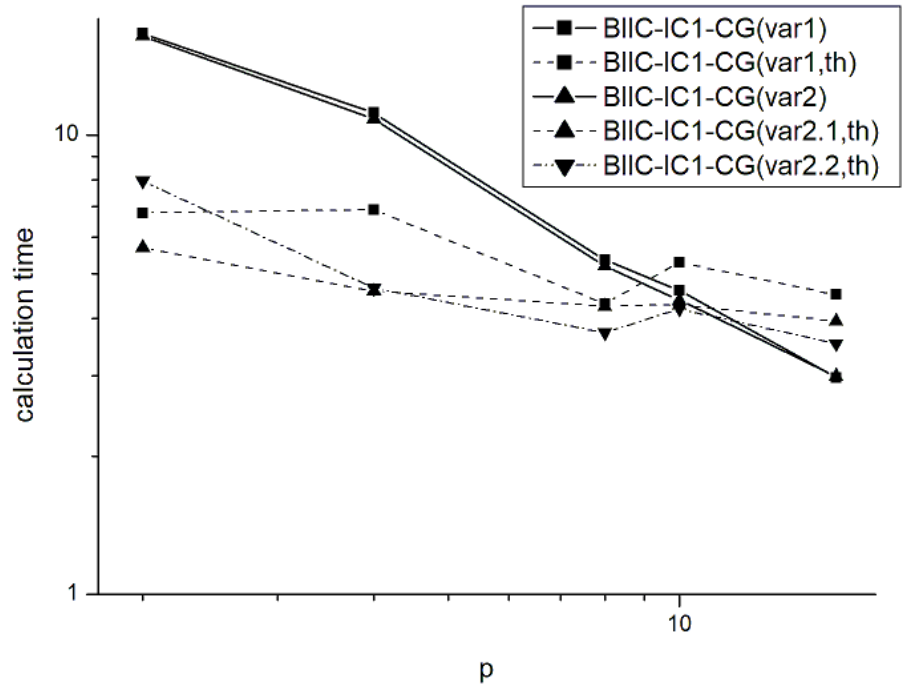

Рис. 3. Времена счета задачи с матрицей **apache2** методом BIIC-IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

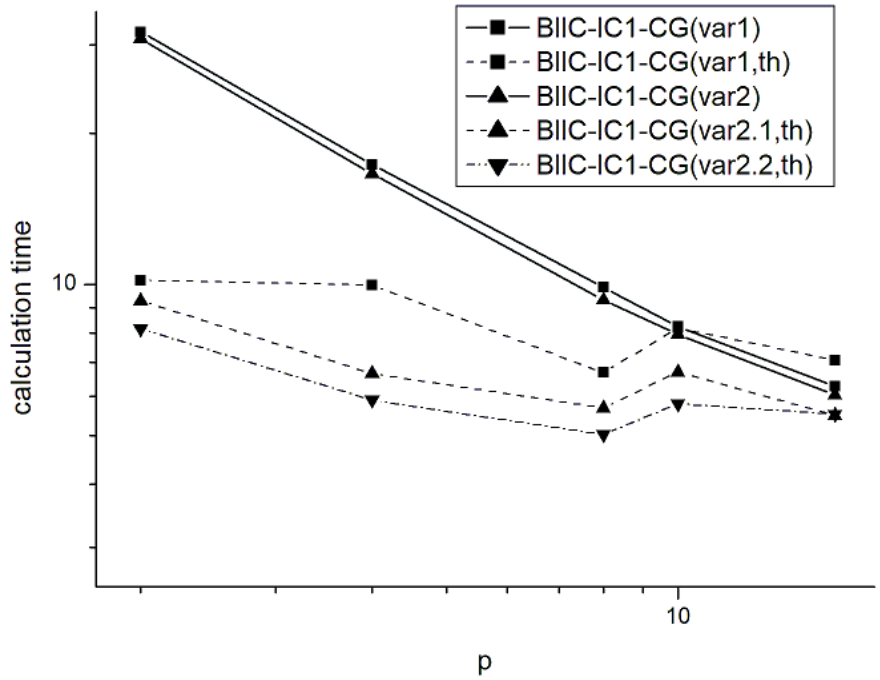

Рис. 4. Времена счета задачи с матрицей **ecology2** методом BIIC-IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

