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

Препринты ИПМ • Препринт № 83 за 2020 г.

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

О.Ю. Милюкова

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1 // Препринты ИПМ им. М.В.Келдыша. 2020. № 83. 28 с. https://doi.org/10.20948/prepr-2020-83
https://library.keldysh.ru/preprint.asp?id=2020-83

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 Якоби IC1

Москва — 2020

Милюкова О.Ю.

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1

В работе предлагаются два способа применения MPI+OpenMP технологии для безытерационного построения и обращения предобусловливателя блочного Якоби в сочетании с неполным разложением с отсечением по параметру IC1. В способе 1 распараллеливания по потокам используется переупорядочение узлов сетки каждой подобласти типа DDO, многопоточные вычисления применяются для подавляющего большинства строк матрицы. В способе 2 для построения предобусловливателя используется уменьшение шаблона разреженности матрицы, многопоточные вычисления применяются для всех строк матрицы. Проводится сравнение времени решения задач с использованием исходной MPI технологии и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции университета Флориды.

Ключевые слова: итерационное решение систем линейных алгебраических уравнений, разреженные матрицы, неполное треугольное разложение Холецкого, параллельное предобусловливание, метод сопряженных градиентов

Olga Yurievna Milyukova

MPI+OpenMP parallel implementation of conjugate gradient method with the Block Jacobi preconditioner combined with IC1

Two non-iterative algorithms based on MPI+OpenMP techniques are proposed for the construction and application of the Block Jacobi preconditioner combined with incomplete parameter-trimmed decomposition IC1. In Algorithm 1, the Domain Decomposition Ordering is used for the organization of OpenMP processing, OpenMP processing is used for almost all rows of the matrix. In Algorithm 2 uses sparsification of matrix structure, OpenMP processing is used for all rows of the matrix. 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 University of Florida collection test problems are presented.

Keywords: iterative solution of linear systems, sparse matrices, incomplete Cholesky factorization, parallel preconditioning, conjugate gradient method

Работа выполнена при финансовой поддержке РФФИ (код проекта 18-07-00841-а).

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

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

Используется факторизованная матрица предобусловливания

$$B \approx A, \quad B = U^T U,$$

где U – верхнетреугольная матрица. В настоящей работе рассматривается метод предобусловливания блочного Якоби неполного треугольного разложения с отсечением по параметру первого порядка (BJIC1). В этом методе сначала находится предобусловливатель блочного Якоби, а затем для каждого блока строится неполное треугольное разложение с отсечением по параметру $0<\tau<<1$ 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 [1]. Заметим, что предобусловливание IC1(τ) имеет ограниченную область применимости [2].

Решение задач с матрицами очень большого размера требует применения параллельных компьютеров. При решении многомерных задач на многопроцессорных вычислительных системах обычно используют подход, называемый декомпозицией области расчета. Для преодоления основной трудности распараллеливания алгоритмов построения и обращения предобусловливателя неполного треугольного разложения, связанной с рекурсивным характером вычислений, часто используют переупорядочение узлов сетки и соответствующую перестановку строк и столбцов матрицы.

Одними из наиболее часто используемых переупорядочений являются упорядочения, связанные с разбиением области расчета (DDO – Domain Decomposition ordering) [3]. Применению такого подхода для крупнозернистого распараллеливания, когда область расчета разбивается на подобласти и расчеты

в каждой подобласти производятся на своем процессоре, посвящено много работ, например [4-11].

В работе [12] предлагается использовать упорядочение типа DDO для построения параллельного варианта метода стабилизированного треугольного разложения второго порядка сопряженных градиентов (IC2S-CG) [13]. В работах [14,15] предложены и исследованы предобусловливатели блочного неполного обратного треугольного разложения второго порядка, в которых специальным образом строятся блоки с налеганием. Для построения предобусловливателей внутри блока в работах [14,15] используется треугольные разложения с отсечением по параметру второго.

Использование OpenMP технологии для параллельной реализации построения и применения факторизованного предобусловливателя требует дальнейшего изучения. В работе [16] (см. также цитированную там литературу) представлено предобусловливание, основанное на ILU-разложении матрицы коэффициентов с последующим построением приближенных обратных для соответствующих нижнего и верхнего треугольных сомножителей. При таком подходе можно использовать OpenMP технологии.

В работах [17-20] было предложено использовать несколько итераций Якоби или блочного Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения. Такой подход позволяет использовать высокий уровень параллелизма (мелкозернистый или распараллеливание алгоритма на потоки). В работе [21] предлагается безытерационный способ применения MPI+OpenMP технологии при обращении факторизованного предобусловливателя. В работе [22] предлагается новый итерационный алгоритм вычисления неполного LU и IC(0), IC(1), IC(2) (для симметричных матриц) разложений, в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно. Этот алгоритм обладает высоким уровнем параллелизма. Численные эксперименты показали, что достаточно нескольких итераций для получения эффективного предобусловливателя. В работах [23, 24] 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 СЛАУ (1.1) предобусловленным методом CG, например, [25-27].

В работе [28] предлагаются два способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0). Они основаны на упорядочении узлов сетки типа DDO внутри каждой подобласти (способ 1) и уменьшении шаблона разреженности матрицы A (способ 2). При этом многопоточные вычисления применяются для подавляющего большинства строк матрицы. С помощью расчетов модельной задачи и ряда задач из коллекции университета Флориды показано, что применение MPI+OpenMP технологии позволяет существенно ускорить

вычисления по сравнению с применением только MPI технологии 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 [15]. 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 предлагаются два безытерационных способа применения MPI+OpenMP технологии при построении и обращении предобусловливателя BJIC1. При этом в способе 1 при построении и обращении предобусловливателя используется переупорядочение узлов сетки каждой подобласти типа DDO, OpenMP технологии применяются для подавляющего большинства строк матрицы. В способе 2 используется уменьшение шаблона разреженности матрицы A для построения предобусловливателя, OpenMP технологии применяются для всех строк при построении матрицы U и обращении предобусловливателя. По существу, способ 2 представляет собой применение MPI+OpenMP технологии при построении и обращении предобусловливателя BJIC1 для числа блоков, кратного числу используемых процессоров и числу используемых потоков. Проводится сравнение времени решения задач с использованием MPI и MPI+OpenMP подходов на примере модельной задачи и ряда задач из коллекции университета Флориды [31].

2. Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1.1). Алгоритм предобусловленного метода CG (см., например, [32]) имеет следующий вид:

Алгоритм 1

$r_0 = b - Ax_0$, $p_0 = w_0 = B^{-1} r_0$, $\gamma_0 = r_0^T p_0$,

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

$q_k = Ap_k$,

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

$x_{k+1} = x_k + \alpha_k p_k$,

$$r_{k+1} = r_k - \alpha_k q_k,$$
$$z_{k+1} = B^{-1} r_{k+1},$$
$$\gamma_{k+1} = r_{k+1}^T z_{k+1},$$
$$\beta_k = \gamma_{k+1} / \gamma_k,$$
$$p_{k+1} = z_{k+1} + \beta_k p_{k+1},$$

где $0 < \varepsilon << 1$, $B = U^T U$.

Этот алгоритм использует операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений и элементарные векторные операции, а также операции обращения треугольных матриц. Принципиальная возможность эффективной параллельной реализации всех операций, кроме операции обращения треугольных матриц, не вызывает сомнений, даже при использовании большого числа процессоров и (или) применения OpenMP технологии.

3. Алгоритм построения предобусловливателя IC1(τ)

Перед построением матрицы предобусловливания в методе IC1(τ)-CG необходимо выполнить масштабирование матрицы A размера $n \times n$:

$$A_0 = (Diag(A))^{-1/2} A (Diag(A))^{-1/2},$$

где $Diag(A)$ – диагональная часть матрицы A. Тогда матрица A_0 имеет единичную диагональ. Далее вместо обозначения A_0 будем использовать A.

Матричная схема метода предобусловливания IC1(τ) имеет вид

$$A = U^T U - E,$$

где U – верхнетреугольная матрица. Покомпонентная запись этого уравнения имеет вид

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

Зафиксируем i и предположим, что все строки матрицы U с номерами, меньшими i, уже известны. Вычислим вспомогательные значения

$$v_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si} u_{sj}, \; j=i,\ldots,n.$$

Выбор элементов e_{ij} матрицы погрешности E осуществляется так, чтобы «убрать» все «малые» значения v_{ij}. В качестве «базового» используется алгоритм:

если $j = i$, то $e_{ij} = 0$,

если $j > i$, то

$e_{ij} = -v_{ij}$, если $|v_{ij}| < \tau \sqrt{v_{ii}}$,

$e_{ij} = 0$, в противном случае.

