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

Препринты ИПМ • Препринт № 48 за 2021 г.

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

О.Ю. Милюкова

MPI+OpenMPI реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. MPI+OpenMPI реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1 // Препринты ИПМ им. М.В.Келдыша. 2021. № 48. 32 с. https://doi.org/10.20948/prepr-2021-48
https://library.keldysh.ru/preprint.asp?id=2021-48

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

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1

Москва — 2021

Милюкова О.Ю.

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1

В работе предлагается новый предобусловливатель для решения систем линейных алгебраических уравнений с симметричной положительно определенной матрицей методом сопряженных градиентов – предобусловливатель блочного неполного обратного разложения Холецкого BIIC в сочетании с треугольным разложением первого порядка «по значению» – BIIC-IC1. Предложен способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC в сочетании со стабилизированным треугольным разложением второго порядка «по значению» – BIIC-IC2S. При этом в предобусловливателе BIIC-IC2S число блоков кратно числу используемых процессоров и числу используемых потоков. Предложены два способа применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1. Проводится сравнение времени решения задач с использованием исходной MPI технологии и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse.

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

Olga Yurievna Milyukova

MPI+OpenMP parallel implementation of conjugate gradient method with preconditioner of block partial inverse triangular decomposition of IC2S and IC1

The paper proposes a new preconditioner for solving systems of linear algebraic equations with a symmetric positively defined matrix by the method of conjugate gradients – Block Incomplete Inverse Cholesky BIIC preconditioner in combination with a triangular first-order decomposition "by value" - BIIC-IC1. The algorithm based on MPI+OpenMP techniques is proposed for the construction and application of the BIIC preconditioner combined with stabilized triangular decomposition of the second order "by value" (BIIC-IS2S). In this case, the BIIC-IC2S preconditioner uses the number of blocks multiple of the number of processors used and the number of threads used. Two algorithms based on MPI+OpenMP techniques are proposed for the construction and application of the BIIC-IC1 preconditioner. Comparative timing results for the MPI+OpenMP and MPI implementations of the proposed preconditioning used with the conjugate gradient method for a model problem and the sparse matrix collections SuiteSparse are presented.

Keywords: sparse matrices, 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]. Для каждого блока предобусловливателя BIIC точное разложение Холецкого заменяется соответствующим приближенным стабилизированным треугольным разложение второго порядка «по значению» IC2S(τ), $0<\tau<<1$ [3]. В работе также используется новый предобусловливатель, в котором для каждого блока предобусловливателя BIIC точное разложение Холецкого заменяется соответствующим приближенным треугольным разложением с отсечением по параметру $0<\tau<<1$ первого порядка IC1(τ). 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], алгоритм построения предобусловливателя IC2S(τ) является безотказным [3].

Блочная версия неполного обратного треугольного разложения BIIC в сочетании с неполным треугольным разложением второго порядка BIIC-IC2(τ) предложена и исследована в работе [6]. В работе [6] для построения предобусловливателя BIIC-IC2(τ) специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение второго порядка IC2(τ) [3]. В работе [7] 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. Методы сопряженных градиентов с предобусловливанием BIIC-IC2(τ) и с предобусловливанием BIIC-IC2S(τ) являются безотказными. Метод сопряженных градиентов с

предобусловливанием BIIC-IC1(τ), предложенный в настоящей работе, имеет ограниченную область применимости, в ряде случаев для безотказности требует чрезмерного уменьшения параметра τ. Методы сопряженных градиентов с предобусловливанием BIIC-IC2(τ) и BIIC-IC2S(τ) были эффективно реализованы на параллельных архитектурах с распределенной памятью [6, 7].

Другим способом преодоления основной трудности распараллеливания алгоритмов построения и обращения предобусловливателя неполного треугольного разложения, связанной с рекурсивным характером вычислений, является использование переупорядочения узлов сетки и соответствующей перестановки строк и столбцов матрицы, например, использование переупорядочений, связанных с разбиением области расчета (DDO - Domain Decompositing Ordering) [8]. Применению такого подхода для крупнозернистого распараллеливания посвящено много работ, например, [9 - 14]. В работе [15] предлагается использовать упорядочение типа DDO для построения параллельного варианта метода стабилизированного треугольного разложения второго порядка сопряженных градиентов (IC2S-CG).

Основное внимание в настоящей работе уделяется применению OpenMP технологии для параллельной реализации алгоритмов метода сопряженных градиентов с предобусловливателями BIIC-IC2S, BIIC-IC1 при проведении расчетов на многопроцессорной вычислительной системе. Здесь и далее параметр τ в названии предобусловливателя опущен.

В работах [16 - 19] 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 (мелкозернистый или распараллеливание алгоритма на потоки). В работе [20] предлагается безытерационный способ применения MPI+OpenMP технологии при обращении факторизованного предобусловливателя. В работе [21] предлагается новый итерационный алгоритм вычисления неполного IC(0), IC(1), IC(2) разложений, в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно. В работах [22, 23] при применении предобусловливания ILU при решении задач с использованием GPU было использовано многоцветное упорядочение.

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 [24 - 26].

В работах [27, 28] предлагаются 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). Эти способы основаны на переупорядочении узлов сетки типа DDO внутри каждой подобласти (способ 1)

и на уменьшении шаблона разреженности матрицы A при построении предобусловливателя (способ 2). 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 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 работах [29, 30], являющиеся обобщением упорядочения [7]. 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 предлагается новый предобусловливатель BIIC-IC1 для решения СЛАУ (1.1) методом сопряженных градиентов. Предлагается безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1 – способ 1. При этом в предобусловливателе BIIC используется вместо p блоков, где p – число процессоров, pm – блоков, где m – число потоков. OpenMP технологии применяются для всех строк матрицы на этапах построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1. В настоящей работе предлагается также безытерационный способ 2 применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1. При этом в предобусловливателе BIIC используется p блоков, а для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO. При использовании способа 2 при построении матрицы предобусловливания для некоторых ее строк осуществляется отсечение по позициям, при построении и обращении предобусловливателя OpenMP технологии применяются для большинства строк матрицы. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателями BIIC-IC2S и BIIC-IC1 с использованием MPI и MPI+OpenMP подходов на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [31].

2. Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1.1). Алгоритм предобусловленного метода сопряженных градиентов (см., например, [32]) 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

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

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

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

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. Неявное блочное предобусловливание

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

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

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

$$V_s^T A V_s = \overline{U}_s^T \overline{U}_s, \quad s=1,\ldots, p. \tag{3.2}$$

Предобусловливатель блочного неполного обратного разложения Холецкого (ВИС) имеет вид [1, 2]

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

В работах [1, 2] показано, что ВПС предобусловливатель обладает свойством *K*-оптимальности в следующем смысле. Пусть Ω – множество разреженных нижнетреугольных матриц *L*, которые имеют ненулевые элементы в позициях $j \in \{j_s(1), ..., j_s(m_s - n_s), k_{s-1}+1, ..., i\}$, $k_{s-1}+1 \le i \le k_s$. Тогда

$$H = \arg \min_{H=L^T L, L \in \Omega} K(HA),$$

где $K(HA)$ – K - число обусловленности матрицы *HA* [1, 2],

$$K(HA) = \left(trace(HA)/n\right)^n / \det(HA).$$

При этом оценка числа итераций предобусловленного метода сопряженных градиентов имеет вид [1, 2]:

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

То есть при $k \ge i_K(\varepsilon)$ $\|Ax_k - b\|_H \le \varepsilon \|Ax_0 - b\|_H$, где $0 < \varepsilon << 1$.

4. Использование IC2S-разложения и IC1-разложения в предобусловливании ВПС

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 стабилизированным треугольным разложение второго порядка «по значению» IC2S [3]

$$V_s^T A V_s = U_s^T U_s + U_s^T R_s + R_s^T U_s - S_s.$$

Здесь U_s – верхнетреугольные матрицы, R_s – строго верхнетреугольные матрицы, $\|R_s\| = O(\tau)$, $\|S_s\| = O(\tau^2)$ и $0 < \tau << 1$ порог отсечения, матрицы V_s определены в (3.1). Подробное описание предобусловливателя IC2S приведено в следующем разделе. Определим предобусловливатель ВПС-IC2S формулой [6, 7]

$$\tilde{H} = \sum_{s=1}^{p} V_s U_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} U_s^{-T} V_s^T. \qquad (4.1)$$

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

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

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

$$\hat{H} = \sum_{s=1}^{p} V_s \hat{U}_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} \hat{U}_s^{-T} V_s^T. \qquad (4.2)$$

5. Алгоритм построения предобусловливателя IC2S(τ)

Предобусловливание с использованием двух порогов отсечения, являющееся более экономичной версией метода Тисменецкого [33], было описано в [34], а соответствующая матричная формулировка, обоснование, теоретический анализ и выбор параметров были даны в работе [35]. Там же была предложена версия этого метода, отличающаяся предварительной равномерной модификацией всех элементов диагонали аналогично подходу [36], см. также [37]. В методе IC2S(τ)-CG перед построением матрицы предобусловливания необходимо выполнить масштабирование матрицы A (см. выше). Далее считаем, что матрица A отмасштабирована.