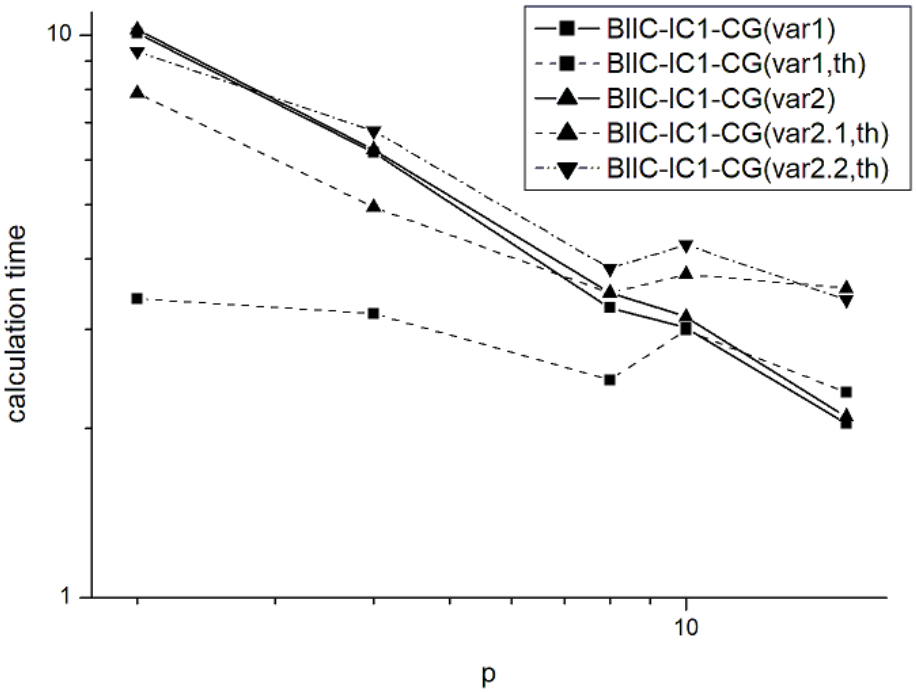

Рис. 5. Времена счета задачи с матрицей **boneS01** методом BIIC-IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

Как видно из таблиц 2 – 6 и рис. 1 – 5, использование способа 1 применения MPI+OpenMP технологии для решения всех задач 1 – 5 позволяет значительно ускорить решение этих задач, по сравнению с использованием только MPI, при

p<10. При p=10, 16 применение OpenMP технологии для решения тестовых задач методом BIIC-IC1-CG становится нецелесообразным.

Как видно из таблиц 2 – 5 и рис. 1 – 4, использование способа 2 применения MPI+OpenMP технологии при разбиениях 1 и 2 позволяет значительно ускорить вычисления по сравнению с использованием только MPI при решении задач с сильно разреженными матрицами: задач 1, 2, 4 при $p \le 10$, а задачи 3 – при p<10. Причем для задач 1 и 4 вычисления с применением MPI+OpenMP технологии с использованием способа 2 почти всегда (при $p \le 16$) происходили быстрее, чем при использовании только MPI.

Как правило, решение задач 1 – 4 с сильно разреженными матрицами с использованием способа 2 применения MPI+OpenMP технологии происходило быстрее, чем с использованием способа 1 применения MPI+OpenMP технологии. Как видно из таблиц 2 – 5 и рис. 1 – 4, при использовании способа 2 применения MPI+OpenMP технологии при разбиении 2 решение тестовых задач 1 – 4 в большинстве случаев происходило быстрее, чем при разбиении 1.

Использование способа 2 применения MPI+OpenMP технологии с разбиением 1 при решении задачи 5 (с более плотной матрицей и маленьким значением τ) методом BIIC-IC1-CG не позволило так же сильно, как в случае более разреженных матриц, уменьшить время счета по сравнению со счетом с использованием только MPI. Причинами этого являются прежде всего большое время вычисления матрицы предобусловливания и большое число узлов сетки, попавших на разделители при разбиении на внутренние подобласти. Ускорение счета благодаря использованию OpenMP технологии способом 2 с разбиением 1 наблюдалось только при p=2,4. Использование способа 2 применения OpenMP технологии с разбиением 2 при решении задачи 5 методом BIIC-IC1-CG оказалось нецелесообразным. Разница в эффективности способа 2 применения MPI+OpenMP технологии при разбиениях 1 и 2 может быть объяснена особенностями разбиения 1. При использовании разбиения 1 разбиение всей области расчета происходило на pm подобластей с помощью алгоритма [29], одной из задач которого является концентрация большей части ненулевых элементов матрицы в ее блочно-диагональной части, содержащей при таком разбиении не менее pm блоков [29]. Заметим, что для сильно разреженных матриц эта проблема не является настолько актуальной.