Алгоритм, совпадающий с «базовым» (с точностью до критерия отсечения внедиагональных элементов), приведен в работе [2]. В настоящей работе при написании программы использовался следующий алгоритм:

Алгоритм 2

1. Инициализация вспомогательной диагональной матрицы:
   for $i=1,\ldots,n$
   $d_i := 1$
   end for
2. Цикл по строкам A для вычисления строк U :
   for $i=1,\ldots,n$
3. Инициализация вектора v при помощи i-й строки A:
   for $j=i+1,\ldots,n$
   $v_j := a_{ij}$
   end for
4. Цикл по уже вычисленным строкам U, R:
   for $s=1,\ldots,i-1$
5. Сделать поправку к вектору v:
   for $j=i+1,\ldots,n$
   $v_j = v_j - u_{si}u_{sj}$
   end for
   end for
6. Нормализация вектора v:
   $u_{ii} := \sqrt{d_i}$
   for $j=i+1,\ldots,n$
   $v_j := v_j / u_{ii}$
   end for
7. 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
8. Выполнение поправки к вспомогательной диагональной матрице:
   for $j=i+1,\ldots,n$
   $d_j := d_j - u_{ij}^2$
   end for
   end for

Напомним, что $a_{ii}=1$. При написании программы для реализации алгоритма 2 был обеспечен доступ по столбцам к элементам матрицы *U*.

4. Предобусловливание при помощи блочного метода Якоби в сочетании с IC1(τ)

Пусть матрица *A* переупорядочена и разбита на блоки, причем на блочной диагонали расположены *p* квадратных блоков размера $n_s \times n_s$, $1 \le s \le p$. Обозначим $k_s = n_1 + ... + n_s$. Определим прямоугольные матрицы

$$W_s = \left\lfloor e_{k_{s-1}+1} \middle| ... \middle| e_{k_s} \right\rfloor,$$

столбцы которых являются единичными *n*-векторами, $k_{s-1}+1,...,k_s$ представляют собой индексы *s*-го блока. Построим матрицы размерами $n_s \times n_s$: $W_s^T A W_s = A_s$. Построим неполные треугольные разложения IC1(τ) для этих матриц: $A_s \approx U_s^T U_s$. В качестве предобусловливателя будем использовать

$$B = \sum_{s=1}^{p} W_s U_s^T U_s W_s^T ,$$

которое будем называть блочное Якоби неполное треугольное разложение первого порядка (BJIC1). Заметим, что предобусловливатель блочного Якоби имеет вид

$$\overline{B} = \sum_{s=1}^{p} W_s A_s W_s^T \approx A .$$

Вычисление элементов матриц U_s (*s*=1,…,*p*) осуществляется аналогично описанному в разделе 3 (см. Алгоритм 2), вместо матрицы *A* используются матрицы A_s.

5. Алгоритм параллельной реализации

Параллельная реализация вычисления и обращения предобусловливателя BJIC1 с использованием только MPI не представляет труда. При вычислении матрицы U_s в каждом процессоре с номером *s*=1,…,*p* не требуется информации, хранящейся в других процессорах. В процессе вычисления матриц U_s все процессоры могут работать одновременно и независимо, пересылок не требуется. Для вычисления матриц U_s^T производится транспонирование матриц U_s. При этом все процессоры могут работать одновременно. Как показывают расчеты, операция транспонирования таких матриц занимает ничтожно малое время по сравнению с остальным временем вычисления предобусловливателя [25, 26].

При выполнении операции

$$z_s = (U_s^T U_s)^{-1} r_s$$

тоже все процессоры могут работать одновременно, пересылок не требуется.

Рассмотрим способ 1 применения OpenMP технологии при вычислении элементов верхнетреугольной матрицы при построении предобусловливателя BJIC1 для подавляющего большинства строк этой матрицы. Разобьем каждую подобласть, вычисления в которой происходят на своем процессоре, на m внутренних подобластей, где m – число используемых нитей (потоков) при применении OpenMP технологии. Как показывают расчеты [28] задач 1-5, приведенных в разделе 6, с точки зрения общего времени решения СЛАУ целесообразно производить разбиение по порядку следования узлов в подобласти на приблизительно равные части.

Будем использовать внутри каждой подобласти упорядочение, предложенное в работе [12], являющееся упорядочением типа DDO. Введем множество узлов разделителей – множество узлов сетки во внутренних подобластях, у которых имеются соседи из внутренних подобластей с бóльшими номерами. Остальные узлы сетки в подобласти 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 уровень, расположенных в подобластях с бóльшими номерами. Остальные узлы разделителей назовем узлами разделителей третьего уровня. Установим следующий порядок следования узлов сетки в подобласти. Сначала идут все «внутренние» узлы подобластей, полученных в результате разбиения узлов сетки в подобласти, в порядке следования номеров внутренних подобластей, причем сохраняется порядок следования узлов внутри каждой 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 узлов внутри подобласти, введенного ранее.

На рис. 1 приведен пример структуры разреженности матрицы $\tilde{A}_s = P_s A_s P_s^T$, где P_s – матрица перестановок, полученной после перестановки строк и столбцов в результате переупорядочения в случае разбиения подобласти с номером s на 4 подобласти (m=4). На рис. 1 использованы обозначения: l – число всех «внутренних» узлов сетки из всех внутренних подобластей подобласти с номером s, $l1$ – число всех «внутренних» узлов сетки из всех внутренних подобластей подобласти с номером s и всех узлов разделителей первого уровня из всех внутренних подобластей этой подобласти, $l2$ – число всех «внутренних» узлов сетки из всех внутренних подобластей и всех узлов разделителей первого и второго уровня из всех внутренних подобластей подобласти с номером s. Строки матрицы, содержащие блочно-диагональные части A_{11}^{κ}, соответствуют «внутренним» узлам внутренней подобласти с номером k (k=1,2,3,4). Строки матрицы, содержащие блочно-диагональные

части A_{22}^{κ} и A_{33}^{κ}, соответствуют узлам сетки соответственно на разделителях первого и второго уровня внутренней подобласти с номером *k*. Строки матрицы, соответствующие блочно-диагональной части A_{44}, соответствуют узлам сетки на разделителях третьего уровня внутренних подобластей подобласти с номером s.

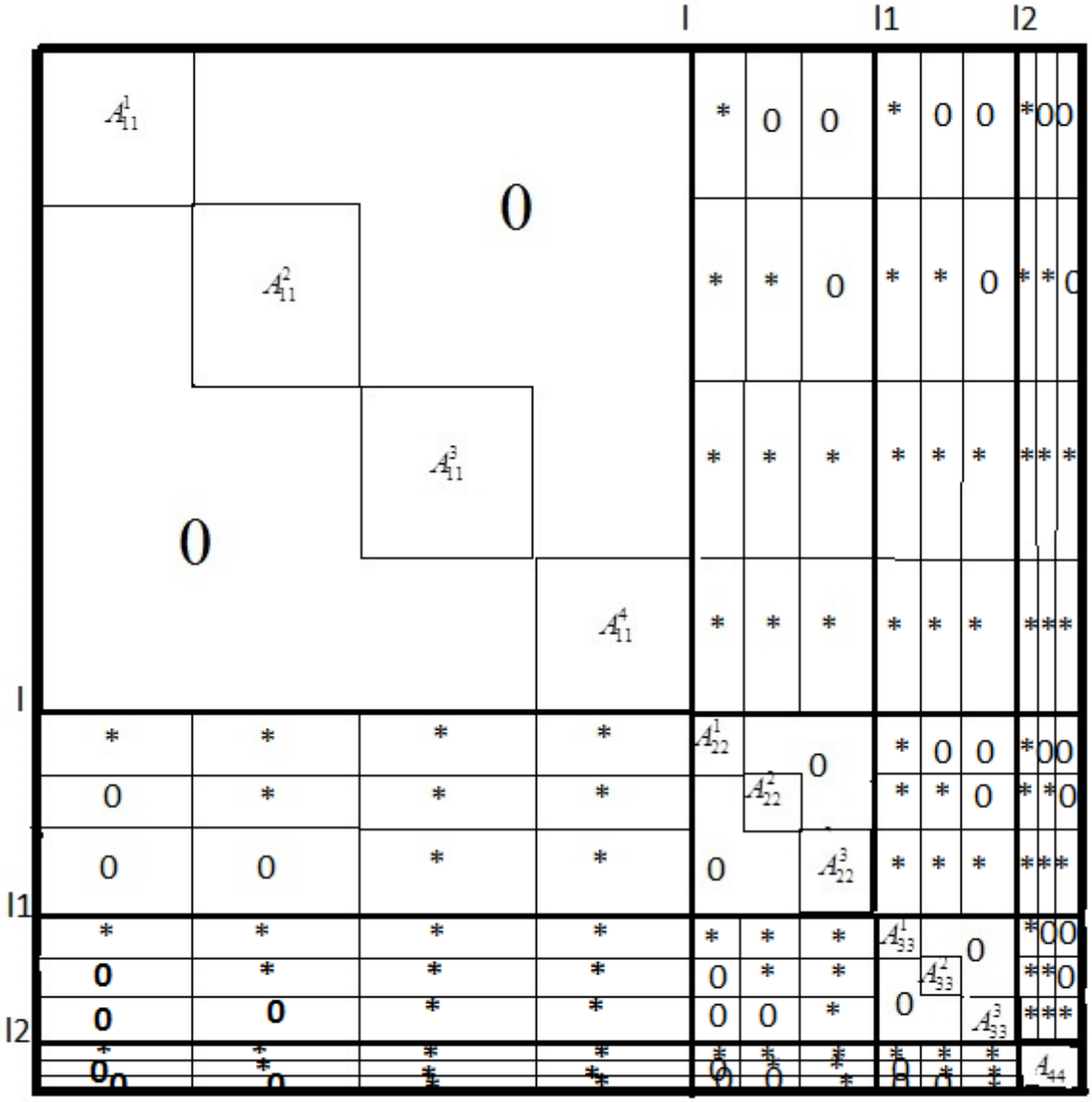

Рис. 1. Пример структуры разреженности матрицы $\tilde{A}_s$, полученной после перестановки столбцов и строк в результате переупорядочения матрицы A_s.

