УДК 519.6

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ С РАЗРЕЖЕННЫМИ СИММЕТРИЧНЫМИ МАТРИЦАМИ

А.В. Попов

Институт кибернетики им. В.М. Глушкова НАН Украины,
03680, Киев, проспект Академика Глушкова, 40.
Тел.: 526 1196, dept150@insyg.kiev.ua

Рассматриваются блочные алгоритмы прямых методов исследования и решения систем линейных алгебраических уравнений с разреженными (ленточными, профильными, блочно-диагональными с окаймлением) симметричными матрицами на компьютерах MIMD-архитектуры. Исследуется эффективность рассматриваемых алгоритмов. Приводятся некоторые результаты численных экспериментов на MIMD-компьютере.

The block algorithms of direct methods for the investigating and solving of systems of linear algebraic equations with sparse (band, profile, block-diagonal with bordering) symmetric matrices on the computers of MIMD-architecture are dealt with. The performance of algorithms being studied is investigated. Some results of numeral tests carried out on MIMD-computer are given.

Введение

При численном решении научно-технических задач во многих случаях возникает необходимость решать задачу (или несколько задач) линейной алгебры – линейную систему (систему линейных алгебраических уравнений, СЛАУ) или алгебраическую проблему собственных значений. Причем, как правило, решение задач линейной алгебры занимает значительную часть (50 % и более) времени решения всей задачи в целом. Например, задачи линейной алгебры возникают при дискретизации краевых задач или задач на собственные значения проекционно-разностным методом (конечных разностей, конечных элементов).

Важной особенностью задач линейной алгебры, возникающих при дискретизации, является высокий порядок их матриц – до миллионов, а то и до десятков миллионов. Это вызвано желанием оперировать более точными дискретными моделями, позволяющими получать приближенные решения более близкие к решениям исходных задач, лучше учитывать локальные особенности рассматриваемого процесса или явления.

Другой важной особенностью является то, что количество ненулевых элементов матриц таких задач составляет kn, где $k << n$, а n – порядок матрицы. Разреженная структура матрицы определяется нумерацией неизвестных задачи и зачастую является ленточной, профильной, блочно-диагональной с окаймлением и т.п. Во многих случаях матрицы дискретных задач симметричны и положительно определены или полуопределены.

Из-за высоких порядков, несмотря на разреженную структуру матриц, решение таких задач требует значительных вычислительных ресурсов, которые могут быть предоставлены современными параллельными компьютерами, в частности компьютерами MIMD-архитектуры. Поэтому является актуальной проблема создания для MIMD-компьютеров эффективных параллельных алгоритмов решения задач линейной алгебры, с разреженными матрицами. Под эффективным алгоритмом решения понимается алгоритм, позволяющий получить достоверное решение задачи с минимальным использованием ресурсов компьютера – процессоров, памяти, времени. Эффективность параллельных алгоритмов оценивается, как правило, с помощью коэффициентов ускорения и эффективности [1]. Важно также определить, какой алгоритм наиболее эффективен для решения конкретной задачи.

Эффективность параллельных алгоритмов в значительной мере зависит от сбалансированности загрузки процессоров, которая при решении задач линейной алгебры во многом определяется способами распределения по процессорам, хранения и обработки матриц и векторов решаемой задачи. При этом необходимо стремиться к равномерной загрузке всех участвующих в решении задачи процессоров в каждый момент времени, минимизируя время пребывания процессоров в состоянии ожидания.

Среди множества задач линейной алгебры с разреженными матрицами ключевое место занимает решение симметричных положительно определенных СЛАУ. Эта задача является составной частью более сложных задач, например, решения СЛАУ с полуопределенной матрицей [2], частичной обобщенной алгебраической проблемы собственных значений [3] и т.д. При решении линейных систем с симметричными положительно определенными матрицами на компьютерах традиционной архитектуры хорошо зарекомендовали себя прямые методы, в том числе метод Холецкого. Здесь рассматриваются параллельные алгоритмы решения СЛАУ с разреженными симметричными матрицами методом Холецкого.

Постановка и метод решения задачи

Для решения методом Холецкого линейной системы порядка n с q правыми частями

$$Ax = b \qquad (1)$$

© А.В. Попов, 2008

с симметричной матрицей, как правило, используется разложение исходной матрицы $A = LL^T$, где L – нижняя треугольная матрица, или его модификация

$$A = LDL^T, \tag{2}$$

где L – нижняя треугольная матрица с единичной главной диагональю, D – диагональная матрица [4].

LDL^T-факторизация открывает более широкие возможности для исследования и решения задач линейной алгебры. Например, использование LDL^T-факторизации позволяет получать решение СЛАУ со знаконеопределенной симметричной матрицей. Другим примером использования LDL^T-разложения является определение количества отрицательных собственных значений действительной симметричной матрицы с помощью свойства последовательности Штурма. Кроме того, при LDL^T-факторизации исключаются операции извлечения квадратных корней при вычислении диагональных элементов нижней треугольной матрицы. В случае использования LDL^T-разложения решение задачи (1) может быть получено в четыре этапа:

- факторизация (2) исходной матрицы СЛАУ;
- прямой ход – решение линейной системы с нижней треугольной матрицей

$$Lz = b; \tag{3}$$

- решение СЛАУ с диагональной матрицей

$$Dy = z; \tag{4}$$

- обратный ход – решение СЛАУ с верхней треугольной матрицей

$$L^T x = y. \tag{5}$$

Необходимо отметить, что в случае LDL^T-разложения разреженной матрицы нижняя треугольная матрица L также является разреженной. Причем матрица L сохраняет характерные особенности структуры исходной матрицы A, хотя общее количество ненулевых элементов, как правило, увеличивается. Так сохраняется количество поддиагоналей ленточной матрицы и профиль матрицы профильной матрицы.

