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

Препринты ИПМ • Препринт № 35 за 2023 г.

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

О.Ю. Милюкова

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

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2023. № 35. 32 с. https://doi.org/10.20948/prepr-2023-35
https://library.keldysh.ru/preprint.asp?id=2023-35

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

О. Ю. Милюкова

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

Москва — 2023

Милюкова О.Ю.

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

В работе рассматриваются два способа применения MPI и MPI+OpenMP технологий 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. Они отличаются способом вычисления матрицы предобусловливания IC(0). Способы применения MPI и MPI+OpenMP технологий основаны на использования 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 методом сопряженных градиентов с предобусловливателем IC(0) с использованием MPI и гибридной MPI+OpenMP технологий на примере модельных задач и ряда задач из коллекции разреженных матриц SuiteSparse, а также сравнение времени решения этих задач с использованием двух способов применения MPI и MPI+OpenMP технологий.

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

Olga Yurievna Milyukova

MPI+OpenMP methods for implementing the conjugate gradient method with the IC(0) preconditioner based on the use of grid node reordering

The paper considers two ways of using MPI and MPI+OpenMP technologies for constructing and inverting 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. They differ in the way in which the preconditioning matrix IC(0) is computed. 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 as well as comparing the time of solving these problems using two methods of using MPI and MPI + OpenMP technology 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

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

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 наличия диагонального преобладания у симметричной положительно определенной матрицы A.

Решение задач с матрицами большого размера требует применения параллельных вычислений. 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, например использование упорядочений, связанных 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 узлов сетки типа DDO для построения параллельного варианта метода неполного треугольного 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.

Проблеме использования высокоуровневого параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. В работах [14-17] 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 [18] 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 [19] 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.

В работах [20 – 22] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0) и IC1(τ) (неполным треугольным разложением первого порядка) [23]. Один из них основан на переупорядочении узлов сетки типа DDO внутри каждой подобласти, другой – на уменьшении шаблона разреженности матрицы A при построении предобусловливателя. С помощью расчетов тестовых задач показано, что применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для умеренного числа узлов суперкомпьютерной системы (порядка нескольких десятков). В работе [24] предложены 2 способа применения OpenMP технологии для параллельной MPI+OpenMP реализации построения и обращения предобусловливателя блочного неполного обратного треугольного разложения первого порядка [25], в одном из которых используется упорядочение узлов сетки типа DDO [12].

В работе [26] предложены способы применения MPI и MPI+OpenMP технологий для построения и обращения предобусловливателя IC(0) в предобусловленном методе сопряженных градиентов. В этой работе для параллельной реализации с применением только MPI используется упорядочение узлов сетки типа DDO [12], для применения OpenMP технологии тоже используется упорядочение узлов сетки типа DDO [12]. В работе [27] предложены способы применения MPI и MPI+OpenMP технологий для построения и обращения предобусловливателя типа неполного треугольного разложения без заполнения [28], основанные на использовании упорядочений узлов сетки, согласованных с разбиением области расчета.

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 [29], являющиеся обобщением упорядочения [30]. 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 рассматриваются 2 способа применения MPI и MPI+OpenMP технологий построения и обращения предобусловливателя IC(0) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей методом сопряженных градиентов. Оба способа применения MPI при построении и обращении предобусловливателя основаны на использовании переупорядочения строк и столбцов матрицы в соответствии с использованием упорядочения узлов сетки всей области расчета типа DDO [12]. В способе 1 применения только MPI, предложенном ранее в работе [26], строится верхнетреугольный множитель матрицы предобусловливания. В способе 2 применения только MPI, предложенном в настоящей работе, в каждом процессоре сначала строятся нижнетреугольные матрицы, являющиеся подматрицами нижнетреугольного множителя предобусловливателя и содержащие строки и столбцы, соответствующие внутренним узлам подобластей, полученных при разбиении для применения MPI. 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, используется упорядочение узлов сетки типа DDO [12]. 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 [31], а также сравнение времени решения этих задач методом IC(0)-CG при использовании рассматриваемых способов применения MPI и MPI+OpenMP технологий.

2. Предобусловленный метод сопряженных градиентов

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

Алгоритм 1

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

для $k = 0, \ldots,$ 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 $z_{k+1} = B^{-1} r_{k+1}, \quad p_0 = w_0 = B^{-1} r_0$. 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. Алгоритмы построения матрицы предобусловливания IC(0)

Алгоритм вычисления матрицы $L = U^T$ в предобусловливателе IC(0) имеет вид [2]:

Алгоритм 2

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

Здесь и далее a_{ij} – элементы матрицы A, l_{ij} – элементы матрицы L.

В настоящей работе будем также использовать алгоритм построения верхнетреугольной матрицы U в предобусловливателе IC(0), предложенный в работе [26]. Это удобнее для реализации распараллеливания алгоритма построения матрицы предобусловливания IC(0) с применением MPI. Перед построением матрицы предобусловливания с использованием этого алгоритма

необходимо обеспечить, чтобы матрица A была отмасштабирована, тогда диагональные элементы этой матрицы равны единице. Алгоритм построения верхнетреугольной матрицы U в предобусловливателе IC(0) имеет вид [26]:

Алгоритм 3

1. Инициализация вспомогательной диагональной матрицы:
   for $i = 1, ..., n$
   $d_i := 1$
   end for
   for $i = 1, ..., n$ (*цикл по строкам* A *для вычисления строк* U)
2. Инициализация вектора v при помощи i-ой строки A:
   for $j = i+1, ..., n$
   $v_j := a_{ij}$
   end for
3. Сделать поправку к вектору v:
   for $t = 1, ..., i-1$
   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 $j = i+1, ..., n$
   $d_j := d_j - u_{ij}^2$
   end for
   end for (*конец цикла по* i)

Здесь и далее u_{ij} – элементы матрицы U, d_i – элементы вспомогательной диагональной матрицы. При вычислении l_{ii} в алгоритме 2 и u_{ii} в п. 4 алгоритма 3 необходимо извлекать квадратный корень из числа, которое определяется в процессе вычислений. Это выражение может оказаться отрицательным. В этом

случае следует для решения СЛАУ (1.1) использовать метод сопряженных градиентов с другим предобусловливателем.