Матричная схема метода предобусловливания IC2S(τ) имеет вид

$$A = U^T U + U^T R + R^T U - S,$$

где U – верхнетреугольная матрица, R – строго верхнетреугольная матрица, $S = 2\tau^2 I + E_{JM}(\tau^2)$, матрица $E_{JM}(\tau^2) = (E_{JM}(\tau^2))^T \geq 0$ формируется, как было предложено в [37], $\|R\| = O(\tau)$ и $\|S\| = O(\tau^2)$. В качестве предобусловливателя используется матрица $U^T U$. Алгоритм вычисления стабилизированного треугольного разложения второго порядка имеет следующий вид [3]. Напомним, что после масштабирования $a_{ii} = 1$.

Алгоритм 2.

1. Инициализация вспомогательной диагональной матрицы:
   for $i=1,\ldots,n$
   $d_i := 1 + 2\tau^2$
   end for
   for $i=1,\ldots,n$ (*цикл по строкам A для вычисления строк U, R)*
2. Инициализация вектора v при помощи i-ой строки A:
   for $j=i+1,\ldots,n$
   $v_j := a_{ij}$
   end for
3. Вычисление поправки к вектору v:
   for $s=1,\ldots,i-1$
   for $j=i+1,\ldots,n$
   $v_j := v_j - u_{si}u_{sj} - u_{si}r_{sj} - r_{si}u_{sj}$
   end for
   end for
4. Прореживание вектора v и выполнение поправки к d:
   for $j=i+1,\ldots,n$
   if $|v_j| \leq \tau^2\sqrt{d_i}$ then
   $d_i := d_i + |v_j|$
   $d_j := d_j + |v_j|$

$v_j := 0$

endif

end for

5. Вычисление u_{ii}:

$u_{ii} := \sqrt{d_i}$

6. Расщепление вектора *v* на *i*-ую строку *U* и *i*-ую строку *R*:

for *j=i+1,...,n*

if $|v_j| \geq \tau u_{ii}$ then

$u_{ij} := v_j / u_{ii}$

else

$r_{ij} := v_j / u_{ii}$

endif

end for

7. Выполнение поправки к диагональной матрице d:

for *j=i+1,...,n*

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

end for

end for (*конец цикла по строкам* A *для вычисления строк U , R*)

6. Алгоритм построения предобусловливателя IC1(τ)

Перед построением матрицы предобусловливания IC1(τ) необходимо выполнить масштабирование матрицы A размера $n \times n$, как описано во введении. Матричная схема метода предобусловливания IC1(τ) имеет вид [4]

$$A = \hat{U}^T \hat{U} - E,$$

где $\hat{U}$ – верхнетреугольная матрица. Покомпонентная запись этого уравнения имеет вид

$$a_{ij} + e_{ij} = u_{ii} u_{ij} + \sum_{s=1}^{i-1} u_{si} u_{sj}, \quad j \geq i, \; i=1,\ldots n,$$

где u_{ij} – элементы матрицы $\hat{U}$. Для каждого i выбор элементов e_{ij} матрицы погрешности E осуществляется так, чтобы «убрать» все «малые» значения $v_{ij} = u_{ii} u_{ij} - e_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si} u_{sj}$, $j=i,\ldots,n$. В настоящей работе при написании программы использовался следующий алгоритм [28]. Напомним, что после масштабирования $a_{ii} = 1$.

Алгоритм 3

1. Инициализация вспомогательной диагональной матрицы:

for *i=1,...,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. 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 алгоритмов 2 и 3 в программе следует обеспечить доступ к элементам матриц U, R и матрицы $\hat{U}$ по строкам и по столбцам этих матриц.

7. Алгоритмы параллельной реализации

Пусть матрица A переупорядочена, отмасштабирована и разбита на p блоков. В настоящей работе, не ограничивая общности, в предобусловливателях BIIC-IC2S и BIIC-IC1 будем использовать налегание 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 $m_s \times m_s$, для этого осуществляются необходимые пересылки строк матрицы A.

Параллельная реализация вычисления предобусловливателей BIIC-IC2S, BIIC-IC1 с использованием только MPI не представляет труда. Для вычисления элементов матриц U_s и $\hat{U}_s$ в (4.1) или (4.2) используются алгоритмы 2 или 3, приведенные выше, в которых вместо матрицы A используются отмасштабированные матрицы A_s, пересылок не требуется. Так же, как в работе [7], матрицы U_s^T и $\hat{U}_s^T$ не вычисляются.

Обращение предобусловливателя: $z = \tilde{H} r$, где $\tilde{H}$ определено в (4.1), происходит следующим образом [7]. Перед началом вычислений элементы вектора r_s с индексами $\{j_s(1), ..., j_s(m_s - n_s)\}$ должны быть пересланы в процессор с номером s. Затем в каждом процессоре с номером s вычисляются

$$z_s = U_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} U_s^{-T} r_s .$$

Используется способ обращения матриц U_s^T, предложенный в [7]. После вычисления z_s следует выполнить необходимые пересылки элементов вектора z_s с индексами $\{j_s(1), ..., j_s(m_s - n_s)\}$ в другие процессоры. Аналогично происходит обращение предобусловливателя BIIC-IC1.

Рассмотрим способ параллельной реализации с использованием MPI+ OpenMP технологии этапов построения и обращения предобусловливателей, называемый далее способом 1. Разбиение всей области расчета произведем сразу на pm подобластей, где p – число используемых процессоров, m – число используемых потоков. 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 предобусловливателях BIIC-IC2S и BIIC-IC1 равно pm.

Для построения предобусловливателей BIIC-IC2S, BIIC-IC1 в каждом процессоре с номером s сначала создаются m матриц $A_t = V_t^T A V_t$, где $t=(s-1)m+1, \ \ldots, \ sm$. При этом совершаются необходимые пересылки строк матрицы A. Затем по матрицам A_t в процессоре с номером s ($s=1, \ \ldots, \ p$) строятся матрицы U_t или $\hat{U}_t$ ($t=(s-1)m+1, \ \ldots, \ sm$). При этом для каждого t вычисления элементов матриц U_t или $\hat{U}_t$ происходят в своем потоке с использованием алгоритмов 2 или 3, в которых вместо матрицы A используются отмасштабированные матрицы A_t. Все рекурсивные вычисления при построении матриц U_t или $\hat{U}_t$ происходят внутри потоков. В программе необходимо задать число «внутренних» блоков с налеганием, равное m, и

инициализировать число нитей, совпадающее с числом «внутренних» блоков. В настоящей работе на этапе построения предобусловливателей для циклов по $t1=1, \ldots, m$ использовалась директива **do** с опцией **schedule static,** внутри циклов по $t1$ осуществлялось построение матриц U_t или $\hat{U}_t$.

При построении матриц A_t ($t=(s-1)m+1, \ldots, sm$) в каждом процессоре с номером s ($s=1, \ldots, p$) тоже используются OpenMPI технологии с числом нитей m. В настоящей работе на этапе построения матриц A_t для внешних циклов по $t1=1, \ldots, m$ использовалась директива **do** с опцией **schedule static**.

Обращение предобусловливателя BIIC-IC2S с использованием формулы

$$z_t = U_t^{-1} \begin{bmatrix} 0\,0 \\ 0\; I_{n_t} \end{bmatrix} U_t^{-T} r_t, \quad t=(s-1)m+1, \ldots, sm, \quad s=1, \ldots, p \qquad (7.1)$$

происходит следующим образом. Перед началом вычислений элементы векторов r_t для $t=(s-1)m+1, \ldots, sm$, необходимые для расчетов и хранящиеся в памяти других процессоров, должны быть пересланы в процессор с номером s. Для каждого $t=(s-1)m+1, \ldots, sm$ вычисления по формуле (7.1) происходят в своем потоке. В настоящей работе при вычислении z_t для выполнения циклов по $t1=1, \ldots, m$ использовалась директива **do** с опцией **schedule static**. Используется способ обращения матриц U_t^T, предложенный в [7]. После вычисления z_t для всех $t=(s-1)m+1, \ldots, sm$, $s=1, \ldots, p$ следует выполнить необходимые пересылки. Аналогично происходит обращение предобусловливателя BIIC-IC1 с использованием MPI+OpenMP технологии.

Заметим, что в случае $p=1$ выше описан способ параллельной реализации с применением OpenMP технологии этапов построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1 для m блоков с налеганием.

Рассмотрим способ 2 применения MPI+OpenMPI технологии обращения и построения предобусловливателя BIIC-IC1, аналогичный способу 1 из работ [27, 28]. Как показали расчеты, использование способа 2 применения MPI+OpenMPI технологии неэффективно при решении СЛАУ (1.1) методом BIIC-IC2S-CG. В способе 2 используется p блоков с налеганием в предобусловливателе BIIC, где p – число процессоров ($p \neq 1$), а внутри каждого блока используется упорядочение узлов сетки нерасширенных подобластей типа DDO [15].

Каждая нерасширенная подобласть с номером s ($s=1, \ldots, p$) разбивается на m «внутренних» подобластей, где m – число используемых нитей (потоков) при применении OpenMP технологии, с приблизительно равным числом узлов сетки. Разбиение в настоящей работе производилось в порядке следования узлов нерасширенных подобластей следующим образом [28]. 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. Можно использовать другие способы разбиения.

