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

Препринты ИПМ • Препринт № 31 за 2020 г.

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

О.Ю. Милюкова

MPI+OpenMPI реализация метода сопряженных градиентов с факторизованным предобусловливателем

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. MPI+OpenMPI 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
https://library.keldysh.ru/preprint.asp?id=2020-31

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М. В. Келдыша
Российской академии наук

О. Ю. Милюкова

MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем

Москва — 2020

Милюкова О.Ю.

MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем

В работе предлагаются два способа применения MPI+OpenMP технологии для безытерационного построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0). При этом многопоточные вычисления применяются для подавляющего большинства строк матрицы. В способе 1 для распараллеливания по потокам используется переупорядочение узлов сетки каждой подобласти типа DDO, а в способе 2 используется уменьшение шаблона разреженности 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 factorized preconditioner

Two non-iterative algorithms based on MPI+OpenMP techniques are proposed for the construction and application of the Block Jacobi preconditioner combined with IC(0) factorization. In the algorithms, OpenMP processing is used for almost all rows of the matrix. In Algorithm 1, the Domain Decomposition Ordering is used for the organization of OpenMP processing, while Algorithm 2 uses sparsification of matrix structure. 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

$$B \approx A,\ B = LL^T,$$

где L – нижнетреугольная матрица. В настоящей работе рассматривается метод предобусловливания блочного Якоби неполного треугольного разложения Холецкого без заполнения (BJIC(0)). В этом методе сначала находится предобусловливатель блочного Якоби, а затем для каждого блока строится неполное треугольное разложение Холецкого без заполнения IC(0) [1]. При использовании IC(0) матрица L имеет структуру разреженности, совпадающую со структурой разреженности нижнетреугольной части матрицы A. Заметим, что предобусловливание IC(0) имеет ограниченную область применимости, теоретическое обоснование применимости сделано для М-матриц [1], для Н-матриц [2], в частности метод применим в случае положительных диагональных элементов матрицы, отрицательных внедиагональных элементов и наличия диагонального преобладания.

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. Основная трудность распараллеливания алгоритмов построения и обращения предобусловливателя неполного треугольного разложения связана с рекурсивным характером вычислений. Для ее преодоления 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 неполного треугольного разложения второго порядка сопряженных градиентов (IC2S-CG) [13]. В работах [14, 15] предлагается предобусловливатель блочного неполного обратного треугольного разложения второго порядка на основе перекрывающихся блоков. При параллельной реализации метода CG с этими предобусловливателями не требуется использование специального упорядочение узлов сетки.

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 сопряженных градиентов, например, [25-27].

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 [28, 29], 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 BJIC(0). При этом OpenMP технологии применяются для подавляющего большинства строк матрицы. В способе 1 при построении и обращении предобусловливателя используется переупорядочение узлов сетки каждой подобласти типа DDO, рассмотренное в работе [12] для распараллеливания по подобластям. В способе 2 при построении и обращении предобусловливателя используется уменьшение шаблона разреженности матрицы A. Проводится сравнение времени решения задач с использованием MPI и MPI+OpenMP подходов на примере модельной задачи и ряда задач из коллекции университета Флориды [30].

2. Предобусловленный метод сопряженных градиентов

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

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

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

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.

Алгоритм вычисления матрицы L в случае построения предобусловливателя IC(0) имеет вид [2] (a_{ij} – элементы матрицы A, l_{ij} – элементы матрицы L).

Алгоритм 1.

Для $i=1,n$

Если $i \neq 1$, то

Для $j=1,i-1$

Если $a_{ij} = 0$, то $l_{ij} = 0$

иначе

$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}$

Если $i \neq 1$, то $l_{ii} = (a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2)^{1/2}$, $l_{11} = \sqrt{a_{11}}$.

3. Предобусловливание при помощи блочного метода Якоби в сочетании с неполным разложением Холецкого без заполнения

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

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

столбцы которых являются единичными n-векторами, $k_{s-1}+1,...,k_s$ представляют собой индексы s-ого блока. Построим матрицы размерами $n_s \times n_s$: $W_s^T A W_s = A_s$. Построим неполные треугольные разложения Холецкого без заполнения для этих матриц: $A_s \approx L_s L_s^T$. В качестве предобусловливателя будем использовать

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

которое будем называть блочное Якоби неполное треугольное разложение Холецкого без заполнения (BJIC(0)). 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 L_s (s=1,…,p) осуществляется аналогично описанному в разделе 2 (см. Алгоритм 1), вместо матрицы A используются матрицы A_s.

