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

Препринты ИПМ • Препринт № 63 за 2022 г.

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

О.Ю. Милюкова

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2022. № 63. 32 с. https://doi.org/10.20948/prepr-2022-63
https://library.keldysh.ru/preprint.asp?id=2022-63

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 IC(0) на основе использования переупорядочения узлов сетки

Москва — 2022

Милюкова О.Ю.

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки

В работе предлагаются способы применения MPI и MPI+OpenMP технологии для построения и обращения предобусловливателя неполного треугольного разложения Холецкого без заполнения IC(0) для решения систем линейных алгебраических уравнений с произвольной симметричной положительно определенной матрицей. Способы применения MPI и MPI+OpenMP технологии основаны на использовании упорядочений узлов сетки, согласованных с разбиением области расчета. Применение OpenMP технологии при построении и обращении предобусловливателя осуществляется для большинства строк матрицы. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем IC(0) с использованием MPI и гибридной MPI+OpenMP технологии на примере модельных задач и ряда задач из коллекции разреженных матриц SuiteSparse.

Ключевые слова: неполное треугольное разложение Холецкого, переупорядочение узлов сетки, параллельное предобусловливание, метод сопряженных градиентов

Olga Yurievna Milyukova

MPI+OpenMP implementation of the conjugate gradient method with IC(0) preconditioner based on the use of mesh node reordering

The paper proposes methods for using MPI and MPI+OpenMP technologies to construct and invert the preconditioner of an incomplete triangular Cholesky decomposition without filling IC(0) for solving systems of linear algebraic equations with an arbitrary symmetric positive definite matrix. Methods of using MPI and MPI+OpenMP technologies are based on the use of grid node orderings consistent with the division of the calculation area. The use of OpenMP technology in the construction and inversion of the preconditioner is carried out for most 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 problems and the sparse matrix collections SuiteSparse are presented.

Keywords: incomplete Cholesky factorization, Domain Decomposition ordering, parallel preconditioning, conjugate gradient method

1. Введение

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

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

с симметричной положительно определенной разреженной матрицей A общего вида

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

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

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

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

Для предобусловливания будем использовать неполное треугольное разложение Холецкого без заполнения IC(0) (Incomplete Cholesky) [1]. При этом используется факторизованная матрица предобусловливания

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

где U – верхнетреугольная матрица. При использовании предобусловливания IC(0) структура разреженности матрицы U совпадает со структурой разреженности верхнетреугольной части матрицы A. Предобусловливание IC(0) имеет ограниченную область применимости, теоретическое обоснование применимости сделано для М-матриц [1] и для Н-матриц [2]. В частности метод IC(0)-CG применим в случае положительных диагональных элементов, отрицательных внедиагональных элементов и наличии диагонального преобладания у симметричной положительно определенной матрицы A.

Решение задач с матрицами большого размера требует применения параллельных компьютеров. Основная трудность распараллеливания алгоритмов построения и обращения предобусловливателя неполного треугольного разложения связана с рекурсивным характером вычислений. Одним из способов ее преодоления является использование переупорядочений узлов сетки и соответствующих перестановок строк и столбцов матрицы. При этом часто используются 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]. Однако большая часть этих работ посвящена распараллеливанию вычислений при проведении расчетов задач с использованием ортогональных сеток.

В работе [12] предлагается использовать упорядочение типа DDO для построения параллельного варианта метода неполного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG) [13]. Здесь и далее $0<\tau<<1$ – параметр отсечения. При этом производится отсечение по позициям для некоторых элементов матрицы предобусловливания.

В работах [14, 15, 16] предложен альтернативный подход, который позволяет преодолеть проблему распараллеливания рекурсивных вычислений при построении и обращении предобусловливателя при решении задачи на многопроцессорной вычислительной системе. В этих работах предложены параллелизуемые предобусловливатели, представляющие собой блочную версию предобусловливания неполного обратного треугольного разложения BIIC [17] в сочетании с неполным треугольным разложением второго порядка IC2(τ) [13] - BIIC-IC2(τ) [14], IC2S(τ) [13] - BIIC-IC2S(τ) [15] и первого порядка IC1(τ) [18] - BIIC-IC1(τ) [16]. Для построения этих предобусловливателей специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение IC2(τ), IC2S(τ) или IC1(τ). Заметим, что методы сопряженных градиентов с предобусловливателями IC2(τ), IC2S(τ), BIIC-IC2(τ) и BIIC-IC2S(τ), в отличие от методов с предобусловливателями IC(0), IC1(τ), BIIC-IC1(τ), являются безотказными для любого фиксированного значения τ.

Проблеме использования высокоуровневого параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. В работах [19 – 22] было предложено использовать несколько итераций Якоби или блочного Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения, что позволило использовать высокий уровень параллелизма. В работе [23] предложен безытерационный способ применения MPI+OpenMP технологии при обращении неявного факторизованного предобусловливателя, т. е. решении двух треугольных систем (в том числе для случая решения СЛАУ (1.1)). В работе [24] предложен новый итерационный алгоритм вычисления неполного треугольного разложения Холецкого IC(0), а также IC(1), IC(2) (неполного треугольного разложения Холецкого с заполнением 1 и 2), в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно.

Заметим, что использование явных предобусловливателей позволяет эффективно применять MPI+OpenMP технологии для параллельного решения системы линейных алгебраических уравнений (1.1) предобусловленным методом сопряженных градиентов, например [25 – 27].

В работах [28 - 30] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0) и IC1(τ). Один из них основан на переупорядочении узлов сетки типа DDO внутри каждой подобласти, другой - на уменьшении шаблона разреженности матрицы A при построении

предобусловливателя. В работах [16, 31] предложен безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC2S(τ) и BIIC-IC1(τ) на основе использования числа блоков в предобусловливателе, кратного числу используемых процессоров и числу используемых потоков. В работах [31, 32] предложен безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ), в котором для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO. При этом в работе [31], в отличие от работы [32], при построении матрицы предобусловливания для некоторых ее строк осуществлялось отсечение по позициям. При применении MPI+OpenMP технологии для построения и обращения предобусловливателя на основе использования упорядочения узлов сетки типа DDO в работах [28-32] OpenMP технологии применялись для большинства строк матрицы. В работах [16, 28-32] с помощью расчетов тестовых задач показано, что применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для умеренного числа узлов суперкомпьютерной системы (порядка нескольких десятков.

В формуле (1.1) предполагается, что матрица A уже переупорядочена, а вместо A_P стоит A ($A = A_P = P\tilde{A}P^T$), где P – матрица перестановок, а $\tilde{A}$ – матрица коэффициентов исходной задачи. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно предложенные в работе [33], являющиеся обобщением упорядочения [15]. Подход, предложенный в этих работах, позволяет одновременно произвести разбиение односвязной 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 способы применения MPI и MPI+OpenMP технологии построения и обращения предобусловливателя IC(0) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей методом сопряженных градиентов. Способ применения MPI при построении и обращении предобусловливателя основан на использовании переупорядочения строк и столбцов матрицы в соответствии с использованием упорядочения узлов сетки всей области расчета типа DDO [12]. При применении MPI+OpenMP технологии проводится дополнительное переупорядочение строк и столбцов матрицы в соответствии с дополнительным переупорядочением внутренних узлов сетки, полученных в результате переупорядочения для применения MPI. При этом тоже используется упорядочение узлов сетки типа DDO [12]. Предлагается безытерационный

способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя IC(0), при котором OpenMP технологии применяются для большинства строк матрицы. В работе проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем IC(0) с использованием MPI и MPI+OpenMP подходов на примере двух модельных задач и ряда задач из коллекции разреженных матриц SuiteSparse [34].

2. Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1.1). Алгоритм предобусловленного метода сопряженных градиентов (см., например, [35]) 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

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

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

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

где $0 < \varepsilon << 1$, B – матрица предобусловливания ($B \approx A$). Этот алгоритм использует операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений, элементарные векторные операции, а также вычисление $z_{k+1} = B^{-1} r_{k+1}$, $p_0 = w_0 = B^{-1} r_0$. Принципиальная возможность эффективной параллельной реализации всех операций, кроме вычисления $B^{-1} r_{k+1}$, $B^{-1} r_0$, не вызывает сомнений, даже при использовании большого числа процессоров и (или) применения OpenMP технологии.

3. Алгоритм построения матрицы предобусловливания IC(0)

В работе [2] приведен алгоритм построения нижнетреугольной матрицы $L = U^T$, который обычно используется для реализации построения предобусловливателя IC(0). Однако в настоящей работе будем производить построение верхнетреугольной матрицы U, так как это удобнее для реализации распараллеливания алгоритма построения матрицы предобусловливания с применением MPI. Для написания этого алгоритма будем использовать алгоритм построения матрицы U в предобусловливателе IC1(τ) для матрицы A [30]. При этом перед построением матрицы предобусловливания необходимо обеспечить, чтобы матрица A была отмасштабирована.