На первом этапе переупорядочения узлов нерасширенной подобласти будем использовать упорядочение типа DDO, предложенное в работе [15], а затем использованное для применения OpenMP технологии в работах [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. Установим следующий порядок следования узлов сетки в нерасширенной 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, однако, как показали расчеты, это не приводит к существенному изменению времени счета задачи.

Определим $\bar{l}_s$ – минимальное значение количества «внутренних» узлов при разбиении нерасширенной подобласти с номером s на «внутренние» подобласти. Предполагается, что $\bar{l}_s \neq 0$. Обозначим $M1 = m\bar{l}_s$. На втором этапе переупорядочения узлов сетки нерасширенной 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 следуют 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.

Установим следующий порядок следования узлов сетки расширенной подобласти с номером s (s=1,…,p). Сначала идут первые m_s-n_s узлов сетки расширенной подобласти при упорядочении, которое было установлено ранее.

Затем следуют узлы сетки нерасширенной подобласти, соответствующей расчетам на процессоре с номером *s*, при введенном ранее переупорядочении, произведенном в два этапа (см. выше).

При использовании способа 2 применения MPI+OpenMP технологии для построения предобусловливателя BIIC-IC1 производится вычисление верхнетреугольного множителя матрицы предобусловливания IC1 для матрицы, полученной после переупорядочения матрицы $A_s = V_s^T A V_s$. Для первых $m_s - n_s$ строк это происходит без применения OpenMP технологии. Затем с применением OpenMP технологии с использованием m нитей производится вычисление следующих $M1 = m\bar{l}_s$ строк. При этом происходит отсечение по позициям элементом $\bar{u}_{ij}$ верхнетреугольного множителя предобусловливателя, если узлы с номерами i и j принадлежат разным «внутренним» подобластям. Такая ситуация может возникнуть из-за предшедствующих вычислений первых $m_s - n_s$ строк. Затем с применением OpenMP технологии с использованием m нитей производится вычисление следующих $M1 = m\bar{l}_s$ строк. Используется цикл по $k2$=1, …, m, внутри которого осуществляется вычисление элементов строк искомой матрицы с номерами $i = m_s - n_s + (k2-1)\bar{l}_s + 1, ..., m_s - n_s + k2\bar{l}_s$. При этом все рекурсивные вычисления происходят внутри потоков. Для выполнения цикла по $k2$ в настоящей работе использовалась директива **do** с опцией **schedule static**. Затем без использования OpenMPI технологии производится вычисление элементов оставшихся строк верхнетреугольного множителя матрицы предобусловливания IC1 при новом окончательном переупорядочении. Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства строк матрицы верхнетреугольного множителя предобусловливателя вычисления происходят с использованием OpenMP технологии. При вычислении элементов строк верхнетреугольного множителя матрицы предобусловливания IC1 используется алгоритм 3. Затем с помощью транспонирования производится определение нижнетреугольного множителя матрицы предобусловливания IC1. На этом этапе OpenMP технологии не применяются.

Первый этап обращения матрицы предобусловливания (обращение нижнетреугольного множителя матрицы предобусловливания IC1) с использованием OpenMP технологии происходит аналогично вычислению верхнетреугольного множителя матрицы предобусловливания IC1. С использованием OpenMP технологии вычисляются M1 элементов искомого вектора с номерами $m_s - n_s + 1, ..., m_s - n_s + M1$ (при новом окончательном упорядочении). На заключительном этапе обращения матрицы предобусловливания (обращении нижнетреугольной матрицы) вычисления происходят в обратном порядке, элементы искомого вектора с номерами $m_s - n_s + M1, ..., m_s - n_s + 1$ вычисляются с использованием OpenMP технологии.

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. После вычисления z_s следует выполнить необходимые пересылки.

Вычисление элементов вектора $q_k = Ap_k$, где k – номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, происходит следующим образом. Матрица A хранится в памяти в распределенном CRS-формате, содержит верхний и нижний треугольники. Сначала производится пересылка значений компонент вектора p_k, необходимых для вычисления части вектора q_k в рассматриваемом процессоре с номером s (s=1,…,p), хранящихся в памяти других процессоров. В случае применения OpenMP технологии для вычисления $(q_k)_i$ (компонент вектора q_k) в каждом процессоре для своей группы индексов i для выполнения цикла по i используется директива **do** с опцией **schedule static.** Заметим, что, как показали расчеты задач методом CG с предобусловливанием Якоби в работах [24, 25], использование параметров **dynamic**; **guided**; **guided, 6** в этой опции, как правило, не позволяет ускорить вычисления по сравнению с использованием параметра **static.** MPI реализация вычислений векторных операций и скалярных произведений в алгоритме 1 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.**

8. Результаты расчетов

Программы, реализующие применение методов BIIC- IC2S-CG, BIIC- IC1-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии, 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 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 [31]. 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;
cfd2 – вычислительная гидродинамика (уравнение давления).

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 [38], Id – количество строк без

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
cfd2	123440	3085406	123440	936464	8	30	0.15+7

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.

Модельную задачу далее будем называть задачей 1. Задачу с матрицей **parabolic_fem** будем называть задачей 2, задачи с матрицами **apache2, ecology2** – соответственно задачей 3, задачей 4. Задачи с матрицами **boneS01, cfd2** будем называть задачами 5, 6.

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+OpenMPI технологии) использовался алгоритм [29]. В качестве параметра отсечения при решении всех задач методом CG с предобусловливанием BIIC-IC2S использовалось значение $\tau = 0.01$. При решении задач 1 – 4 методом CG с предобусловливанием BIIC-IC1 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 этой задачи. При решении задачи 6 методом BIIC-IC1-CG для безотказности метода BIIC-IC1-CG необходимо использовать чрезмерно малое значение τ , поэтому решение этой задачи методом BIIC-IC1-CG не проводилось.

В таблицах 2 – 7 приведены числа итераций и времена счета методом BIIC-IC2S-CG тестовых задач 1 – 6 при использовании для параллельной реализации MPI и MPI+OpenMP технологии способом 1. В таблицах 8 – 11 приводятся результаты расчетов методом BIIC-IC1-CG тестовых задач 1 – 4 с применением MPI+OpenMP технологии способами 1, 2, а в таблице 12 – задачи 5 с применением MPI+OpenMP технологии способом 1. Использование способа 2 применения OpenMP технологии при решении задачи 5 (с более плотной матрицей и маленьким значением τ) методом BIIC-IC1-CG не позволило

существенно уменьшить время счета по сравнению со счетом с использованием только MPI из-за большого времени вычисления предобусловливателя.

При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 12, 16 нитей. В таблицах 2–12 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей (Th). Приведены также коэффициенты ускорения счета благодаря использованию OpenMP технологии на том же числе процессоров (обозначены μ) и коэффициенты ускорения счета по сравнению со счетом на 4 процессорах без использования OpenMP технологии (обозначены η).

Под временем вычислений в таблицах 2–12 подразумевается время счета итерационного процесса в сумме с временем вычисления предобусловливателя.

Таблица 2. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **5_1048576** на p процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	It =342, 17.4	It =343,9.93 1.75	It =375, 5.96 2.91	It =374,4.5 3.86	It =386,3.52 4.94
MPI+OpenMP μ	Th =12 It=421,5.73 3.04	Th =6 It =421,4.62 2.15	Th =3 It =421,3.8 1.57	Th =3 It =414,4.63 0.97	Th =3 It =442,3.9 0.9

Таблица 3. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **parabolic_fem** на p процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	It =296, 8.79	It =323,5.22 1.68	It =346, 2.94 2.98	It =359,3.24 2.71	It=357, 1.59 5.52
MPI+OpenMP μ	Th =12 It =368, 3.22 2.73	Th =6 It =368,2.63 1.98	Th =3 It =368, 2.03 1.45	Th =3 It =399,3.06 1.05	Th =3 It=424,2.33 0.68

Таблица 4. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **apache2** на p процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	It =403,15.85	It =418,10.5 1.51	It =451, 5.17 3.07	It=461, 4.61 3.44	It =481,2.99 5.3
MPI+OpenMP μ	Th =12 It =509,5.63 2.81	Th =6 It =509,4.71 2.23	Th =3 It =509,3.86 1.34	Th =3 It =560,5.28 0.87	Th =3 It=581,3.99 0.75

Таблица 5. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **ecology2** на *p* процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	*It* =560,27.55	*It* =608,14.8 1.86	*It* =623, 8.32 3.31	*It* =634,7.19 3.83	*It*=677, 5.48 5.03
MPI+OpenMP μ	*Th* =16 *It* =544,7.49 3.68	*Th* =6 *It* =698,6.71 2.21	*Th* =3 *It* =698,5.45 1.53	*Th* =3 *It* =714,7.72 0.93	*Th* =3 *It* =737,5.48 1.0