4. Алгоритмы параллельной реализации построения и обращения предобусловливателя IC(0) с использованием MPI

Пусть матрица A переупорядочена и отмасштабирована. Разобьем каким-либо образом произвольную (возможно, трехмерную) область расчета на p подобластей, где p – предполагаемое число используемых процессоров, выберем некоторую нумерацию подобластей и будем использовать упорядочение узлов сетки, предложенное в работе [12]. Для этого введем множество узлов разделителей – множество узлов сетки в подобластях, у которых имеются соседи из подобластей с бóльшим 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, бóльшими, чем номер рассматриваемой подобласти. Узел разделителя назовем узлом разделителя второго уровня, если в шаблоне этого узла нет узлов разделителей более высокого уровня, чем первый, расположенных в подобластях с бóльшим номером. Остальные узлы разделителей назовем узлами разделителей третьего уровня.

Используя определение узлов разделителей первого, второго и третьего уровней и доказательство от противного, получим, что в шаблонах узлов разделителей первого уровня могут быть только внутренние узлы из подобластей с тем же или бóльшими 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 бóльшими номерами, узлы разделителей первого уровня из подобластей с тем же или бóльшими 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 следования 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.

На рис. 1 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* –

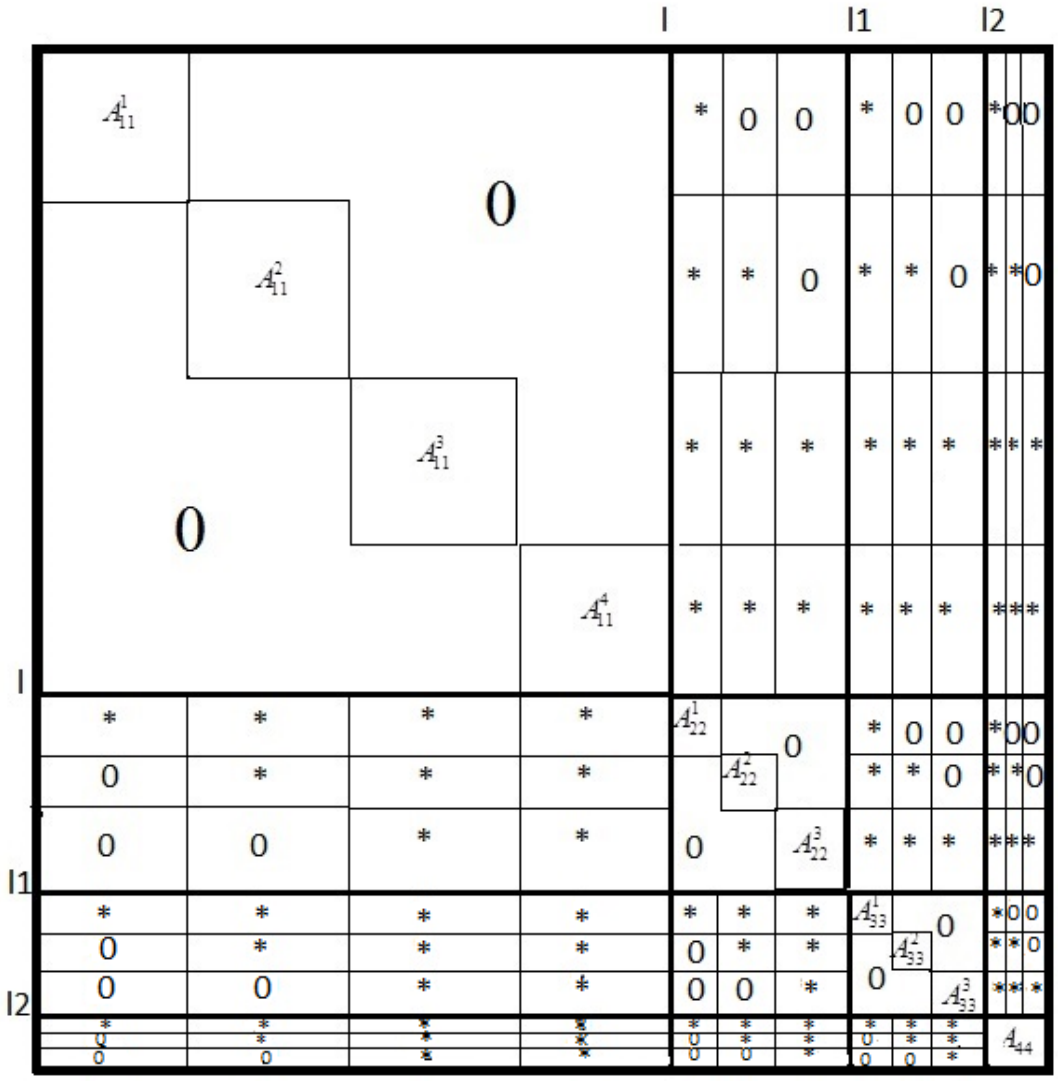

Рис. 1. Вид структуры разреженности матрицы *A*, полученной после перестановки строк и столбцов в результате переупорядочения.

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.

Напомним, что структура разреженности матрицы *U* в предобусловливателе IC(0) совпадает со структурой разреженности верхнетреугольной части матрицы *A*.

Рассмотрим 2 способа параллельной реализации построения и обращения матрицы предобусловливания IC(0) с использованием только MPI.

В способе 1 параллельной реализации вычисления матрицы предобусловливания IC(0) с использованием только MPI вычисляются элементы матрицы *U*. Перед вычислением элементов матрицы *U* необходимо произвести переупорядочение строк и столбцов матрицы *A* в соответствии с новым упорядочением узлов сетки. Определение элементов матрицы *U* начинается с вычисления ее элементов в строках, соответствующих внутренним узлам всех подобластей. Используется алгоритм, аналогичный алгоритму 3, 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*. Поскольку в строках матрицы *U*, соответствующих внутренним узлам сетки, нет ненулевых элементов в столбцах, соответствующих внутренним узлам из других подобластей, вычисление элементов матрицы *U* в строках, соответствующих внутренним узлам, может происходить одновременно и независимо во всех процессорах.

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 4.

Алгоритм 4.

1. for *s = 1, ..., p*
   Вычислить V_{ij}^S для *i = l+1, ..., n, j = i+1, ..., 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
```

$$V_{ij} = \sum_{s=1}^{p} V_{ij}^{s}$$

```
   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 в 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 неокончательных значений d_j для j, соответствующих номерам узлов разделителей. При этом используется операция MPI_AllReduce.

