2П

УДК 519.6

А.В. Попов
Институт кибернетики им. В.М. Глушкова НАН Украины, г. Киев,
dept150@insyg.kiev.ua, Украина

Решение задач линейной алгебры с разреженными симметричными матрицами на MIMD-компьютере

Рассматриваются блочные алгоритмы исследования и решения задач линейной алгебры с разреженными (узкими ленточными, с окаймлением и т.п.) симметричными матрицами на компьютерах MIMD-архитектуры. Исследуется эффективность рассматриваемых алгоритмов. Приводятся некоторые результаты численных экспериментов на MIMD-компьютере.

Введение

При решении научно-технических задач во многих случаях возникает необходимость решать задачу (или несколько задач) линейной алгебры. 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).

Матрицы таких дискретных задач имеют разреженную структуру и могут быть сведены путем оптимизации нумерации неизвестных к ленточным, профильным, блочно-трехдиагональным и т.п.. Во многих случаях матрицы дискретных задач симметричны и положительно определены или полуопределены.

Важной особенностью возникающих при дискретизации задач линейной алгебры является высокий порядок их матриц – от десятков и сотен тысяч 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.

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, о которых говорилось выше. Под эффективным алгоритмом решения понимается алгоритм, позволяющий получить достоверное решение задачи с минимальным использованием ресурсов компьютера – процессоров, оперативной памяти, времени. Эффективность параллельных алгоритмов оценивается, как правило, с помощью коэффициентов ускорения и эффективности [1].

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. На компьютерах

традиционной архитектуры хорошо зарекомендовали себя алгоритмы приведения матрицы задачи к одному из стандартных видов (например, к нижней или верхней треугольной матрице). Для таких алгоритмов характерен постепенно уменьшающий от шага к шагу размер обрабатываемой части матрицы. Достаточно хорошую сбалансированность загрузки процессоров обеспечивают параллельные версии этих алгоритмов, использующих т.н. циклические схемы распределения и обработки матриц [2]. Такие циклические алгоритмы позволяют добиться примерно равного объема вычислений и обменов в каждом процессоре и практически исключить влияние эффекта Гайдна.

В [3] предложен параллельный блочно-циклический алгоритм метода Холесского для решения системы линейных алгебраических уравнений (СЛАУ) с ленточными симметричными матрицами. Однако исследование этого алгоритма показало, что он эффективен при достаточно большой ширине ленты матрицы, а для узких ленточных матриц (когда ширина ленты намного меньше порядка матрицы) время решения СЛАУ при росте числа используемых процессоров практически не сокращается. Это связано со значительным уменьшением количества арифметических операций (пропорционально квадрату полуширины ленты) при сохранении общего количества обменов, что приводит к значительному снижению сбалансированности загрузки процессоров.

Оказалось, что для узких ленточных матриц целесообразно вернуться к нециклическому распределению блоков матрицы по процессорам. Такое распределение дополняется переупорядочением строк и столбцов матрицы. Этот подход получил название «дели и побеждай» (divide-and-conquer) [4]. Следствием переупорядочивания является значительный рост числа арифметических операций – примерно в 4 раза. Поэтому коэффициент эффективности алгоритма не может превысить величины 0,25.

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 [5] или частичной обобщенной алгебраической проблемы собственных значений [6]. Поэтому здесь исследуется параллельный блочный алгоритм метода Холесского из группы алгоритмов divide-and-conquer для разреженных симметричных матриц, которые могут быть сведены к матрицам с окаймлением.

Блочное представление разреженной матрицы

Для решения СЛАУ

$$Ax = b \tag{1}$$

с разреженной симметричной матрицей методом Холесского обычно путем перестановки строк и столбцов такую матрицу приводят к одному из стандартных видов: ленточному, профильному, с окаймлением 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, и более подробно остановимся на случае узкой ленточной матрицы.

Обозначим: n – порядок матрицы и СЛАУ, $2m+1$ – ширина ленты матрицы, q – количество правых частей СЛАУ, p – количество используемых для решения СЛАУ процессоров.