4. Алгоритм параллельной реализации

Параллельная реализация вычисления и обращения предобусловливателя BJIC(0) с использованием только MPI не представляет труда. При вычислении матрицы L_s в каждом процессоре с номером s=1,…,p не требуется информации,

хранящейся в других процессорах. В процессе вычисления матрицы L_s все процессоры могут работать одновременно и независимо, пересылок не требуется. Для вычисления матрицы L_s^T производится транспонирование матрицы L_s. При этом все процессоры могут работать одновременно, пересылок не требуется. Как показывают расчеты, операция транспонирования таких матриц занимает ничтожно малое время по сравнению с остальным временем вычисления предобусловливателя [25, 26].

При выполнении операции

$$z = (L_s L_s^T)^{-1} r$$

тоже все процессоры могут работать одновременно, пересылок не требуется.

Рассмотрим способ 1 применения OpenMP технологии при вычислении элементов нижнетреугольной матрицы при построении предобусловливателя BJIC(0) 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 (проведенные при $m \le 8$) задач, приведенных в разделе 5, 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 бóльшим номером. 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.

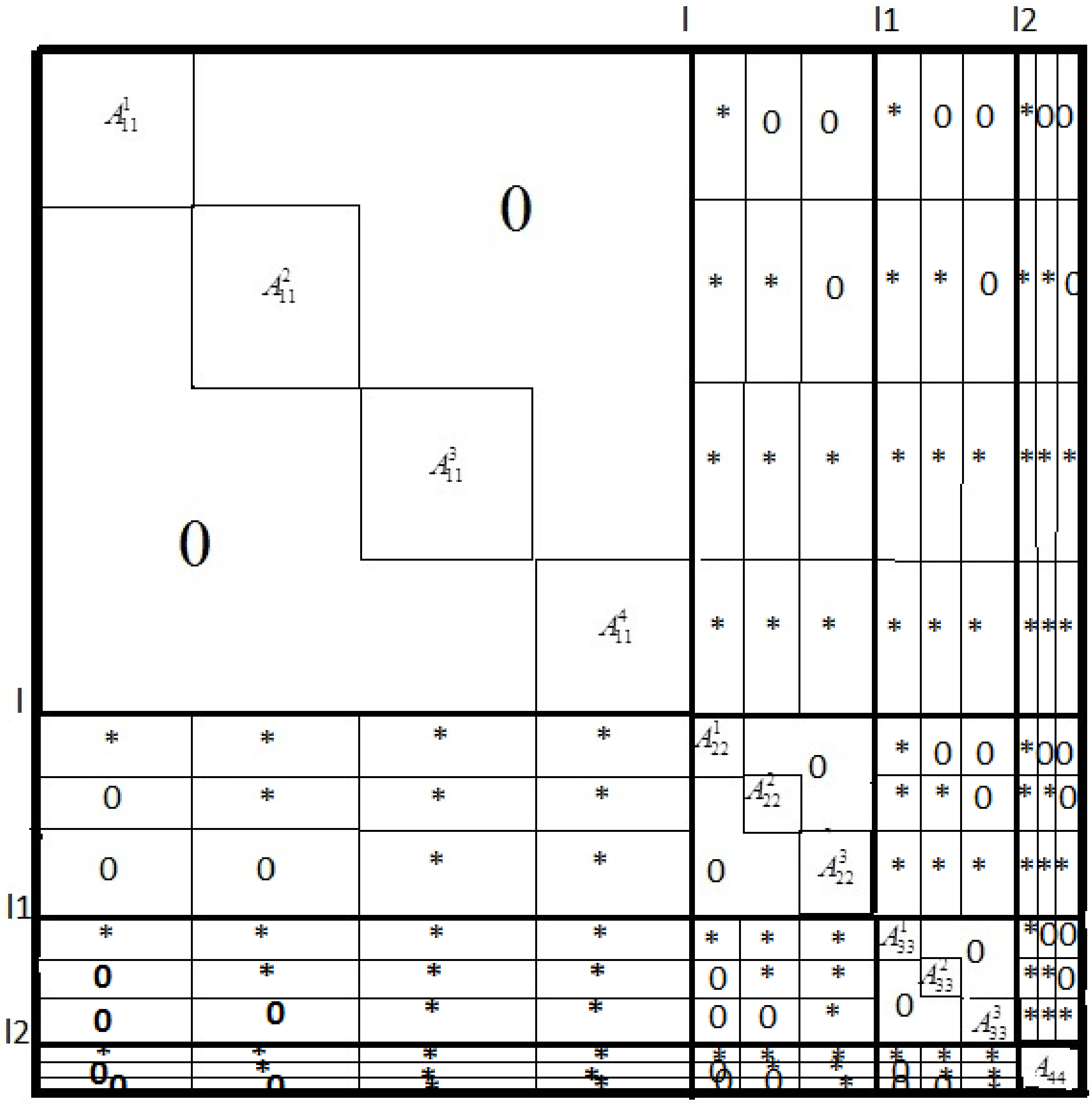

Рис. 1. Пример структуры разреженности матрицы $\tilde{A}_s$, полученной после перестановки столбцов и строк в результате переупорядочения.