Благодаря совпадению структуры разреженности матрицы U со структурой разреженности верхнетреугольной части матрицы A вычисление элементов матрицы U в строках, соответствующих узлам разделителей первого уровня, происходит во всех процессорах одновременно с использованием алгоритма 5. Напомним, что перед началом вычисления элементов матрицы U всем d_i было присвоено значение 0.

Алгоритм 5

for $s = 1, \ldots, p$

1. Инициализация вспомогательной диагональной матрицы:

```
for i = ī1(s),..., ī2(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*-й 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, ..., n*

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

end for

end for (*конец цикла по i*)

end for (*конец цикла по s*)

В алгоритме 5 $\bar{i}_1(s)$, $\bar{i}_2(s)$ – номера первой и последней строк на разделителях первого уровня в подобласти с номером *s*.

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

Вычисление элементов матрицы U в строках, соответствующих узлам разделителей второго уровня, производится аналогично вычислению элементов этой матрицы в строках, соответствующих узлам разделителей первого уровня. При этом в п. 2 алгоритма 5 добавляется еще одно слагаемое: $\bar{V}_{ij}$, где $\bar{V}_{ij}$ – элементы матрицы $\bar{V}$. Затем вычисляются элементы вспомогательной матрицы $\tilde{V}$, что делается аналогично вычислению элементов матриц V, $\bar{V}$, и производится суммирование по всем процессорам вычисленных там значений Δd_j для номеров j, соответствующих узлам разделителей третьего уровня, и добавление Δd_j к вычисленным ранее неокончательным значениям d_j.

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.

Перед обращением матрицы предобусловливания необходимо произвести переупорядочение элементов вектора r^{k-1}, где k – номер итерации в 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 [30] и подробно описанный в работе [26]. Будем вычислять элементы вектора $z = D_U \hat{w}^k$, где D_U – диагональная 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 6

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).

После этого осуществляется пересылка значений слагаемых z_j для индексов 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.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей первого уровня с использованием алгоритма, аналогичного алгоритму 6, что осуществляется во всех процессорах одновременно и независимо. Затем осуществляются необходимые пересылки и необходимая коррекция значений z_j, аналогично описанному выше. Вычисление значений $\hat{w}_i^k$ в узлах разделителей второго уровня происходит аналогично вычислению значений $\hat{w}_i^k$ в узлах разделителей первого уровня.

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 предобусловливания сначала с использованием матрицы U происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем – второго и первого уровней, 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 выполнения всех пересылок при обращении матрицы предобусловливания используются операции MPI_Recv и MPI_Send. После завершения вычисления вектора w^k следует произвести переупорядочение его элементов.

В способе 2 параллельной реализации вычисления матрицы предобусловливания IC(0) с применением только MPI сначала в каждом процессоре с использованием алгоритма 2 вычисляются элементы подматриц матрицы L ($L = U^T$), содержащих строки, соответствующие внутренним узлам сетки. Затем с помощью транспонирования вычисляются подматрицы матрицы U, содержащие строки и столбцы, соответствующие внутренним узлам сетки. Это осуществляется во всех процессорах одновременно и независимо. Затем с использованием алгоритма, аналогичного алгоритму 3, вычисляются подматрицы матрицы U, содержащие строки, соответствующие внутренним узлам сетки, и столбцы, соответствующие узлам разделителей, а также подматрицы матрицы U, содержащие строки, соответствующие узлам разделителей. Это осуществляется аналогично вычислению матрицы U с использованием способа 1 применения MPI. Затем создается матрица U, необходимая на втором этапе обращения матрицы предобусловливания, с помощью объединения созданных ранее частей этой матрицы.