Схемы распределения по процессорам разреженных матриц

Все прямые методы решения СЛАУ базируются на разложении матрицы задачи в произведение матриц стандартных видов, например, нижней и верхней треугольных матриц, LDL^T-разложение (2). Для алгоритмов таких разложений характерен постепенно уменьшающий от шага к шагу размер обрабатываемой части матрицы. Достаточно хорошую сбалансированность загрузки процессоров обеспечивают параллельные версии этих алгоритмов, в которых используются так называемые циклические схемы распределения и обработки матриц (см. напр. [5]). Такие циклические алгоритмы во многих случаях позволяют добиться примерно равного объема вычислений и обменов, выполняемых каждым процессором (параллельным процессом), и практически исключить влияние эффекта Гайдна.

Примером циклической схемы распределения по процессорам элементов матрицы является строчная циклическая схема. В этом случае элементы строк матрицы располагаются циклически в p процессорах, участвующих в вычислениях: если в процессоре с логическим номером q располагаются элементы r-й строки матрицы, то в следующем $(q+1)$-м – $(r+1)$-й строки. Обобщением данной схемы является одномерная (строчная) блочная циклическая схема: если в q-м процессоре располагаются элементы строк матрицы с номерами $sr+1,\ldots, sr+r$, то в $(q+1)$-м – строк с номерами $s(r+1)+1,\ldots, s(r+1)+s$, где s – количество строк в блоке (можно говорить о r-й и $(r+1)$-й строках квадратных матричных блоков порядка s).

С целью уменьшения количества арифметических операций для решения СЛАУ (1) с разреженной симметричной матрицей методом Холецкого путем перестановки строк и столбцов (т.е. перенумерации неизвестных) такую матрицу приводят к одному из стандартных видов: ленточному, профильному и т.п.. Существует множество алгоритмов оптимизации структуры разреженной матрицы и приведения ее к соответствующему стандартному виду. Их рассмотрение выходит за рамки данной работы. Будем предполагать, что разреженная матрица уже приведена к одному из таких компактных видов. Рассмотрим линейные системы с матрицами следующих структур: ленточной, профильной или блочно-диагональной с окаймлением.

В случае регулярной структуры разреженной матрицы параллельные алгоритмы, в которых используется одномерная блочная циклическая схема распределения элементов матриц, позволяют добиться примерно равного объема вычислений и обменов, выполняемых каждым параллельным процессом в каждый момент времени. Такой регулярной структурой является в первую очередь ленточная структура матрицы при очевидном условии, что полуширина ленты матрицы m превосходит произведение sp. В случае регулярной профильной структуре матрицы системы, варьируя значения s и p, можно также практически сбалансировать загрузку параллельных процессов в каждый момент времени, если также $c_e/n > sp$, где c_e – общее количество поддиагональных элементов в профиле матрицы.

Однако использование циклических схем распределения и обработки разреженных матриц далеко не всегда позволяет исключить эффект Гайдна. Например, в случае узких ленточных матриц при достаточно большом количестве параллельных процессов невозможно сбалансировать загрузку процессов в каждый

момент времени, так как количество арифметических операций, выполняемых каждым процессом, существенно отличается и может оказаться равным нулю для некоторых процессов. Также в случае блочно-диагональных матриц с окаймлением циклические схемы распределения не позволяют сбалансировать загрузку процессов.

На рис. 1 показан вид блочно-диагональной матрицы с окаймлением. Дискретные задачи с матрицами такого вида получаются при решении краевых задач методом конечных элементов или методом конечных разностей. Для этого область разбивается на подобласти, число которых равно количеству процессоров и проводится дискретизация. Неизвестные дискретной задачи нумеруются в следующем порядке: вначале неизвестные, принадлежащие только одной подобласти, последовательно по подобластям, затем в таком же порядке нумеруются неизвестные, принадлежащие двум подобластям, далее трем и т.д.. Таким образом, квадратные блоки A_i (рис. 1) связаны с неизвестными, принадлежащими одной подобласти, а блоки окаймления (B_{ij}, C_i, V_{ij}) – с остальными неизвестными (часть блоков окаймления может быть нулевыми). В каждый процессор (с логическим номером i) помещаются по одному квадратному блоку A_i, квадратному блоку C_i, и ненулевые внедиагональные прямоугольные блоки B_{ji}, V_{ki}, второй индекс которых совпадает с индексом диагональных блоков. Для эффективного распараллеливания необходимо чтобы размеры квадратных блоков A_i на порядок или более превосходили размеры квадратных блоков C_i.