Для решения СЛАУ на параллельном компьютере ее матрица и правая часть должны быть распределены по процессорам. В рассматриваемом случае узкая ленточная матрица разбивается на p приблизительно равных блоков строк, каждый из которых помещается в отдельный процессор. Соответствующим образом распределяется по процессорам и правая часть. На рис. 1 представлено 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 4 блока (рис. 1): 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, что ленточные блоки были независимы один от другого. Заметим, что хранятся только диагональные и поддиагональные элементы матрицы.

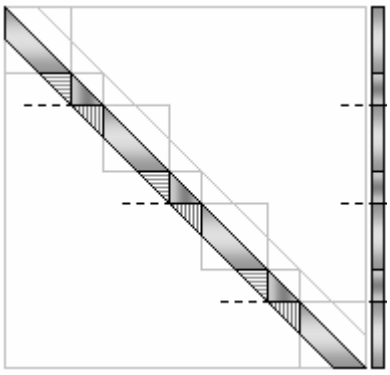

Рисунок 1

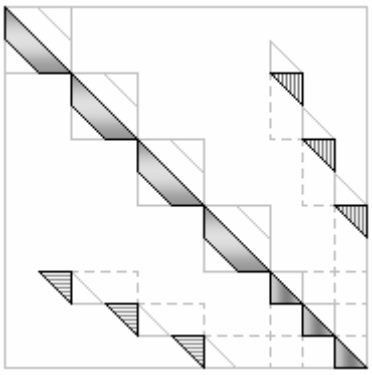

Рисунок 2

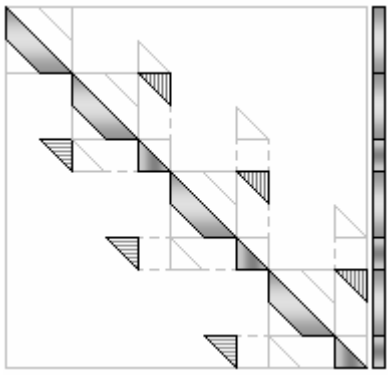

Рисунок 3

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

$$A'x' = b' \qquad (2)$$

с переупорядоченными матрицей $A' = PAP$, правой частью $b' = Pb$ и решением $x' = Px$. Матрица A' может быть блочно-диагональной с окаймлением (рис. 2) или иметь вид, представленный на рис. 3. Речь идет о виртуальном переупорядочивании, так как реально никакие перестановки не проводятся. Поэтому в дальнейшем не потребуется переупорядочивать полученное решение.

Если матрица задачи (1) сразу имеет вид, представленный на рис. 2 или рис. 3, то перестановки не проводятся, а задача (2) совпадает с задачей (1). В k-й процессор в этом случае в порядке нумерации помещается по одному блоку A_k, C_k, S_k и U_k (или U_k^T). В ряде случаев исходная матрица может быть приведена к матрице с окаймлением, например, к виду, представленному на рис. 4. При этом предполагается, что квадратные диагональные блоки A_k и C_k могут иметь разные порядки, но порядок любого блока C_k

намного меньше порядка любого блока A_j. Тогда в один процессор помещаются блоки с одним и тем же индексом, т.е. диагональные блоки A_k и C_k и ненулевые внедиагональные блоки S_k, T_k, U_k и F_k (для рис. 4).

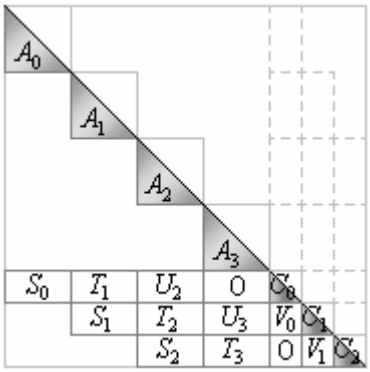

Рисунок 4

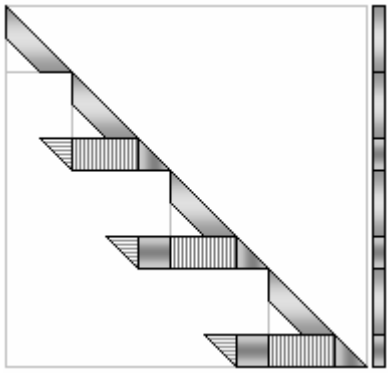

Рисунок 5

Параллельный блочный алгоритм метода Холесского

Так как переупорядоченная матрица A' сохраняет свойства исходной матрицы, в том числе симметричность, то для решения СЛАУ (2) можно использовать метод Холесского, а исходя из возможных приложений, его LDL^{T}-версию. Таким образом, решение задачи (1) можно получить, выполнив следующие шаги:

– виртуальное переупорядочение матрицы и правой части $A' = PAP$, $b' = Pb$;

– факторизация матрицы системы (2)

$$LDL^{\mathsf{T}} = A'; \tag{3}$$

– вычисление решения системы (2)

$$Lz = b', \quad y = D^{-1}z, \quad L^{\mathsf{T}}x' = y; \tag{4}$$

– виртуальное переупорядочение решения $x = Px'$.

Еще раз заметим, что первый и последний шаги являются виртуальными и в действительности не выполняются. Структура матрицы L представлена на рис. 5.

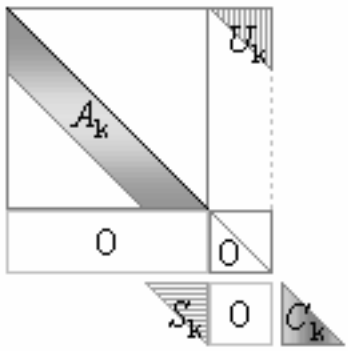

Рисунок – 6

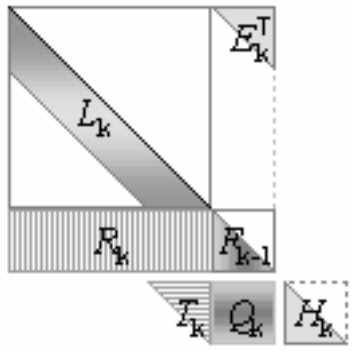

Рисунок – 7

Рассмотрим алгоритм LDL^{T}-разложения симметричной матрицы n-го порядка PAP. Структура распределенной в k-й процессор подматрицы матрицы A' представлена на рис. 6, причем в нулевом процессоре отсутствует блок U_0, а в последнем – блоки C_{p-1} и S_{p-1}. На рис. 7 представлена 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 рассматриваемого алгоритма является появление ненулевых блоков Q_k – квадратного порядка m и R_k – прямоугольного размера $m \times n_k$ вместо треугольного блока U_k.

Таким образом, исходя из (3), имеем:

$$L_k D_k L_k^T = A_k, \quad k = \overline{0, p-1}, \qquad T_k D_k L_k^T = S_k, \quad k = \overline{0, p-2} \tag{5}$$

$$R_k D_k L_k^T = \breve{U}_k^T, \quad k = \overline{1, p-1}, \tag{6}$$

$$V_k = -T_k D_k R_k^T, \quad Q_k \overline{D}_{k-1} F_{k-1}^T = V_k, \quad k = \overline{1, p-2}, \tag{7}$$

$$\begin{gathered} G_{k+1} = -R_{k+1} D_{k+1} R_{k+1}^T, \quad E_k = C_k - T_k D_k T_k^T, \quad k = \overline{0, p-2}, \\ H_0 \equiv E_0, \quad H_k = E_k - Q_k \overline{D}_{k-1} Q_k^T, \quad k = \overline{1, p-2}, \\ F_{k-1} \overline{D}_{k-1} F_{k-1}^T = H_{k-1} + G_k, \quad k = \overline{1, p-1}, \end{gathered} \tag{8}$$

где D_k и $\overline{D}_k$ – диагональные матрицы, а элементы блоков $\breve{U}_k$ определяются формулой $\breve{u}_{i,j}^{(k)} = \left\{ u_{i,j}^{(k)}, i \le j \mid 0, i > j \right\}$. Анализ этих формул показывает, что в разложении (5) блока A_k и в вычислении блоков T_k из (5), R_k из (6), V_k из (7), G_k и E_k из (8) участвуют блоки исходной и факторизованной матриц, расположенные в одном процессоре. Поэтому они вычисляются в процессорах, где они хранятся, параллельно и без обменов.