Перед обращением матрицы предобусловливания необходимо произвести переупорядочение элементов вектора r^{k-1}, где k – номер итерации в предобусловленном методе сопряженных градиентов. В способе 2 параллельной реализации выполнение первого этапа обращения матрицы предобусловливания начинается с обращения подматриц матрицы L, содержащих строки и столбцы, соответствующие внутренним узлам сетки. При этом процессоры работают одновременно и независимо. После вычисления таким образом $\hat{w}_i^k$ для индексов i, соответствующих внутренним узлам сетки, происходит вычисление слагаемых z_j для номеров j, соответствующих узлам разделителей. При этом используются столбцы матрицы U с номерами, соответствующими узлам разделителей, из строк этой матрицы с номерами, соответствующими внутренним узлам. Эти вычисления тоже происходят

одновременно и независимо во всех процессорах. 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. Вычисление элементов вектора $\hat{w}^k$, соответствующих узлам разделителей, осуществляется так же, как в способе 1.

Вычисление элементов вектора w^k осуществляется так же, как в способе 1, и описано выше. После завершения вычисления вектора w^k следует произвести переупорядочение его элементов.

5. Алгоритмы параллельной реализации построения и обращения предобусловливателя IC(0) с использованием MPI+OpenMP

Рассмотрим 2 способа применения MPI+OpenMP технологии при построении и обращении предобусловливателя IC(0), соответствующих рассмотренным выше способам применения MPI. 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 ($s = 1, 2, \ldots, p$) 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 [25].

Произведем переупорядочение внутренних узлов сетки в подобластях, полученных при разбиении всей области расчета на p подобластей ($p \neq 1$). 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. 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 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.

Рассмотрим 2 способа использования MPI+OpenMP технологии, соответствующие 2 способам применения MPI, описанным выше. В обоих способах применения MPI+OpenMP технологии использовалось упорядочение узлов сетки, описанное в начале этого раздела.

При использовании MPI+OpenMP способом 1 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 *k*2 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.

При использовании способа 1 применения MPI+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. 2 приведен вид структуры разреженности матрицы $L1_S$ для случая m = 4 и разбиения множества внутренних узлов в каждой подобласти, используемого в

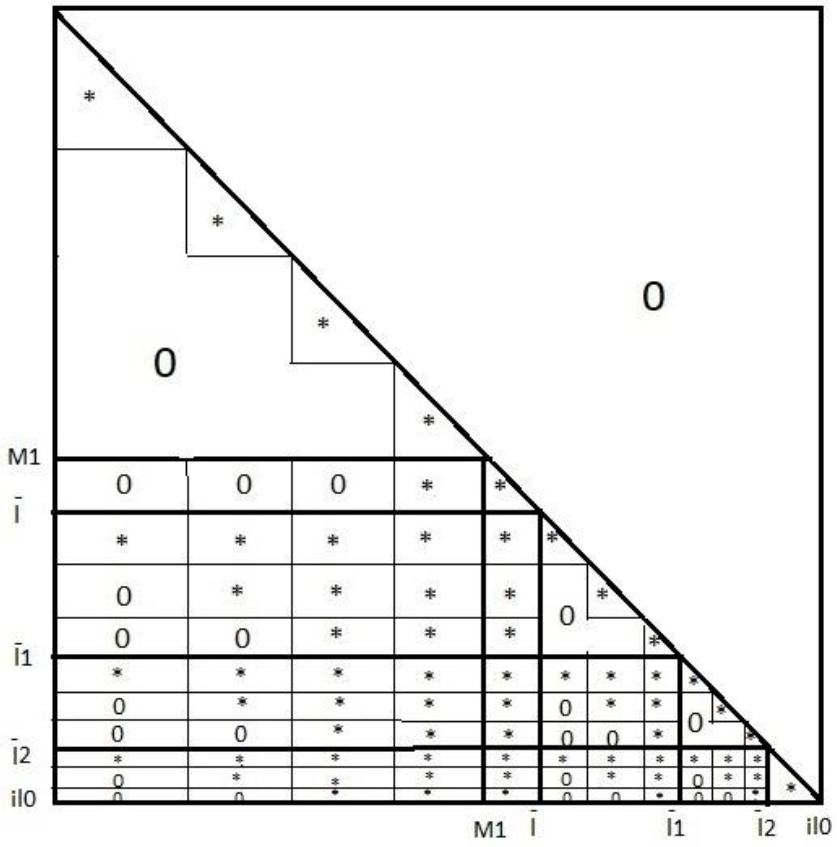

Рис. 2. Вид структуры разреженности матрицы $L1_S$ для случая *m=4.*

настоящей работе. На рис. 2 $\bar{l}$ – 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 первых $il0(s)$ строк матрицы U_S с номерами столбцов с индексами, соответствующими узлам разделителей при упорядочении для крупнозернистого распараллеливания.

При использовании MPI+OpenMP способом 2 на этапе построения матрицы предобусловливания вычисления начинаются с построения подматриц матрицы L, содержащих строки и столбцы, соответствующие внутренним узлам сетки, и обозначенных выше $L1_S$. Это осуществляется с использованием алгоритма 2. При построении первых $M1$ строк матриц $L1_S$ используются OpenMP технологии, используется цикл по $k2 = 1, \ldots, m$, внутри которого осуществляется вычисление элементов строк искомой матрицы с локальными номерами $1, \ldots, M1$. 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 матриц $L1_S$.

После этого осуществляется вычисление элементов подматриц матрицы U, состоящих из строк, соответствующих внутренним узлам сетки, и столбцов в этих строках с номерами, соответствующими узлам разделителей, обозначенных выше $U2_S$, а также строк матрицы U, соответствующих узлам разделителей. На этих этапах в настоящей работе OpenMP технологии не применялись.

Обращение матрицы предобусловливания в способах 1 и 2 применения MPI+OpenMP технологии происходило одинаковым образом. 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 $i = 1, \ldots M1 = m\bar{l}_S$ (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 $i = M1+1, \ldots, il0(s)$. Затем с