Рассмотрим также случай СЛАУ с узкими ленточными матрицами. Для эффективного распараллеливания решения таких задач необходимо провести виртуальное переупорядочивание матрицы. Для этого узкая ленточная матрица разбивается на p (количество процессоров) приблизительно равных блоков строк, каждый из которых помещается в отдельный процессор. Соответствующим образом распределяется по процессорам и правая часть. На рис. 2 показано такое распределение для случая $p = 4$ (блоки разделены пунктирными линиями). После этого в каждом процессоре проводится разбиение на блоки помещенной в него подматрицы (части матрицы A). В общем случае для процессора с логическим номером k выделяется четыре блока (рис. 2): один (обозначим его A_k) порядка n_k – ленточный симметричный с полушириной ленты m, второй (C_k) порядка m – квадратный симметричный, два верхних треугольных блока порядка m – S_k и U_k. Исключение составляют нулевой (нет одного из треугольных блоков) и последний (нет квадратного и одного треугольного блока) процессоры. Такое разбиение проводится так, чтобы ленточные блоки были независимы один от другого. Заметим, что как обычно хранятся только диагональные и поддиагональные элементы матрицы.

Следующим шагом является виртуальное переупорядочивание матрицы – перестановка строк и столбцов, содержащих элементы блоков C_k ($k = \overline{0,\ p-2}$). Соответствующим образом виртуально переупорядочивается и правая часть. В результате вместо (1) получаем задачу

$$A'x' = b' \tag{6}$$

с переупорядоченными матрицей $A' = PAP$, правой частью $b' = Pb$ и решением $x' = Px$. Матрица A' получается блочно-диагональной с окаймлением (рис. 3), квадратные блоки A_k, которой являются ленточными матрицами, а из внедиагональных блоков ненулевыми являются только блоки $B_{k+1,k}$, $B_{k+1,k+1}$. При этом справа в блоке $B_{k+1,k}$ расположен треугольный блок S_k и все остальные элементы равны нулю, а в блоке $B_{k+1,k+1}$ слева расположен нижний треугольный блок U_k^T и все остальные элементы равны нулю. Еще раз заметим, что переупорядочивание виртуальное, так как реально никакие перестановки не проводятся. Поэтому в дальнейшем не потребуется переупорядочивать полученное решение.

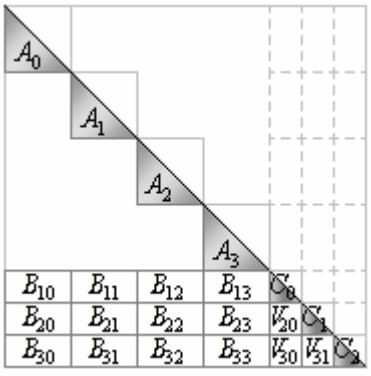

Рис. 1

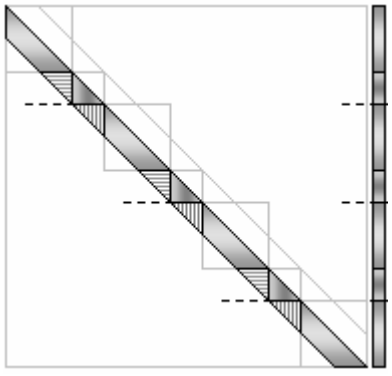

Рис. 2

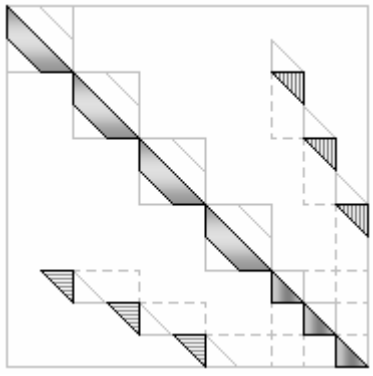

Рис. 3

Параллельный блочно-циклический алгоритм метода Холецкого

В работе [6] предложен параллельный блочно-циклический алгоритм метода Холецкого для решения СЛАУ с ленточными симметричными матрицами, который использует вышеописанную одномерную блочно-циклическую схему распределения элементов матрицы, правой части и решения системы (1). Этот алгоритм можно использовать и для решения СЛАУ с профильными симметричными матрицами.

Вычислительная схема параллельного блочно-циклического алгоритма разложения (2) как ленточной, так и профильной матрицы одна и та же, отличаясь только определением величины m_i (см. далее). Итак, для $I = 0, 1, 2, \ldots, N$, где $n = Ns+t$, $1 \le t \le s$, выполняются следующие шаги (действия):

– при $I > 0$ вычисляются в соответствующем процессоре элементы $g_{i+rs,j}$ ($g_{i,j} \equiv l_{i,j}d_j$; $i, j = Is–s+1,\ldots, Is$; $r = 1,..., p$);

– вычисление в процессоре, где хранится (I+1)-я строка квадратных блоков (порядка s) матрицы A, элементов $l_{i, i-k}$ ($k = 1,..., m_i$) и d_i, где $i = Is+1,\ldots, Is+s$;

– при $I > 0$ вычисление ненулевых элементов $g_{i,j}$ ($i = Is+ps+1,\ldots, n$; $j = Is-s+1,\ldots, Is$) квадратных блоков (порядка s) I-го столбца блоков в соответствии с их распределением по процессорам;

– рассылка из процессора, где хранится (I+1)-я строка блоков матрицы A, во все остальные процессоры диагональных элементов d_i ($i = Is+1,\ldots, Is+s$) и элементов (I+1)-й строки блоков матрицы L;

– проверка положительной определенности ($d_i > 0$) или невырожденности ($|d_i| > masheps$), $i = Is+1,\ldots, Is+s$, матрицы A.