Заметим, что при использовании способов 1 и 2 применения OpenMP технологии по-разному происходит изменение числа итераций с ростом числа потоков.

На рисунках 6 – 8 приведены графики ускорений счета тестовых задач методом BIIC-IC1(τ)-CG с применением MPI (сплошные линии) и MPI+OpenMP (штриховые линии) по сравнению со счетом на 2 процессорах с применением только MPI. При построении графиков на рис. 6 – 8 использовалось время счета задач на p процессорах с применением только MPI с разбиением области расчета на p подобластей методом [29].

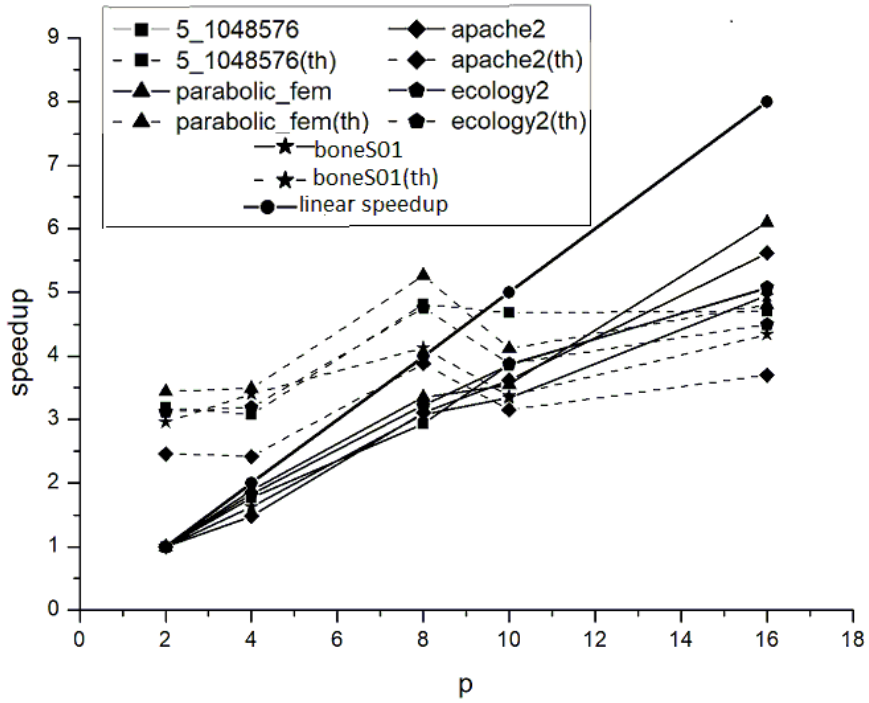

Рис. 6. Ускорения счета тестовых задач методом BIIC-IC1(τ)-CG при применении MPI и при применении MPI+OpenMP технологии способом 1