Таблица 6. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **boneS01** на *p* процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	*It*=301, 12.28	*It* =324,6.89 1.78	*It* =332, 3.91 3.14	*It* =350,3.27 3.75	*It* =357,2.19 5.6
MPI+OpenMP μ	*Th* =12 *It* =366, 3.62 3.39	*Th* =6 *It* =366,3.26 2.11	*Th* =3 *It* =366,2.69 1.45	*Th* =3 *It* =380,3.21 1.02	*Th* =3 *It* =390,2.5 0.88

Таблица 7. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **cfd2** на *p* процессорах без использования и с использованием OpenMP технологии

P	2	4	8	10	16
MPI, η	*It* =296, 8.61	*It*=333, 5.14 1.67	*It* =386,3.20 2.69	*It* =393,2.61 3.29	*It* =428,2.09 4.12
MPI+OpenMP μ	*Th* =12 *It*=436, 3.15 2.73	*Th* =6 *It*=436,2.76 1.86	*Th* =3 *It* =436,2.36 1.35	*Th* =3 *It*=462,2.61 1.0	*Th* =3 *It*=515,2.23 0.94

В случае использования способа 2 применения MPI+OpenMP технологии время вычисления предобусловливателя включает также время построения

Таблица 8. Числа итераций и времена счета методом BIIC-IC1-CG задачи с матрицей **5_1048576** на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ 1 MPI, η	*It* =401,19.55	*It*=401,11.02 1.77	*It* =435,6.67 2.93	*It* =427,5.04 3.87	*It* =444,3.86 5.06
MPI+OpenMP μ	*Th*=12 *It* =469,6.13 3.19	*Th*=4 *It*=444, 6.33 1.74	*Th*=3 *It* =469,4.05 1.65	*Th*=3 *It* =493,4.17 1.2	*Th*=3 *It* =493,4.16 0.93
способ 2 MPI, η	*It* =401,18.94	*It*=401,10.76 1.76	*It* =435,6.4 2.96	*It* =427,4.8 3.94	*It* =444,3.77 5.02
MPI+OpenMP μ	*Th*=12 *It* =426,5.3 3.57	*Th*=6 *It*=408 3.56 3.02	*Th*=3 *It* =450,3.22 1.98	*Th*=3 *It* =459,3.7 1.29	*Th*=3 *It* =479,3.36 1.12

Таблица 9. Числа итераций и времена счета методом BIIC-IC1-CG задачи с матрицей **parabolic_fem** на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ 1 MPI, η	*It*=415, 10.98	*It* =418,5.79 1.89	*It* =440, 3.27 3.35	*It* =463,3.09 3.55	*It*=353, 1.80 6.1
MPI+OpenMP μ	*Th*=12 *It* =454, 3.19 3.44	*Th*=4 *It* =453,3.14 1.84	*Th*=3 *It* =454, 2.08 1.57	*Th*=4 *It* =497,2.66 1.16	*Th*=3 *It* =509,2.28 0.79
способ 2 MPI, η	*It*=415, 10.34	*It* =418,5.7 1.81	*It* =440, 3.25 3.18	*It* =463,3.00 3.44	*It*=453, 1.78 5.8
MPI+OpenMP μ	*Th*=12 *It* =424, 2.86 3.63	*Th*=6 *It*=434,1.96 2.9	*Th*=3 *It* =456, 1.78 1.82	*Th*=3 *It* =477,2.3 1.3	*Th*=3 *It* =481,1.67 1.06

Таблица 10. Числа итераций и времена счета методом BIIC-IC1-CG задачи с матрицей **apache2** на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ 1 MPI, η	*It* =489,16.69	*It*=501,11.21 1.48	*It* =529, 5.36 3.11	*It* =540,4.60 3.62	*It* =542,2.97 5.62
MPI+OpenMP μ	*Th*=8 *It*=542,6.78 2.46	*Th*=8 *It*=611,6.89 1.63	*Th*=3 *It*=577, 4.3 1.25	*Th*=3 *It*=618,5.29 0.86	*Th*=3 *It*=645,4.5 0.66
способ 2 MPI, η	*It* =489,16.48	*It*=501,10.87 1.52	*It* =529, 5.20 3.17	*It* =540,4.38 3.76	*It* =542,2.99 5.51
MPI+OpenMP μ	*Th*=6 It=507,7.26 2.26	*Th*=6 It=529,4.06 2.67	*Th*=3 It=593,3.14 1.65	*Th*=3 It=618,3.83 1.14	*Th*=3 It=609,2.84 1.05

Таблица 11. Числа итераций и времена счета методом BIIC-IC1-CG задачи с матрицей **ecology2** на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ 1 MPI, η	*It* =697,31.84	*It*=721,17.34 1.83	*It* =740, 9.87 3.22	*It*=751,8.25 3.86	*It*=790, 6.27 5.08
MPI+OpenMP μ	*Th*=12 *It* =809,10.19 3.12	*Th*=4 *It* =790,9.98 1.74	*Th*=3 *It*=809,6.68 1.48	*Th*=4 *It* =846,8.2 1.01	*Th*=3 *It* =854,7.07 0.88
способ 2 MPI, η	*It* =697,30.87	*It*=721,16.64 1.85	*It* =740, 9.32 3.31	*It*=751,7.96 3.88	*It*=790, 6.03 5.12
MPI+OpenMP μ	*Th*=12 *It* =728,7.21 4.28	*Th*=6 *It* =758,5.15 3.23	*Th*=6 *It* =764,4.58 2.03	*Th*=3 *It* =783,5.8 1.37	*Th*=3 *It* =832,4.98 1.21

переупорядочения. Так как при применении способа 2 используются как верхнетреугольный, так и нижнетреугольный множитель предобусловливателя IC1, то в этом случае сравнение времени счета с применением MPI+OpenMP технологии производилось с временем счета с применением только MPI с использованием обоих множителей матрицы предобусловливания IC1.

Таблица 12. Числа итераций и времена счета методом BIIC-IC1-CG задачи с матрицей **boneS01**на *p* процессорах без применения и с применением OpenMP технологии

P	2	4	8	10	16
способ 1 MPI, η MPI+OpenMP μ	*It*=301, 10.09 *Th* =12 It=365, 3.4 2.96	*It* =348,6.19 1.63 *Th* =12 It=402,3.2 1.93	*It* =335, 3.28 3.08 *Th* =3 It=365, 2.44 1.34	*It* =376,3.02 3.34 *Th* =3 It=378,2.99 1.01	*It* =380,2.04 4.95 *Th* =3 It=404,2.32 0.88

Еще раз подчеркнем, что использование способа 2 применения MPI+OpenMP технологии, связанного с переупорядочением узлов внутри подобластей, в случае метода BIIC-IC2S-CG не оправдывает себя. Его применение для решения задачи 1 позволяет лишь немного ускорить вычисления только в случае двух процессоров (см. таблицу 13). Это связано с большим временем вычисления матрицы предобусловливания. Следует ожидать, что использование способа 2 применения MPI+OpenMP технологии при решении остальных тестовых задач методом BIIC-IC2S-CG тоже окажется нецелесообразным.

Таблица 13. Числа итераций и времена счета методом BIIC-IC2S-CG задачи с матрицей **5_1048576** на *p* процессорах без использования и с использованием OpenMP технологии способом 2

P	2	4	8	10
способ 2 MPI, η MPI+OpenMP μ	*It* =342,16.86 *Th*=12 *It* =353,14.84 1.14	*It* =343,9.6 1.76 *Th*=12 *It*=394, 9.96 0.96	*It* =375,5.71 2.95 *Th*=12 *It* =405,7.13 0.8	*It* =374,4.39 3.84 *Th*=10 *It* =404,6.08 0.72

На рисунках 1–6 приведены графики зависимости времени счета тестовых задач от числа процессоров в логарифмическом масштабе с использованием только MPI (линии красного цвета) и с использованием MPI+OpenMP технологии (линии синего цвета) для метода BIIC-IC2S-CG.

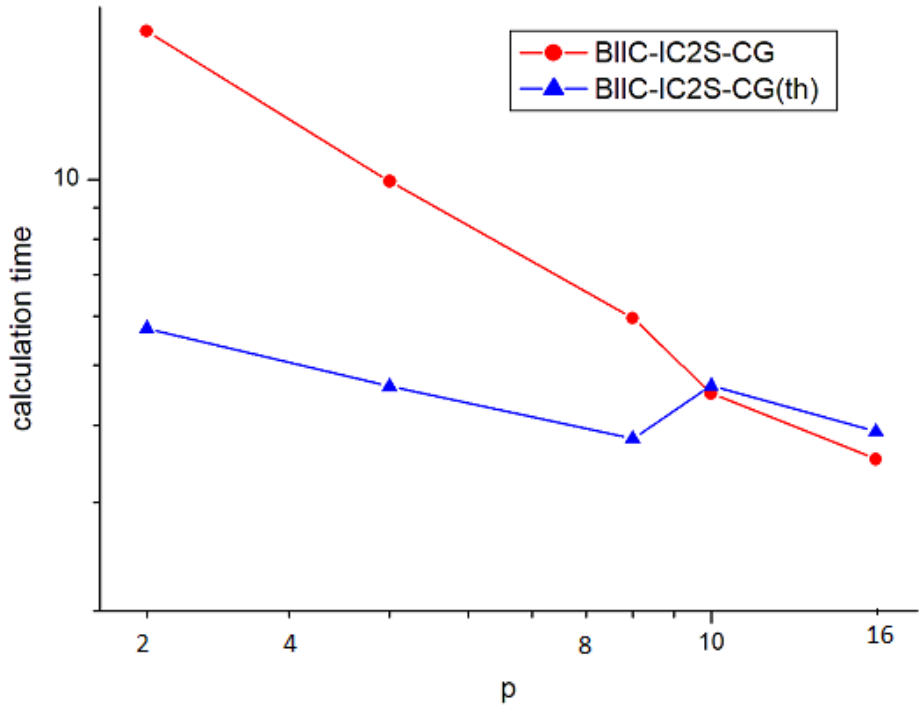

Рис. 1. Времена счета задачи с матрицей **5_1048576** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