Здесь в случае ленточной матрицы $m_i = \min\{ m, i-1 \}$, а в случае профильной матрицы $m_i = i-k$, где k – второй индекс первого (крайнего левого) элемента i-й строки матрицы.

Приведенная вычислительная схема позволяет сохранить одномерную блочно-циклическую схему распределение по процессорам элементов матриц L и D. Элементы результата разложения (матриц L и D) сохраняются в одном массиве: значение $1/d_i$ хранится на месте элемента $l_{i,i} \equiv 1$ матрицы L. Так как все элементы матриц L и D передаются в каждый из процессоров, то для этих матриц можно изменить (увеличить) количество строк в блоках – величину s. Если эта величина не изменяется, то результат может быть помещен на место исходной матрицы.

Правые части и решение задачи (1) распределяются циклически в соответствии с распределением матриц L и D. Параллельные алгоритмы решения систем (3) и (5) с треугольными матрицами для случая профильных матриц такие же, как в [6], но в вычислениях участвуют лишь ненулевые элементы матрицы. Решение системы (4) с диагональной матрицей выполняется без обменов, так как соответствующие компоненты диагональной матрицы, правых частей и решения хранятся в одном и том же процессоре.

Критериями эффективности параллельного алгоритма, как правило, служат коэффициенты ускорения S_p и эффективности E_p [1, 5]:

$$S_p = \frac{T_1}{T_p}, \quad E_p = \frac{S_p}{p}, \tag{7}$$

где T_1 – время решения задачи на одном процессоре, T_p – время решения той же задачи с использованием p параллельных процессов (процессоров). Оценив количество арифметических операций на одном процессоре в однопроцессорном и многопроцессорном вариантах и количество операций обмена и объем передаваемой и принимаемой информации, можно получить априорные оценки этих коэффициентов. В этих оценках используются величины: τ_c – отношение среднего времени синхронизации процессов при обмене к среднему времени выполнения процессором одной арифметической операции с плавающей запятой, τ_o – отношение среднего времени обмена одним машинным словом между двумя процессами к среднему времени выполнения процессором одной арифметической операции с плавающей запятой.

Эффективность параллельного алгоритма решения СЛАУ с ленточной или с профильной матрицей определяется в основном эффективностью алгоритма LDL^T-разложения ленточной симметричной матрицы, так как количество арифметических операций при факторизации матрицы оценивается величиной $O(nm^2)$, а при вычислении решения – $O(nm)$. Для ленточных матриц оценки коэффициентов эффективности приведены в [6]. Эти же оценки справедливы и для случая профильных матриц, если понимать под m введенную в предыдущем разделе величину c_e/n. Таким образом, ускорение и эффективность вышеприведенного параллельного алгоритма LDL^T-разложения при условиях, что $n \ge sp^2$ и $m \ge sp$, оценивается величинами

$$S_p \approx \frac{p(m+2)}{m+2+2s(p-1)+p\log_2 p\,\tau_{Bcast}}, \qquad E_p \approx \left(1+\frac{2s(p-1)}{m+2}+\frac{p\log_2 p}{m}\tau_{Bcast}\right)^{-1}, \tag{8}$$

где $\tau_{Bcast} = \tau_o + \frac{\tau_c}{sm}$. Оценки (8) свидетельствует, что коэффициенты ускорения и эффективности не зависят от порядка матрицы.

Параллельный блочный алгоритм метода Холецкого

Из оценок (8) следует, что коэффициенты ускорения и эффективности блочно-циклического алгоритма уменьшаются с уменьшением ширины ленты. Для узких ленточных матриц (когда ширина ленты намного меньше порядка матрицы) время решения СЛАУ при росте числа используемых процессоров практически не сокращается.

Оказалось, что для узких ленточных матриц целесообразно вернуться к вышеописанному нециклическому распределению элементов матрицы по процессорам. Такое распределение проводится путем виртуального

приведения узкой ленточной матрицы к блочно-диагональной матрице с окаймлением. Такой подход получил название “дели и побеждай” (divide-and-conquer) [7]. На рис. 4 показаны блоки распределенной в *k*-й процессор подматрицы матрицы A' преобразованной задачи (6); причем в нулевом процессоре отсутствует блок U_0, а в последнем – блоки C_{p-1} и S_{p-1}. На рис. 5 показана структура нижней треугольной матрицы разложения L, а на рис. 6 – структура вычисляемой в k-м процессоре подматрицы матрицы L (в нулевом процессоре присутствуют только блоки L_0, T_0, а в последнем отсутствуют блоки Q_{p-1} и T_{p-1}; блоки E_k и H_k участвуют только в промежуточных вычислениях и не входят в матрицу L). Важной особенностью LDL^T-разложения такой преобразованной матрицы является появление ненулевых блоков: Q_k – квадратного порядка m – и R_k – прямоугольного размера $m \times n_k$ – вместо треугольного блока U_k.