использованием матриц $U2_s$ вычисляются слагаемые элементов $z_j = \hat{w}_j U_{jj}$ вектора z для значений j, соответствующих узлам разделителей. Используется алгоритм, аналогичный алгоритму 6.

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.

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 матриц U_s (s = 1, 2, …, p). 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. 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, …, 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.

MPI+OpenMP реализация вычисления элементов вектора $q_k = Ap_k$, где k – номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, и вычислений векторных операций и скалярных произведений в алгоритме 1 кратко описаны в работе [26]**.**

6. Результаты расчетов

Программы, реализующие применение метода IC(0)-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI и MPI+OpenMP технологий. 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 задач 1 и 2. В качестве модельной задачи 1 рассматривалась разностная задача Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке, причем n = 1048576. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы **5_1048576**). Модельная задача 2 – разностная задача Дирихле для уравнения Пуассона в единичном кубе на равномерной ортогональной сетке, причем n = 830584. Использовалась стандартная 7-точечная аппроксимация лапласиана (имя матрицы **7_830584**). Для тестирования рассматриваемых параллельных методов использовались также три матрицы из коллекции разреженных матриц SuiteSparse [31]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

apache2 – трехмерная конечно-разностная схема;

parabolic_fem – уравнение диффузии-конвекции с постоянным переносом;

ecology2 – приложение теории электрических цепей к задаче передачи генов.

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 [33], Id – количество строк без диагонального преобладания, Ip – количество положительных внедиагональных элементов, NZA – число ненулевых элементов матрицы A, $nz_{\min}$, $nz_{\max}$ – минимальное и максимальное числа ненулевых элементов в строках матрицы A, n – число неизвестных в системе уравнений (1.1).

Таблица 1

Свойства некоторых матриц из Флоридской коллекции

Матрица	n	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_0)$
apache2	715176	4817870	2	0	4	8	0.12+7
parabolic_fem	525825	3674625	0	1048576	3	7	0.20+6
ecology2	999999	4995991	1124	0	3	5	0.63+8

Решалось уравнение $Ax = b$, где A – матрица, полученная после переупорядочения, связанного с разбиением на подобласти для использования MPI с помощью алгоритма [29], 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 $\varepsilon = 10^{-8}$.

В таблицах 2–6 приведены числа итераций и время счета методом IC(0)-CG тестовых задач при применении MPI и MPI+OpenMP технологий. Под временем вычислений в таблицах 2–6 подразумевается время счета в секундах итерационного процесса в сумме со временем вычисления

предобусловливателя. При этом под временем вычисления предобусловливателя при использовании способа 1 применения MPI подразумевалось время вычисления верхнетреугольной части матрицы предобусловливания, при использовании способа 1 применения MPI+OpenMP технологии – время вычисления верхнетреугольной части матрицы предобусловливания в сумме со временем вычисления нижнетреугольной части матрицы предобусловливателя для строк и столбцов, соответствующих внутренним узлам сетки в подобластях (с использованием операции транспонирования, см. выше). Под временем вычисления предобусловливателя при использовании способа 2 применения MPI подразумевалось время вычисления нижнетреугольной части матрицы предобусловливания для строк, соответствующих внутренним узлам сетки, транспонирование получившейся матрицы и вычисление верхнетреугольной части матрицы предобусловливания для строк, соответствующих внутренним узлам сетки, со столбцами, соответствующими узлам разделителей, а также время вычисления верхнетреугольной части матрицы предобусловливания для строк, соответствующих узлам разделителей. При использовании способа 2 применения MPI+OpenMP технологии под временем вычисления матрицы предобусловливания подразумевалось суммарное время вычисления указанных в предыдущем предложении этапов с использованием OpenMP технологии, как было описано в предыдущем разделе.