После этого образуется приведенная блочно-трехдиагональная симметричная матрица, порядок которой $(p-1)m$. Диагональные блоки ее – это суммы $E_{k-1} + G_k$, $k = 1, p-1$, расположенных в соседних процессорах блоков, а поддиагональными являются блоки V_k из (7). Следовательно, по процессорам оказываются распределенными не только элементы приведенной матрицы, но и составляющие их слагаемые. Если порядок этой матрицы $(p\text{-}1)m << n$, то распараллеливать процесс LDL^T-разложения приведенной матрицы нецелесообразно, а можно провести его следующим образом: последовательно для $k = \overline{1, p-1}$ выполняется:

- прием от процессора с логическим номером $k-1$ блока H_{k-1} из (8),
- факторизация диагонального блока согласно последней строки в (8),
- вычисление блока Q_k из (7),
- вычисление блока H_k из (8),
- передача блока H_k из (8) процессору с логическим номером $k+1$ ($k = \overline{0, p-2}$).

При таком подходе 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 сред.

Если условие $(p\text{-}1)m << n$ 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 (2) согласно формулам (4) распадается на три подзадачи.

Блочный алгоритм решения системы $Lz = b'$ аналогичен алгоритму разложения матрицы A' и записывается следующим образом:

$$L_k Z_k = B_k, \quad k = \overline{0, p\text{-}1}, \tag{9}$$

$$\begin{gathered} c_k = b_k - T_k \hat{z}_k, \quad g_{k+1} = -R_{k+1} Z_{k+1}, \quad k = \overline{0, p-2}, \\ h_0 \equiv c_0, \quad h_k = c_k - Q_k z_{k-1}, \quad k = \overline{1, p-2}, \\ F_{k-1} z_{k-1} = h_{k-1} + g_k, \quad k = \overline{1, p-1}, \end{gathered} \tag{10}$$

где B_k, Z_k – блоки размера $n_k \times q$ правой части и решения соответственно, b_k, z_k – блоки размера $m \times q$ правой части и решения соответственно, $\hat{z}_k$ – подблок блока Z_k размера $m \times q$, содержащий последние m элементов каждого вектора решения из Z_k. Вычисление блоков Z_k, c_k и g_k проводится параллельно без обменов, а блоков z_{k-1} и h_k по такому алгоритму: последовательно для $k = \overline{1, p-1}$ выполняется:

– прием от процессора с логическим номером $k-1$ блока h_{k-1} из (10),
– вычисление блока z_{k-1} согласно (10),
– вычисление блока h_k из (10),
– передача блока h_k процессору с логическим номером $k+1$ ($k = \overline{0, p-2}$).

Такой непараллельный алгоритм целесообразно использовать всегда, так как количество арифметических операций здесь относительно невелико и распараллеливание оказывается неэффективным.

Вычисление $y = D^{-1} z$ полностью распараллеливается (без обменов), так как части диагональной матрицы D хранятся в тех же процессорах, что и соответствующие им блоки y и z, и выполняется по формулам

$$Y_k = D_k^{-1} Z_k, \quad k = \overline{0, p\text{-}1}; \qquad \hat{y}_k = \hat{q}_{k-1} \hat{z}_k, \quad k = \overline{0, p\text{-}2}.$$

Блочный алгоритм обратного хода, т.е. решения системы $L^{\mathrm{T}} x' = y$ записывается так

$$\begin{gathered} w_{p-1} = y_{p-2}, \quad w_k = y_{k-1} - Q_k^{\mathrm{T}} x_k, \quad k = \overline{p\text{-}2, 1}, \\ F_k^{\mathrm{T}} x_k = w_{k+1}, \quad k = \overline{p\text{-}2, 0}, \end{gathered} \tag{11}$$

$$\begin{gathered} W_k = Y_k - R_k^{\mathrm{T}} x_{k-1}, \quad k = \overline{p-1, 1}, \quad W_0 \equiv Y_0, \\ \hat{W}_{p-1} \equiv W_{p-1}, \quad \hat{W}_k = W_k - T_k^{\mathrm{T}} x_k, \quad k = \overline{p-2, 0}, \\ L_k^{\mathrm{T}} X_k = \hat{W}_k, \quad k = \overline{p-1, 0}, \end{gathered} \tag{12}$$