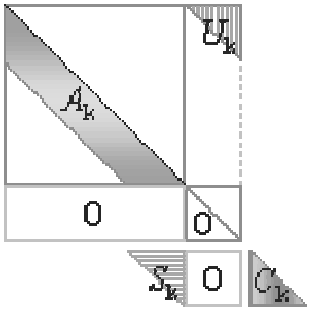

Рис. 4

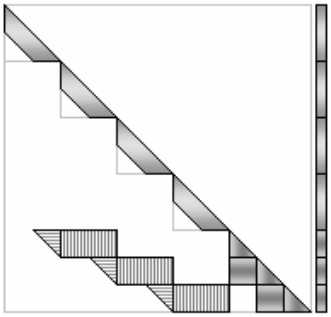

Рис. 5

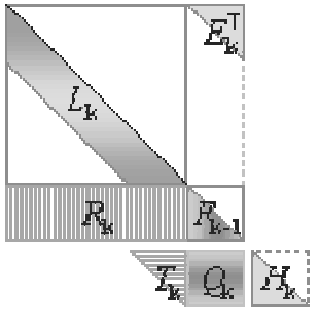

Рис. 6

Рассмотрим алгоритм LDL^T-разложения симметричной блочно-диагональной матрицы с окаймлением *n*-го порядка. Обозначим B_k – столбец блоков B_{ik}, а C – квадратный блок, состоящий из блоков C_k и V_{jk}. Тогда в соответствии с (2) имеем:

$$L_k D_k L_k^T = A_k, \quad R_k D_k L_k^T = B_k, \quad k = \overline{0, p-1}, \tag{9}$$

$$W_k = R_k D_k R_k^T, \quad k = \overline{0, p-1}. \tag{10}$$

Учитывая структуру блочно-диагональной матрицы с окаймлением, при вычислении блока разложения R_k используются только размещенные в том же процессоре, что и блок R_k, блок исходной матрицы B_k и блоки разложения L_k и D_k. При этом если является нулевой одна из строк блока B_k, то нулевой оказывается соответствующая строка блока R_k, что позволяет уменьшить количество арифметических операций для разложения исходной матрицы. Аналогичные замечания справедливы относительно промежуточного блока W_k из (10). Вычисления (9), (10) проводятся параллельно и без обменов. Кроме того, структура квадратных блоков A_k и прямоугольных блоков B_k может быть оптимизирована с целью минимизации в каждом процессоре количества арифметических операций для выполнения вычислений (9), (10).

Дальнейшие вычисления требуют выполнения операций обмена как для получения приведенной симметричной матрицы G, в общем случае плотной, порядок которой обозначим n_r, так и ее LDL^T-разложения:

$$G = C - \sum_{k=0}^{p-1} W_k, \quad F\overline{D}F^T = G. \tag{11}$$

Структура симметричной матрицы G и нижней треугольной матрицы ее LDL^T-разложения F определяется структурой исходной матрицы с учетом оптимизации структур блоков A_k и B_k ($k = \overline{0, p-1}$). Так в случае узкой ленточной исходной матрицы, т.е. решения системы (6), приведенная матрица G является блочно-трех-диагональной симметричной матрицей, порядок которой $n_r = (p-1)m$ (в этом случае все матрицы из (11) являются двумерными блочными и состоят из $(p-1)\times(p-1)$ квадратных блоков).

Если $n_r << n$, то распараллеливать процесс LDL^T-разложения приведенной матрицы нецелесообразно. В этом случае можно, например, сформировать приведенную матрицу в одном из процессоров и в нем же провести ее LDL^T-разложение.

В случае узкой ленточной исходной матрицы диагональные блоки $G_{k,k}$ приведенной матрицы – это разности расположенных в соседних процессорах блоков C_{k-1}-E_{k-1} и E_k, а поддиагональными являются блоки P_k (здесь блоки C_k и E_k являются блоками с индексами (k,k) матриц C и W_k соответственно, а блок P_k – блоком с индексом $(k-1,k)$ матрицы W_k). Тогда, если $n_r << n$, то далее последовательно для $k = \overline{1, p-1}$ в процессоре с логическим номером k выполняется:

– прием от процессора с логическим номером $k-1$ блока H_{k-1}, где $H_0 \equiv E_0$, $H_k = E_k - Q_k \overline{D}_{k-1} Q_k^T$, $k = \overline{1, p-2}$,

– факторизация диагонального блока $F_{k-1}\overline{D}_{k-1}F_{k-1}^T = H_{k-1} + G_k$,

– вычисление блока Q_k из $Q_k \overline{D}_{k-1} F_{k-1}^T = P_k$, если $k < p - 1$,

– вычисление блока H_k,

– передача блока H_k процессору с логическим номером $k + 1$, если $k < p - 1$.

В таком случае выполняется минимальное количество обменов, причем только между парами процессоров, логические номера которых отличаются на единицу. Такие обмены достаточно хорошо выполняются в большинстве используемых на MIMD-компьютерах коммуникационных средах.