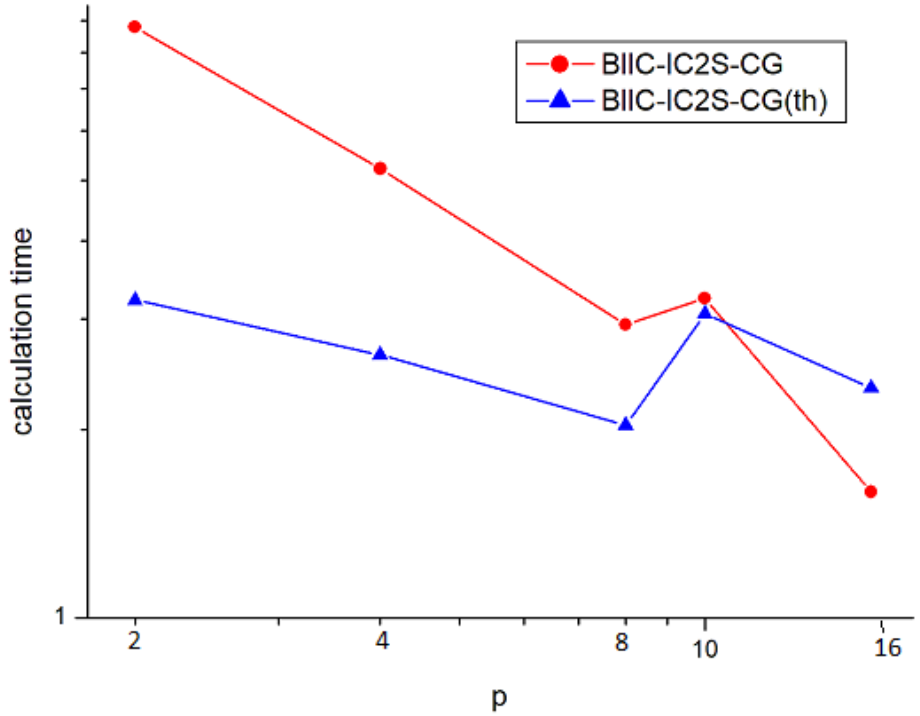

Рис. 2. Времена счета задачи с матрицей **parabolic_fem** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

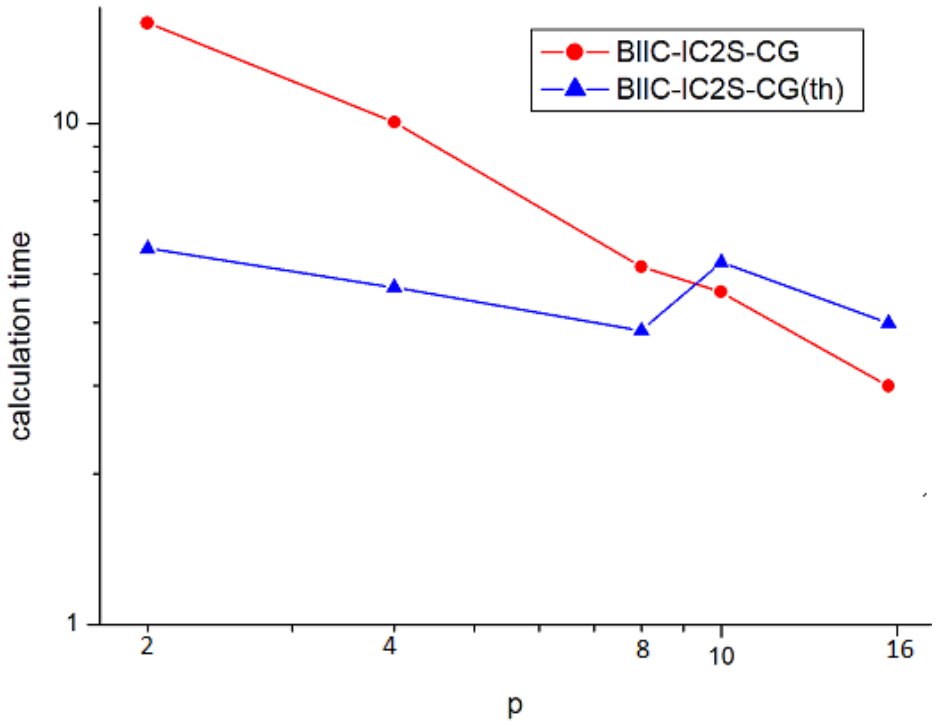

Рис. 3. Времена счета задачи с матрицей **apache2** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

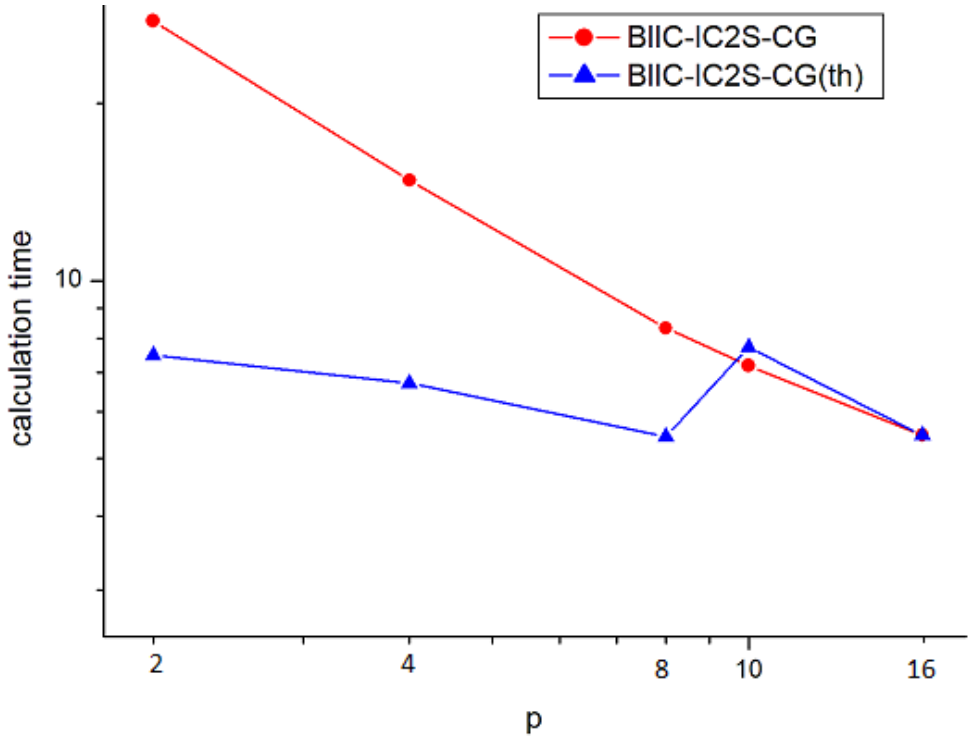

Рис. 4. Времена счета задачи с матрицей **ecology2** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

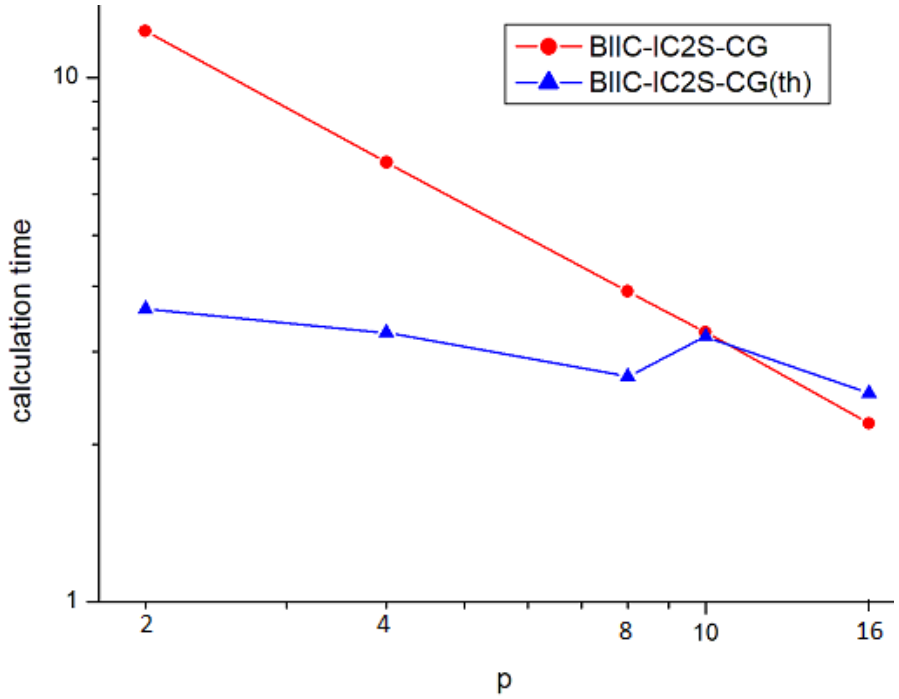

Рис. 5. Времена счета задачи с матрицей **boneS01** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