В настоящей работе при написании программы использовался следующий алгоритм. Напомним, что после масштабирования $a_{ii}=1$. Здесь и далее a_{ij} - элементы матрицы A, u_{ij} - элементы матрицы U, d_i - элементы вспомогательной диагональной матрицы.

Алгоритм 2

1. Инициализация вспомогательной диагональной матрицы:
   for $i=1,\ldots,n$
   $d_i := 1$
   end for
   for $i=1,\ldots,n$ (*цикл по строкам* A *для вычисления строк* U)
2. Инициализация вектора v при помощи i-ой строки A:
   for $j=i+1,\ldots,n$
   $v_j := a_{ij}$
   end for
3. Сделать поправку к вектору v:
   for $t=1,\ldots,i-1$
   for $j=i+1,\ldots,n$
   if $a_{ij} \neq 0$ then
   $v_j = v_j - u_{ti}u_{tj}$
   endif
   end for
   end for
4. Вычисление элементов u_{ii} и u_{ij} при $j>i$:
   $u_{ii} := \sqrt{d_i}$,
   for $j=i+1,\ldots,n$
   $u_{ij} = v_j / u_{ii}$
   end for
5. Выполнение поправки к вспомогательной диагональной матрице:
   for $j=i+1,\ldots,n$
   $d_j := d_j - u_{ij}^2$
   end for
   end for (*конец цикла по* i)

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

При вычислении u_{ii}- диагональных элементов матрицы U в п. 4 необходимо извлекать квадратный корень из числа, которое определяется в процессе вычисления элементов строк этой матрицы с номерами, не превосходящими i. Это выражение может оказаться отрицательным. В этом случае следует для решения СЛАУ (1.1) использовать метод сопряженных градиентов с другим предобусловливателем.

4. Алгоритм параллельной реализации построения и обращения предобусловливателя с использованием MPI

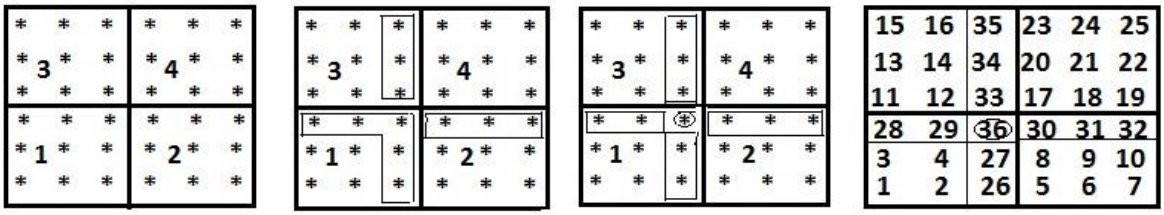
Пусть матрица A переупорядочена и отмасштабирована. Разобьем произвольную (возможно трехмерную) область расчета на p подобластей каким-либо образом, выберем некоторую нумерацию подобластей и будем использовать упорядочение узлов сетки, описанное в работе [12]. Введем множество узлов разделителей – множество узлов сетки в подобластях, у которых имеются соседи из подобластей с большим номером. Остальные узлы сетки будем называть внутренними. Узел разделителя назовем узлом разделителя первого уровня, если в шаблоне этого узла нет узлов разделителей из других подобластей с номерами, большими, чем номер рассматриваемой подобласти. Узел разделителя назовем узлом разделителя второго уровня, если в шаблоне этого узла нет узлов разделителей более высокого, чем первый, уровня, расположенных в подобластях с большим номером. Остальные узлы разделителей назовем узлами разделителей третьего уровня.

Используя определение узлов разделителей первого, второго и третьего уровня и доказательство от противного, получим, что в шаблонах узлов разделителей первого уровня могут быть только внутренние узлы из подобластей с тем же или большими номерами и узлы разделителей из той же подобласти и подобластей с меньшими номерами. В шаблонах узлов разделителей второго уровня могут быть внутренние узлы из подобластей с тем же или большими номерами, узлы разделителей первого уровня из подобластей с тем же или большими номерами, узлы разделителей более высокого уровня, чем первый, но только из рассматриваемой подобласти и подобластей с меньшими номерами. Рассуждая от противного, можно доказать, что в шаблонах узлов разделителей первого уровня не может быть узлов разделителей первого уровня из других подобластей; в шаблонах узлов разделителей второго уровня не может быть узлов разделителей второго уровня из других подобластей.

Установим следующий порядок следования узлов сетки. Сначала идут все внутренние узлы подобластей в порядке следования номеров подобластей, причем сохраняется порядок следования узлов внутри каждой подобласти, введенный ранее. Затем идут узлы разделителей первого уровня в порядке следования номеров подобластей с сохранением порядка следования узлов

внутри каждой подобласти, введенного ранее. Далее следуют узлы разделителей второго уровня в порядке следования номеров подобластей с сохранением ранее введенного порядка узлов внутри подобластей. И, наконец, идут узлы разделителей третьего уровня в порядке следования номеров подобластей и с сохранением ранее введенного порядка следования узлов внутри каждой подобласти.

Так, например, для двумерной задачи Дирихле на пятиточечном шаблоне в квадратной области расчета на рис. 1. а) приведен пример разбиения области расчета, на рис. 1 б) выделены все узлы разделителей, на рис. 1. в) узлы разделителей первого уровня обведены прямоугольными контурами, узел разделителей второго уровня обведен в кружок, узлы разделителей третьего уровня отсутствуют. На рис. 1. г) указан новый порядок следования узлов сетки для рассматриваемой двумерной задачи.



а) б) в) г)

Рис. 1. Разбиение двумерной области расчета на 4 подобласти, расположение внутренних узлов подобластей и узлов разделителей и новый порядок следования узлов сетки для рассматриваемой двумерной задачи.

На рис. 2 приведен вид структуры разреженности матрицы *A*, полученной после перестановки строк и столбцов в результате переупорядочения в случае разбиения области расчета на 4 подобласти. Использованы обозначения: *l* – число всех внутренних узлов сетки из всех подобластей, *l*1 - число всех внутренних узлов сетки из всех подобластей и всех узлов разделителей первого уровня из всех подобластей, *l*2 - число всех внутренних узлов сетки из всех подобластей и всех узлов разделителей первого и второго уровня из всех подобластей. Строки матрицы, содержащие блочно-диагональные части A_{11}^{s}, соответствуют внутренним узлам сетки и хранятся в процессоре с номером *s*. Строки матрицы, содержащие блочно-диагональные части A_{22}^{s} и A_{33}^{s}, соответствуют узлам сетки соответственно на разделителях первого и второго уровня из подобласти с номером *s* и хранятся в процессоре с номером *s*. Строки матрицы, соответствующие блочно-диагональной части A_{44}, соответствуют узлам сетки на разделителях третьего уровня и хранятся в соответствующих процессорах.