Если условие $n_r << n$ не выполняется, то процесс разложения приведенной матрицы необходимо распараллеливать. При этом выбор параллельного алгоритма определяется как параметрами задачи (1), так и математическими и коммуникационными свойствами MIMD-компьютера, на котором эта задача решается.

Вычисление решения системы (1) или (6) с блочно-диагональной матрицей с окаймлением согласно (3)-(5) распадается на следующие подзадачи:

$$L_k z_k = b_k, \quad w_k = R_k z_k, \quad y_k = D_k^{-1} z_k, \quad k = \overline{0, p\text{-}1}, \tag{12}$$

$$g = \overline{b} - \sum_{k=0}^{p-1} w_k, \quad F\overline{z} = g, \quad \overline{y} = \overline{D}^{-1}\overline{z}, \quad F^T \overline{x} = \overline{y}, \tag{13}$$

$$L_k^T x_k = y_k - R_k^T \overline{x}, \quad k = \overline{p-1, 0}, \tag{14}$$

где b_k, x_k, y_k, z_k – блоки размера $n_k \times q$, а $\overline{b}, \overline{x}, \overline{y}, \overline{z}$ – блоки размера $n_r \times q$ правой части и решений систем (5), (4), (3) соответственно, g – правая часть приведенной системы $G\overline{x} = g$.

Аналогично разложению матрицы системы все вычисления (12) выполняются параллельно для каждого k в процессоре с логическим номером k и без обменов. После этого, как и в алгоритме разложения матрицы, формируется правая часть g и вычисляется решение $\overline{x}$ приведенной системы согласно (13). Полученное решение $\overline{x}$ или его части рассылается всем процессорам, а вычисления (14) для получения блоков решения x_k, как и (12), выполняются параллельно и без обменов для каждого k в процессоре с логическим номером k.

Алгоритм формирования правой части и вычисления решения приведенной системы аналогичен алгоритму LDL^T-разложении. Если $n_r << n$, то решение приведенной системы проводится или в одном процессоре после формирования в нем правой части g, или последовательно в процессоре с логическим номером k для $k = \overline{1, p-1}$ и $k = \overline{p-1, 1}$ с использованием обменов между парами процессоров, логические номера которых отличаются на единицу. В противном случае вычисления (13) необходимо распараллеливать и параллельные алгоритмы аналогичны используемому параллельному алгоритму LDL^T-разложении приведенной матрицы.

Как и в случае блочно-циклического алгоритма решение СЛАУ с ленточными или профильными матрицами эффективность рассматриваемого параллельного блочного алгоритма решения СЛАУ с блочно-диагональными матрицами с окаймлением определяется эффективностью LDL^T-разложения матрицы системы. Если блочно-диагональной матрицей с окаймлением является матрица задачи (1), то операции (9),(10) распараллеливаются полностью, т.е. для данных операций коэффициент эффективности распараллеливания равен 1. Коэффициент эффективности распараллеливания формирования и разложения матрицы приведенной системы, т.е. операций (11), зависит от выбранного алгоритма и всегда меньше 1. Коэффициенты ускорения и эффективности рассматриваемого параллельного блочного алгоритма LDL^T-разложения можно оценить так

$$S_p \approx p\left(1 + (p-1)\tau_n + p\tau_к\right)^{-1}, \qquad E_p \approx \left(1 + (p-1)\tau_n + p\tau_к\right)^{-1}, \tag{15}$$

где τ_n – отношение времени LDL^T-разложения матрицы приведенной системы на p процессорах ко времени LDL^T-разложения матрицы системы (1) на одном процессоре, $\tau_к$ – отношение времени затрачиваемого на операции обмена данными между p процессорами ко времени LDL^T-разложения матрицы системы (1) на одном процессоре.

Рассмотрим теперь оценки коэффициентов ускорения и эффективности рассматриваемого параллельного блочного алгоритма LDL^T-разложения узкой ленточной матрицы. Следствием переупорядочивания исходной матрицы является значительный рост числа арифметических операций – примерно в четыре раза. Поэтому коэффициент эффективности алгоритма не может превысить величины 0,25. В однопроцессорном случае количество арифметических операций при LDL^T-разложении оценивается величиной $nm^2 + O(nm)$, а при вычислении решения СЛАУ – величиной $2nmq + O(nq)$. Соответственно, в многопроцессорном случае количество арифметических операций, выполняемых одним процессором, оценивается как $4(n_k+m)m^2 + O(nm) + O(m^3)$ для LDL^T-разложении и $4(n_k+m)mq + O(nq) + O(m^2q)$ для вычисления решения. Кроме того, при использовании p процессоров для LDL^T-разложения выполняется $p-1$ передач и приемов по $(m+1)m/2$ 64-битовых машинных слов, а для вычисления решения $2(p-1)$ передач и приемов по mq слов. Тогда коэффициент ускорения для LDL^T-разложения оценивается величиной

$$S_{\text{p}} \approx \frac{p}{4}\left(1-\frac{1}{m+1}+(p-1)\left(\frac{(7p-18)m}{12n}+\frac{p}{4n}\left(\frac{\tau_{\text{c}}}{m^2}+\tau_{\text{o}}\right)\right)\right)^{-1}, \qquad (16)$$

а для вычисления решения

$$S_{\text{p}} \approx \frac{p}{2}\left(1-\frac{1}{8m+2}+(p-1)\left(\frac{(3p-4)m}{4n}+\frac{p}{2n}\left(\frac{\tau_{\text{c}}}{2mq}+\tau_{\text{o}}\right)\right)\right)^{-1}. \qquad (17)$$

Коэффициенты эффективности легко получить, используя второю формулу из (7).