Можно доказать, что при вычислении элементов верхнетреугольного множителя предобусловливателя $\tilde{U}_s$ в строках, соответствующих «внутренним» узлам, не требуется значений элементов матрицы $\tilde{U}_s$ в строках, соответствующих «внутренним» узлам из других внутренних подобластей подобласти с номером s, s=1,…,p.

На рис. 2 приведен вид структуры разреженности матрицы $\tilde{U}_s$, полученной для матрицы $\tilde{A}_s$, структура разреженности которой изображена на рис.1.

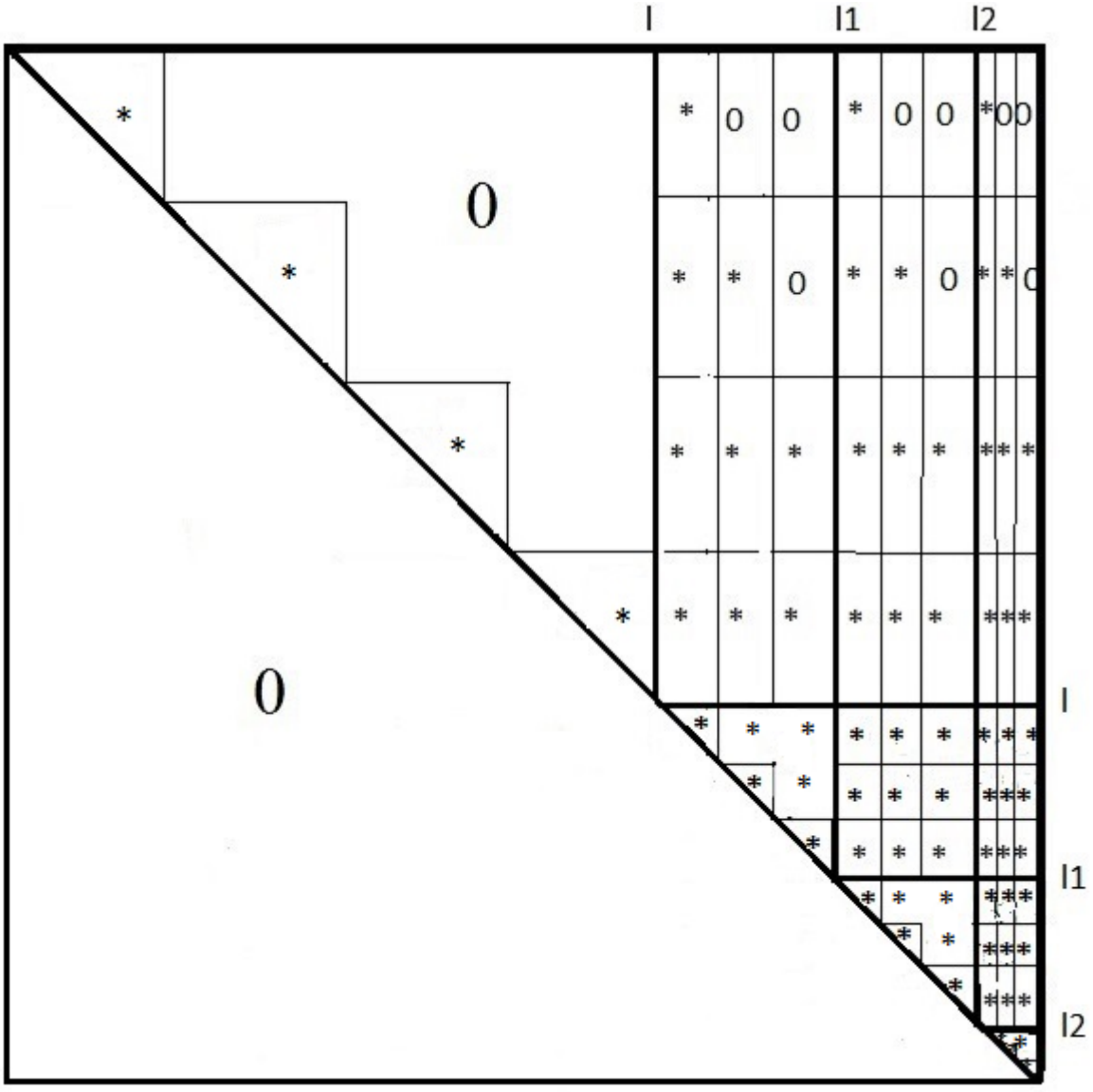

Рис. 2**.** Структура разреженности матрицы $\tilde{U}_s$ для матрицы $\tilde{A}_s$, структура разреженности которой изображена на рис.1.

Итак, разбиение каждой подобласти производится на *m* подобластей. Определим $\bar{l}_s$ – минимальное значение количества «внутренних» узлов при разбиении подобласти на внутренние подобласти. Предполагается, что $\bar{l}_s \neq 0$. Обозначим $M1 = m\bar{l}_s$. Заметим, что *M*1 зависит от s, *M*1= *M*1(s). Переупорядочим узлы в подобласти следующим образом. Сначала идут $\bar{l}_s$ узлов первой внутренней подобласти, затем $\bar{l}_s$ узлов второй внутренней подобласти и т. д. до $\bar{l}_s$ узлов *m*-ой внутренней подобласти включительно.

Далее идут оставшиеся «внутренние» узлы внутренних подобластей в порядке возрастания номеров подобластей. Затем идут узлы разделителей в порядке, установленном ранее. Вычисление элементов матриц $\overline{U}_s$ для матрицы $\overline{A}_s = \overline{P}_s \widetilde{A}_s \overline{P}_s^T$, где $\overline{P}_s$ – матрица перестановок, происходит следующим образом. При вычислении первых *M1* строк с использованием алгоритма 2 будем применять OpenMP технологии. Для цикла по *i* =1,*M*1 (см. алгоритм 2 в разделе 3) будем использовать директиву do с опцией schedule static с числом нитей *m.* При этом каждой нити (потоку) достаются для расчетов соответствующие $\bar{l}_s$ подряд идущие при новом упорядочении строк, рекурсивные вычисления производятся при расчетах внутри каждого потока. После вычисления $m \times \bar{l}_s$ строк матрицы $\overline{U}_s$ необходимо вычислить значения $d_i = 1 + \sum_{k=1}^{m} d_i^k$ ($i = m \times \bar{l}_s + 1, \ldots, n_s$), где d_i^k – значения, вычисленные в *k*-ом потоке (при проведении расчетов в *k*–ой внутренней подобласти). Вычисления остальных строк матрицы $\overline{U}_s$ с использованием алгоритма 2 (кроме п. 1) производятся без применения OpenMP технологии. Как правило, *M1* не очень сильно меньше числа узлов в подобластях n_s, *s*=1,...,*p*. Поэтому можно надеяться на хорошую эффективность использования такого подхода, если матрицы $\overline{U}_s$ не будут очень сильно заполненными.

Заметим, что можно не проводить дополнительное переупорядочение $\overline{A}_s = \overline{P}_s \widetilde{A}_s \overline{P}_s^T$, а применять OpenMP технологии для построения предобусловливателя для матрицы $\widetilde{A}_s$. При этом необходимо организовать внешний цикл по k2=1,…,*m,* внутри которого будет осуществляться цикл по *i=ibeg, iend*, где $ibeg = \sum_{k=1}^{k2-1} l_k + 1$, $iend = \sum_{k=1}^{k2} l_k$, l_k – количество «внутренних» узлов во внутренней подобласти с номером *k* (при разбиении подобласти с номером s) с использованием алгоритма 2. OpenMP технологии следует применять для цикла по k2=1, …, *m*. Однако, как показали расчеты задач, приведенных в разделе 6, такой подход, как правило, приводит к бóльшему времени решения СЛАУ.

Для вычисления элементов матрицы $\overline{U}_s^T$ производится транспонирование матрицы $\overline{U}_s$, при этом не применяются OpenMP технологии.

При обращении матриц $\overline{U}_s$ и $\overline{U}_s^T$ будем использовать OpenMP технологии для подавляющего большинства строк. Вычисление элементов вектора $w_s = \overline{U}_s^{-T} r_s$ происходит аналогично вычислению элементов матрицы $\overline{U}_s$. Вычисление элементов вектора $z_s = \overline{U}_s^{-1} w_s$ происходит в обратном порядке, причем последние при расчете *M1* элементов этого вектора вычисляются с использованием OpenMP технологии.