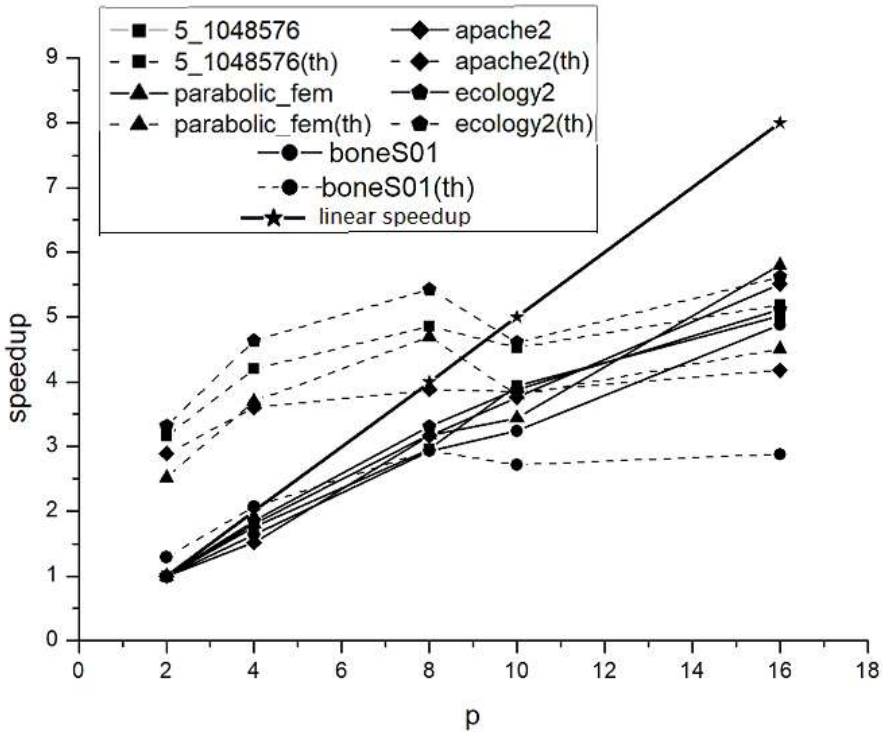

Рис. 7. Ускорения счета тестовых задач методом BIIC-IC1(τ)-CG при применении MPI и при применении MPI+OpenMP технологии способом 2 при разбиении 1 (var2.1)

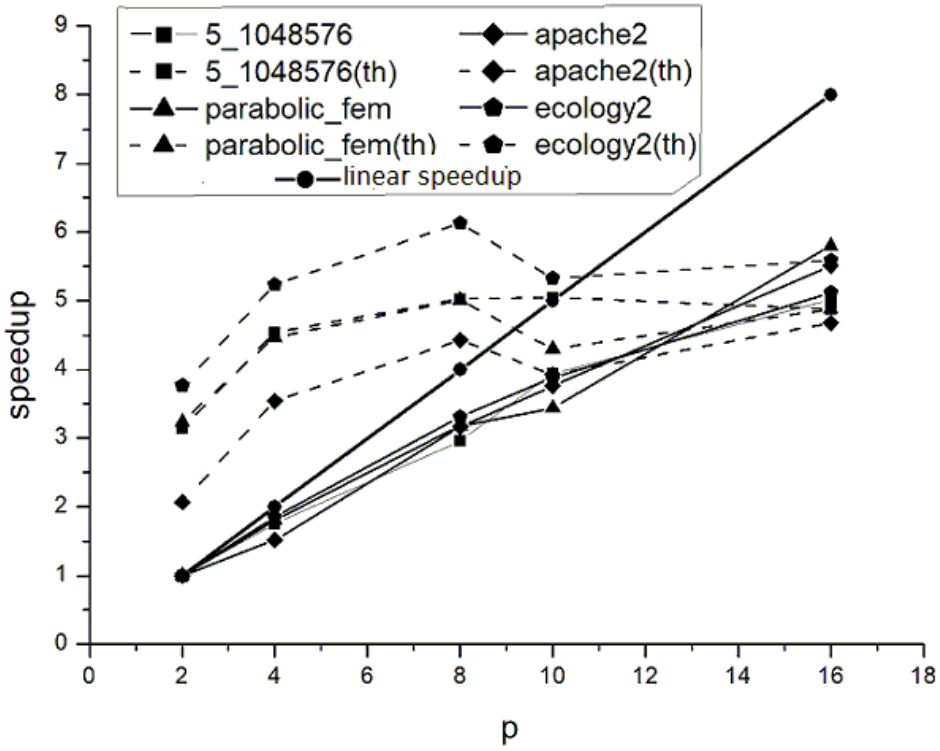

Рис. 8. Ускорения счета тестовых задач методом BIIC-IC1(τ)-CG при применении MPI и при применении MPI+OpenMP технологии способом 2 при разбиении 2 (var2.2)

Как видно из рис. 6, использование способа 1 применения MPI+OpenMP технологии для решения тестовых задач 1,2,4,5 позволило получить сверхлинейное ускорение при p<10, а задачи 3 - при p<8.