Строки матрицы, соответствующие блочно-диагональной части A_{44}, соответствуют узлам сетки на разделителях третьего уровня внутренних подобластей подобласти с номером s.

Можно доказать, что при вычислении элементов нижнетреугольного множителя предобусловливателя в строках, соответствующих «внутренним» узлам, не требуется значений элементов этой нижнетреугольной матрицы в строках, соответствующих «внутренним» узлам из других внутренних подобластей подобласти с номером s, s=1,…,p. Можно доказать, что при вычислении элементов нижнетреугольного множителя предобусловливателя в строках, соответствующих узлам разделителей первого (второго) уровня, не требуется значений элементов этой нижнетреугольной матрицы в строках, соответствующих узлам разделителей первого (второго) уровня из других внутренних подобластей подобласти с номером s, s=1,…,p.

Итак, разбиение каждой подобласти производится на m подобластей. Определим $\bar{l}$ – минимальное значение количества «внутренних» узлов при разбиении подобласти на внутренние подобласти. Предполагается, что $\bar{l} \neq 0$. Обозначим $M1 = m\bar{l}$. Вычисление элементов матриц $\overline{L}_s$ для матрицы $\overline{A}_s = \overline{P}_s \widetilde{A}_s \overline{P}_s^T$ происходит следующим образом. При вычислении первых *M1* строк будем использовать OpenMP технологии. Для цикла по i =1,M1 (см. алгоритм 1 в разделе 2) будем использовать директиву **do** с опцией **schedule static.** При этом каждой ните (потоку) достаются для расчетов соответствующие $\bar{l}$ подряд идущие при новом упорядочении строк, рекурсивные вычисления производятся при расчетах внутри каждого потока. Вычисления остальных строк матрицы производятся без использования OpenMP технологии. Сначала вычисляются элементы в строках матриц, соответствующих оставшимся «внутренним» узлам при разбиении на внутренние подобласти, а далее в порядке, установленном при переупорядочении узлов внутри подобласти, описанном выше. 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. Для вычисления матрицы $\overline{L}_s^T$ производится транспонирование матрицы $\overline{L}_s$, при этом не применяются OpenMP технологии.

Способ использования OpenMP технологии при проведении вычислений элементов строк матрицы $\overline{L}_s$, соответствующих «внутренним» узлам сетки в подобласти, может быть применен при проведении вычислений элементов строк этой матрицы, соответствующих узлам разделителей первого и второго уровня. Однако, расчеты задач, приведенных в разделе 5, показали, что использование подхода, применяемого для «внутренних» узлов подобластей подобласти, к узлам разделителей первого уровня нецелесообразно с точки зрения времени вычислений. Тем более, это будет нецелесообразно для узлов

разделителей второго уровня. Возможно, это окажется целесообразным в случаях более плотных матриц.

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

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

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 $\hat{l}$ – минимальное число узлов в каждой внутренней подобласти ($\hat{l} \approx n_s / m$), $M2 = \hat{l} m$. При использовании способа 2 при построении матрицы L_s изменим шаблон разреженности этой матрицы. Будем использовать следующий алгоритм вычисления элементов нижнетреугольной матрицы.

Алгоритм 2.

Для $i=1, n_s$

Если $i \neq 1$, то

Для $j=1,i-1$

Если $a_{ij} = 0$ или узлы i и j принадлежат разным внутренним подобластям подобласти, то $l_{ij} = 0$,

иначе $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}$.

Если $i \neq 1$, то $l_{ii} = (a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2)^{1/2}$, $l_{11} = \sqrt{a_{11}}$.

Здесь a_{ij} - элементы матрицы A_s. Перед вычислением l_{ij} с использованием алгоритма 2 следует для элементов матрицы A_s выполнить: $a_{ij} := 0$, если узлы i и j принадлежат разным внутренним подобластям подобласти с номером s, s=1,…,p.

Переупорядочим узлы подобласти (если это необходимо) следующим образом. Сначала идут первые $\hat{l}$ узлов первой внутренней подобласти, затем первые $\hat{l}$ узлов второй внутренней подобласти, итак далее до m-той подобласти включительно. После этого следуют оставшиеся узлы подобластей в порядке следования, установленном до переупорядочения внутри подобласти.

При вычислении первых *M2* строк матрицы $\hat{L}_s$ – сомножителя для предобусловливателя $\hat{L}_s \hat{L}_s^T$ матрицы $\hat{A}_s = \hat{P}_s A_s \hat{P}_s^T$, где $\hat{P}_s$ – матрица перестановки, $\hat{P}_s = I$ (I – единичная матрица), если перестановка не

осуществлялась, будем использовать OpenMP технологии для подавляющего большинства строк. Для цикла по *i* от 1 до *M2* в алгоритме 2, применяемом для вычисления матрицы $\hat{L}_s$, будем использовать директиву **do** с опцией **schedule static**. При этом рекурсивные вычисления происходят внутри потока. Остальные строки ($i=M2+1,\ldots,n_s$ при новом упорядочении) будем вычислять без использования OpenMP технологии. Здесь s=1,…,p.