где X_k – блоки размера $n_k \times q$, а x_k – блоки размера $m \times q$ решения x' задачи (2), элементы блоков $\hat{T}_k$ (размера $n_k \times m$) определяются формулой $\hat{t}_{i,j}^{(k)} = \left\{ t_{i,j}^{(k)}, i \le j - n_k + m \mid 0, i > j - n_k + m \right\}$. Вначале последовательно для $k = \overline{p-2, 0}$ выполняется

– прием процессором с логическим номером k от процессора с логическим номером $k+1$ блока x_k из (11),
– вычисление в процессоре с логическим номером k блока w_k согласно (11) ($k = \overline{p-2, 1}$),
– вычисление в процессоре с логическим номером $k+1$ блока x_k из (11),

– передача из процессора с логическим номером $k+1$ блока x_k процессору с логическим номером k.

После этого параллельно без обменов в каждом процессоре по формулам (12) вычисляется соответствующий блок X_k решения системы (2).

Если исходная матрица является матрицей с окаймлением, например, имеет вид, аналогичный рис. 4, то соответствующим образом изменяются формулы (6)–(8) для вычисления LDL^T-разложения матрицы, а также формулы (10) и (12) для вычисления решения системы. При этом важным обстоятельством является то, что каждый прямоугольный блок типа блока R_k вычисляется по аналогичной (6) формуле и никаких дополнительных операций не требуется. Однако возникает необходимость в дополнительных обменах для формирования распределенной по процессорам приведенной системы.

Эффективность блочного алгоритма метода Холесского

Критериями определения эффективности параллельного алгоритма, как правило, служат коэффициенты ускорения S_p и эффективности E_p [1], [2]:

$$S_p = \frac{T_1}{T_p}, \quad E_p = \frac{S_p}{p}, \tag{13}$$

где T_1 – время решения задачи на одном процессоре, T_p – время решения той же задачи с использованием p 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 процессоров при обмене к среднему времени выполнения процессором одной арифметической операции с плавающей запятой, τ_o – отношение среднего времени обмена одним машинным словом между двумя процессорами к среднему времени выполнения процессором одной арифметической операции с плавающей запятой.

Остановимся на априорных оценках коэффициента ускорения для рассматриваемого блочного алгоритма решения СЛАУ с узкой ленточной симметричной матрицей. В этом случае при LDL^T-разложении переупорядоченной матрицы появляются дополнительные блоки R_k и Q_k, что влечет увеличение количества арифметических операций приблизительно в 4 раза при разложении матрицы и в 2 раза при вычислении решения системы. Таким образом, 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-разложения и $4(n_k + m)mq + O(nq) + O(m^2 q)$ 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

$$S_{\text{p}} \approx p\left(\left(\frac{2m+1}{m+1}\right)^2 + (p-1)\left(\frac{(7p-18)m}{3n} + \frac{p}{n}\left(\frac{\tau_{\text{c}}}{m^2} + \tau_{\text{o}}\right)\right)\right)^{-1}, \quad (14)$$

а для вычисления решения

$$S_{\text{p}} \approx p\left(\frac{8m+1}{4m+1} + (p-1)\left(\frac{(3p-4)m}{2n} + \frac{p}{n}\left(\frac{\tau_{\text{c}}}{2mq} + \tau_{\text{o}}\right)\right)\right)^{-1}. \quad (15)$$

Коэффициенты эффективности легко получить, используя вторую формулу из (13).

В оценках коэффициентов ускорения и эффективности учитывается количество операций, но скорость выполнения арифметических операций зависит от организации вычислений, например, от времени обращения к памяти. Это время существенно отличается для обращения к основной памяти, к кэш-памяти и к регистрам процессора. Поэтому при программной реализации алгоритма особое внимание должно быть уделено организации вычислений с прямоугольными блоками R_{k}, на которые приходится около 75 % арифметических операций.