Как видно из рис. 7, использование способа 2 применения MPI+OpenMP технологии с разбиением 1 для решения тестовых задач 1,2,4 позволило получить сверхлинейное ускорение при p<10, задачи 3 - при p<8, а задачи 5 только при p=2,4. Как видно из рис. 8, использование способа 2 применения MPI+OpenMP технологии с разбиением 2 для решения тестовых задач 1,4 позволило получить сверхлинейное ускорение при p<16, а задач 2, 3 - при p<10.

Заметим, что алгоритм построения матрицы предобусловливания BIIC-IC1(τ) значительно проще, чем алгоритм построения матрицы предобусловливания BIIC-IC2S(τ) [8], время вычисления предобусловливателя BIIC-IC1(τ) при использовании только MPI и способа 1 применения MPI+OpenMP технологии должно быть меньше, чем время вычисления предобусловливателя BIIC-IC2S(τ). Однако скорость сходимости итерационного процесса метода BIIC-IC2S(τ)-CG должна быть выше, чем в методе BIIC-IC1-CG. Кроме того, метод BIIC-IC2S(τ)-CG является безотказным. На рисунке 9 приведены времена счета модельной задачи с матрицей **5_1048576** для различного числа процессоров при использовании для ее решения методов BIIC-IC2S(τ)-CG и BIIC-IC1(τ)-CG, где τ=0.01, с

применением MPI и MPI+OpenMP технологий. Результаты расчетов методом BIIC-IC2S(τ)-CG взяты из работы [3].

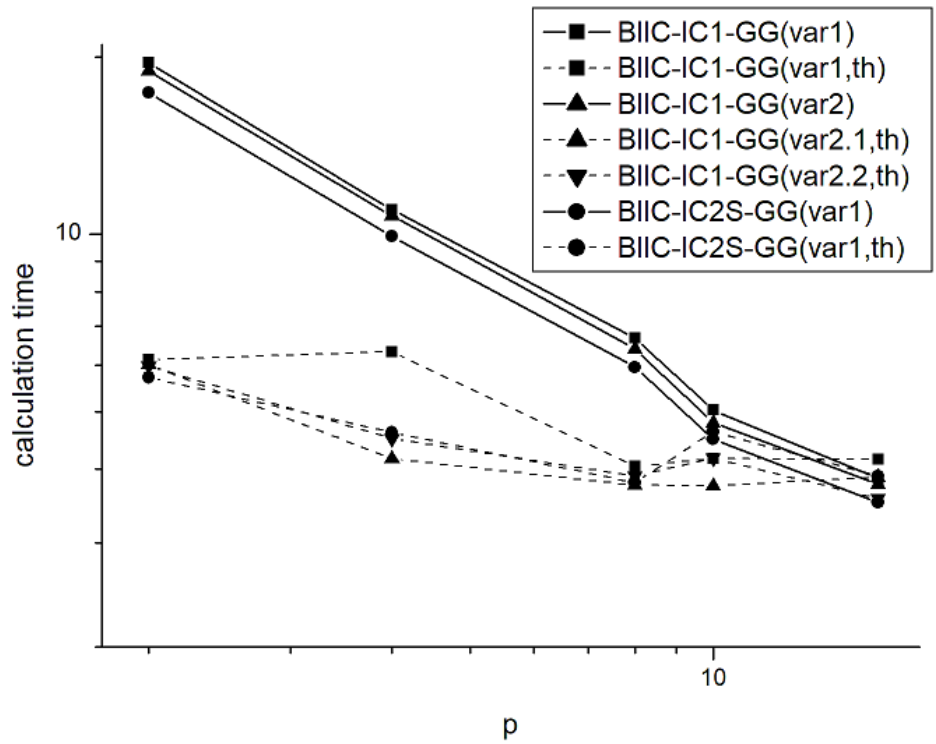

Рис. 9. Времена счета задачи с матрицей **5_1048576** методами BIIC-IC1(τ)-CG, BIIC-IC2S(τ)-CG-CG с использованием MPI и MPI+OpenMP технологии

Как видно из рисунка 9, решение модельной задачи с матрицей **5_1048576** методом BIIC-IC2S(τ)-CG происходит быстрее, чем методом BIIC-IC1(τ)-CG, в случае применения только MPI и MPI+OpenMP способом 1 для всех значений *p*. В случае применения MPI+OpenMP способом 2 при решении задачи методом BIIC-IC1(τ)-CG время счета не так сильно отличается от времени счета методом BIIC-IC2S(τ)-CG с применением MPI+OpenMP способом 1. Причем в случае var2.1,th только для p=2,8, а в случае var2.2,th только для p=2 решение этой задачи методом BIIC-IC2S(τ)-CG происходит быстрее, чем методом BIIC-IC1(τ)-CG. Заметим, что использование способа 2 применения MPI+OpenMP технологии при решении задач методом BIIC-IC2S(τ)-CG нецелесообразно [3].