Вычисление элементов вектора $w=\hat{L}_s^{-1}r$ происходит аналогично вычислению элементов матрицы $\hat{L}_s$. Вычисление элементов вектора $z=\hat{L}_s^{-T}w$ происходит в обратном порядке, причем последние при расчете *M2* строк вычисляются с использованием OpenMP технологии.

Заметим, что при использовании способа 2 применение OpenMP технологии, как правило, происходит для бóльшего числа строк, чем при использовании способа 1. Однако при использовании способа 2 возможно более заметное увеличение числа итераций предобусловленного метода сопряженных градиентов из-за изменения шаблона разреженности.

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

Заметим, что аналогичный способ применения MPI+OpenMP технологии может быть применен для параллельной реализации вычислений при решении СЛАУ (1.1) методом IC(0)-CG, если распараллеливание по процессорам производится с использованием упорядочения типа DDO. Заметим, что итерации метода BJIC-CG(0) могут сходиться медленнее, чем итерации IC(0)-CG, и их рост с ростом числа процессоров может быть быстрее.

5. Результаты расчетов

Все программы, реализующие применение метода BJIC(0)-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 [30]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:
apache2 – трехмерная конечно-разностная схема;
parabolic_fem – уравнение диффузии-конвекции с постоянным переносом;
ecology2 – приложение теории электрических цепей к задаче передачи генов;
thermal2 – стационарная термальная задача.

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

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 2, задачей 3, задачей 4 и задачей 5.

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

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 [29]. При использовании способа 2 применения OpenMP технологии переупорядочения внутри подобластей не производились, первые m-1 внутренних подобластей содержали по $\lfloor n_s / m \rfloor$ узлов, s=1,…,p.

В таблицах 2–6 приведены числа итераций и времена счета методом BJIC(0)-CG модельной задачи и задач с матрицами из коллекции университета Флориды при использовании для параллельной реализации MPI и MPI+OpenMP технологии. В случае использования MPI+OpenMP технологии расчеты производились с использованием 3, 4, 6 и 8 нитей (потоков). В таблицах 2-6 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей (Th). Приведены также коэффициенты ускорения счета благодаря использованию OpenMP технологии (обозначены μ) на том же числе процессоров и коэффициенты ускорения счета по сравнению со счетом на 4 процессорах (обозначены η) без использования OpenMP технологии.

Как видно из таблиц 2-6, при использовании метода BJIC(0)-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.

Таблица 2

Числа итераций и времена счета методом BJIC(0)-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=1206,22.2	It=1257,12.42 1.81	It=1270,7.3 3.04	It=1318,3.67 6.05	It=1328,1.76 12.61
MPI+ OpenMP (сп.1) μ ,η	Th=6 It=1258,5.60 3.96, 3.96	Th=3 It=1233, 5.09 2.44, 4.36	Th=3 It=1295,5.87 1.24, 3.78	Th=3 It=1298, 4.07 0.9, 5.45	Th=4 It=1377, 2.0 0.88, 11.1
MPI+ OpenMP (сп.2) μ ,η	Th=6 It=1308,5.36 4.14, 4.14	Th=6 It=1357, 5.23 2.37, 4.24	Th=3 It=1356, 5.9 1.24, 3.76	Th=3 It=1390, 4.3 0.85, 5.16	Th=3 It=1400,2.13 0.82, 10.4

Таблица 3

Числа итераций и времена счета методом BJIC(0)-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=1849, 73.5	It=1886, 28.42 2.58	It=1934,14.3 5.14	It=1972,8.17 8.99	It=2026,4.21 17.45
MPI+ OpenMP (сп.1) μ ,η	Th=6 It=1892, 17.1 4.3, 4.3	Th=6 It=1941,11.06 2.57, 6.64	Th=3 It=1996,10.6 1.35, 6.93	Th=4 It=1971,8.45 0.96, 8.69	Th=4 It=2061,4.64 0.91, 15.84
MPI+ OpenMP (сп.2) μ ,η	Th=6 It=1985,16.4 4.48, 4.48	Th=6 It=2057,10.97 2.59, 6.7	Th=3 It=2036,10.8 1.32, 6.8	Th=3 It=2111,9.22 0.88, 7.97	Th=3 It=2140,4.88 0.86, 15.06