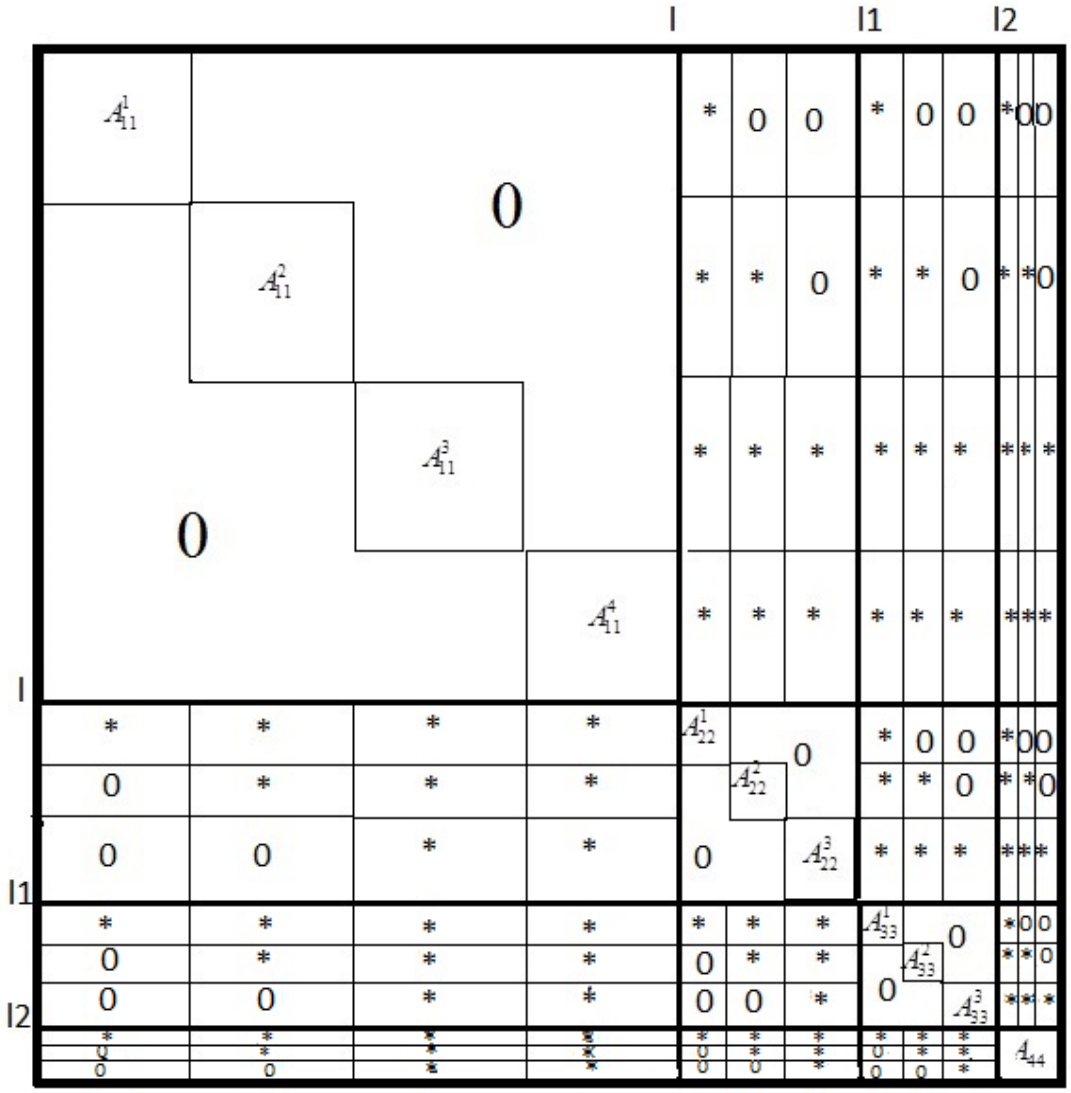


Рис. 2. Вид структуры разреженности матрицы A, полученной после перестановки строк и столбцов в результате переупорядочения.

Как указывалось во введении, структура разреженности матрицы U в предобусловливателе IC(0) совпадает со структурой разреженности верхнетреугольной части матрицы A. На рис. 3 приведен вид структуры разреженности матрицы U в предобусловливателе IC(0) для матрицы A, структура разреженности которой изображена на рис. 1, при условии, что блок A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков.

Перед вычислением матрицы U необходимо произвести переупорядочение строк и столбцов матрицы A в соответствии с новым упорядочением узлов сетки. При этом можно ограничиться созданием только верхнетреугольной части переупорядоченной матрицы. Определение элементов матрицы U

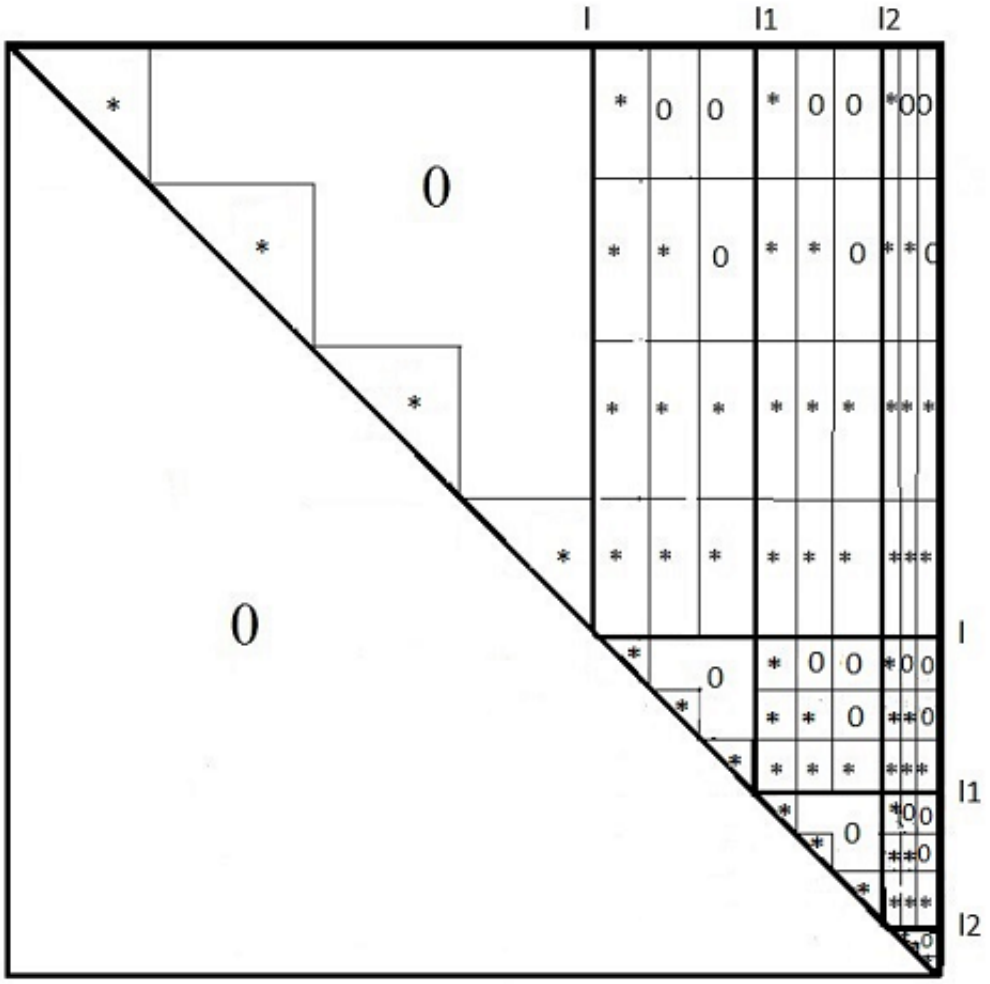


Рис. 3. Структура разреженности матрицы *U* в предобусловливателе IC(0) для матрицы *A*, структура разреженности которой изображена на рис. 2.

начинается с вычисления ее элементов во внутренних узлах всех подобластей. Заметим, что в каждом процессоре с номером *s* (*s*=1, 2, …, *p*), производящем расчеты в своей подобласти, при вычислении элементов u_{ij} для *j*, соответствующих узлам сетки из других подобластей, для вычисления пункте 3 алгоритма 2 не требуется знания u_{tj}. Это связано с тем, что $u_{ti} = 0$ для $t \le i_2(s-1)$, где $i_2(s-1)$- номер последнего внутреннего узла в подобласти, предшествующей рассматриваемой подобласти с номером *s* (см. рис. 3). Поэтому вычисление элементов матрицы *U* в строках, соответствующих внутренним узлам подобластей, может происходить во всех процессорах одновременно и независимо. Используется алгоритм, аналогичный алгоритму 2, в котором циклы по *i* и по *t* происходят по внутренним узлам соответствующих подобластей. Перед началом вычисления матрицы *U* следует положить $d_i := 0$ для *i*=1,2, …,*n*.

Перед переходом к вычислениям элементов матрицы U в строках, соответствующих узлам разделителей первого уровня, следует вычислить вспомогательную матрицу V, элементы которой V_{ij} определяются с помощью алгоритма 3.

Алгоритм 3.

1.for $s=1,\ldots,p$

Вычислить V_{ij}^s для $i=l+1,\ldots,n, j=i+1,\ldots,n$:

```
for i=l+1,...,n
  for j=i+1,...,n
    V_ij^s := 0
    if (a_ij ≠ 0) then
      for t = i_1(s), i_2(s)
        V_ij^s = V_ij^s + u_ti u_tj
      end for
    end if
  end for
 end for
end for (конец цикло по s)
```

2. Вычислить V_{ij} для $i=l+1,\ldots,n, j=i+1,\ldots,n$:

```
for i=l+1,...,n,
  for j=i+1,...,n
  V_ij = Σ_{s=1}^{p} V_ij^s
  end for
end for
```

Здесь и далее $i_1(s)$, $i_2(s)$ - номера первой и последней строк, соответствующих внутренним узлам в подобласти с номером s. При параллельной реализации пункта 1 алгоритма 3 каждый процессор производит вычисления в своей подобласти. Для реализации вычислений пункте 2 в каждом процессоре с номером s производится сборка строк матриц V^s с номерами $i \in \Psi_s$, где Ψ_s - множество узлов разделителей в подобласти с номером s, и вычисление $V_{ij} = \sum_{s=1}^{p} V_{ij}^s$ для строк с номерами $i \in \Psi_s$. В результате в каждом процессоре находятся те строки матрицы V, которые необходимы для дальнейших расчетов

в этом процессоре. Для пересылок используются операции MPI_Recv и MPI_ Send.