На рисунке 10 приведены времена счета задачи с матрицей **boneS01** для различного числа процессоров при использовании для ее решения методов BIIC-IC2S(τ)-CG и BIIC-IC1(τ)-CG с применением MPI и MPI+OpenMP технологии способом 1. При этом в методе BIIC-IC1(τ)-CG используется максимальное значение τ=0.005, при котором удается произвести вычисление элементов матрицы предобусловливания (см. раздел 3), в безотказном при любых значениях τ методе BIIC- IC2S(τ)-CG используются значения τ=0.005,

0,01, 0.02. Значения τ, использованные в расчетах, указаны в скобках для каждой кривой на графике на рис. 10 (в рамочке).

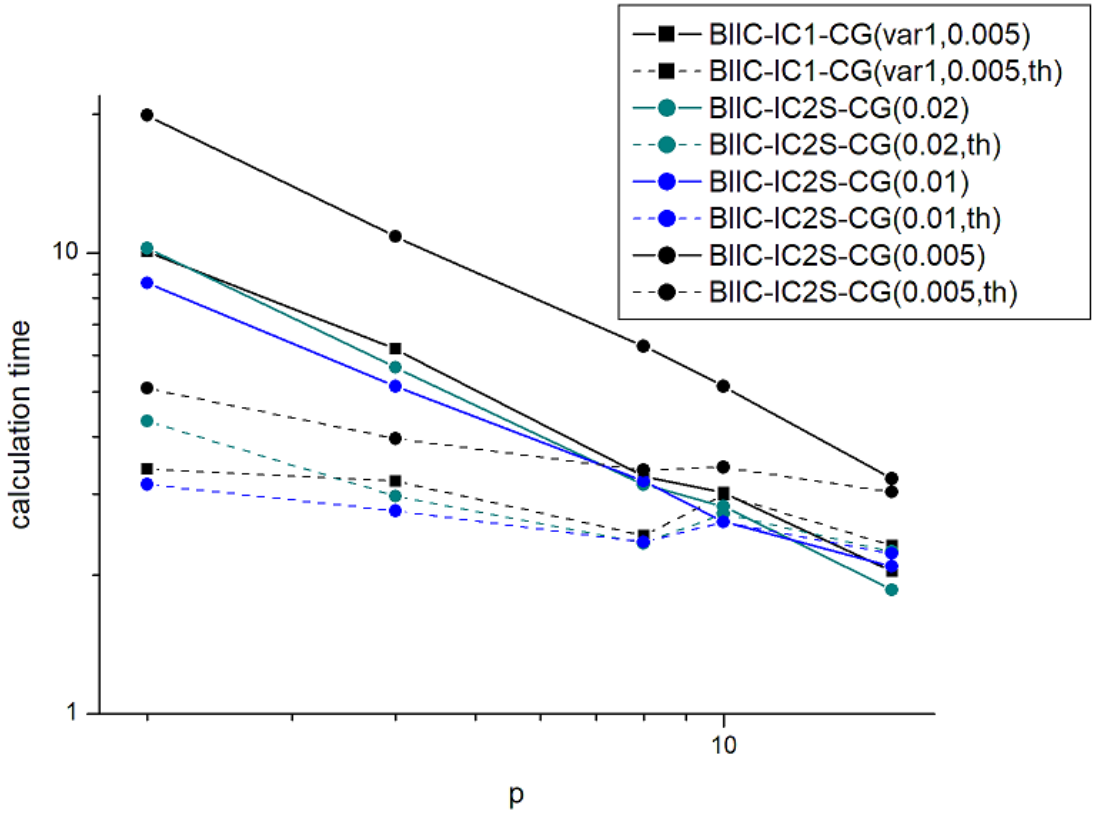

Рис. 10. Времена счета задачи с матрицей **boneS01** методами BIIC-IC1(τ)-CG, BIIC-IC2S(τ)-CG-CG с использованием MPI и MPI+OpenMP технологии

При одинаковых значениях τ=0.005 решение задачи с матрицей **boneS01** методом BIIC-IC2S(τ)-CG происходило даже медленнее, чем методом BIIC-IC1(τ)-CG, как при использовании только MPI, так и при использовании MPI+OpenMP. Это связано с большим временем вычисления предобусловливателя в методе BIIC-IC2S(τ)-CG. При этом число итераций метода BIIC-IC2S(τ)-CG было меньше числа итераций BIIC-IC1(τ)-CG.