Таблица 4

Числа итераций и времена счета методом BJIC(0)-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=943, 9.56	It=994, 5.41 1.76	It=1009, 2.92 3.27	It=1041, 1.71 5.59	It=1072,0.87 10.98
MPI+ OpenMP (сп.1) μ , η	Th=6 It=962, 2.56 3.73, 3.73	Th=6 It=1034, 2.52 2.15, 3.79	Th=3 It=1024, 2.42 1.21, 3.95	Th=4 It=1078, 1.98 0.86, 4.82	Th=4 It=1102,1.05 0.81, 9.1
MPI+ OpenMP (сп.2) μ , η	Th=6 It=1003, 2.38 4.02, 4.02	Th=6 It=1108, 2.49 2.17, 3.83	Th=3 It=1049, 2.44 1.2, 3.91	Th=4 It=1159, 2.08 0.82, 4.59	Th=3 It=1170,1.13 0.81, 9.1

Таблица 5

Числа итераций и времена счета методом BJIC(0)-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=922, 14.98	It=1048, 7.76 1.93	It=1083, 4.43 3.38	It=1367, 2.82 5.31	It=1438,1.55 9.66
MPI+ OpenMP (сп.1) μ , η	Th=6 It=1065, 4.84 3.09, 3.09	Th=3 It=1113, 3.86 2.01, 3.88	Th=4 It=1177, 3.96 1.12, 3.78	Th=3 It=1452, 3.57 0.79, 4.19	Th=3 It=1550,1.96 0.79, 7.64
MPI+ OpenMP (сп.2) μ , η	Th=6 It=1271, 4.64 3.22, 3.22	Th=3 It=1219, 3.83 2.02, 3.91	Th=3 It=1372, 4.15 1.06, 3.6	Th=3 It=1655, 3.96 0.71, 3.78	Th=3 It=1825,2.30 0.67, 6.51

На рисунках 2-6 представлены графики зависимости времени счета задач 1-5 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 6

Числа итераций и времена счета методом BJIC(0)-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=2214, 35.6	It=2225,19.28 1.84	It=2302,12.1 2.94	It=2315,5.84 6.1	It=2367,3.11 11.45
MPI+ OpenMP (сп.1) μ , η	Th=6 It=2252, 8.48 4.20, 4.20	Th=4 It=2268, 8.06 2.39, 4.41	Th=3 It=2336,9.84 1.23, 3.61	Th=4 It=2367,6.62 0.88, 5.37	Th=4 It=2503,3.88 0.8, 9.17
MPI+ OpenMP (сп.2) μ , η	Th=6 It=2337, 8.20 4.34, 4.34	Th=6 It=2404, 8.02 2.4, 4.43	Th=3 It=2404,9.33 1.3, 3.81	Th=4 It=2509,6.81 0.86, 5.22	Th=3 It=2583,4.25 0.73, 8.37

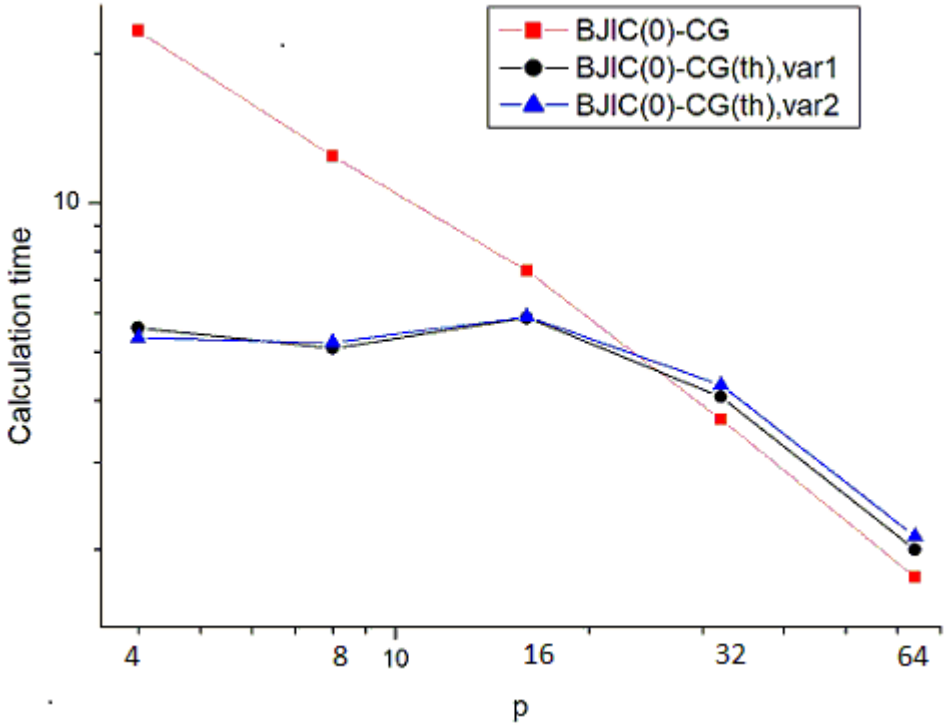

Рис. 2. Времена счета модельной задачи методом BJIC(0)-CG с использованием MPI и MPI+OpenMP технологии.