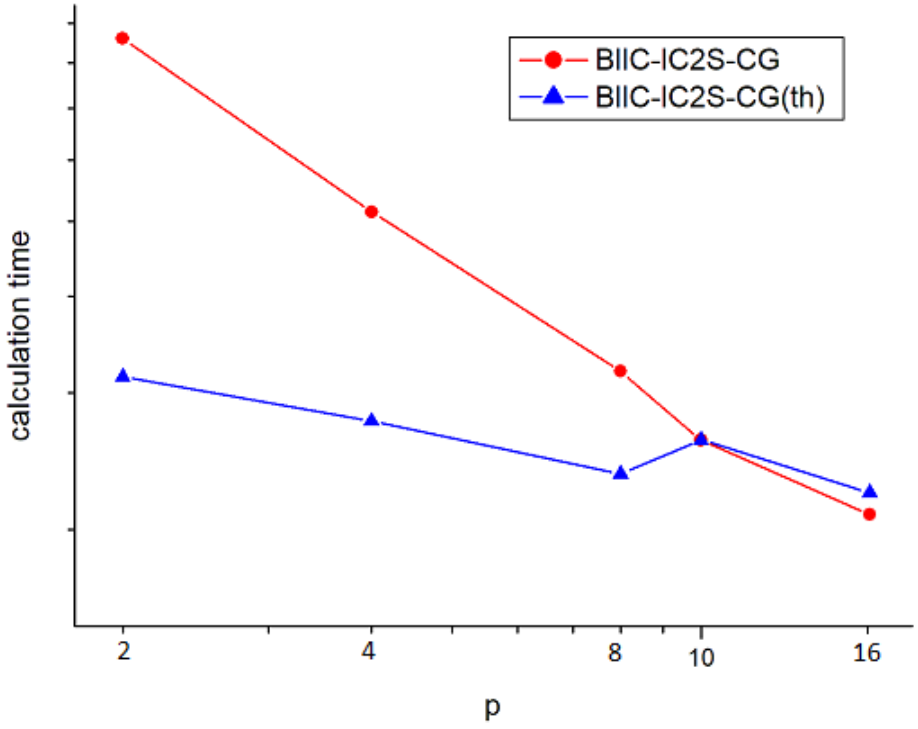

Рис. 6. Времена счета задачи с матрицей **cfd2** методом BIIC-IC2S-CG с использованием MPI и MPI+OpenMP технологии

На рисунках 7-11 приведены графики зависимости времени счета тестовых задач от числа процессоров в логарифмическом масштабе с использованием

только MPI (линии красного цвета) и с использованием MPI+OpenMP технологии (линии синего цвета) для метода BIIC-IC1S-CG.

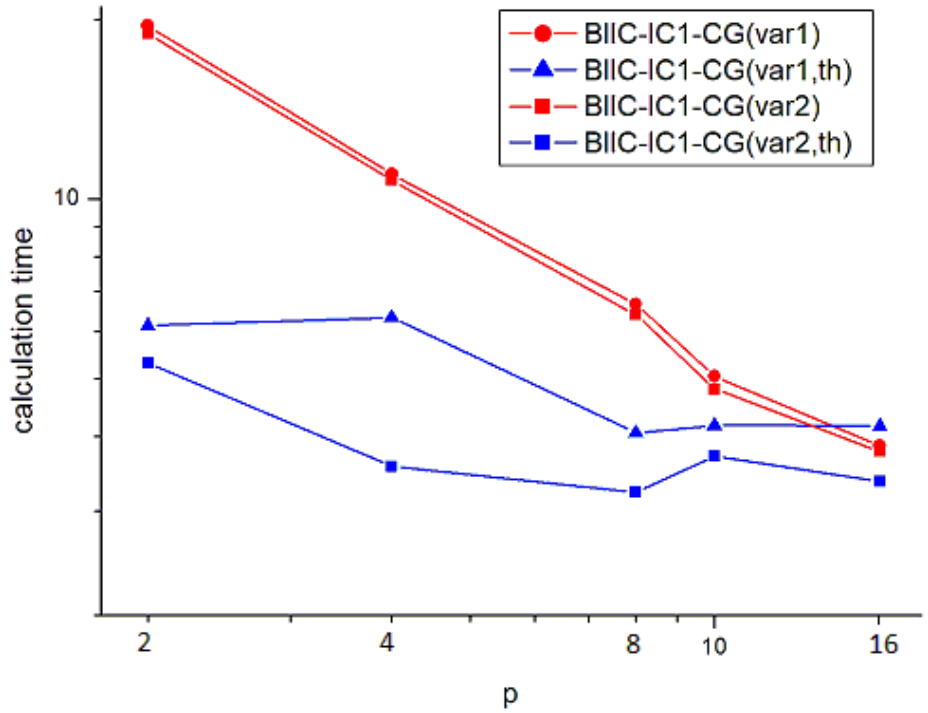

Рис. 7. Времена счета задачи с матрицей **5_1048576** методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

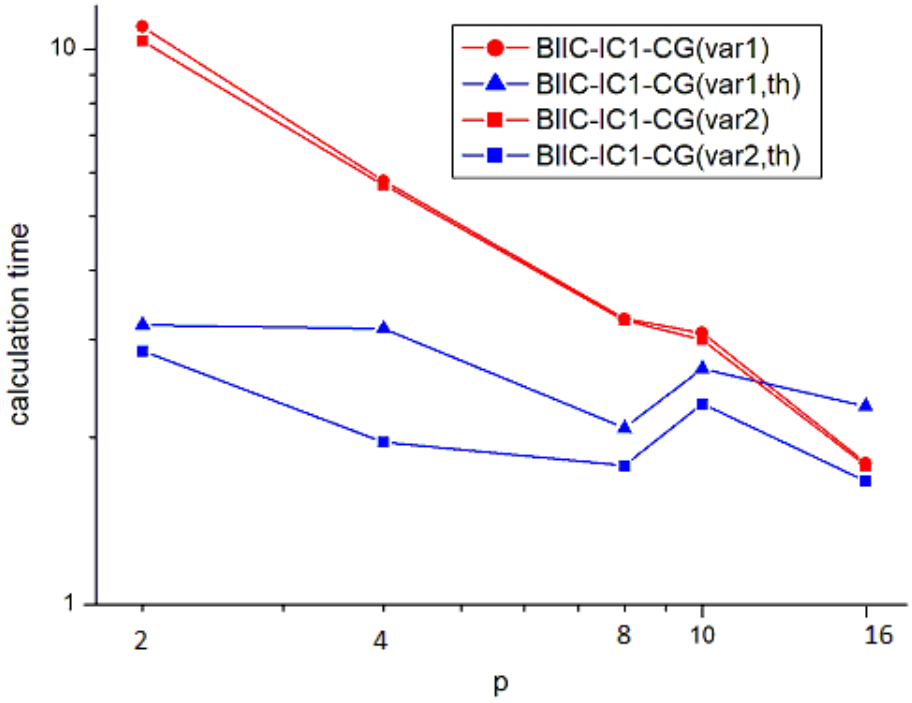

Рис. 8. Времена счета задачи с матрицей **parabolic_fem** методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

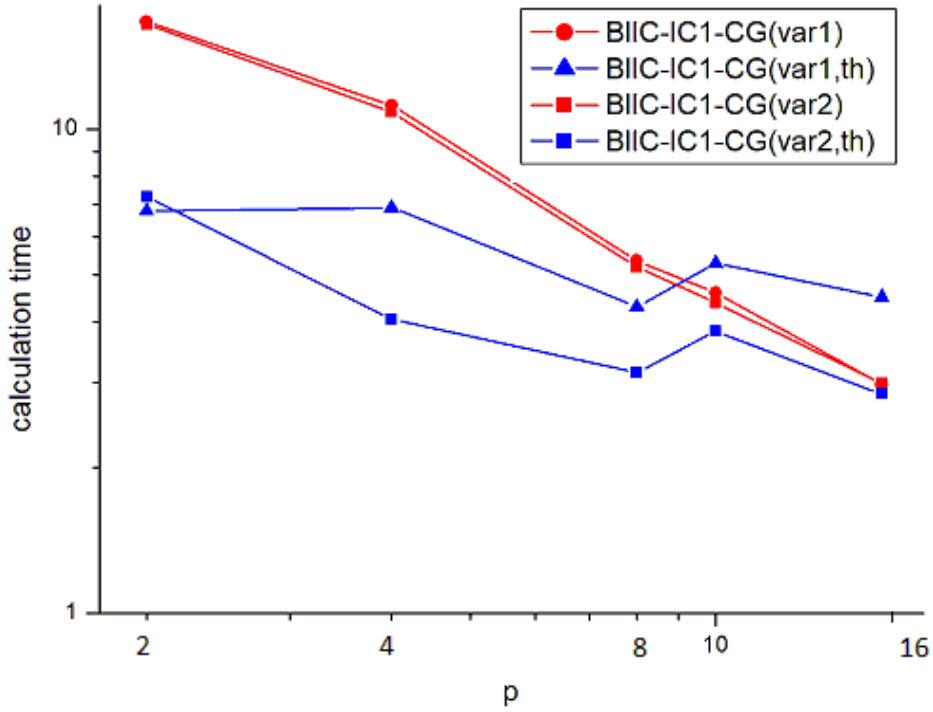

Рис. 9. Времена счета задачи с матрицей **apache2** методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