Оценки (15) свидетельствуют, что в случае блочно-диагональной с окаймлением исходной матрицы основное влияние на эффективность рассматриваемого параллельного алгоритма оказывает формирование и решение приведенной системы. Если же исходная матрица является узкой ленточной, то повысить эффективность алгоритма можно за счет организации вычислений (в том числе обращений к памяти различного быстродействия) с прямоугольными блоками R_{k}, так как на эти вычисления приходится приблизительно ¾ всех арифметических операций.

Результаты численных экспериментов

Рассматриваемые параллельные алгоритмы метода Холецкого для решения СЛАУ с разреженными симметричными матрицами – блочно-циклические для ленточных или профильных матриц и блочный для узкой ленточной матрицы – программно реализованы на языке С в среде параллельного программирования MPI [8]. Для экспериментального исследования эффективности этих алгоритмов проведены численные эксперименты на MIMD-компьютере кластерного типа – интеллектуальной рабочей станции Инпарком-16 (разработка Института кибернетики им. В.М. Глушкова НАН Украины и ГНПП „Электронмаш“).

В работе [9] приведены некоторые результаты экспериментального исследования эффективности блочно-циклического алгоритма для ленточных матриц, которые хорошо согласуются с оценками (8).

Для экспериментального исследования эффективности блочно-циклического алгоритма для профильных матриц осуществлялось решение ряда задач с различной степенью заполненности матрицы ненулевыми элементами. Далее на диаграммах (рис. 7) показаны ускорения, получаемые при решении СЛАУ с профильной матрицей на различном количестве процессоров или при различном количестве строк в блоке. Параметры системы: порядок n = 44 436, максимальная полуширина ленты m = 4 476, заполненность ленты матрицы – 21 %.

Использование блочно-циклического алгоритма для профильных матриц вместо этого же алгоритма для ленточных матриц для решения СЛАУ, порядок которой 1 332 000, максимальная полуширина ленты 1 005, заполненность ленты матрицы 85 %, позволило сократить время решения в 1,32 раза.

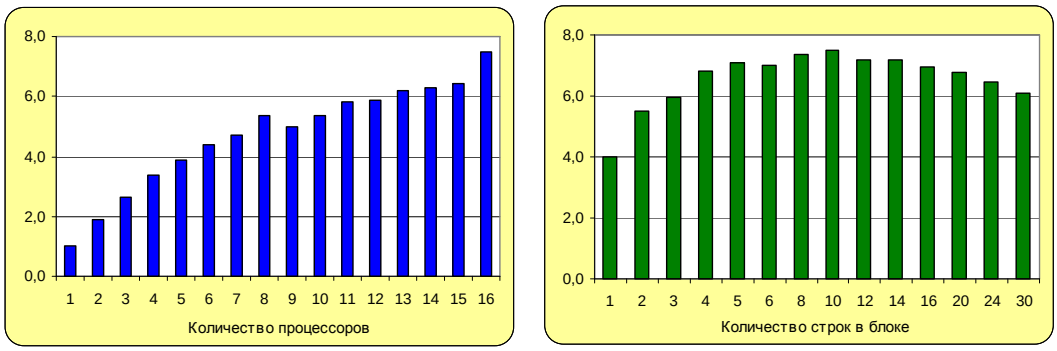

Рис. 7

Для экспериментального исследования эффективности блочного алгоритма решения СЛАУ с узкими ленточными матрицами осуществлялось решение нескольких СЛАУ с одной правой частью. На рис. 8 показаны графики зависимости от числа используемых процессоров ускорений, получаемых при решении четырех СЛАУ: задача 1 – n = 150 000, m = 151; задача 2 – n = 150 000, m = 301; задача 3 – n = 600 000, m = 151; задача 4 – n = 56 942, m = 143. На левой диаграмме (рис. 9) представлены отношения времени решения на 1, 8, 12 и 16 процессорах к времени решения на 4 процессорах одной и той же СЛАУ (q = 1, m = 151) при различных порядках матрицы. А на правой диаграмме (рис. 9) – отношения времени решения на 1, 8, 12 и 16 процессорах к времени решения на 4 процессорах одной и той же СЛАУ (q = 1, n = 600 000) при различных значениях полуширины ленты матрицы. Для некоторых вариантов решение на одном процессоре не проводилось, так как для размещения данных не хватало оперативной памяти процессора.

Полученные результаты хорошо согласуются с априорными оценками (16) и (17), что позволяет оценить значения выражений $\tau_{\text{c}} + m^2\tau_{\text{o}}$ и $\tau_{\text{c}} + 2mq\tau_{\text{o}}$ для конкретного MIMD-компьютера. В дальнейшем данные значения

могут использоваться при оптимизации количества процессоров для эффективного решения конкретной задачи с помощью рассматриваемого алгоритма, а также при выборе самого алгоритма решения. Так приведенные результаты свидетельствуют, что использование рассматриваемого алгоритма для решения задачи 2 малоэффективно. Для задач с таким соотношением порядка и полуширины ленты матрицы целесообразно использовать циклический алгоритм, например описанный в [6].