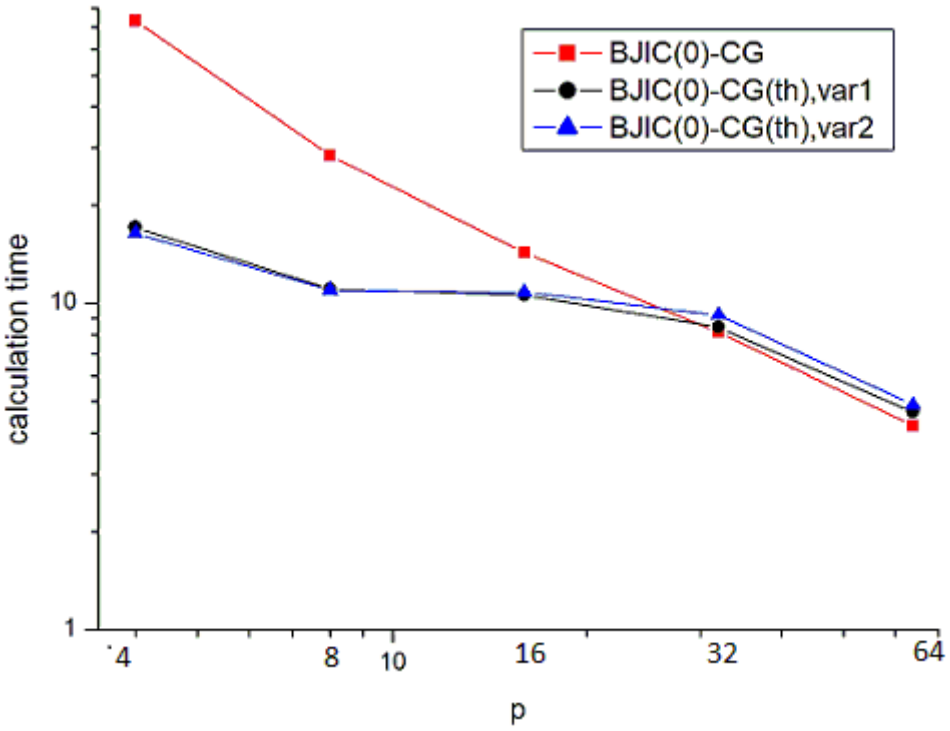

Рис. 3. Времена счета задачи с матрицей **thermal2** методом BJIC(0)-CG с использованием MPI и MPI+OpenMP технологии.

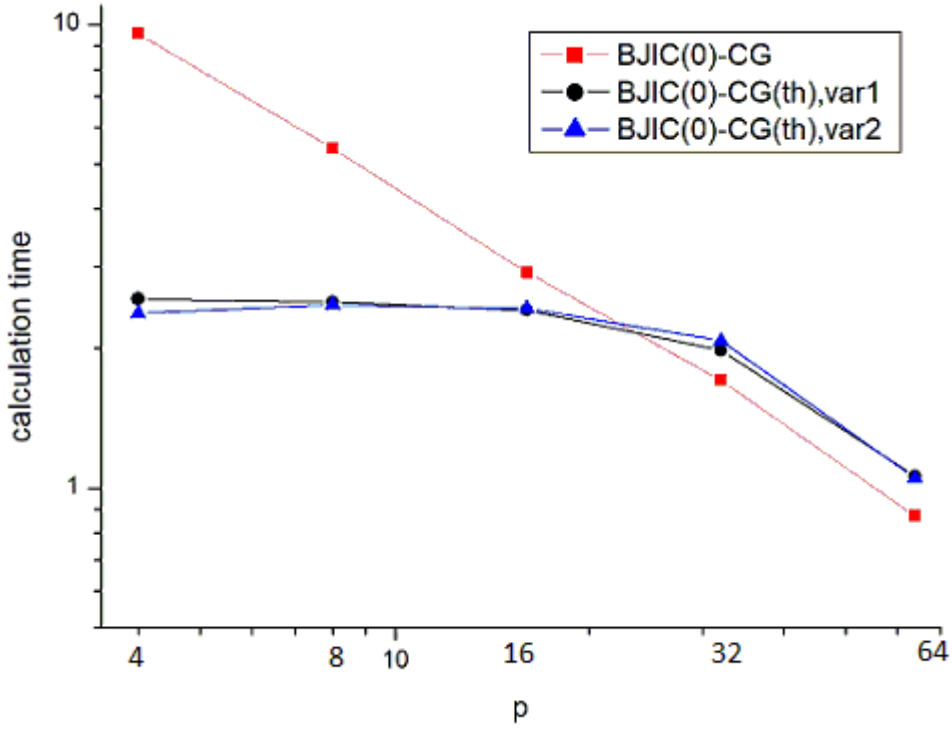

Рис. 4. Времена счета задачи с матрицей **parabolic_fem** методом BJIC(0)-CG с использованием MPI и MPI+OpenMP технологии.