Кроме того, производится суммирование по всем процессорам вычисленных там (см. п. 5 алгоритма 2) неокончательных значений d_j для номеров j, соответствующих номерам узлов разделителей. При этом используется операция MPI_AllReduce.

Благодаря тому, что структура разреженности матрицы U совпадает со структурой разреженности верхнетреугольной части матрицы A, вычисление элементов матрицы U в строках, соответствующих узлам разделителей первого уровня, происходит во всех процессорах одновременно. Используется алгоритм 4. Напомним, что перед началом вычисления элементов матрицы U всем d_i было присвоено значение 0.

Алгоритм 4

for s=1, …,p

1. Инициализация вспомогательной диагональной матрицы:

for $i = \bar{i}_1(s),...,\bar{i}_2(s)$

$d_i := d_i + 1$

end for

for $i = \bar{i}_1(s),...,\bar{i}_2(s)$

2. Инициализация вектора v при помощи i-й строки A и i-о строки V:

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

$v_j := a_{ij} + V_{ij}$

end for

3. Сделать поправку к вектору v:

for $t=\bar{i}_1(s),...,i-1$

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

if $a_{ij} \neq 0$ then

$v_j = v_j - u_{ti}u_{tj}$

endif

end for

end for

4. Вычисление элементов u_{ii} и u_{ij} при $j>i$:

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

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

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

end for

5. Выполнение поправки к вспомогательной диагональной матрице:

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

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

end for

end for (*конец цикла по* i)

end for (*конец цикла по* s)

В алгоритме 4 $\bar{i}_1(s), \bar{i}_2(s)$ - номера первой и последней строк на разделителях первого уровня в подобласти с номером s.

Перед переходом к вычислениям элементов матрицы U в строках, соответствующих узлам разделителей второго уровня, следует вычислить элементы вспомогательной матрицы $\bar{V}$, что осуществляется с использованием алгоритма, аналогичного алгоритму 3. Кроме того, нужно произвести суммирование по всем процессорам вычисленных там значений Δd_j (изменений значений d_j, полученных на этапе 5 алгоритма 4) для j, соответствующих номерам узлов разделителей второго и третьего уровня, и добавить Δd_j к вычисленным ранее неокончательным значениям d_j. При этом в настоящей работе используется операция MPI_AllReduce.

Вычисление элементов матрицы U в строках, соответствующих узлам разделителей второго уровня, производится аналогично вычислению элементов этой матрицы в строках, соответствующих узлам разделителей первого уровня, причем в п.2 алгоритма 4 добавляется еще одно слагаемое: $\bar{V}_{ij}$, где $\bar{V}_{ij}$ - элементы матрицы $\bar{V}$. Перед вычислением элементов матрицы U в строках, соответствующих узлам разделителей третьего уровня, следует вычислить элементы матрицы $\tilde{V}$. Это делается аналогично вычислению элементов матриц V, $\bar{V}$. Кроме того, нужно произвести суммирование по всем процессорам вычисленных там значений Δd_j для номеров j, соответствующих узлам разделителей третьего уровня, и добавить Δd_j к вычисленным ранее неокончательным значениям d_j. При этом в настоящей работе используется операция MPI_AllReduce.

Если блок A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков, то вычисление элементов матрицы U в строках, соответствующих узлам разделителей третьего уровня, производится аналогично. Если подматрица A_{44} не является блочно диагональной с нулевыми элементами вне блоков, то этот этап вычислений можно не распараллеливать, а производить вычисление элементов матрицы U в строках,

соответствующих узлам разделителей третьего уровня, во всех процессорах одновременно по одним и тем же формулам. Потом можно произвести отсечение некоторых элементов матрицы U по позициям или при обращении матрицы предобусловливания вычисления для узлов разделителей третьего уровня не распараллеливать. Так как узлов разделителей третьего уровня относительно мало, эти этапы вычислений должны происходить достаточно быстро.

Заметим, что если матрица A_{44} не имеет блочно-диагональную структуру, то для обеспечения лучшей параллелизуемости можно производить отсечение на разделителях третьего уровня по позициям не только как в предобусловливате IC(0), а еще дополнительно, как бы заменив при этом некоторые значения a_{ij} нулем.

Перед обращением матрицы предобусловливания необходимо произвести переупорядочение элементов вектора r^{k-1}, где k – номер итерации в предобусловленном методе сопряженных градиентов. Обращение матрицы предобусловливания состоит из двух этапов: $\hat{w}^k = U^{-T} r^{k-1}$ и $w^k = U^{-1}\hat{w}^k$. На первом этапе будем использовать алгоритм обращения транспонированной матрицы, предложенный первоначально в работе [15]. Будем вычислять элементы вектора $z = D_U \hat{w}^k$, где D_U – диагональгая часть матрицы U, а затем по элементам вектора z вычислять элементы вектора $\hat{w}^k$. Алгоритм параллельной реализации вычисления элементов векторов z и $\hat{w}^k$, соответствующих внутренним узлам сетки подобластей, а также слагаемых элементов вектора z, соответствующих узлам разделителей, имеет следующий вид:

Алгоритм 5.

1. Вычислить первоначальные значения z_j:

for $j=1,n$

$z_j = r_j^{k-1}$

end for

for s=1,2, …,p

2. Вычислить z_j для $j=i_1(s)$, …, $i_2(s)$, и слагаемые элементов z_j для $j>l$:

for $i = i_1(s),...,i_2(s)$

$save = z_i / u_{ii}$

for $j=i+1,n$

$z_j = z_j - u_{ij} \times save$

end for

end for

3. Вычислить значение $\hat{w}_i^k$ для $i = i_1(s),...,i_2(s)$:

for $i = i = i_1(s),...,i_2(s)$

$\hat{w}_i^k = z_i / u_{ii}$

end for

end for (*конец цикла по s*).

В алгоритме 5 i и j – глобальные индексы.

После этого осуществляется пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, расположенных на разделителях, в процессоры, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах. Для пересылок используются операции MPI_Recv и MPI_ Send.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей первого уровня с использованием алгоритма, аналогичного алгоритму 5. Вычисления на этом этапе происходят во всех процессорах одновременно и независимо. Затем осуществляется пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах. Для пересылок используются операции MPI_Recv и MPI_ Send.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей второго уровня с использованием алгоритма, аналогичного алгоритму 5. Вычисления на этом этапе происходят во всех процессорах одновременно и независимо. Затем осуществляется пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, в процессоры, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей третьего уровня. Если A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков, то вычисления на этом этапе происходят во всех процессорах одновременно и независимо.

На этапе 2 обращения матрицы предобусловливани сначала с использованием матрицы U происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем второго, первого уровня, а далее во внутренних узлах подобластей. При вычислении w_i^k в узлах разделителей каждого уровня и во внутренних узлах все процессоры работают одновременно. После расчета на разделителях каждого уровня происходит пересылка найденных значений w_i^k этого уровня в процессоры, в которых эти значения нужны для дальнейшего расчета. Для пересылок используются операции MPI_Recv и MPI_ Send.

После завершения вычисления вектора w^k следует произвести переупорядочение его элементов.

5. Алгоритм параллельной реализации построения и обращения предобусловливателя с использованием MPI+OpenMP

Рассмотрим способ применения MPI+OpenMP технологии при построении и обращении предобусловливателя IC(0). Сначала произведем разбиение всей области расчета на p подобластей и переупорядочение всех узлов сетки типа DDO, как описано в предыдущем разделе. Затем в каждой получившейся подобласти с номером s разобьём внутреннюю часть подобласти, состоящую из внутренних узлов, полученных при переупорядочении, произведенном для MPI реализации, на m подобластей, где m – предполагаемое число используемых потоков, с приблизительно равным числом узлов сетки. В настоящей работе разбиение осуществлялось в порядке следования внутренних узлов подобластей, установленном ранее, следующим образом. Пусть $\hat{l}_s = [n_s / m]$, первые m-1 «внутренних» подобластей содержат $\hat{l}_s$ узлов внутренних узлов, последняя «внутренняя» подобласть содержит $n_s - (m-1)\hat{l}_s$ внутренних узлов. Здесь и далее кавычки перед и после слова «внутренняя» означают, что речь идет о разбиении для мелкозернистого распараллеливания или распараллеливания на потоки.

Заметим, что можно производить разбиение всей области расчета сразу на pm подобластей каким-либо способом, и обеспечить, чтобы каждый процессор производил вычисления в соответствующих m подряд идущих подобластях как, например, в работе [16].