Заметим, что можно использовать другое упорядочение узлов сетки в подобласти типа DDO.

Рассмотрим способ 2 параллельной реализации с использованием OpenMP технологии этапов построения и обращения предобусловливателя. Как и в способе 1, разобьем каждую подобласть, соответствующую вычислениям на своем процессоре, на m приблизительно равных подобластей, где m – число используемых нитей (потоков). Пусть $\hat{l}_s = \lceil n_s / m \rceil$, первые m-1 внутренних подобластей содержат $\hat{l}_s$ узлов, последняя внутренняя подобласть содержит $n_s - (m-1)\hat{l}_s$ узлов. При использовании способа 2 при построении матрицы U_s изменим шаблон разреженности матрицы A_s, получим матрицу $\hat{A}_s$, элементы $\hat{a}_{ij}$ которой определяются следующим образом:

$\hat{a}_{ij} = 0$, если $a_{ij} = 0$ или узлы i и j принадлежат разным внутренним подобластям подобласти,

$\hat{a}_{ij} = a_{ij}$ в противном случае.

Здесь a_{ij} – элементы матрицы A_s. Заметим, что использование этого алгоритма эквивалентно использованию блочного предобусловливания в сочетании с IC1(τ) для матрицы A для числа блоков, кратного m и кратного p.

При написании программы вычисления элементов матрицы U_s организуем внешний цикл по k2=1,2,…,m, внутри которого будет цикл по i=(k2-1)$\hat{l}_s$+1,…,iend, где iend=k2*$\hat{l}_s$ при k2$\neq m$, iend=n_s при k2=m (s=1,…,p). Для цикла по k2 будем использовать директиву **do** с опцией **schedule static**. Для вычисления элементов матрицы U_s будем в каждом потоке использовать алгоритм 2. При этом рекурсивные вычисления происходят внутри потока. Заметим, что при использовании способа 2 применения OpenMP технологии переупорядочения внутри подобластей не производятся.

Вычисление элементов вектора $w_s = U_s^{-T} r_s$ происходит аналогично вычислению элементов матрицы U_s. Вычисление элементов вектора $z_s = U_s^{-1} w_s$ происходит в обратном порядке.

Заметим, что при использовании способа 2 применение OpenMP технологии происходит для бóльшего числа строк матрицы (для всех строк), чем при использовании способа 1. Однако при использовании способа 2 возможно более заметное увеличение числа итераций предобусловленного метода сопряженных градиентов из-за дополнительного изменения шаблона разреженности матрицы A при построении предобусловливателя. Эти факторы по-разному сказываются на времени расчетов.

Матрица A хранится в памяти в распределенном CRS-формате и содержит как верхний, так и нижний треугольник. Параллельная реализация умножения матрицы на вектор с использованием MPI+OpenMP технологии в этом случае, а также параллельная реализация с использованием MPI+OpenMP технологии вычислений векторных операций и скалярных произведений хорошо известны.

Заметим, что способ, аналогичный способу 1 использования MPI+OpenMP технологии, может быть применен для параллельной реализации вычислений при решении СЛАУ (1.1) методом IC1(τ)-CG, если распараллеливание по процессорам производится с использованием упорядочения типа DDO. При этом применение OpenMP технологии можно производить только для внутренних узлов подобластей, полученных при разбиении всей области расчета для распараллеливания по процессорам. Заметим, что итерации метода BJIC1-CG могут сходиться медленнее, чем итерации IC1(τ)-CG, их рост с ростом числа процессоров может быть быстрее.

6. Результаты расчетов

Программы, реализующие применение метода BJIC1-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии, расчеты производились 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 университета Флориды [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;
thermal2 – стационарная термальная задача;
boneS01 – модель трубчатой кости;
offshore – диффузия электрического поля (3-мерная тетраэдральная сетка);
cfd2 – вычислительная гидродинамика (уравнение давления).

В таблице 1 приведены некоторые свойства этих матриц, причем значения

Таблица 1

Свойства некоторых матриц из Флоридской коллекции

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
thermal2	1228045	8580313	381319	840	1	11	0.45+7
ecology2	999999	4995991	1124	0	3	5	0.63+8
boneS01	127224	5516602	127222	2064830	12	67	0.13+8
Offshore	259789	4242673	194617	1813920	5	31	0.34+5
cfd2	123440	3085406	123440	936464	8	30	0.15+7

$Cond(A_0)$, где $A_0 = (D_A)^{-1/2} A (D_A)^{-1/2}$ – матрица системы уравнений после масштабирования, взяты из работы [33], 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.

Модельную задачу далее будем называть задачей 1. Задачу с матрицей **apache2** далее будем называть задачей 2, задачи с матрицами **parabolic_fem**, **thermal2**, **ecology2** – соответственно задачей 3, задачей 4 и задачей 5. Задачи с матрицами **boneS01, offshore, cfd2** будем называть задачами 6, 7, 8.

Решалось уравнение $Ax = b$, где $A = A_0$, правая часть $b_i \equiv 1$, начальное приближение $x_0 \equiv 0$, счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета при решении всех задач использовался способ [30]. В качестве параметра отсечения при решении задач 1-5 использовалось значение $\tau = 0.01$. При решении задачи 6 (матрица **boneS01)** $\tau = 0.005$, при решении задачи 7 (матрица **offshore)** $\tau = 0.0001$, при решении задачи 8 (матрица **cfd2)** $\tau = 0.00001$. Выбор таких значений τ при решении задач 6, 7, 8 был продиктован требованием безотказности метода BJIC1-CG.

В таблицах 2 – 9 приведены числа итераций и времена счета методом BJIC1-CG модельной задачи и задач с матрицами из коллекции университета Флориды при использовании для параллельной реализации MPI и MPI+OpenMP технологии. Под временем вычислений подразумевается время счета итерационного процесса в сумме с временем вычисления предобусловливателя и переупорядочений в подобластях (если они производятся). В случае использования MPI+OpenMP технологии при решении задач 1-6 расчеты производились с использованием 3, 4, 6 и 8 нитей (потоков), а при решении задач 7, 8 использовались 3, 4, 6, 8, 12, 16 нитей. В таблицах 2-9 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).

Как видно из таблиц 2-6, при использовании метода BJIC1-CG для решения модельной задачи и задач из коллекции университета Флориды наблюдается некоторый (не всегда монотонный) рост числа итераций с ростом числа процессоров. Это связано с использованием блочного метода Якоби при построении предобусловливателя, а также с использованием переупорядочения узлов сетки типа DDO внутри подобластей или изменением шаблона разреженности для применения OpenMP технологии.

Заметим, что, как показали расчеты, наблюдается рост числа итераций с ростом числа процессоров при фиксированном числе потоков при использовании обоих способов применения MPI+OpenMP технологии, а также

рост числа итераций с увеличением числа потоков при фиксированном числе процессоров. Как показали расчеты задач для небольшого числа процессоров, в большинстве случаев наблюдается монотонное уменьшение времени вычисления предобусловливателя с увеличением числа используемых нитей для обоих способов применения MPI+OpenMP технологии.

Таблица 2

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **5_1048576** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=446,11.34	It=490, 6.65 1.7	It=520, 4.09 2.77	It=578,2.32 4.88	It=634,1.16 9.77
MPI+ OpenMP (сп.1) μ , η	Th=6 It=466,3.3 3.43	Th=6 It=521,3.17 2.09 3.57	Th=3 It=538, 3.41 1.19 3.32	Th=4 It=600,2.55 0.91 4.44	Th=3 It=652,1.36 0.85 8.33
MPI+ OpenMP (сп.2) μ , η	Th=6 It=631, 3.67 3.09	Th=3 It=594, 3.4 1.95 3.33	Th=3 It=668, 3.88 1.05 2.92	Th=3 It=774, 3.07 0.75 3.69	Th=3 It=831,1.68 0.69 6.75

Таблица 3

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **thermal2** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=964, 47.53	It=1012, 18.87 2.51	It=1050,9.67 4.91	It=1049,5.44 8.73	It=1183,3.1 15.33
MPI+ OpenMP (сп.1) μ , η	Th=6 It=979,11.51 4.12	Th=6 It=1034,7.84 2.41 6.06	Th=3 It=1062,7.21 1.34 6.59	Th=3 It=1116,5.90 0.92 8.06	Th=4 It=1215,3.36 0.92 14.15
MPI+ OpenMP (сп.2) μ , η	Th=6 It=1119, 11.7 4.06	Th=6 It=1200,7.97 2.36 5.96	Th=3 It=1157,7.41 1.3 6.41	Th=3 It=1283,6.82 0.8 6.96	Th=3 It=1357,3.83 0.81 12.4

На рисунках 3-10 представлены графики зависимости времени счета задач 1-8 от числа процессоров в логарифмическом масштабе с использованием только MPI (красные линии) и с использованием MPI+OpenMP технологии (черные и синие линии). Черные линии соответствует расчетам с использованием способа 1, синие линии соответствует расчетам с использованием способа 2.