При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 10 и 12 нитей. В таблицах 2–6 приведены оптимальные по числу нитей для каждого p с точки зрения времени

Таблица 2

Числа итераций и время счета методом IC(0)-CG задачи с матрицей **5_1048576** на p процессорах без применения и с применением OpenMP технологии

P	3	5	8	10	16
MPI, способ 1 η	1125, 11.79	1031, 6.31 *1.87*	1034, 4.32 *2.73*	1031, 3.38 *3.49*	1122, 3.16 *3.79*
MPI, способ 2 η	1125, 13.61	1031, 7.53 *1.81*	1034, 5.11 *2.67*	1031, 4.03 *3.37*	1122, 3.81 *3.57*
MPI+OpenMP способ 1 μ, η	*Th*=8 1186, 5.47 *2.15, 2.15*	*Th*=4 1168, 4.17 *1.5, 2.73*	*Th*=3 1164, 3.47 *1.25, 3.49*	*Th*=4 1113, 3.24 *1.04, 3.64*	*Th*=3 1202, 3.76 *0.84, 3.14*
MPI+OpenMP способ 2 μ, η	*Th*=8 1186, 5.38 *2.53, 2.53*	*Th*=4 1168, 4.17 *1.81, 3.26*	*Th*=3 1164, 3.48 *1.47, 3.91*	*Th*=4 1113, 3.24 *1.24, 4.2*	*Th*=3 1202, 3.76 *1.01, 3.62*

вычислений результаты и соответствующие им значения числа использованных нитей, обозначенные в этих таблицах *Th*. В таблицах 2–6 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 трех процессорах.

Таблица 3

Числа итераций и время счета методом IC(0)-CG задачи с матрицей **7_830584** на *p* процессорах без применения и с применением OpenMP технологии

p	3	5	8	10	16
MPI, способ 1	129, 1.84	132, 1.36	134, 1.11	135, 0.63	123, 0.37
η		*1.35*	*1.66*	*2.92*	*4.97*
MPI, способ 2	129, 2.03	132, 1.49	134, 1.19	135, 0.75	123, 0.44
η		*1.36*	*1.71*	*2.71*	*4.61*
MPI+OpenMP	*Th*=8	*Th*=4	*Th*=3	*Th*=3	*Th*=3
способ 1	153, 0.99	142, 0.83	145, 0.77	147, 0.63	146, 0.53
μ, η	*1.86, 1.86*	*1.64, 2.22*	*1.44, 2.39*	*1.0, 2.92*	*0.7, 3.47*
MPI+OpenMP	*Th*=8	*Th*=6	*Th*=3	*Th*=3	*Th*=3
способ 2	153, 0.95	152, 0.84	145, 0.8	147, 0.61	146, 0.51
μ, η	*2.14, 2.14*	*1.77, 2.42*	*1.49, 2.54*	*1.23, 3.32*	*0.86, 3.98*

Таблица 4

Числа итераций и время счета методом IC(0)-CG задачи с матрицей **apache2** на *p* процессорах без применения и с применением OpenMP технологии

p	3	5	8	10	16
MPI, способ 1	1297, 9.51	871, 5.34	948, 2.87	1294, 3.27	956, 1.84
η		*1.78*	*3.31*	*2.91*	*5.17*
MPI, способ 2	1297, 11.24	871, 6.15	948, 3.37	1294, 3.94	956, 2.2
η		*1.83*	*3.33*	*2.85*	*5.11*
MPI+OpenMP	*Th*=6	*Th*=4	*Th*=3	*Th*=4	*Th*=3
способ 1	1351, 4.78	980, 3.32	1019, 2.29	1367, 3.21	1068, 2.34
μ, η	*1.99, 1.99*	*1.66, 2.86*	*1.25, 4.15*	*1.02, 2.96*	*0.79, 4.06*
MPI+OpenMP	*Th*=8	*Th*=4	*Th*=3	*Th*=4	*Th*=3
способ 2	1373, 4.82	980, 3.33	1019, 2.26	1367, 3.15	1069, 2.32
μ, η	*2.33, 2.33*	*1.85, 3.37*	*1.49,4.97*	*1.25, 3.57*	*0.95, 4.84*

Заметим, что неожиданно большое число итераций метода IC(0)-CG при решении задачи с матрицей **apache2** на 3 и на 10 процессорах связано с особенностями разбиения трехмерной области расчета на подобласти методом [29]. Этим же можно объяснить неожиданно маленькое число итераций метода IC(0)-CG при решении задачи с матрицей **7_830584** на 16 процессорах.

Таблица 5

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

p	3	5	8	10	16
MPI, способ 1 η	874, 6.54	918, 3.98 *1.64*	914, 2.34 *2.79*	903, 2.02 *3.24*	902, 1.33 *4.92*
MPI, способ 2 η	873, 7.51	917, 4.68 *1.6*	914, 2.76 *2.72*	903, 2.35 *3.2*	902, 1.57 *4.78*
MPI+OpenMP способ 1 μ, η	*Th*=12 913, 2.57 *2.54, 2.54*	*Th*=6 938, 2.11 *1.88, 3.1*	*Th*=6 940, 1.49 *1.57, 4.37*	*Th*=4 926, 1.49 *1.42, 4.39*	*Th*=3 924, 1.39 *0.96, 4.7*
MPI+OpenMP способ 2 μ, η	*Th*=12 913, 2.55 *2.94, 2.94*	*Th*=6 937, 2.09 *2.24, 3.59*	*Th*=6 940, 1.48 *1.86, 5.07*	*Th*=4 926, 1.49 *1.58, 5.04*	*Th*=3 924, 1.36 *1.15, 5.52*

Таблица 6

Числа итераций и время счета методом IC(0)-CG задачи с матрицей **ecology2** на *p* процессорах без применения и с применением OpenMP технологии

P	3	5	8	10	16
MPI, способ 1 η	2041, 19.97	2081, 11.81 *1.69*	2081, 7.57 *2.64*	2056, 6.36 *3.14*	2092, 5.19 *3.83*
MPI, способ 2 η	2039, 23.82	2181, 14.05 *1.69*	2181, 8.84 *2.69*	2055, 7.64 *3.12*	2092, 6.28 *3.79*