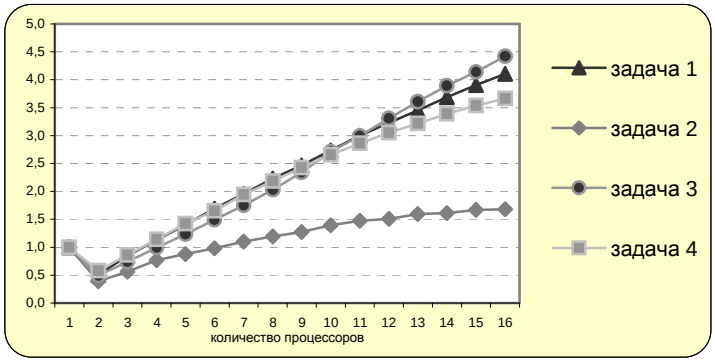

Рис. 8

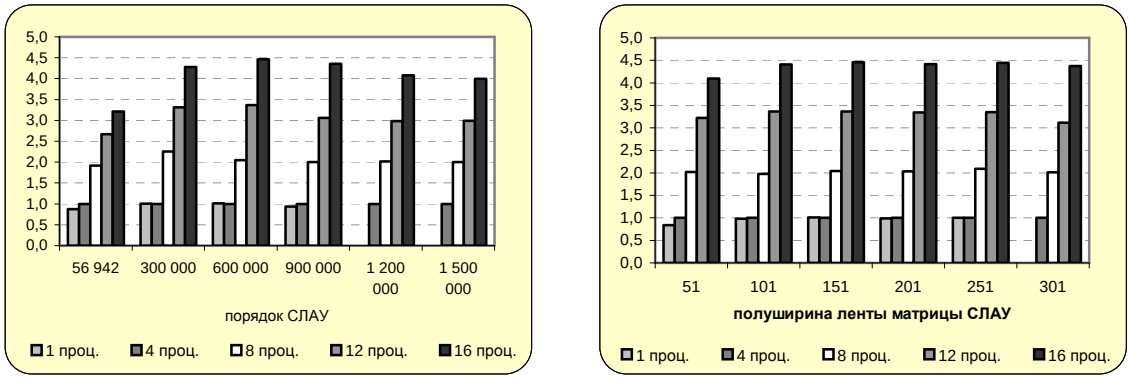

Рис. 9

Выводы

Рассмотренные в работе параллельные алгоритмы позволяют эффективно решать на MIMD-компьютере линейные системы с разреженными симметричными матрицы с регулярной структурой трех видов – ленточными с различной шириной ленты, профильными и блочно-диагональными с окаймлением. Для задач с такими матрицами можно уменьшить количество арифметических операций, если исключить вычисление всех априорно нулевых элементов нижней треугольной матрицы LDL^{T}-разложения исходной матрицы. Однако это часто приводит к существенному нарушению сбалансированности рассмотренных параллельных алгоритмов. Поэтому целесообразно продолжить разработку эффективных параллельных алгоритмов для решения СЛАУ с матрицами нерегулярной разреженной структуры, что позволит как расширить множество эффективно решаемых на MIMD-компьютере задач, так и повысить эффективность решения рассмотренных здесь задач.

1. *Молчанов И.Н., Химич А.Н., Попов А.В. и др.* Об эффективной реализации вычислительных алгоритмов на MIMD-компьютерах // Искусственный интеллект. – 2005. – № 3. – С. 175–184.
2. *Попов А.В., Химич А.Н.* Исследование и решение первой основной задачи теории упругости // Компьютерная математика. – 2003. – № 2. – С. 105–114.
3. *Молчанов И.Н., Попов А.В., Химич А.Н.,* Алгоритм решения частичной проблемы собственных значений для больших ленточных матриц // Кибернетика и системный анализ. – 1992. – № 2. – С. 141–147.
4. *Уилкинсон Дж. Х., Райнш К.* Справочник алгоритмов на языке Алгол. Линейная алгебра. – М.: Машиностроение, 1976. – 389 с.
5. *Численные* методы для многопроцессорного вычислительного комплекса ЕС / *В.С. Михалевич, Н.А. Бик, …, А.Н. Химич и др.* / Под редакцией И.Н. Молчанова. – М.: Изд. ВВИА им. Н.Е. Жуковского, 1986. – 401 с.
6. *Попов А.В., Химич А.Н.* Параллельный алгоритм решения системы линейных алгебраических уравнений с ленточной симметричной матрицей // Компьютерная математика. – 2005. – № 2. – С. 52–59.
7. http://www.netlib.org/scalapack
8. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – СПб.: БХП-Петербург, 2004. – 608 с.
9. *Зубатенко В.С., Майстренко А.С., Молчанов И.Н. и др.* Исследование некоторых параллельных алгоритмов решения задач линейной алгебры на MIMD-компьютерах // Искусственный интеллект. – 2006. – № 3. – С. 129–138.