Таблица 4

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **parabolic_fem** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=446, 5.96	It=519, 3.52 1.69	It=553, 2.06 2.89	It=595, 1.22 4.88	It=659, 0.66 9.03
MPI+ OpenMP (сп.1) μ , η	Th=6 It=494, 2.04 2.92	Th=3 It=517, 1.88 1.87 3.17	Th=3 It=573, 1.79 1.15 3.33	Th=4 It=627, 1.47 0.83 4.05	Th=4 It=700, 0.86 0.76 6.93
MPI+ OpenMP (сп.2) μ , η	Th=6 It=601, 1.92 3.1	Th=3 It=605, 1.95 1.8 3.05	Th=3 It=657, 1.87 1.1 3.18	Th=3 It=726, 1.65 0.74 3.61	Th=4 It=906,1.018 0.64 5.84

Таблица 5

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **apache2** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=582, 11.65	It=678, 6.2 1.87	It=745, 3.84 3.03	It=907, 2.39 4.87	It=996,1.53 7.61
MPI+ OpenMP (сп.1) μ , η	Th=6 It=589, 3.66 3.18	Th=3 It=682, 3.26 1.9 3.57	Th=3 It=751, 3.15 1.21 3.69	Th=4 It=915, 2.74 0.87 4.25	Th=4 It=1009,1.65 0.93 7.06
MPI+ OpenMP (сп.2) μ , η	Th=6 It=1032, 4.65 2.5	Th=3 It=963, 3.68 1.68 3.16	Th=3 It=1139, 4.06 0.94 2.86	Th=3 It=1321, 3.77 0.63 3.09	Th=4 It=1009,1.65 0.93 7.06

Заметим, что в ряде случаев при решении задач 1-8 наблюдалось сверхлинейное ускорение при использовании только MPI. Это связано с тем,

что сравнение производилось с временем счета на 4 процессорах, когда уже было произведено разбиение области расчета на 4 подобласти. Число итераций и время счета зависят, в частности, от способа разбиения на подобласти и способа упорядочения узлов расчетной сетки.

Таблица 6

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **ecology2** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=809, 17.85	It=816, 9.71 1.84	It=901, 6.46 2.76	It=957, 3.31 5.39	It=1044, 1.9 9.39
MPI+ OpenMP (сп.1) μ , η	Th=6 It=843, 4.86 3.67	Th=3 It=840, 4.57 2.12 3.9	Th=3 It=930, 5.09 1.27 3.51	Th=4 It=1030,3.88 0.85 4.6	Th=4 It=1109,2.33 0.81 7.66
MPI+ OpenMP (сп.2) μ , η	Th=6 It=1063, 5.26 3.39	Th=3 It=1035, 5.14 1.88 3.47	Th=3 It=1123,6.00 1.08 2.97	Th=3 It=1296,4.76 0.69 3.75	Th=3 It=1437,2.9 0.65 6.15

Таблица 7

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **boneS01** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	8 2	16 4	32 8	64 16
MPI η	It=498, 7.74	It=449, 4.05 1.91	It=557, 2.36 3.28	It=625, 1.41 5.49	It=694, 0.85 9.11
MPI+ OpenMP (сп.1) μ , η	Th=3 It=451, 5.22 1.48	Th=3 It=490, 3.28 1.23 2.36	Th=3 It=569, 2.71 0.87 2.85	Th=6 It=590,1.65 0.85 4.69	Th=4 It=670,0.99 0.86 7.81
MPI+ OpenMP (сп.2) μ , η	Th=6 It=785, 2.61 2.96	Th=3 It=725, 2.49 1.62 3.11	Th=3 It=804,2.26 1.04 3.42	Th=3 It=850,1.77 0.79 4.37	Th=3 It=943,1.05 0.81 7.37

Как видно из таблиц 2-6 и рис. 3-7, применение способа 1 использования MPI+OpenMP технологии позволяет производить расчет задач 1-5 быстрее, чем

при использовании только MPI, при $p \le 16$. Как видно из таблиц 2-4, 6 и рис. 3-5, 7, применение способа 2 использования MPI+OpenMP технологии позволяет производить расчет задач 1-3, 5 быстрее, чем при использовании только MPI, при $p \le 16$. Из таблицы 5 и рис. 6 видно, что при решении задачи 4 (матрица **apache2)** применение способа 2 использования MPI+OpenMP технологии позволяет производить расчет быстрее, чем при использовании только MPI, при $p \le 8$. Применение способа 1 использования MPI+OpenMP технологии приводит к лучшим результатам, чем применение способа 2, при всех значениях $p \le 64$ в случаях решения задач 1, 2, 4, 5. При решении задачи 3

Таблица 8

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **offshore** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	16 4	32 8	64 16	100 25
MPI η	It=669, 42.1	It=765, 11.62 3.62	It=838, 7.31 5.75	It=795, 5.29 7.95	It=719, 2.41 17.46
MPI+ OpenMP (сп.1) μ , η	Th=8 It=669, 37.01 1.14	Th=3 It=778, 12.40 0.94 3.39	Th=12 It=837, 7.27 1.005 5.79	Th=12 It=795,3.76 1.41 11.19	Th=8 It=720,2.08 1.16 20.24
MPI+ OpenMP (сп.2) μ , η	Th=16 It=1187, 6.91 6.09	Th=16 It=1216, 6.83 1.7 6.16	Th=16 It=1187,3.69 1.98 11.4	Th=16 It=1174,2.13 2.48 19.76	Th=8 It=945,1.16 2.08 36.29

Таблица 9

Числа итераций и времена счета методом BJIC1-CG задачи с матрицей **cfd2** на p процессорах без использования и с использованием OpenMP технологии

P p/4	4 1	16 4	32 8	64 16	100 25
MPI η	It=612,161.9	It=989, 40.96 3.95	It=1081,15.72 10.29	It=1342,7.21 22.45	It=1364,3.88 41.72
MPI+ OpenMP (сп.1) μ , η	Th=12 It=611, 137.4 1.18	Th=4 It=989, 38.8 1.05 4.17	Th=8 It=1140, 11.45 1.37 14.13	Th=6 It=1342,5.33 1.35 30.37	Th=4 It=1360,3.39 1.14 47.75
MPI+ OpenMP (сп.2) μ , η	Th=16 It=1188,11.42 14.17	Th=16 It=1782,10.08 4.06 16.06	Th=16 It=2082,4.31 3.64 37.56	Th=8 It=2132, 2.5 2.88 64.76	Th=6 It=2182,1.86 2.08 87.04

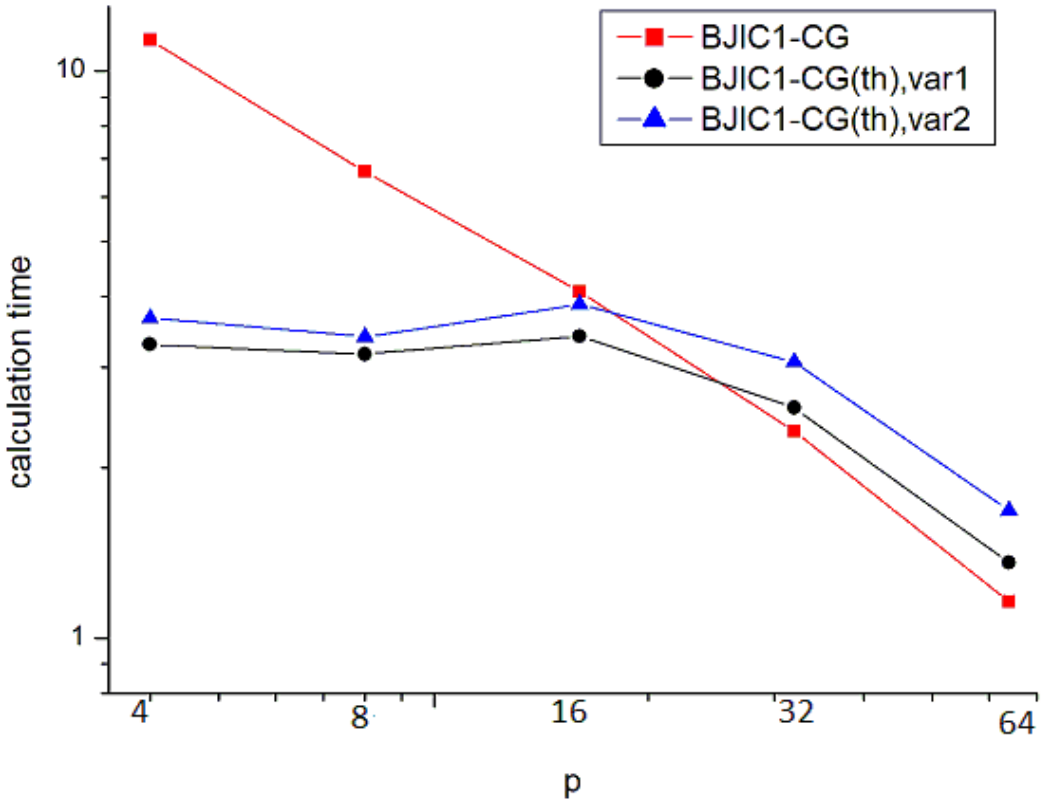

Рис. 3. Времена счета модельной задачи методом BJIC1-CG
с использованием MPI и MPI+OpenMP технологии