Произведем переупорядочение внутренних узлов сетки в подобластях, полученных при разбиении всей области расчета на p подобластей, где p – число процессоров ($p \neq 1$). Для переупорядочения внутренних узлов в подобластях будем использовать упорядочение типа DDO [12], которое было

подробно описано в предыдущем разделе и используется для переупорядочения узлов сетки всей области расчета для крупнозернистого распараллеливания. В результате будут определены «внутренние» узлы и узлы «разделителей». Здесь и далее кавычки перед и после слов «внутренние» и «разделители» означают, что речь идет о внутренних узлах и узлах разделителей при упорядочении для мелкозернистого распараллеливания. Заметим, что при этом к «внутренним» узлам будут относиться узлы, у которых имеются соседи, являющиеся разделителями для переупорядочения для крупнозернистого распараллеливания, но при этом нет соседей, из «подобластей» с большим номером при построении переупорядочения для мелкозернистого распараллеливания. Множество узлов «разделителей» разобьем на множество узлов «разделителей» первого, второго и третьего уровней аналогично тому, как описано в предыдущем разделе при построении переупорядочения для крупнозернистого распараллеливания.

Определим $\bar{l}_s$ – минимальное число «внутренних» узлов при разбиении множества внутренних узлов (необязательно таком, как описано выше) из подобласти с номером s на «внутренние» подобласти. Предполагается, что $\bar{l}_s \neq 0$. Обозначим $M1 = m\bar{l}_s$. Установим следующий порядок следования узлов сетки подобласти с номером s (s= 1, …, p). Сначала идут $\bar{l}_s$ «внутренних» узлов первой «внутренней» подобласти, затем $\bar{l}_s$ «внутренних» узлов второй «внутренней» подобласти и т. д., наконец, $\bar{l}_s$ «внутренних» узлов m-й «внутренней» подобласти. Затем идут оставшиеся «внутренние» узлы «внутренних» подобластей в порядке следования номеров «внутренних» подобластей и в порядке следования узлов внутри «внутренних» подобластей, введенном ранее. Затем идут узлы «разделителей» первого уровня, полученные при упорядочении множества внутренних узлов подобласти с номером s, затем узлы «разделителей» второго уровня при этом упорядочении, а затем узлы «разделителей» третьего уровня. При этом для каждого уровня «разделителей» узлы следуют с сохранением порядка следования «внутренних» подобластей и порядка следования узлов внутри подобласти, введенного ранее.

Затем идут узлы разделителей первого, второго и третьего уровней при упорядочении для крупнозернистого распараллеливания. При этом сохраняется порядок следования узлов на разделителях, введенный для крупнозернистого распараллеливания.

Таким образом, при использовании MPI+OpenMP технологии производится дополнительное переупорядочение только узлов сетки, являющихся внутренними при упорядочении для использования только MPI.

Заметим, что, как показали расчеты тестовых задач методом сопряженных градиентов с предобусловливанием блочного Якоби в сочетании с IC1(τ) (BJIC1(τ)) [30], применение OpenMP технологии для вычисления верхнетреугольного множителя предобусловливателя и обращения матрицы предобусловливателя BJIC1(τ) для всех «внутренних» узлов «внутренних»

подобластей в подавляющем большинстве случаев нецелесообразно. Как показали расчеты тестовых задач, нецелесообразным является также использование OpenMP технологий при вычислении верхнетреугольного множителя предобусловливателя и обращении матрицы предобусловливателя BJIC1(τ) для строк, соответствующих узлам «разделителей». Заметим, что можно использовать другое упорядочение узлов сетки типа DDO.

При использовании MPI+OpenMP применение OpenMP технологии в каждом процессоре с номером s (s=1,…,p) осуществляется при построении первых $M1$ строк верхнетреугольного множителя матрицы предобусловливания IC(0) при новом упорядочении узлов сетки, используется цикл по $k2$=1, …, m, внутри которого осуществляется вычисление элементов строк искомой матрицы с локальными номерами 1, …, $M1$. При этом все рекурсивные вычисления происходят внутри потоков. Для выполнения цикла по $k2$ в настоящей работе использовалась директива **do** с опцией **schedule static**. Затем без использования OpenMP технологии производится вычисление элементов оставшихся строк этой матрицы с использованием только MPI. Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства строк верхнетреугольного множителя матрицы предобусловливания IC(0) вычисление его элементов происходит с использованием OpenMP технологии.

Перед началом итерационного процесса в каждом процессоре с номером s строится матрица $L1_S$ размера $il0(s)\times il0(s)$, транспонированная к матрице $U1_S$, где матрица $U1_S$ содержит первые $il0(s)$ строк и первые $il0(s)$ столбцов матрицы U_S. Здесь U_S – часть матрицы U, хранящаяся в процессоре с номером s, $il0(s)$ – количество внутренних узлов в подобласти с номером s. На рис. 4 приведен вид структуры разреженности матрицы $L1_S$ для случая m=4 и произвольного разбиения множества внутренних узлов в каждой подобласти с минимальным числом «внутренних» узлов в четвертой «внутренней» подобласти. На рис. 4 $\bar{l}$ – число всех «внутренних» узлов во всех m «внутренних» подобластях, $\bar{l}1$ – число всех «внутренних» узлов во всех m «внутренних» подобластях в сумме с числом всех узлов «разделителей» первого уровня, $\bar{l}2$ – число всех «внутренних» узлов во всех m «внутренних» подобластях в сумме с числом всех «разделителей» первого и второго уровня. Создадим также матрицы $U2_S$, элементы которых совпадают с элементами из первые $il0(s)$ строк матрицы U_S с номерами столбцов с индексами, соответствующими узлам разделителей при упорядочении для крупнозернистого распараллеливания.

При обращении предобусловливателя, состоящем из двух этапов, указанных выше, перед началом вычислений в каждом процессоре с номером s производится переупорядочение элементов вектора r_S^{k-1} – части вектора r^{k-1},

хранящейся в процессоре с номером s. Первый этап обращения матрицы предобусловливания ($\hat{w}^k = U^{-T} r^{k-1}$) начинается с обращения нижнетреугольных матриц $L1_S$. В каждом процессоре с применением OpenMP технологии вычисляются $M1$ элементов $\hat{w}_i^k$ с локальными индексами i=1, … $M1 = m\bar{l}_s$ (при новом упорядочении), в настоящей работе использовалась директива **do** с опцией **schedule static**. Далее без применения OpenMP технологии производится вычисление элементов $\hat{w}_i^k$ с локальными номерами i=$M1$+1, …, $il0$(s). Затем с использованием матрицы $U2_S$ вычисляются слагаемые элементов $z_j = \hat{w}_j U_{jj}$ вектора z для значений j, соответствующих узлам разделителей. При этом используется алгоритм, аналогичный алгоритму 5.

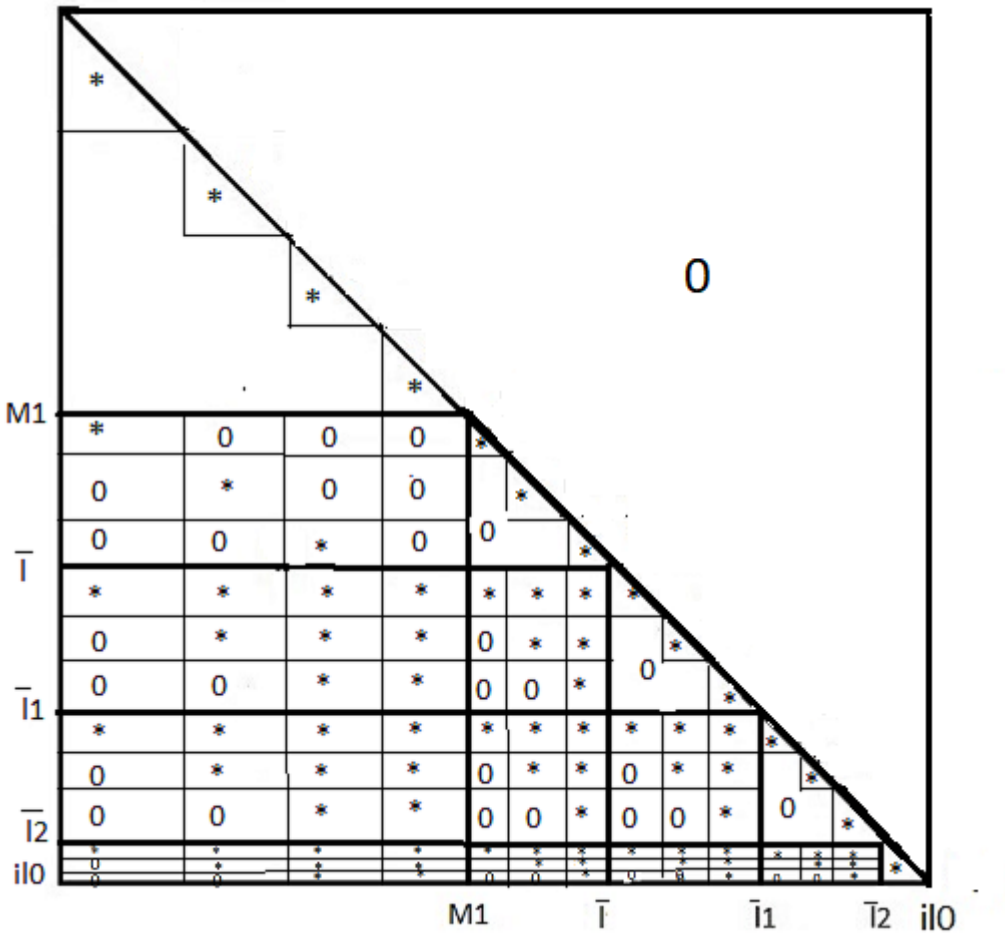


Рис. 4. Вид структуры разреженности матрицы $L1_S$ для случая m=4 и минимального числа «внутренних» узлов в четвертой «внутренней подобласти».