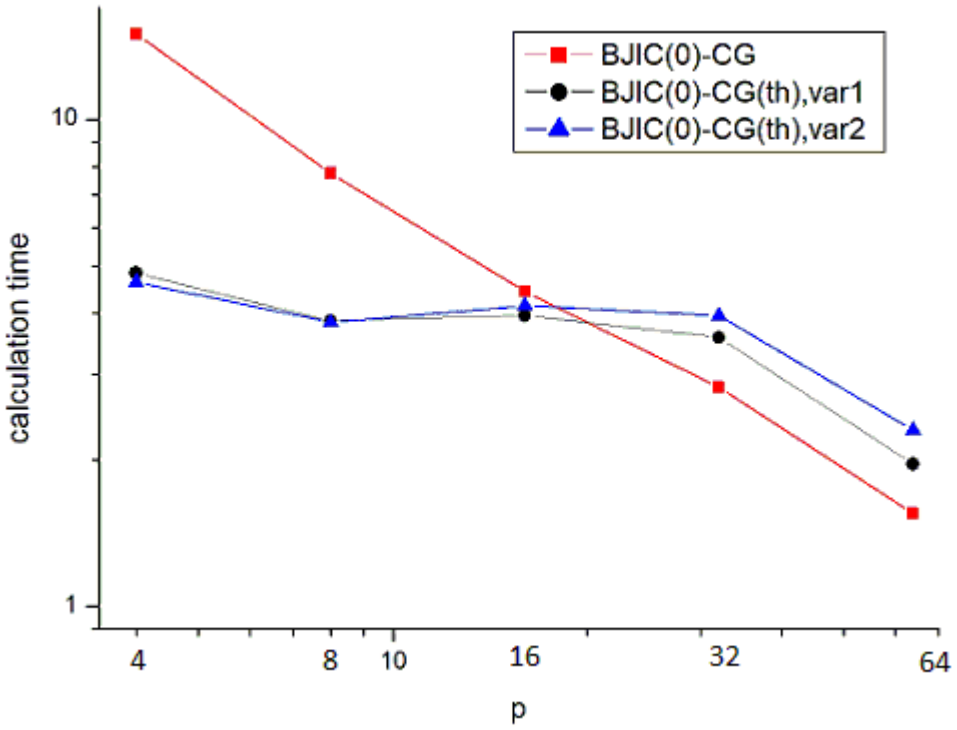

Рис. 5. Времена счета задачи с матрицей **apache2** методом BJIC(0)-CG с использованием MPI и MPI+OpenMP технологии.

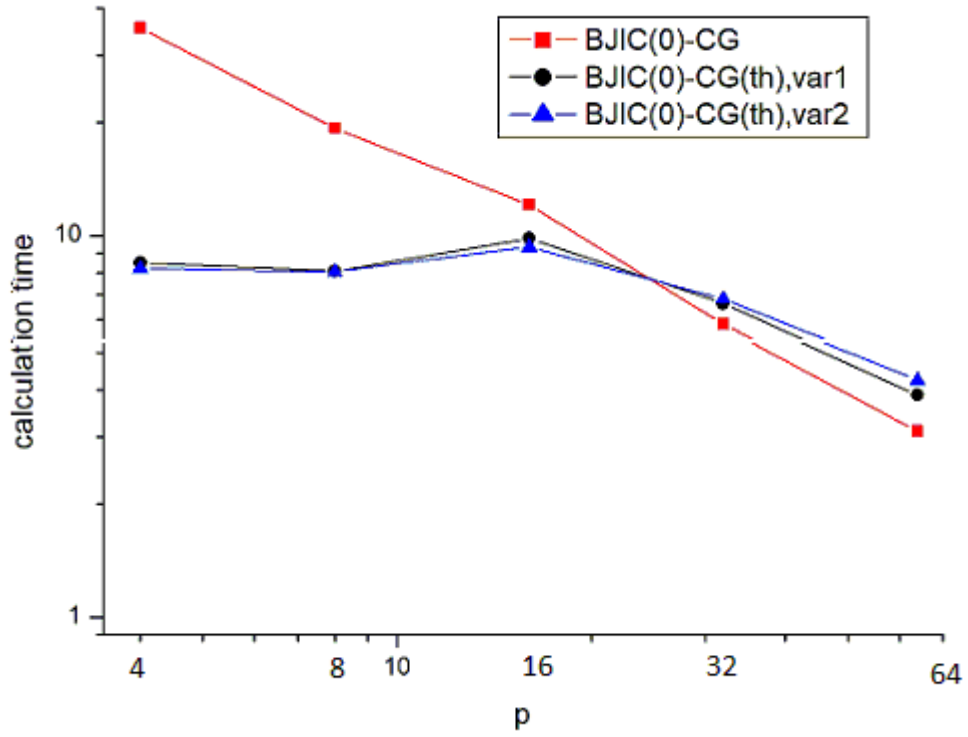

Рис. 6. Времена счета задачи с матрицей **ecology2** методом BJIC(0)-CG с использованием MPI и MPI+OpenMP технологии.