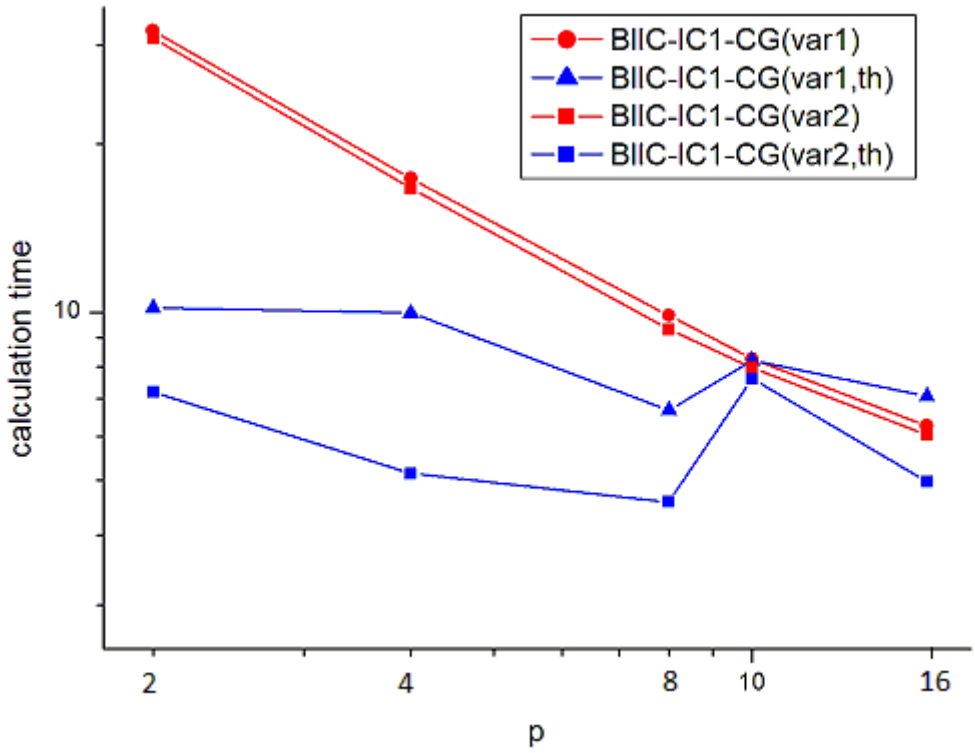

Рис. 10. Времена счета задачи с матрицей **ecology2** методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии способами 1 и 2

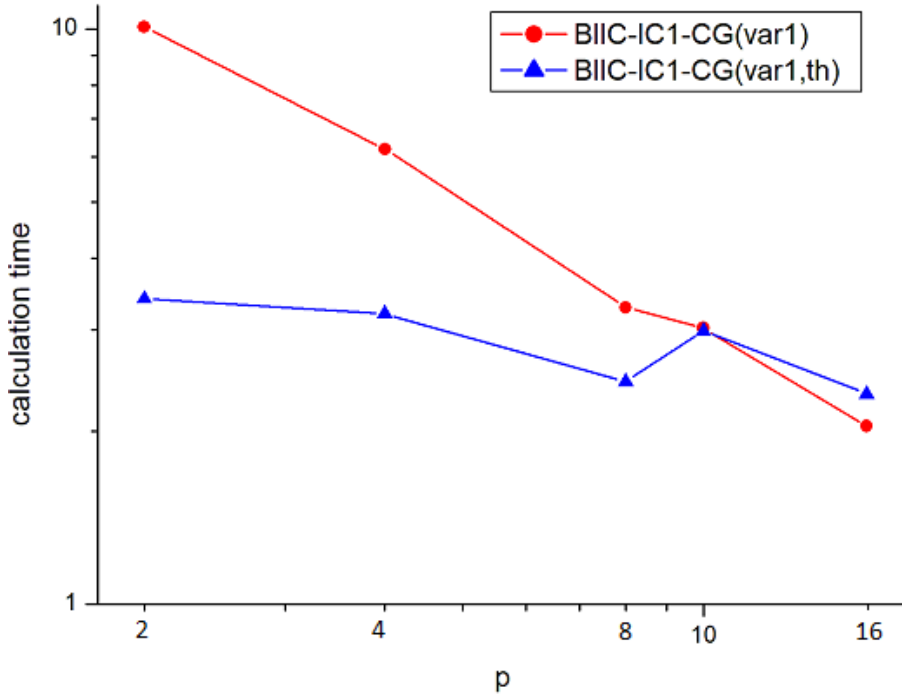

Рис. 11. Времена счета задачи с матрицей **boneS01** методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии способом 1

Как видно из таблиц 2 – 7 и рис. 1 – 6, применение MPI+OpenMP технологии с использованием способа 1 позволяет значительно быстрее, чем при применении только MPI, производить решение всех тестовых задач методом BIIC-IC2S-CG при p<10. При p=10, 16 применение OpenMP технологии для решения тестовых задач методом BIIC-IC2S-CG становится нецелесообразным.

Как видно из таблиц 8 – 12 и рис. 7 – 11, применение MPI+OpenMP технологии с использованием способов 1 и 2 для решения задач 1 – 4, а также применение способа 1 для решения задачи 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 в большинстве случаев становится нецелесообразным.

Как видно из таблиц 8 – 11 и рис. 7 – 10, использование способа 2 применения OpenMP технологии для решения тестовых задач 1 – 4 часто приводит даже к меньшему времени вычислений, чем использование способа 1. Заметим, что при использовании способов 1 и 2 по-разному происходит изменение числа итераций с применением OpenMP технологии.

Заметим, что алгоритм построения матрицы предобусловливания BIIC-IC1 значительно проще, чем алгоритм построения матрицы предобусловливания BIIC-IC2S, время вычисления предобусловливателя BIIC-IC1 при использовании способа 1 меньше, чем время вычисления предобусловливателя BIIC-IC2S, особенно для случая не очень сильно разреженных матриц. Однако скорость сходимости итерационного процесса

метода BIIC-IC2S-CG должна быть выше, чем в методе BIIC-IC1-CG. Кроме того, метод BIIC-IC2S-CG является безотказным.

На рисунке 12 приведены времена счета модельной задачи с матрицей **5_1048576** для различного числа процессоров при использовании для ее решения методов BIIC-IC2S-CG и BIIC-IC1-CG с применением MPI и MPI+OpenMP технологий. Как видно из рисунка 12, решение модельной задачи с матрицей **5_1048576** методом BIIC-IC2S-CG происходит быстрее, чем методом BIIC-IC1-CG, в случае применения только MPI для всех значений p; а в случае применения MPI+OpenMP способом 1 – для p <10, В случае применения MPI+OpenMP способом 2 при всех значениях *p* решение этой задачи методом BIIC-IC1-CG происходит быстрее, чем методом BIIC-IC2S-CG.

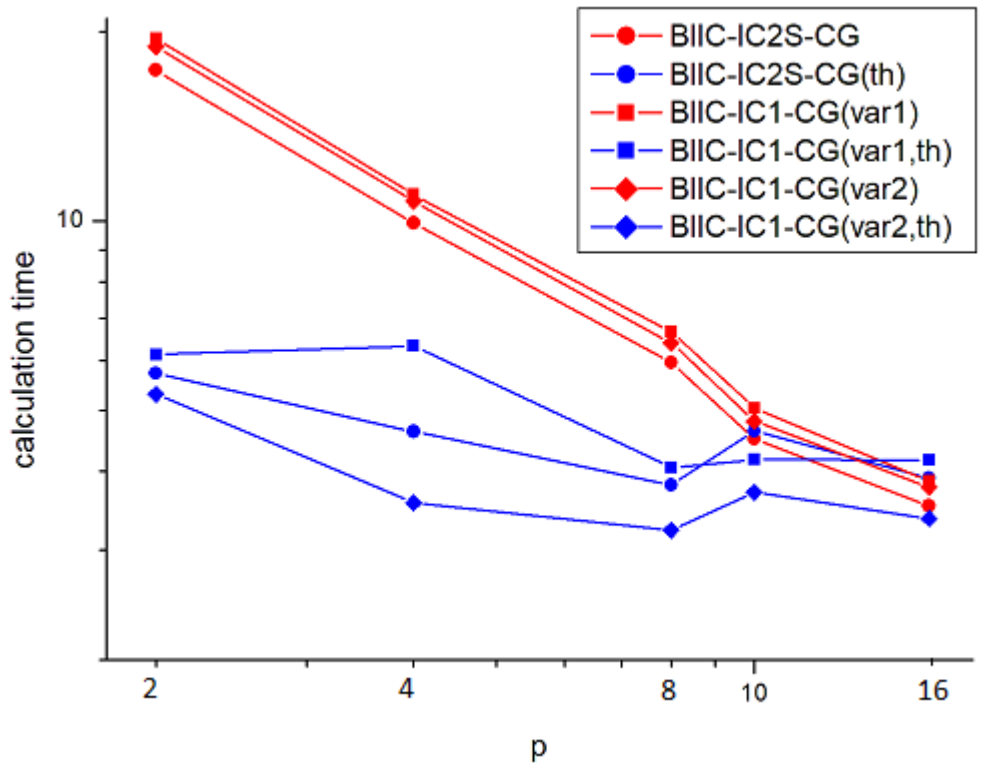

Рис. 12. Времена счета задачи с матрицей **5_1048576** методами BIIC-IC2S-CG BIIC-IC1-CG с использованием MPI и MPI+OpenMP технологии