После этого осуществляется пересылка значений слагаемых z_j для индексов j, соответствующих узлам сетки из соседних подобластей на разделителях, в процессоры, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах. Для пересылок используются операции MPI_Recv и MPI_ Send.

Затем происходит вычисление значений $z_j = U_{jj}\hat{w}_j^k$ и для индексов j, соответствующих узлам разделителей всех уровней при упорядочении, введенном для крупнозернистого распараллеливания, и вычисление $\hat{w}_i^k$ в узлах разделителей всех уровней. Это осуществляется так же, как в случае применения только MPI, и описано в предыдущем разделе.

На этапе обращении верхнетреугольных матриц вычисления происходят в обратном порядке с использованием матрицы U_S. Сначала происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем второго и первого уровня, что осуществляется так же, как при использовании только MPI. После расчета на разделителях каждого уровня происходит пересылка найденных значений w_i^k этого уровня в процессоры, в которых эти значения нужны для дальнейшего расчета. Для пересылок используются операции MPI_Recv и MPI_ Send. Потом происходит вычисление w_i^k во внутренних узлах сетки (полученных при упорядочении для MPI распараллеливания). При этом элементы искомого вектора с локальными номерами $i=M1, \ldots,1$ (при новом упорядочении) вычисляются с использованием OpenMP технологии. В настоящей работе при этом использовалась директива **do** с опцией **schedule static**. После вычисления вектора w^k при новом упорядочении следует вернуться к первоначальному упорядочению для элементов этого вектора.

Заметим, что при применении OpenMP технологии не имеет значения вид структуры разреженности блока матрицы $U1_S$ для строк и столбцов, соответствующих узлам «разделителей» третьего уровня, так как на этом этапе построения (и обращения) матрицы предобусловливания OpenMP технологии не используются.

Заметим, что кроме описанного выше способа применения MPI+OpenMP технологии для распараллеливания вычислений при построении и обращении матрицы предобусловливания можно аналогичным образом применять OpenMP технологии для распараллеливания вычислений при построении и обращении матрицы предобусловливателя на разделителях первого уровня, полученных при переупорядочении для применения крупнозернистого распараллеливания. Но вряд ли будет целесообразным применять OpenMP технологии для

распараллеливания вычислений при построении и обращении матрицы предобусловливания для элементов, соответствующих узлам разделителей второго и третьего уровня.

Вычисление элементов вектора $q_k = Ap_k$, где k – номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, если матрица A хранится в памяти в распределенном CRS-формате, содержит верхний и нижний треугольники, с использованием MPI+OpenMP хорошо изучено, подробно описано, например, в работах [26, 30]. При применении MPI + OpenMP подхода для умножения матрицы на вектор в настоящей работе использовалась директива **do** с опцией **schedule static.** MPI+OpenMP реализация вычислений векторных операций и скалярных произведений в алгоритме 1 тоже хорошо известна. При применении MPI+OpenMP подхода для вычислений векторных операций и частичных сумм в скалярных произведениях в настоящей работе использовалась директива **do** с опцией **schedule static.**

6. Результаты расчетов

Программы, реализующие применение метода IC(0)-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии. Расчеты проводились на многопроцессорном вычислительном кластере К60, установленном в ЦКП ИПМ им. М.В. Келдыша РАН. Тестирование и сравнение методов производилось с помощью решения модельных задачи 1 и 2. В качестве модельной задачи 1 рассматривалась разностная задача Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке, причем n=262144. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы **5_262144_2D**). Модельная задача 2 – разностная задача Дирихле для уравнения Пуассона в единичном кубе на равномерной ортогональной сетке, причем n=262144. Использовалась стандартная 7-точечная аппроксимация лапласиана (имя матрицы **7_262144_3D**). Для тестирования рассматриваемых параллельных методов использовались также две матрицы из коллекции разреженных матриц SuiteSparse [34]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

apache1 – трехмерная конечно-разностная схема;

g2_circuit – моделирование интегральных схем.

В таблице 1 приведены некоторые свойства этих матриц, причем значения $Cond(A_0)$, где $A_0 = (D_A)^{-1/2} A (D_A)^{-1/2}$ – матрица системы уравнений после масштабирования, взяты из работы [36], Id – количество строк без диагонального преобладания, Ip – количество положительных внедиагональных элементов, NZA – число ненулевых элементов матрицы A, $nz_{\min}$, $nz_{\max}$ –

минимальное и максимальное числа ненулевых элементов в строках матрицы *A*, N – число неизвестных в системе уравнений (1.1).

Таблица 1

Свойства некоторых матриц из коллекции разреженных матриц SuiteSparse

Матрица	N	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_S)$
apache1	80800	542184	0	0	4	7	0.10+7
g2_circuit	150102	726674	0	28000	2	6	0.23+6

Модельные задачи 1 и 2 далее будем называть задачами 1 и 2. Задачи с матрицами **apache1** и **g2_circuit** будем называть соответственно задачей 3, задачей 4.

Решалось уравнение $Ax = b$, где A – матрица, полученная после переупорядочения, связанного с разбиением на подобласти для использования MPI, и масштабирования исходной матрицы, с единичной правой частью ($b_i \equiv 1$), начальное приближение $x_0 = 0$. Счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета на p подобластей использовался алгоритм [33] (разбиение 1). Для изучения влияния разбиения области расчета на скорость сходимости итерационного процесса и эффективность применения MPI и MPI+OpenMP технологии разбиение квадратной области расчета на 4 подобласти при решении задачи 1 производилось также с помощью деления области расчета в обоих пространственных направлениях на две одинаковые части (разбиение 2). При этом использовалось лексикографическое упорядочение.

В таблицах 2 – 5 приведены числа итераций и времена счета методом IC(0)-CG тестовых задач 1 – 4 при использовании для параллельной реализации разбиения 1 и применении MPI и MPI+OpenMP технологии. Под временем вычислений в таблицах 2 – 6 подразумевается время счета в секундах итерационного процесса в сумме с временем вычисления предобусловливателя.

При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 10, 12 и 16 нитей. В таблицах 2 – 5 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей, которые приведены в таблицах в скобках. В таблицах 2 – 5 приведены курсивом коэффициенты μ ускорения счета задач благодаря использованию OpenMP технологии, а также коэффициенты η ускорения счета по сравнению со счетом с использованием только MPI на двух процессорах.

Заметим, что неожиданно большое число итераций метода IC(0)-CG при решении задачи с матрицей **apache1** на двух процессорах связано с особенностями разбиения области расчета на подобласти методом [33].