MPI+OpenMP способ 1 μ, η	*Th*=6 2123, 9.81 *2.04, 2.04*	*Th*=4 2107, 7.07 *1.67, 2.82*	*Th*=3 2156, 5.71 *1.33, 3.5*	*Th*=4 2142, 5.88 *1.08,3.4*	*Th*=3 2122, 5.82 *0.89, 3.43*
MPI+OpenMP способ 2 μ, η	*Th*=6 2123, 9.78 *2.44, 2.44*	*Th*=4 2107, 7.25 *1.94, 3.28*	*Th*=3 2156, 5.66 *1.59, 4.21*	*Th*=4 2142, 5.84 *1.31, 4.08*	*Th*=3 2122, 5.69 *1.1, 4.19*

Заметим, что применение OpenMP технологии при обоих способах использования MPI+OpenMP технологии не позволяет ускорить вычисление матрицы предобусловливания в методе IC(0)-CG. Ускорение решения СЛАУ происходит исключительно за счет ускорения итерационного процесса, причем время счета итерационного процесса составляет основную часть времени решения СЛАУ с этими матрицами. Заметим, что при вычислении матрицы предобусловливания при использовании способа 2 применения MPI и MPI+OpenMP технологий вычисление матриц $L1_S$, $U1_S$, как правило, занимало меньше половины времени вычисления матрицы предобусловливания. При использовании MPI+OpenMP способом 1 почти всегда основную часть времени вычисления матрицы предобусловливания составило время вычисления верхнетреугольного множителя этой матрицы.

На рисунках 3–6 приведены графики зависимости времени счета тестовых задач методом IC(0)-CG от числа процессоров p в логарифмическом масштабе с использованием только MPI и разбиения на p подобластей с помощью алгоритма [29] (линии черного цвета), а также с использованием MPI+OpenMP технологии (линии синего цвета).

Как видно из таблиц 2–6 и рис. 3–7, применение MPI+OpenMP технологии способом 1 для решения задач с матрицами **5_1048576**, **7_830584**, **apache2, ecology2** позволило заметно ускорить решение этих задач по сравнению с использованием только MPI при $p \le 8$. Как видно из таблиц 2–6 и рис. 3–7, применение MPI+OpenMP технологии способом 2 для решения задач с матрицами **5_1048576**, **7_830584**, **apache2**, **ecology2** позволило заметно

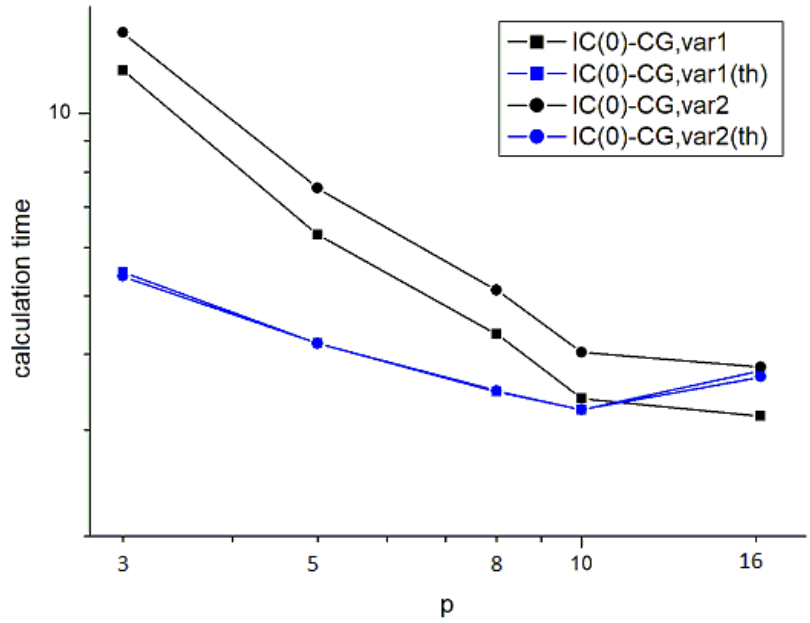

Рис. 3. Время счета задачи с матрицей **5_1048576** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологий способами 1 и 2.

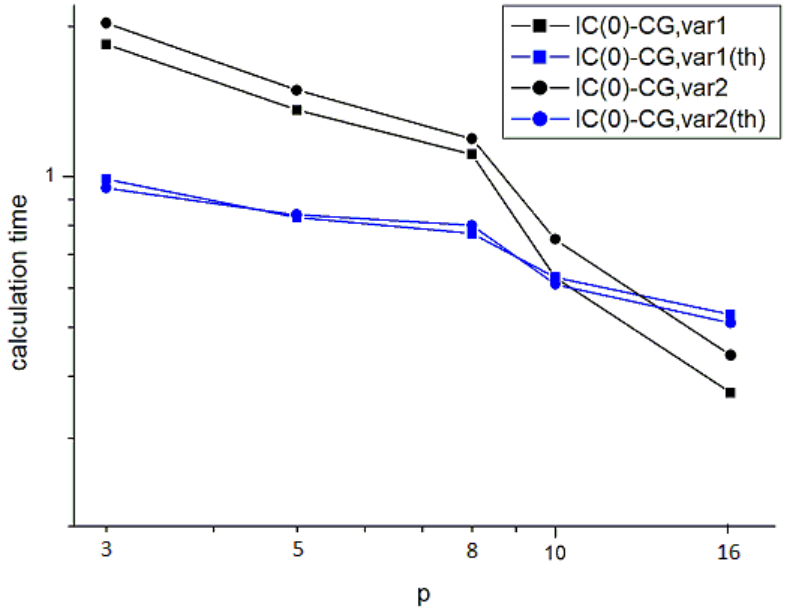

Рис. 4. Время счета задачи с матрицей **7_830584** методом IC1(0)-CG с использованием MPI и MPI+OpenMP технологий способами 1 и 2.

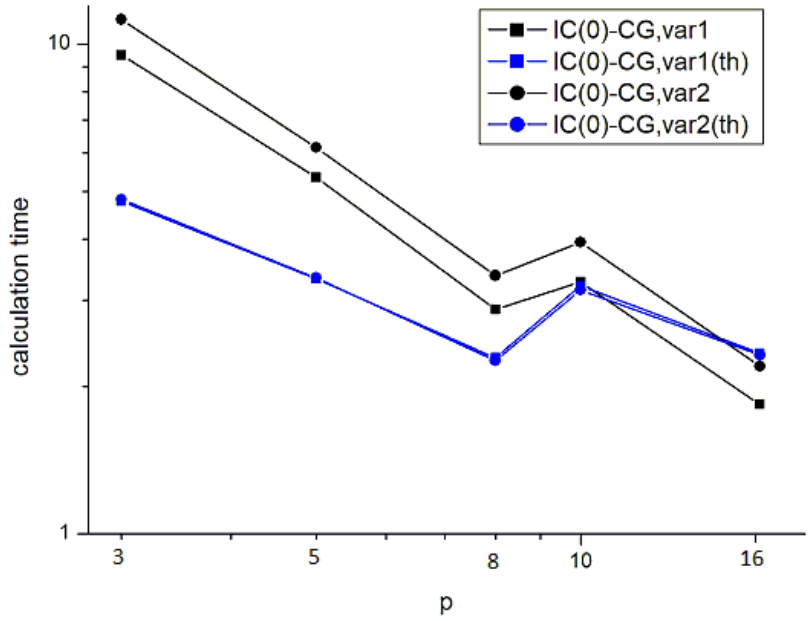

Рис. 5. Время счета задачи с матрицей **apache2** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологий способами 1 и 2.

ускорить решение этих задач, по сравнению с использованием только MPI способом 2 при $p \le 10$. Применение MPI+OpenMP технологии способами 1 и 2 для решения задачи с матрицей **parabolic_fem** позволило заметно ускорить ее решение по сравнению с использованием только MPI при $p \le 10$. Применение OpenMP технологии с использованием способов 1 и 2 применения MPI+OpenMP на 16 процессорах стало нецелесообразным.

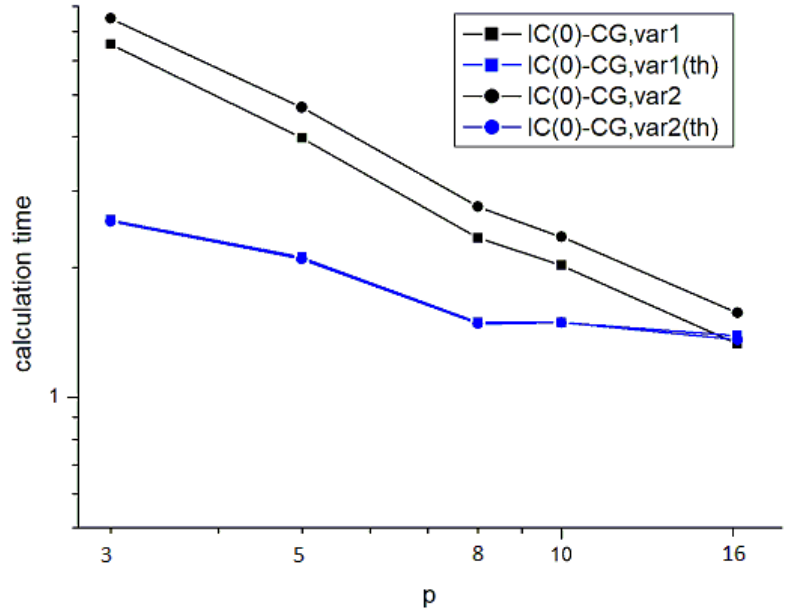

Рис. 6. Время счета задачи с матрицей **parabolic_fem** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологий способами 1 и 2.

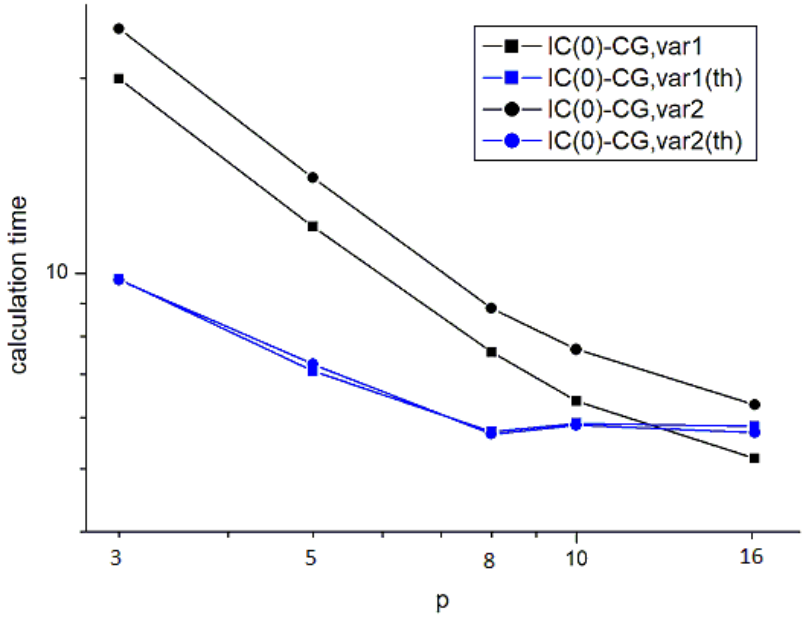

Рис. 7. Время счета задачи с матрицей **ecology2** методом IC(0)-CG с использованием MPI и MPI+OpenMP технологий способами 1 и 2.

Как видно из таблиц 2–6 и рис. 3–7, решение тестовых задач с применением способа 1 использования только MPI происходило быстрее, чем с применением способа 2 использования только MPI, очевидно, за счет меньшего времени, необходимого на обращение матрицы предобусловливания на каждой итерации. Как видно из таблиц 2–6 и рис. 3–7, решение тестовых задач с применением способов 1 и 2 использования MPI+OpenMP происходило

примерно с одинаковой скоростью, для реализации итерационного процесса используется одна и та же программа.

На рисунках 8, 9 приведены графики зависимости ускорения счета от числа процессоров при решении тестовых задач с использованием только MPI и

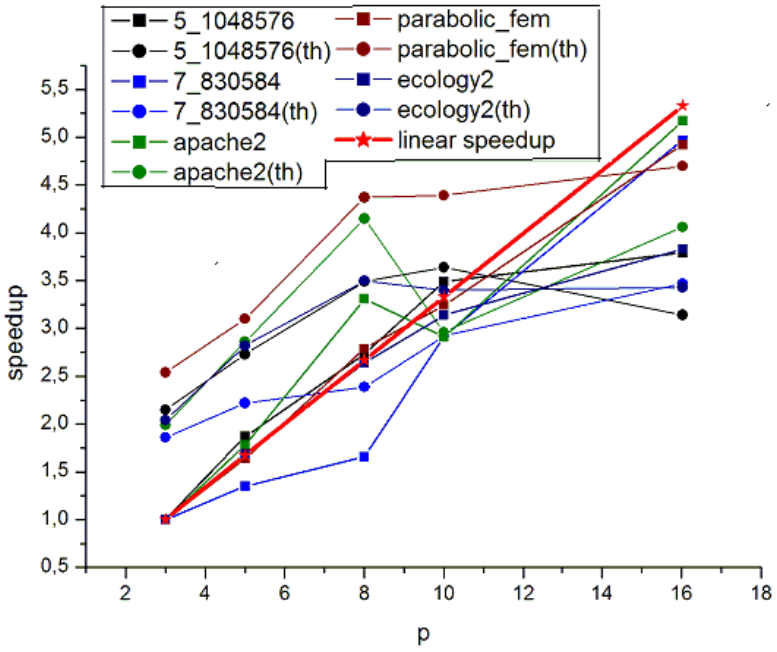

Рис. 8. Ускорение счета задач при использовании MPI и MPI+OpenMP способом 1 по сравнению со счетом на 3 процессорах с использованием MPI.

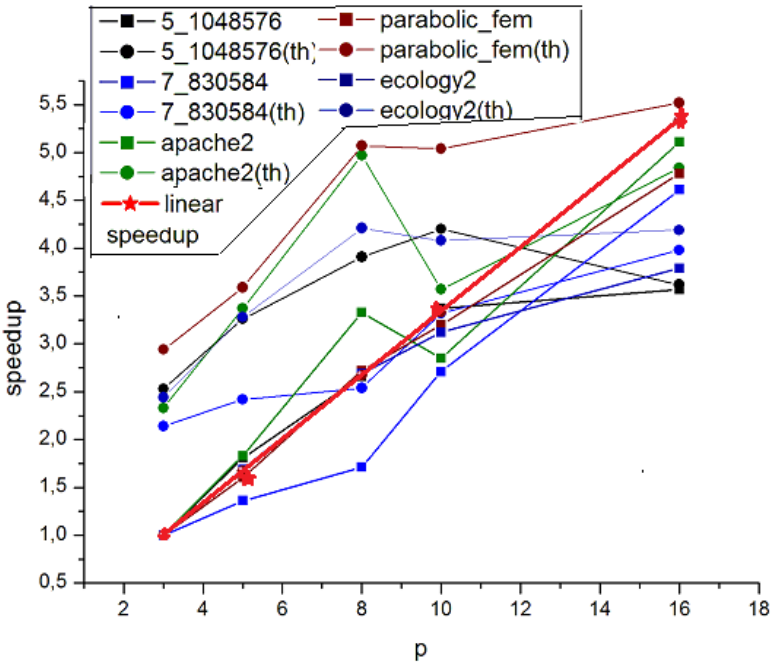