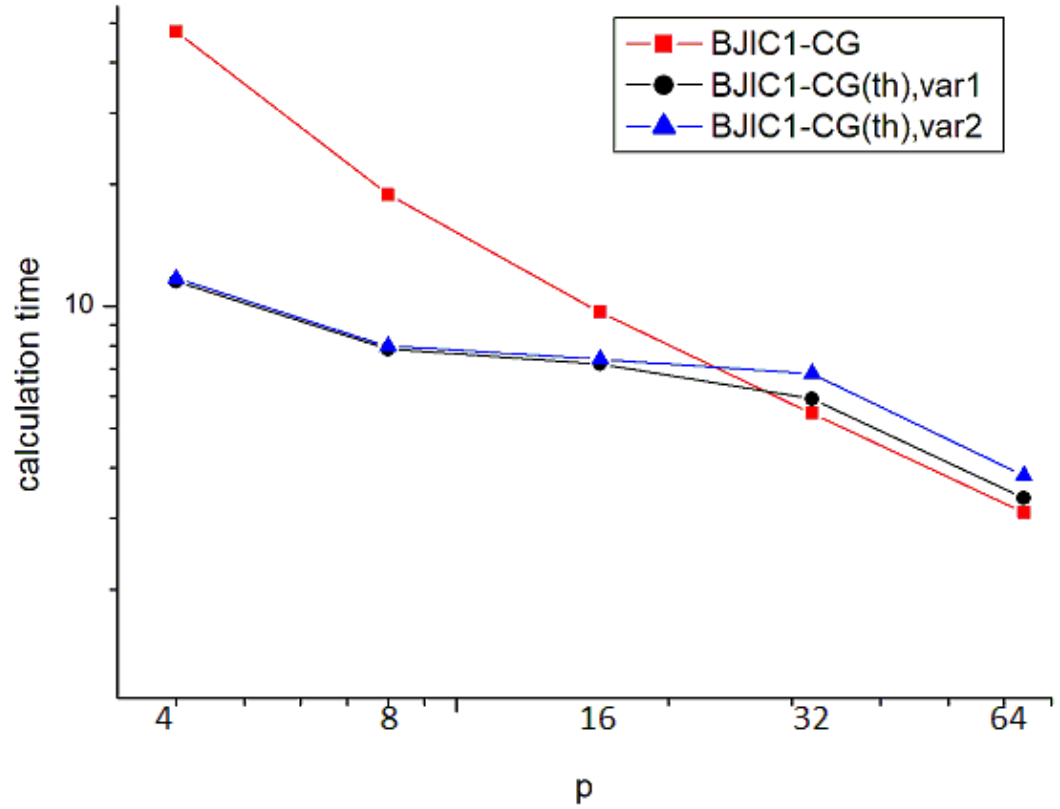

Рис. 4. Времена счета задачи с матрицей **thermal2** методом BJIC1-CG
с использованием MPI и MPI+OpenMP технологии

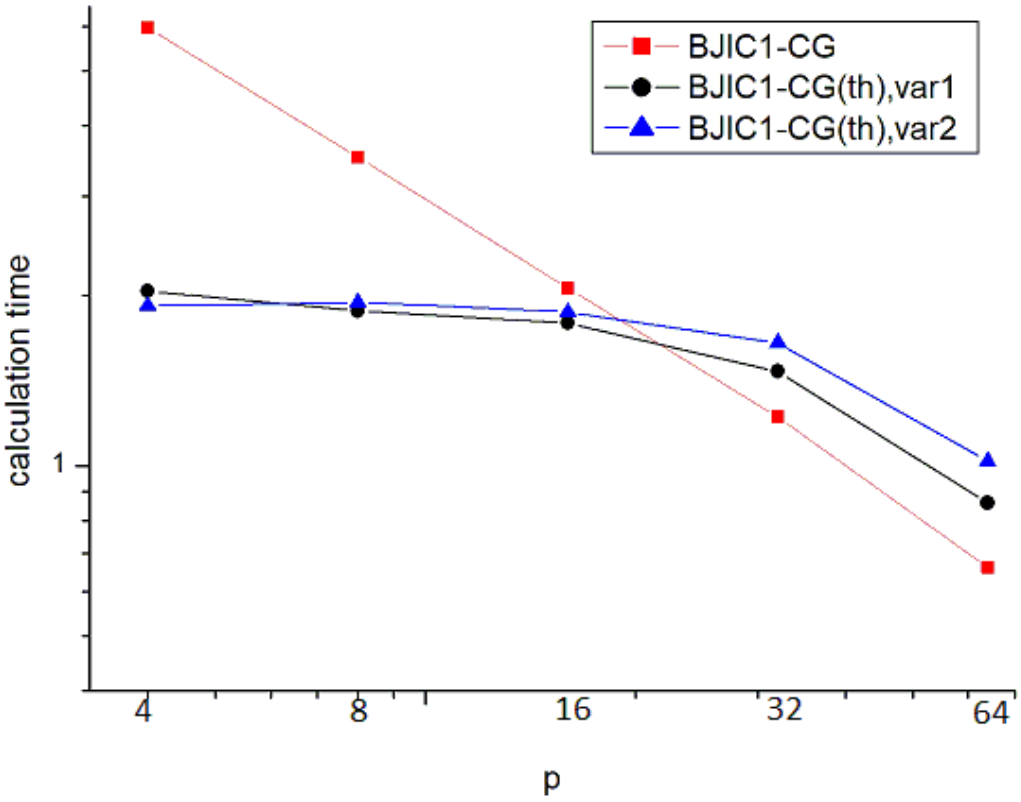

Рис. 5. Времена счета задачи с матрицей **parabolic_fem** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

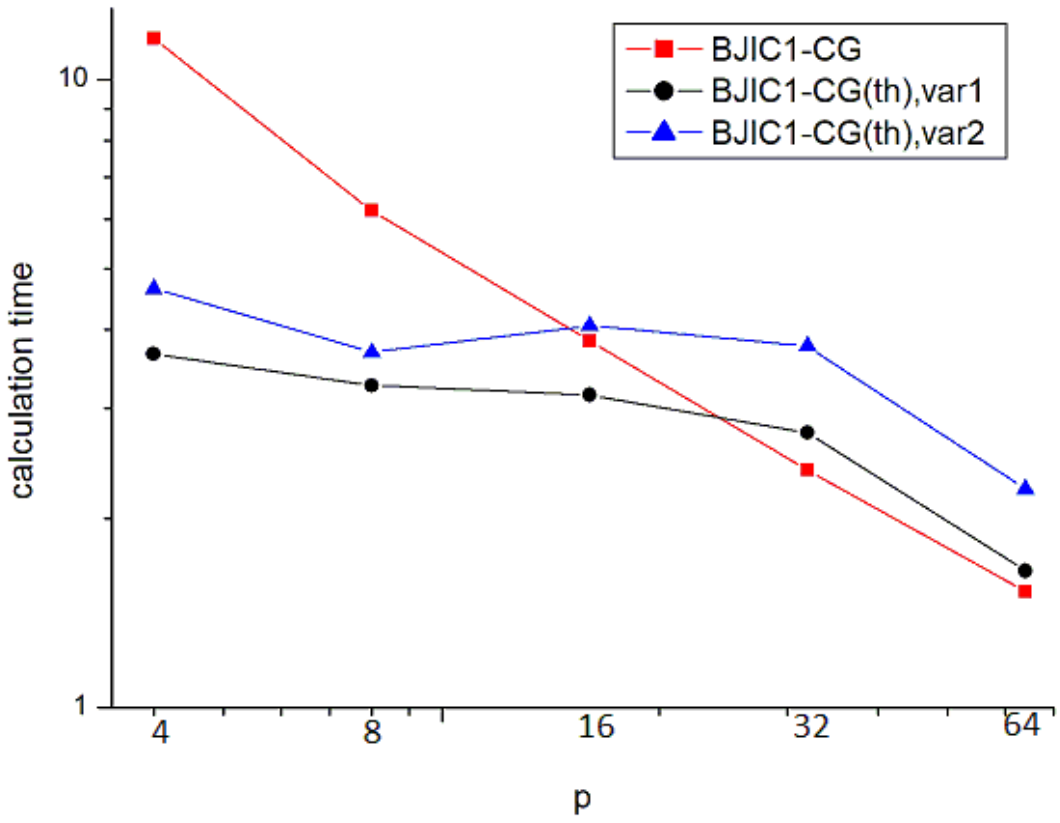

Рис. 6. Времена счета задачи с матрицей **apache2** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