Как видно из таблиц 2-6 и рис. 2-6, при p=4, 8, 16 применение MPI+OpenMP технологии позволяет производить расчеты всех задач быстрее, чем использование только MPI. При p=4 для всех задач применение OpenMP технологии позволяет сильно ускорить вычисления, использование способа 2 применения OpenMP технологии приводит к меньшему времени вычислений, чем использование способа 1. Это связано с дополнительным временем для построения и использования переупорядочения, а также с тем, что число строк матрицы, для которых применяются OpenMP технологии, при использовании способа 2 больше, чем при использовании способа 1. При p=8 для задач 2-5 использование способа 2 позволило производить расчеты быстрее, чем использование способа 1; однако, разница во временах счета задач 1-5 невелика. При p=16 уже использование способа 1 при решении задач 1-4 приводило к меньшему времени счета по сравнению с временем счета при использовании способа 2, хотя в случае задач $1-3$ разница невелика.

Уменьшение времени счета при использовании способа 1 по сравнению со временем счета при использовании способа 2 связано с более слабым ростом числа итераций при использовании способа 1, чем при способе 2.

При p=32, 64 использование способа 1 применения OpenMP технологии приводило к заметно меньшему времени счета задач 1-5, чем использование способа 2. Однако, при p=32, 64 применение OpenMP технологии при решении задач 1-5 оказалось нецелесообразным, так как не уменьшало время счета по сравнению с временем счета с использованием только MPI.

Итак, при решении задач методом BJIC-CG(0) применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению применением только MPI в случае использования не слишком большого числа процессоров (при $p < 32$). Уменьшение эффекта от использования OpenMP технологии с увеличением числа процессоров объясняется также уменьшением числа строк матрицы, приходящихся на каждый процессор.

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 (в случае отмасштабированной матрицы это означает отсутствие предобусловливания), начиная с некоторого числа процессоров становится неэффективно по сравнению с применением только MPI [25, 26]. Так, например, как показали расчеты задачи с матрицей **5_1048576** на кластере К60, начиная с 32 процессоров, время счета методом J-CG при использовании MPI+OpenMP технологии больше времени счета при использовании только MPI. Как показали расчеты этой задачи методом J-CG на многопроцессорной вычислительной системе МВС 10П, установленной в МСЦ РАН, начиная с 32 процессоров время счета при использовании MPI+OpenMP технологии по сравнению с временем счета при использовании только MPI значительно больше [25]. Поэтому потеря эффективности использования MPI+OpenMP

технологии по сравнению с использованием только MPI при применении метода BJIC(0)-CG не выглядит столь неожиданной.

Заметим, что в ряде случаев при решении задач 1-5 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.

Заметим, что в настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, заметно больше. Следует ожидать, что потеря эффективности от использования OpenMP технологии 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 того же времени счета.

Заметим, что предложенный в настоящей работе подход применения MPI+OpenMP технологии для построения и обращения предобусловливателя IC(0) может быть обобщен на случай построения и обращения предобусловливателя ILU(0), применяемого при решении задач с несимметричной матрицей.

В дальнейшем предполагается обобщение рассматриваемых в настоящей работе подходов на случаи построения и применения предобусловливателя IC с отсечением по значению первого и второго порядка (IC1, IC2).

6. Заключение

В работе предложены два способа применения MPI+OpenMP технологии для безытерационного построения и обращения предобусловливателя блочного Якоби в сочетании с неполным разложением Холецкого без заполнения. 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). При прочих равных условиях с ростом числа вычислителей использование способа 1 становится более предпочтительным по сравнению со способом 2.

Список литературы

1. Meijering J.A., van der Vorst H.A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix // Math. Comp. 1977. V.31. P. 148-162.
2. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. М.: Мир. 1991. 365 с.
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. д-ра 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 // in 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 multiproces530sor, 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 // in: 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. 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.
29. Капорин И.Е., Милюкова О.Ю. Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных

градиентов // Препринты ИПМ им. М.В.Келдыша. 2017. № 37. 28 с. doi:10.20948/prepr-2017-37.
30. Davis T., Hu Y.F. University of Florida sparse matrix collection.//ACM Trans. on Math.~Software. 2011. V.38, N.1// http://www.cise.ufl.edu/research/sparse/matrices
31. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
32. Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т.52. № 2. С.1-26.

Оглавление