Решение этой задачи методом BIIC-IC2S-CG при бóльших значениях τ (τ=0.01, 0.02), когда метод BIIC-IC1(τ)-CG не является безотказным, при использовании только MPI и MPI+OpenMP происходило быстрее, чем методом BIIC-IC1(τ)-CG при τ=0.005, кроме случая p=2, τ=0.02. То есть использование метода BIIC-IC2S(τ)-CG позволяет осуществить решение задачи с матрицей **boneS01** быстрее, чем методом BIIC-IC1(τ)-CG, благодаря безотказности метода BIIC-IC2S(τ)-CG при бóльших значениях τ.

Уменьшение эффекта от использования OpenMP технологии с увеличением числа процессоров объясняется, в частности, уменьшением числа строк матрицы, приходящихся на каждый процессор, т. е. уменьшением вычислительной работы в каждом процессоре. В работах [3, 28] на примере

решения модельной задачи, описанной в начале раздела 5, при различных значениях размерности матрицы n методом BIIC-IC2S(τ)-CG с нулевым налеганием (являющимся методом CG предобусловливанием блочного Якоби в сочетании с IC2S(τ)) показано, что при фиксированном числе процессоров ускорение счета благодаря использованию OpenMP технологии растет с ростом n. В настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, значительно больше. Следует надеяться, что потеря эффективности от применения OpenMP технологии при решении задач с большими размерами матриц наступит при значительно большем числе процессоров.

6. Заключение

В работе рассмотрен предобусловливатель BIIC-IC1(τ) для решения СЛАУ (1.1) методом сопряженных градиентов и способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ) с числом блоков, кратным числу процессоров и числу используемых потоков – способ 1. Предложен новый способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ) с числом блоков, кратным числу процессоров, в котором для мелкозернистого распараллеливания используется переупорядочение узлов сетки внутри расширенных подобластей типа DDO – способ 2. При применении MPI+OpenMP технологии способом 2 использованы два различных способа разбиения области расчета на подобласти. С помощью расчетов модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse показано, что использование способа 1 применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI при решении СЛАУ (1.1) методом BIIC-IC1(τ)-CG для не слишком большого числа узлов суперкомпьютерной системы, использование способа 2 применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для не слишком большого числа узлов суперкомпьютерной системы при решении задач с достаточно разреженными матрицами. При использовании способа 2 применения MPI+OpenMP технологии для решения задачи с более заполненной матрицей удалось получить небольшое ускорение счета по сравнению со счетом с применением только MPI для маленького числа процессоров, причем только при хорошем разбиении области расчета.

Список литературы

1. Kaporin I.E. New convergence results and preconditioning strategies for conjugate gradient method // Numer. Linear Algebra and Appls. 1994. V. 1. N 2. P. 179-210.