Если $m >> q$ и проводится однократное решение СЛАУ, то основной объем вычислений приходится на LDL^{T}-разложения матрицы системы и его эффективность практически совпадает с эффективностью решения задачи в целом. Если же при решении задачи проводится однократное LDL^{T}-разложение матрицы и многократное вычисление решений при различных правых частях, например, как при решении алгебраической задачи на собственные значения методом итераций на подпространстве, то на эффективность решения всей задачи обе части алгоритма оказывают примерно одинаковое влияние.

Если исходным представлением матрицы является блочно-диагональное с окаймлением, т.е. переупорядочивание не проводится, то в оценках (14), (15) первые слагаемые становятся близкими к единице и основное влияние на эффективность алгоритма оказывает формирование и решение приведенной системы.

Результаты численных экспериментов

Рассматриваемый параллельный блочный алгоритм метода Холесского для решения СЛАУ с узкой ленточной симметричной матрицей программно реализован на языке С в среде параллельного программирования MPI [7]. Для экспериментального исследования эффективности этого алгоритма проведены численные эксперименты на MIMD-компьютере кластерного типа – интеллектуальной рабочей станции «Инпарком-16» (разработка Института кибернетики им. В.М. Глушкова НАНУ и ГНПП «Электронмаш»). Проводилось решение СЛАУ на различном количестве процессоров для различных параметров (порядок и ширина ленты) матрицы.

На рис. 8 представлены графики зависимости от числа используемых процессоров ускорений, получаемых при решении четырех СЛАУ. Решались СЛАУ с одной правой частью и со следующими параметрами матрицы:

Таблица 1

	задача 1	задача 2	задача 3	задача 4
n	150 000	150 000	600 000	56 942
m	151	301	151	143

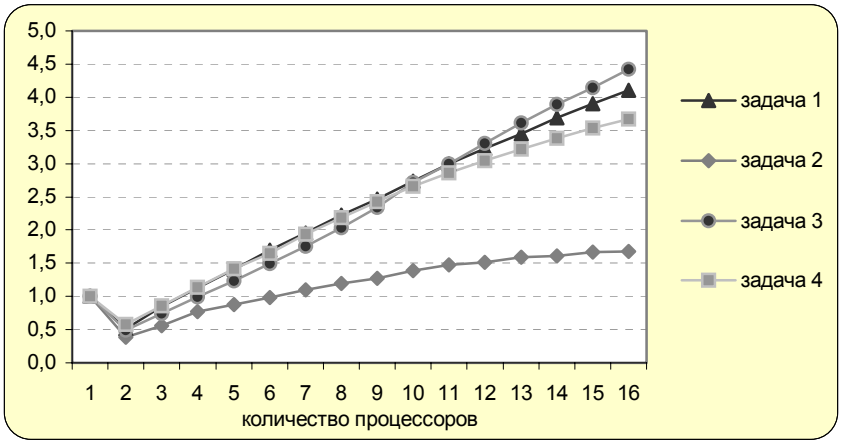

Рисунок – 8

На диаграмме (рис. 9) представлены отношения времени решения на 1, 8, 12 и 16 процессорах ко времени решения на 4 процессорах одной и той же СЛАУ ($q = 1$, $m = 151$) при различных порядках матрицы. А на диаграмме (рис. 10) – отношения времени решения на 1, 8, 12 и 16 процессорах ко 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.