Таблица 2

Числа итераций и времена счета методом IC(0)-CG задачи с матрицей **5_262144_2D** на *p* процессорах без применения и с применением OpenMP технологии

p	2	3	5	8
MPI η	456, 1.14	517, 0.99 *1.15*	569, 0.7 *1.62*	578, 0.46 *2.47*
MPI+OpenMP μ, η	569, 0.55(12) *2.07, 2.07*	612, 0.55(8) *1.8, 2.07*	593, 0.46(4) *1.52, 2.48*	609, 0.34(3) *1.35, 3.35*

Таблица 3

Числа итераций и времена счета методом IC(0)-CG задачи с матрицей **5_262144_3D** на *p* процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI η	88, 0.34	90, 0.34 *1.0*	91, 0.29 *1.17*	90, 0.20 *1.7*	91, 0.14 *2.42*
MPI+OpenMP μ, η	104, 0.2(8) *1.7, 1.7*	104, 0.21(6) *1.61, 1.61*	101, 0.2(4) 109, 0.2(6) *1.45, 1.7*	100, 0.16(3) 103, 0.16(4) *1.25, 2.12*	100,0.13(3) 104,0.13(4) *1.07, 2.61*

Таблица 4

Числа итераций и времена счета методом IC(0)-CG задачи с матрицей **apache1** на *p* процессорах без применения и с применением OpenMP технологии

p	2	3	5	8
MPI η	1395, 1.36	455 ,0.36 *3.78*	534, 0.27 *5.03*	639, 0.21 *6.47*
MPI+OpenMP μ, η	1448, 0.64(12) *2.12, 2.12*	706, 0.32(4) *1.12, 4.25*	756,0.28(3) *0.96, 4.85*	886,0.23(3) *0.91, 5.91*

Таблица 5

Числа итераций и времена счета методом IC(0)-CG задачи с матрицей **g2_circuit** на *p* процессорах без применения и с применением OpenMP технологии

P	2	3	5	8
MPI η	747, 1.28	756, 0.97 *1.32*	757, 0.62 *2.06*	758, 0.44 *2.29*
MPI+OpenMP μ, η	787, 0.65(6) *1.96, 1.96*	817, 0.57(4) *1.7, 2.24*	825, 0.48(3) *1.29, 2.66*	821, 0.37(3) *1.19, 3.46*

Эффективность (в процентах) использования MPI по сравнению со счетом на 2 процессорах с использованием MPI, вычисляемая по формуле $e = 2/(p\eta)\times 100\%$, на 8 процессорах в задачах 1, 3, 4 составляла более 57%, в задаче 3 на 6 процессорах e=81%. При решении всех тестовых задач на 5 процессорах эффективность превышала 60%.

На рисунках 5 – 8 приведены графики зависимости времени счета тестовых задач 1 – 4 методом IC(0)-CG от числа процессоров p в логарифмическом масштабе с использованием только MPI и разбиения на p подобластей с помощью алгоритма [33] (линии черного цвета), а также с использованием MPI+OpenMP технологии (линии синего цвета).

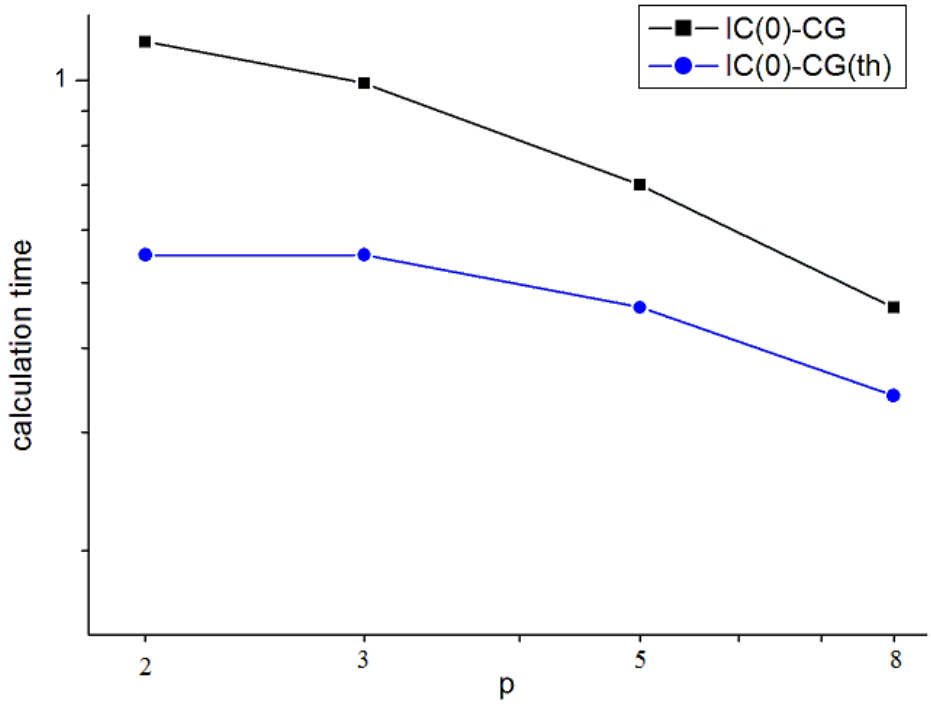


Рис. 5. Времена счета задачи с матрицей **5_262144_2D** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

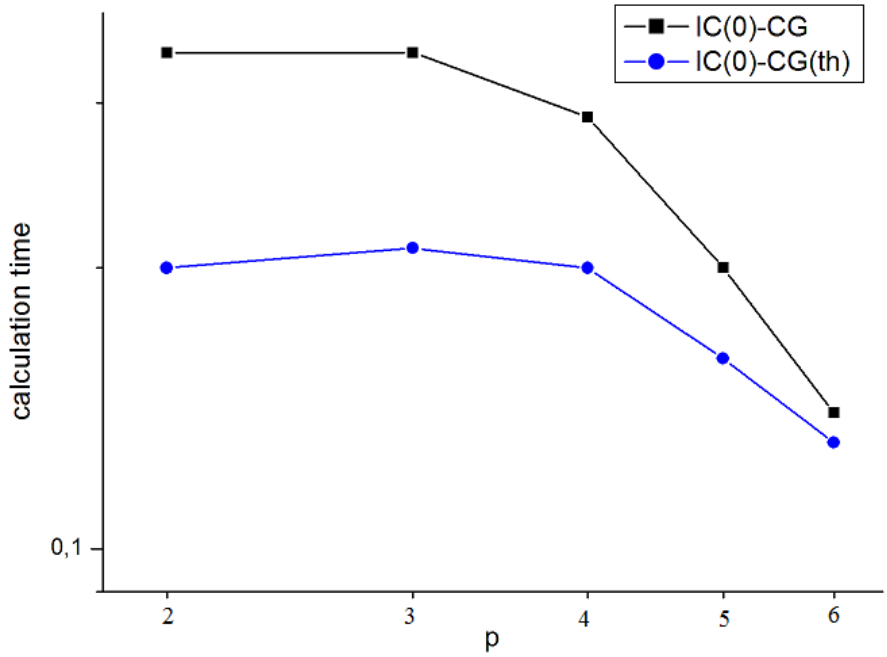


Рис. 6. Времена счета задачи с матрицей **7_262144_3D** методом IC1(0)-CG с использованием MPI и MPI+OpenMP технологии.

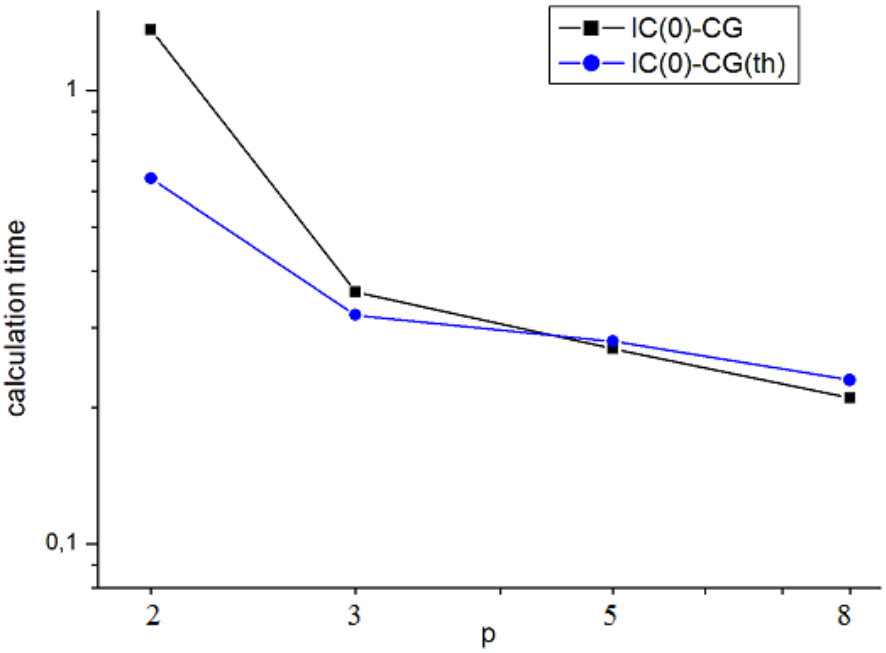


Рис. 7. Времена счета задачи с матрицей **apache1** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