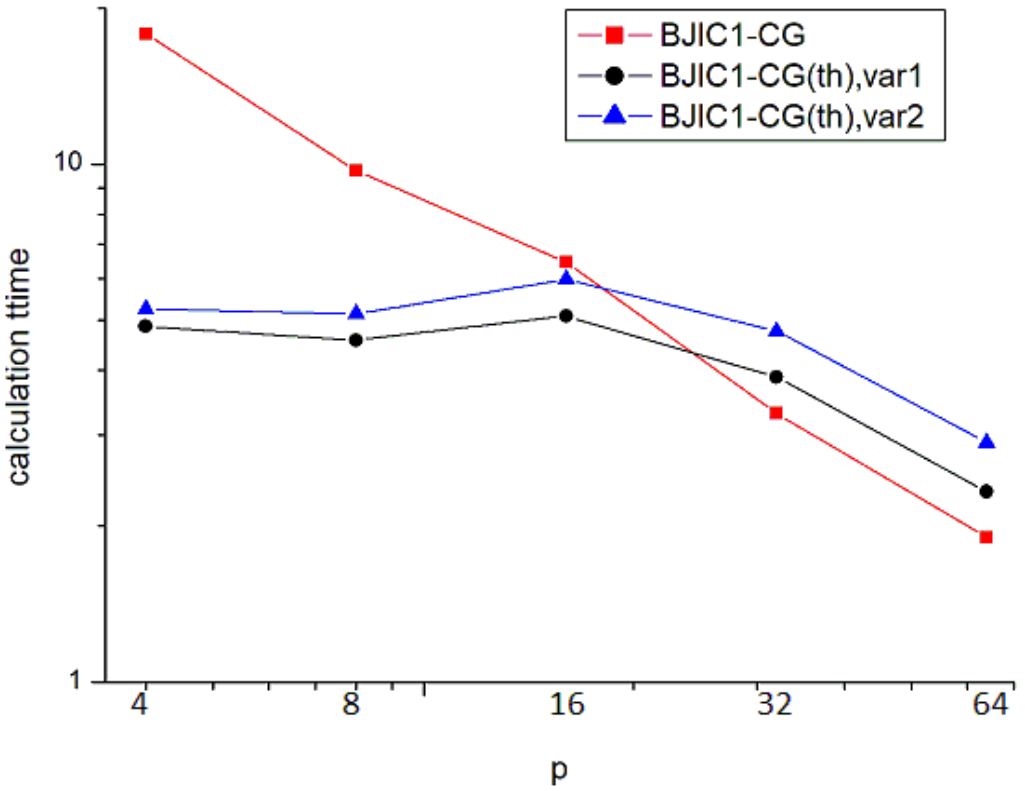

Рис. 7. Времена счета задачи с матрицей **ecology2** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

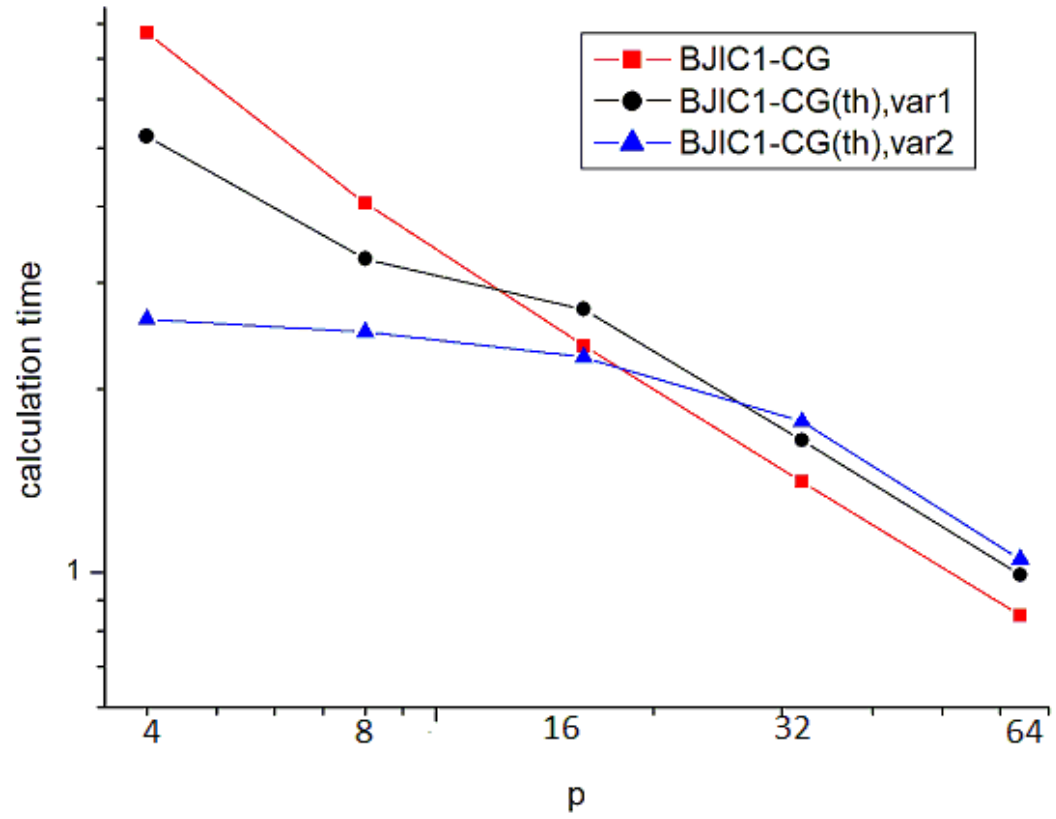

Рис. 8. Времена счета задачи с матрицей **boneS01** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

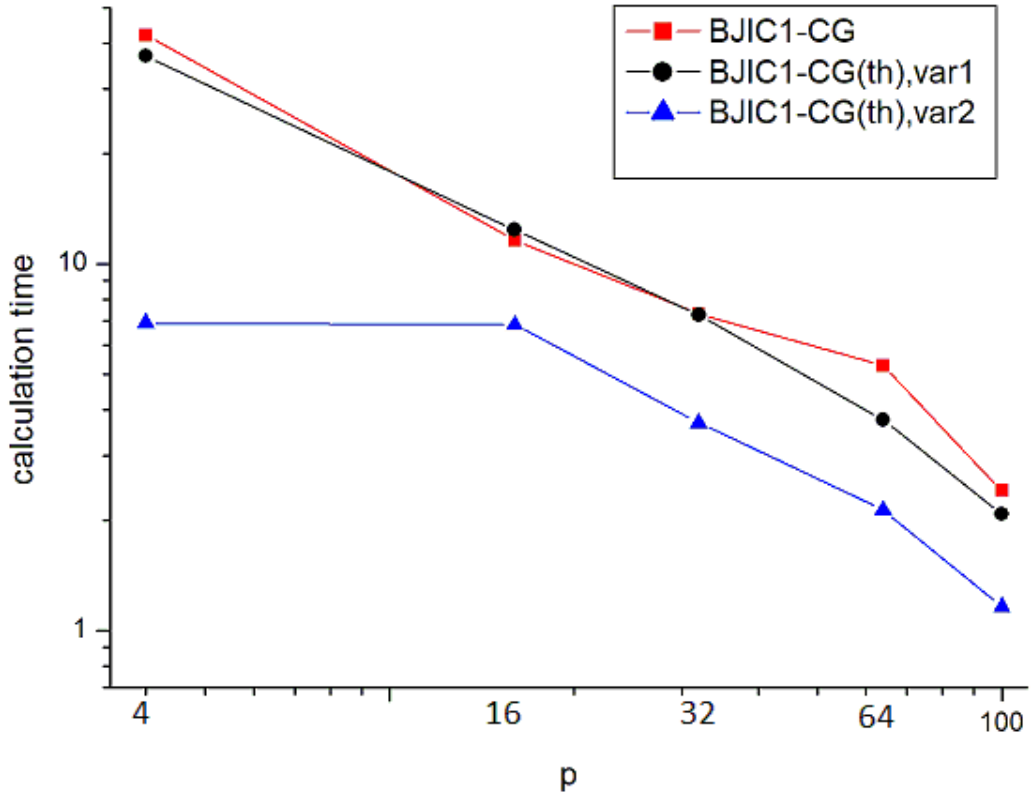

Рис. 9. Времена счета задачи с матрицей **offshore** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