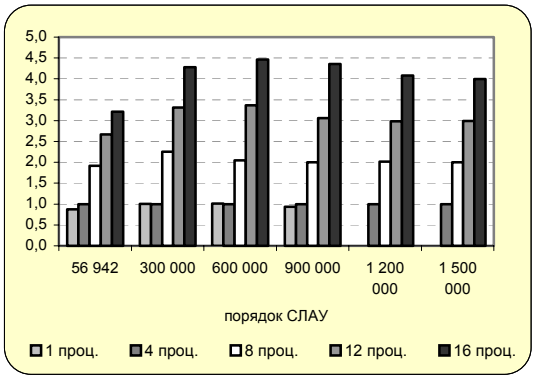

Рисунок – 9

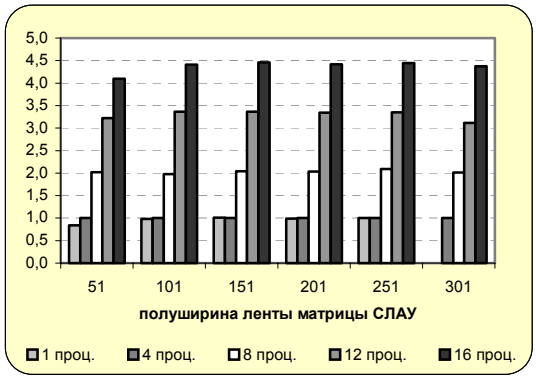

Рисунок - 10

Полученные результаты хорошо согласуются с априорными оценками (14) и (15), что позволяет оценить значения выражений $\tau_c + m^2\tau_o$ и $\tau_c + 2mq\tau_o$ для конкретного MIMD-компьютера. В дальнейшем эти 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 [3]. Так для задачи, порядок которой 600 000, а полуширина ленты 501, использование циклического алгоритма сокращает время решения на 8 процессорах почти в 3 раза. В то же время для задачи того же порядка, но с полушириной ленты матрицы, равной 151, циклический алгоритм работает в 2,4 раза медленнее (также на 8 процессорах).

Выводы

Проведенные исследования показывают целесообразность разработки нециклических параллельных алгоритмов для решения задач линейной алгебры с узкими ленточными матрицами. С целью повышения их эффективности особое внимание при программной реализации необходимо уделить организации вычислений с прямоугольными блоками R_k, а также организации решения приведенной системы. Если MIMD-компьютер состоит из достаточно большого числа процессоров, то возникает необходимость в распараллеливании процесса решения приведенной системы.

При решении задач с разреженными матрицами целесообразно еще до формирования матрицы так нумеровать неизвестные, чтобы матрица сразу получалась блочно-диагональной с окаймлением (например, как на рис. 2 или рис. 4). Причем эта нумерация должна обеспечивать минимизацию общего количества арифметических операций при решении задачи.

Литература

1. Молчанов И.Н., Химич А.Н., Попов А.В. Об эффективной реализации вычислительных алгоритмов на MIMD-компьютерах // Искусственный интеллект. – 2005. – № 3. – С. 175-184.
2. Численные методы для многопроцессорного вычислительного комплекса ЕС / В.С. Михалевич, Н.А. Бик, Б.Н. Брусникин, А.Н. Химич / Под редакцией И.Н. Молчанова. – М.: Издание ВВИА им. проф. Н.Е. Жуковского, 1986. – 401 с.
3. Попов А.В., Химич А.Н. Параллельный алгоритм решения системы линейных алгебраических уравнений с ленточной симметричной матрицей // Компьютерная математика. – 2005. – № 2. – С. 52-59
4. http://www.netlib.org/scalapack
5. Попов А.В., Химич А.Н. Исследование и решение первой основной задачи теории упругости // Компьютерная математика. – 2003. – № 2. – С. 105-114.
6. 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.
7. Воеводин В.В., Воеводин В.Вл. Параллельные вычисления. – СПб.: БХП-Петербург, 2004. – 608 с.

О.В. Попов

Розв'язування задач лінійної алгебри з розрідженими симетричними матрицями на MIMD-комп'ютері

Розглядаються блочні алгоритми дослідження та розв'язування задач лінійної алгебри з розрідженими (вузькими стрічковими, з обрамленням і т.п.) симетричними матрицями на комп'ютерах MIMD-архітектури. Досліджується ефективність даних алгоритмів. Наводяться деякі результати чисельних експериментів на MIMD-комп'ютері.

A.V. Popov

The Solving of Linear Algebra Problems with Sparse Symmetric Matrices on MIMD-Computer

Block algorithms for the investigating and solving of linear algebra problems with sparse (narrow band, with bordering, etc.) symmetric matrices on 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.

Статья поступила в редакцию 27.06.2006.