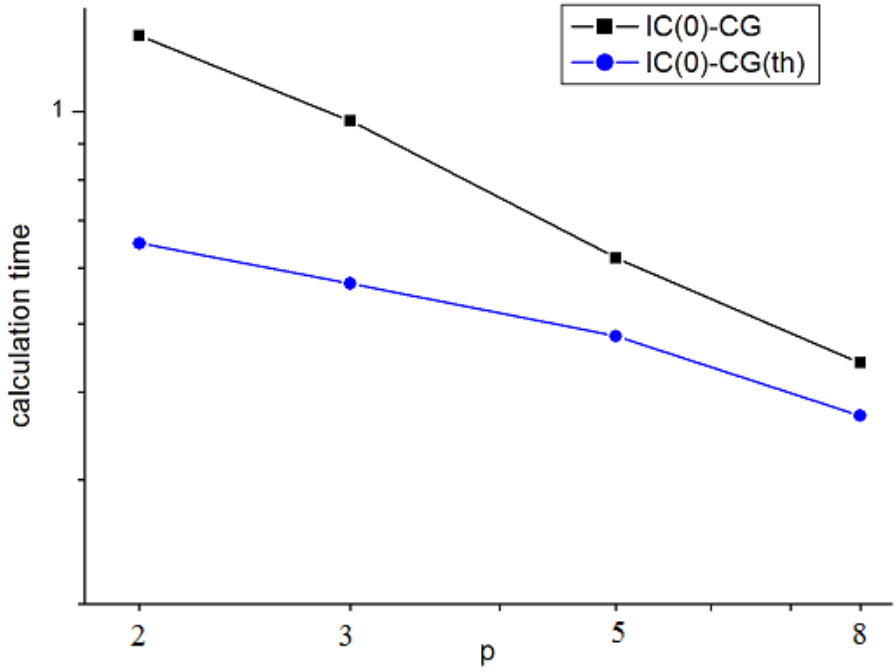


Рис. 8. Времена счета задачи с матрицей **g2_circuit** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

Как видно из таблиц 2, 3, 5 и рис. 5, 6, 8, применение MPI+OpenMP технологии для решения задач 1, 2, 4 позволяет заметно ускорить решение этих задач, по сравнению с использованием только MPI, при всех использованных значениях p. Как видно из таблицы 4 и рисунка 7, применение OpenMP технологии для решения задачи с матрицей **apache1** значительно меньшего размера позволяет ускорить вычисления только при p=2, 3.

В таблице 6 приведены числа итераций и времена счета на 4 процессорах модельной задачи 1 при двух различных способах разбиения области расчета.

Таблица 6

Числа итераций и времена счета методом IC(0)-CG задачи с матрицей **5_262144_2D** на 4 процессорах без применения и с применением OpenMP технологии

p=4	Разбиение 1	Разбиение 2
MPI	543, 0.83	407, 0.9
MPI+OpenMP μ	616, 0.43(6) 1.93	412, 0.34(6) 2.64

Как видно из таблицы 6, при использовании только MPI и MPI+OpenMP технологии числа итераций и времена счета при разбиении 2 значительно меньше, чем при разбиении 1, ускорение счета благодаря применению OpenMP технологии выше при разбиении 2, чем при разбиении 1.

На рисунке 9 приведены графики ускорений счета тестовых задач 1, 2, 4

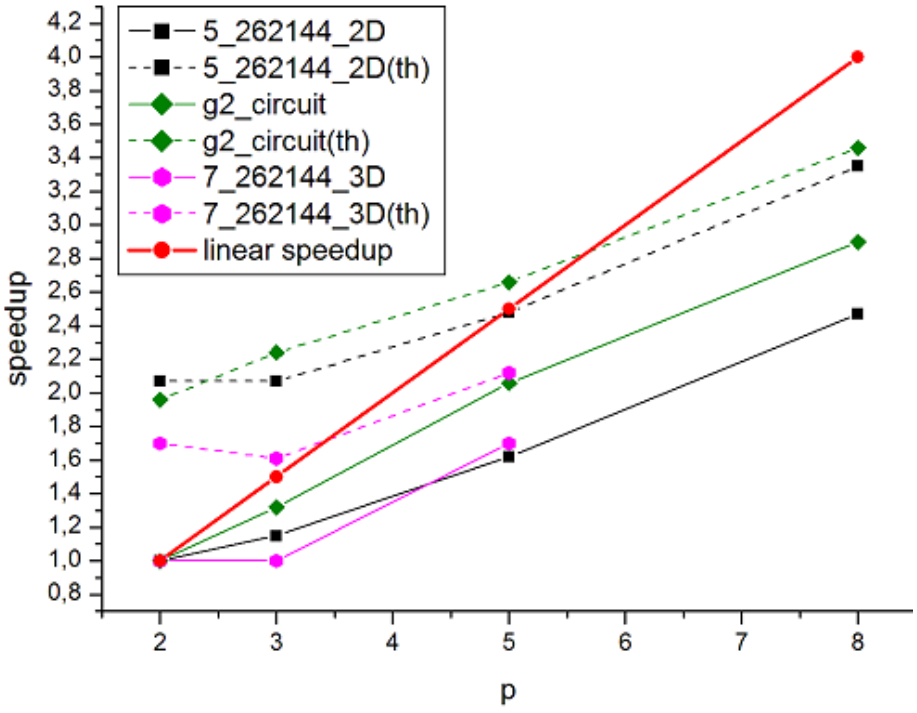


Рис. 9. Ускорения счета тестовых задач 1, 2, 4 методом IC(0)-CG при применении MPI и при применении MPI+OpenMP технологии.

методом IC(0)-CG с применением MPI (сплошные линии) и MPI+OpenMP (штриховые линии) по сравнению со счетом на 2 процессорах с применением только MPI.

Как видно из рис. 9 и таблиц 2, 3, 5, применение MPI+OpenMP технологии для решения тестовых задач 1,2 позволило получить сверхлинейное ускорение при p=2, 3, а при решении задачи 4 – при p=2, 3 ,5.

Уменьшение эффекта от использования OpenMP технологии с увеличением числа процессоров объясняется, в частности, уменьшением числа строк матрицы, приходящихся на каждый процессор, т. е. уменьшением вычислительной работы в каждом процессоре. В работе [30] на примере решения модельной задачи 1, описанной в начале раздела 6, при различных значениях размерности матрицы n методом CG предобусловливанием блочного Якоби в сочетании с IC2S(τ) показано, что при фиксированном числе процессоров ускорение счета благодаря использованию OpenMP технологии растет с ростом n. В настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, значительно больше. Следует надеяться, что потеря эффективности от применения OpenMP технологии при решении задач с большими размерами матриц наступит при значительно большем числе процессоров.

7. Заключение

В работе предложены способы применения MPI и MPI+OpenMP технологии построения и обращения предобусловливателя IC(0) при решении СЛАУ (1.1) методом сопряженных градиентов с произвольной симметричной положительно определенной матрицей.

Способ применения MPI при построении и обращении предобусловливателя основан на использовании переупорядочения строк и столбцов матрицы в соответствие с использованием упорядочения узлов сетки всей области расчета типа DDO. При применении MPI+OpenMP технологии проводится дополнительное переупорядочение строк и столбцов матрицы в соответствии с дополнительным переупорядочением внутренних узлов сетки, полученных в результате переупорядочения для применения MPI. При построении и обращении матрицы предобусловливания OpenMP технологии применяются для большинства строк матрицы.

С помощью расчетов двух модельных задач и двух задач из коллекции SuiteSparse на небольшом числе процессоров показано, что применение MPI происходит с эффективностью более 57% по сравнению со счетом на 2 процессорах, а использование OpenMP технологии позволяет заметно ускорить вычисления для небольшого числа процессоров.

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. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателями блочного неполного обратного треугольного разложения второго и первого порядка // ВАНТ. Серия Математическое моделирование физических процессов. 2022. Вып.1. С. 48-61.
17. Kaporin I.E. New convergence results and preconditioning strategies for conjugate gradient method // Numer. Linear Algebra and Appls. 1994. V. 1. N 2. P. 179-210.
18. Munksgaard N. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. № 6. P. 206-219.
19. Anderson E. C., Saad Y. Solving sparse triangular systems on parallel computers // International J. of High Speed Computing. 1989. V.1. P. 73–96.
20. Hammond S. W., Schreiber R. Efficient ICCG on a shared memory multiproces530sor, International J. High Speed Computing 4. 1992. P. 1–21.
21. 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.
22. 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.

23. Chow E., A. Patel A. Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. V. 37. P. 169-193.
24. 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.
25. Капорин И.Е., Милюкова О.Ю. MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями // Препринты ИПМ им. М.В. Келдыша РАН № 8. Москва. 2018 г. 28 с. doi:10.20948/prepr-2018-8.
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. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем // Препринты ИПМ им. М.В. Келдыша. 2020. № 31. 22 с. https://doi.org/10.20948/prepr-2020-31.
29. 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.
30. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями // Математическое моделирование.2021.Т. 33. № 10. с.19-39.
31. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1 // Препринты ИПМ им. М.В. Келдыша РАН. 2021. № 48. 32 с.
32. Милюкова О.Ю. Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC1 // Препринты ИПМ им. М.В. Келдыша РАН. 2022, № 2. 30 с.
33. 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.
34. 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.
35. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
36. 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