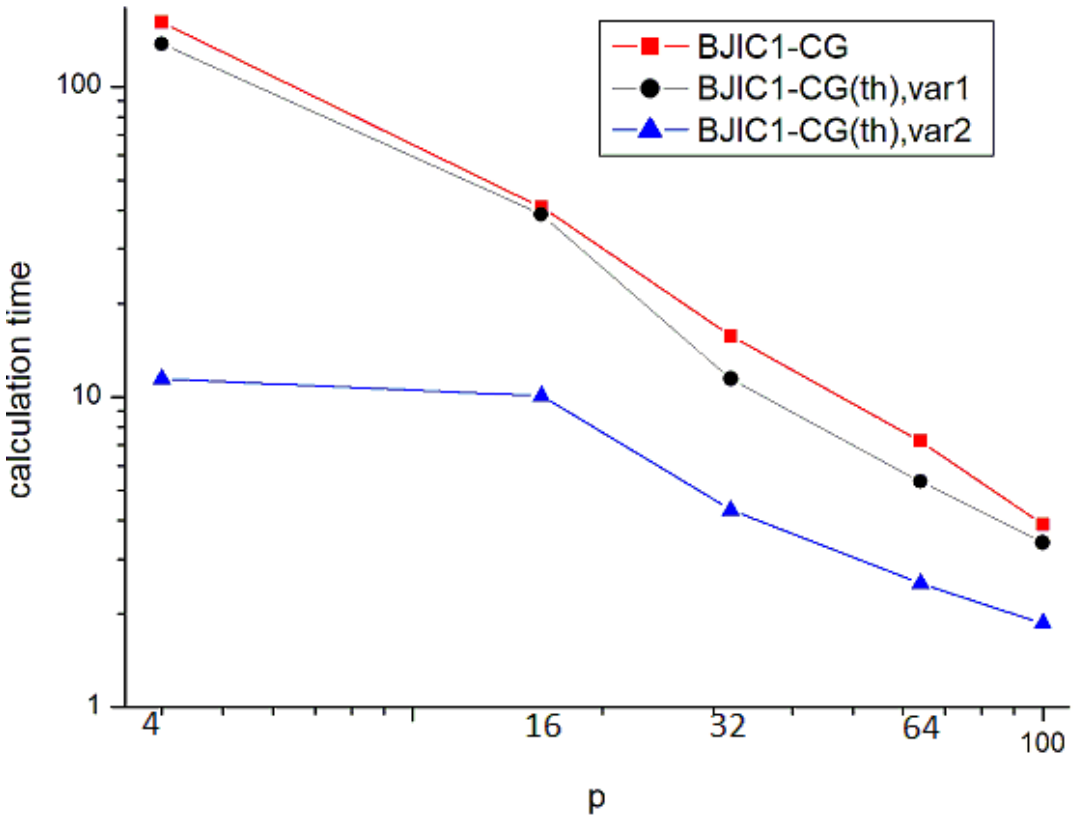

Рис. 10. Времена счета задачи с матрицей **cfd2** методом BJIC1-CG с использованием MPI и MPI+OpenMP технологии

(с матрицей **parabolic_fem**) применение способа 1 использования MPI+OpenMP технологии приводит к лучшим результатам, чем применение способа 2, при всех значениях $4 < p \le 64$.

Перейдем теперь к анализу результатов расчетов задач с более плотными матрицами **boneS01, offshore, cfd2 (**задачи 6, 7, 8). В случае решения задачи 6 (матрица **boneS01**, $\tau = 0.005$) применение способа 1 использования MPI+OpenMP технологии позволяет производить расчет быстрее, чем при использовании только MPI, при p=4,8, а применение способа 2 позволяет производить расчет быстрее, чем при использовании только MPI, при p=4,8,16. Применение способа 1 использования MPI+OpenMP технологии приводит к лучшим результатам, чем применение способа 2, только при p=32,64.

Решение задач 7,8 производилось с маленькими значениями τ ($\tau = 0.0001$ и $\tau = 0.00001$) для обеспечения безотказности методов. При решении этих задач при всех значениях $p \le 100$ применение способа 2 использования MPI+OpenMP технологии позволяет производить расчет много быстрее, чем при использовании только MPI. Применение способа 1 использования MPI+OpenMP технологии позволяет производить расчеты задач 7, 8 в подавляющем большинстве случаев немного быстрее, чем при использовании только MPI. Однако разница времен счета с применением OpenMP технологии и без ее применения невелика. Применение способа 1 использования OpenMP технологии при решении последних двух задач можно считать нецелесообразным. Это связано прежде всего с необходимостью использовать очень маленькие значения τ для обеспечения безотказности метода, а следовательно, получением более плотных матриц $\overline{U}_s$ ($s=1,...p$). Это приводит к большому времени расчета при вычислении предобусловливателя, прежде всего за счет большого времени вычислений его значений на разделителях, когда не используются OpenMP технологии.

Итак, при решении задач методом BJIC1-CG с достаточно большими значениями порога отсечения τ применение обоих способов использования MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI в случае не слишком большого числа процессоров (при $p < 32$ или при p<16). При решении задач методом BJIC1-CG в случае достаточно заполненной матрицы СЛАУ и необходимости использовать очень маленькие значения порога отсечения применение способа 2 использования MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению применением только MPI для любого умеренного числа процессоров при $p \le 100$.

Заметим, что решение задач с большими и сильно разреженными матрицами с применением MPI+OpenMP технологии методом CG с предобусловливанием Якоби (J-CG), когда $B = D_A$, где $D_A -$ диагональная часть матрицы A, начиная с некоторого числа процессоров становится неэффективно по сравнению с применением только MPI [25, 26]. Поэтому потеря

эффективности использования MPI+OpenMP технологии по сравнению с использованием только MPI при применении метода BJIC1-CG в этом случае не выглядит столь неожиданной.

Уменьшение эффекта от использования OpenMP технологии с увеличением числа процессоров объясняется также уменьшением числа строк матрицы, приходящихся на каждый процессор. 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 от применения OpenMP технологии при решении задач с использованием не слишком малого значения τ наступит при значительно большем числе процессоров.

Заметим, что применение OpenMP технологии позволяет в ряде случаев уменьшить количество использованных процессоров при параллельной реализации расчетов с помощью MPI+OpenMP подхода для получения приблизительно того же времени счета.

В дальнейшем предполагается обобщение рассматриваемых в настоящей работе подходов на случаи построения и применения предобусловливателя с отсечением по значению второго порядка.

6. Заключение

В работе предложены два способа применения MPI+OpenMP технологии для безытерационного построения и обращения предобусловливателя блочного Якоби в сочетании с неполным разложением с отсечением по параметру первого порядка IC1(τ). Они основаны на упорядочении узлов сетки типа DDO внутри каждой подобласти (способ 1) и уменьшении шаблона разреженности матрицы A (способ 2). При использовании способа 1 многопоточные вычисления применяются для подавляющего большинства строк матрицы, при использовании способа 2 – для всех строк матрицы. С помощью расчетов модельной задачи и ряда задач из коллекции университета Флориды показано, что использование обоих способов применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI технологии для умеренного числа узлов суперкомпьютерной системы (порядка нескольких десятков) в случае не слишком малого порога отсечения τ. При необходимости производить решение СЛАУ с очень маленьким значением порога отсечения использование только второго способа применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением MPI технологии для любого умеренного числа использованных в расчетах процессоров.

Список литературы

1. Munksgaard N. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. № 6. P. 206-219.
2. Tuff A.D., Jennings A. An iterative method for large systems of linear structural equations // J. numer. Methods engrg. 1973. №7 – P.175-183.
3. Duff I.S., Meurant G.A. The effect of ordering on preconditioned conjugate gradients // BIT. 1989. V. 29. P. 625-657.
4. Doi S. On parallelism and convergence of incomplete LU factorizations //Applied Numerical Mathematics: Transactions of IMACS. 1991. V. 7. № 5. P.417–436.
5. Notay Y. An efficient parallel discrete PDE solver // Parallel Computing. 1995. V.21. P.1725-1748.
6. Milyukova O. Yu. Parallel approximate factorization method for solving discreate elliptic equations // Parallel Computing. 2001. №27. P.1365-1379.
7. Милюкова О.Ю. Параллельные итерационные методы с факторизованной матрицей предобусловливания для решения эллиптических уравнений. Диссерт. на соиск. степ. докт. физ.-мат. наук. Москва. 2004. 219 С.
8. 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.
9. Hysom D., Pothen A. A scalable parallel algorithm for incomplete factor preconditioning // SIAM J. Sci. Comput. 2001. V. 22. P. 2194-2215.
10. Karypis G., Kumar V. Parallel threshold-based ILU factorization // Proceedings of the ACM/IEEE Conference on Supercomputing. ACM, New York, IEEE, Washington.DC. 1997.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. Anzt H., Huckle T.K., Brackle J., Dongarra J. Incomplete Sparse Approximate Inverses for Parallel Preconditioning // Parallel Computing. 2018. V.71. P.1–22.
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. N. 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. 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.
24. Li R., Saad Y. GPU-Accelerated Preconditioned Iterative Linear Solvers // Tech. Report UMSI-2010-112. University of Minnesota Supercomputing Institute. 2010. Minneapolis. MN.
25. Капорин И.Е., Милюкова О.Ю. MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями // Препринты ИПМ им. М.В. Келдыша. 2018. № 8. 28 с. doi:10.20948/prepr-2018-8
26. 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.
27. 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.
28. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем// Препринты ИПМ им. М.В. Келдыша. 2020. № 31. 22 с. http://doi.org/10.20948/prep-2020-31 http://library.keldysh.ru/preprint.asp?id=2020-31.

29. 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.
30. 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.
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.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