2. Капорин И.Е. О предобусловливании метода сопряженных градиентов при решении дискретных аналогов дифференциальных задач // Дифференц. ур-ния. 1990. Т. 26. № 7. С. 1225-1236.
3. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1 // Препринты ИПМ им. М.В. Келдыша. № 48, 2021. 32 с.
4. Munksgaard N. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. № 6. P. 206-219.
5. Tuff A.D., Jennings A. An iterative method for large systems of linear structural equations. // In. j. numer. Methods engrg. 1973. № 7. P.175-183.
6. Капорин И.Е., Коньшин И.Н. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Ж. вычисл. матем. и матем. физики. 2001. Т. 41. № 4. С. 515–528.
7. Kaporin I.E. High quality preconditionings of a general symmetric positive definite matrix based on its $U^TU+U^TR+R^TU$ - decomposition // Numer. Lin. Alg. Appl. 1998. V. 5. P.483-509.
8. Капорин И.Е., Милюкова О.Ю. Массивно-параллельный алгоритм предобусловленного метода сопряженных градиентов для численного решения систем линейных алгебраических уравнений // Сб. трудов отдела проблем прикладной оптимизации ВЦ РАН (под ред. В.Г.Жадана). М.: Из-во ВЦ РАН. 2011. С. 132-157.
9. Duff I.S., Meurant G.A. The effect of ordering on preconditioned conjugate gradients // BIT. 1989. V. 29. P. 625-657.
10. Doi S. On parallelism and convergence of incomplete LU factorizations //Applied Numerical Mathematics: Transactions of IMACS. 1991. V. 7. № 5. P.417–436.
11. Notay Y. An efficient parallel discrete PDE solver // Parallel Computing. 1995. V.21. P.1725-1748.
12. Milyukova O. Yu. Parallel approximate factorization method for solving discreate elliptic equations // Parallel Computing. 2001. № 27. P.1365-1379.
13. Милюкова О.Ю. Некоторые параллельные итерационные методы с факторизованными матрицами предобусловливания для решения эллиптических уравнений на треугольных сетках // Ж. вычисл. матем. и матем. физики. 2006. Т.46. №6. С.1096-1112.
14. Hysom D., Pothen A. A scalable parallel algorithm for incomplete factor preconditioning // SIAM J. Sci. Comput. 2001. V. 22. P. 2194-2215.
15. Magolu Monga Made M., van der Vorst H. A., Spectral analysis of parallel incomplete factorizations with implicit pseudo-overlap // Numer. Linear Algebra Appl. 2002. № 9. P. 45–64.
16. Милюкова О.Ю. Сочетание числовых и структурных подходов к построению неполного треугольного разложения второго порядка в параллельных методах предобусловливания //Журн. вычисл. матем. и матем. физ. 2016. Т. 56. N5. С.711-729.

17. Anderson E. C., Saad Y. Solving sparse triangular systems on parallel computers // International J. of High Speed Computing. 1989. V.1. P. 73–96.
18. Hammond S. W., Schreiber R. Efficient ICCG on a shared memory multiprocessor // International J. High Speed Computing 4. 1992. P. 1–21.
19. Wolf M. M., Heroux M. A., Boman E. G. Factors impacting performance of 535 multithreaded sparse triangular solve // Proceedings of the 9th International Conference on High Performance Computing for Computational Science. VECPAR'10. Springer-Verlag. Berlin. Heidelberg. 2011. P. 32-44.
20. Chow E., Anzt H., Scott J., Dongarra, J. Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning // Journal of Parallel and Distributed Computing. 2018. № 119. P. 219-230.
21. Chow E., A. Patel A. Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. V. 37. P. 169-193.
22. Cayrols S., Duff I., Lopes F. Parallelization of the solve phase in a task-based Cholesky solver using a sequential task flow model // Technical Report RAL-TR-2018-008. Science & Technology Facilities Council. UK. 2018. 27 P.
23. Капорин И.Е., Милюкова О.Ю. MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями // Препринты ИПМ им. М.В. Келдыша. № 8, 2018. 28 с. doi:10.20948/prepr-2018-8.
24. Капорин И.Е., Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными явными предобусловливателями // ВАНТ. Серия Математическое моделирование физических процессов. 2018. Вып.4. С. 57-69.
25. Chow E. Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns // Internat. J. High Performance Comput. Appl. 2001. V.15. N.1. P. 56-74.
26. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем. // Препринты ИПМ им. М.В. Келдыша. 2020. № 31. 22 с. https://doi.org/10.20948/prepr-2020-31.
27. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1. – М.: Препринты ИПМ им. М.В. Келдыша. 2020. № 83. 28 с. https://doi.org/10.20948/prepr-2020-83.
28. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями// Математическое моделирование.2021.Т. 33. № 10. с.19-39
29. Капорин И.Е., Милюкова О.Ю. Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов // Препринты ИПМ им. М.В.Келдыша. 2017. № 37. 28 с. doi:10.20948/prepr-2017-37.
30. Davis T., Hu Y.F. University of Florida sparse matrix collection.//ACM Trans. on Math.~Software. 2011. V.38, N.1// http://www.cise.ufl.edu/research/sparse/matrices.

31. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
32. Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т.52. № 2. С.1-26.

Оглавление