Как видно из таблиц 3 – 5 и 9 – 11, решение задач 2 – 4 методом BIIC-IC2S-CG с применением MPI и MPI+OpenMP в подавляющем большинстве случаев происходит быстрее, чем методом BIIC-IC1-CG с применением MPI и MPI+OpenMP способом 1. Для задач 1 – 4 число итераций метода BIIC-IC2S-CG меньше числа итераций метода BIIC-IC1-CG как без использования, так и при использовании OpenMP технологии способом 1. При использовании способа 2 применения OpenMP технологии решение задач 1 – 4 методом BIIC-IC1-CG в подавляющем большинстве случаев происходит быстрее, чем методом BIIC-IC2S-CG.

Как видно из таблиц 6 и 12, решение задачи 5 с более заполненной матрицей методом BIIC-IC2S-CG с применением MPI и MPI+OpenMP происходит немного медленнее, чем методом BIIC-IC1-CG. Для задачи 5 число итераций метода BIIC-IC2S-CG всегда меньше числа итераций метода BIIC-IC1-CG в случае использования только MPI.

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.

Для изучения влияния размера матрицы на эффективность использования OpenMP технологии было проведено решение модельных задач 2 – 4 – разностных задач Дирихле для уравнения Пуассона в единичной квадратной области на равномерной ортогональной сетке с возрастаюшим числом узлов. В матрице с именем **5_16384** n=16384**,** в матрице **5_65536** – n= 65536, в матрице **5_262144** – n= 262144. Решение этих задач производилось методом BIIC-IC2S-CG без использования налегания (q=0). В этом случае мы имеем дело с предобусловливанием блочного Якоби в сочетании с IC2S (BJIC2). Результаты расчетов приведены в таблице 14.

Как видно из таблицы 14, с увеличением размера матрицы для численного решения задачи Дирихле для уравнения Пуассона в единичном квадрате эффективность использования OpenMP технологии для фиксированного числа процессоров возрастает.

Таблица 14. Числа итераций и времена счета методом BJIC2S-CG модельных задач 2-4 на p процессорах без использования и с использованием OpenMP технологии

P	4	8	16	32
5_16384 MPI, η MPI+OpenMP μ	*It* =79, 0.11 *Th*=6 *It* =164, 0.11 1.0	*It* =103, 0.083 1.32 *Th*=3 *It*=161, 0.097 0.85	*It* =117, 0.045 2.44 *Th*=3 *It*=178, 0.105 0.42	*It* =135, 0.029 3.79 *Th*=3 *It* =214, 0.070 0.41
5_65536 MPI, η MPI+OpenMP μ	*It* =131, 0.32 *Th*=6 *It* =249, 0.23 1.39	*It* =154, 0.239 1.33 *Th*=3 *It* =230, 0.197 1.21	*It* =183, 0.136 2.35 *Th*=3 *It* =277, 0.22 0.61	*It* =201, 0.069 4.64 *Th*=3 *It*=325, 0.144 0.47
5_262144 MPI, η MPI+OpenMP μ	*It* =211, 1.47 *Th*=6 *It* =391, 0.78 1.88	*It* =258, 0.95 1.54 *Th*=3 It=364, 0.67 1.41	*It* =286, 0.54 2.72 *Th*=3 It=436, 0.73 0.73	*It* = 321, 0.33 4.45 *Th*=3 *It* =477, 0.54 0.61

В настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, значительно больше. Можно ожидать, что потеря эффективности применения OpenMP технологии при решении задач наступит при значительно большем числе процессоров.

9. Заключение

В работе предложен новый предобусловливатель BIIC-IC1(τ) для решения СЛАУ (1.1) методом сопряженных градиентов. Предложен способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1 с числом блоков, кратным числу процессоров и числу используемых потоков. Предложен способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1 с числом блоков, кратным числу процессоров, с использованием переупорядочения узлов сетки внутри подобластей типа DDO и отсечения по позициям некоторых элементов матрицы предобусловливания. 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) методами BIIC-IC2S-CG BIIC-IC1-CG для не слишком большого числа узлов суперкомпьютерной системы. Использование способа 2 применения MPI+OpenMP технологии при решении задач с достаточно разреженными матрицами методом BIIC-IC1-CG позволяет существенно ускорить вычисления по сравнению с применением только MPI для не слишком большого числа узлов суперкомпьютерной системы.

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. 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.
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. 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.
8. Duff I.S., Meurant G.A. The effect of ordering on preconditioned conjugate gradients // BIT. 1989. V. 29. P. 625-657.
9. Doi S. On parallelism and convergence of incomplete LU factorizations //Applied Numerical Mathematics: Transactions of IMACS. 1991. V. 7. № 5. P.417–436.
10. Notay Y. An efficient parallel discrete PDE solver // Parallel Computing. 1995. V.21. P.1725-1748.
11. Milyukova O. Yu. Parallel approximate factorization method for solving discreate elliptic equations // Parallel Computing. 2001. №27. P.1365-1379.
12. 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.
13. Hysom D., Pothen A. A scalable parallel algorithm for incomplete factor preconditioning // SIAM J. Sci. Comput. 2001. V. 22. P. 2194-2215.
14. 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.
15. 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.
16. Anderson E. C., Saad Y. Solving sparse triangular systems on parallel computers // International J. of High Speed Computing. 1989. V.1. P. 73–96.
17. Hammond S. W., Schreiber R. Efficient ICCG on a shared memory multiproces530sor, International J. High Speed Computing 4. 1992. P. 1–21.
18. Wolf M. M., Heroux M. A., Boman E. G. Factors impacting performance of 535 multithreaded sparse triangular solve // in: Proceedings of the 9th International Conference on High Performance Computing for Computational Science. VECPAR'10. Springer-Verlag. Berlin. Heidelberg. 2011. P. 32-44.
19. 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. N. 119. P. 219-230.
20. Chow E., A. Patel A. Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. V. 37. P. 169-193.

21. 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.
22. Heuveline V., Lukarski D., Weiss J.-P. Enhanced Parallel ILU(p)-Based Preconditioners for Multi-Core CpuS and GpuS - the Power(Q)-Pattern Method // Tech. Report EMCL-2011-08. Karlsruhe Institute of Technology. Karlsruhe. Germany. 2011.
23. Li R., Saad Y. GPU-Accelerated Preconditioned Iterative Linear Solvers // Tech. Report UMSI-2010-112. University of Minnesota Supercomputing Institute. 2010. Minneapolis. MN.
24. Капорин И.Е., Милюкова О.Ю. MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями // Препринты ИПМ им. М.В. Келдыша РАН № 8. Москва. 2018 г. 28 с. doi:10.20948/prepr-2018-8.
25. 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.
26. 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.
27. 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.
28. 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.
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. 28с. doi:10.20948/prepr-2017-37.
30. Kaporin I.E. Reordering and splitting of sparse matrices into overlapping blocks for massively parallel preconditioning of iterative methods//Presented at NUMGRID-2012. A.A.Dorodnicyn Computing Center RAS. Moscow. Dec.17-18. 2012.
31. 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.
32. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
33. Tismenetsky M. A new preconditioning technique for solving large sparse linear systems // Linear Algebra Appls. 1991. V. 154-156. P. 331-353.
34. Suarjana M., Law K.H. A robust incomplete factorization based on value and space constraints // Int. J. Numer. Methods Engrg. 1995. V. 38. P. 1703-1719.
35. Manteuffel T.A. An incomplete factorization technique for positive definite linear systems // Math.Comput. 1980. V. 34. P. 473-497.

36. Yamazaki I., Bai Z., Chen W., Scalettar R. *A* High-Quality Preconditioning Technique for Multi-Length-Scale Symmetric Positive Definite Linear Systems // Numer. Math. Theor. Meth. Appl. 2009. V. 2. N. 2. P. 469-484.
37. Jennigs A., Malik G.M. Partial elimination // J. Inst. Math. Appl. 1977. V.20. P. 307-316.
38. 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