Рис. 9. Ускорение счета задач при использовании MPI и MPI+OpenMP способом 2 по сравнению со счетом на 3 процессорах с использованием MPI.

MPI+OpenMP по сравнению с временем решением этих задач на 3 процессорах с использованием только MPI.

Как видно из рис. 8, 9, применение MPI+OpenMP технологии способами 1 и 2 позволяет решать задачи с матрицами **5_1048576**, **parabolic_fem**, **ecology2** со сверхлинейным ускорением при $p \le 10$, а задачу с матрицей **7_830584** – при p = 3, 5. Применение MPI+OpenMP технологии позволяет решать задачу с матрицей **apache2** со сверхлинейным ускорением способом 1 при $p \le 8$, а способом 2 – при $p \le 10$.

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 [22] на примере решения модельной задачи 1, описанной в начале раздела 6, при различных значениях размерности матрицы n методом CG с предобусловливанием блочного Якоби в сочетании с стабилизованным неполным треугольным разложением второго порядка [13] 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 бóльшем числе процессоров.

7. Заключение

В работе рассмотрены два способа применения MPI и MPI+OpenMP технологий построения и обращения предобусловливателя IC(0) при решении СЛАУ (1.1) методом сопряженных градиентов с произвольной симметричной положительно определенной матрицей, которые отличаются способом вычисления матрицы предобусловливания, а при применении только MPI – еще и способом обращения матрицы предобусловливания.

Способы применения MPI при построении и обращении предобусловливателя основаны на использовании переупорядочения строк и столбцов матрицы в соответствии 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 трех задач из коллекции SuiteSparse показано, что использование OpenMP технологии позволяет заметно ускорить вычисления для небольшого числа процессоров. Решение тестовых задач с использованием только MPI способом 1 происходило, как правило, быстрее, чем с использованием только MPI способом 2. Решение тестовых задач с использованием способов 1 и 2 применения MPI+OpenMP технологии происходило примерно с одинаковой скоростью.

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. Anderson E. C., Saad Y. Solving sparse triangular systems on parallel computers // International J. of High Speed Computing. 1989. V.1. P. 73–96.
15. Hammond S. W., Schreiber R. Efficient ICCG on a shared memory multiproces530sor, International J. High Speed Computing 4. 1992. P. 1–21.
16. 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.
17. 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.
18. Chow E., A. Patel A. Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. V. 37. P. 169-193.
19. 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.
20. 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.
21. 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.
22. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями // Математическое моделирование. 2021. Т. 33. № 10. с.19-39.
23. Munksgaard N. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. № 6. P. 206-219.
24. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателями блочного неполного обратного треугольного разложения первого порядка // Ж. вычисл. мат. и програм. 2022. Т. 23. Вып. 3. С. 191-206. Doi: 10.26089/NumMet.v23r312
25. 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.
26. 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.

27. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В. Келдыша. 2023. № 18. 29 с. https://doi.org/10.20948/prepr-2023-18.
28. Kershow D. The Incomplete choleski-conjugate gradient method for the iterative solution of systems of linear equations // J. Comp. Phys. 1978. V. 26. P. 43-65.
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. https://doi.org/10.20948/prepr-2017-37.
30. 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.
31. Davis T., Hu Y.F. University of Florida sparse matrix collection / ACM Trans. on Math.~Software. 2011. V. 38, N. 1. http://www.cise.ufl.edu/research/sparse/matrices.
32. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
33. Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т. 52. № 2. С. 1-26.

Оглавление