УДК 519.852:519.876

В. А. Богаенко, канд. техн. наук,

В. И. Кудин, д-р техн. наук

Институт кибернетики имени В.М. Глушкова НАН Украины, г. Киев

О СВОЙСТВАХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СХЕМ МЕТОДА БАЗИСНЫХ МАТРИЦ

В работе рассматриваются параллельные алгоритмы анализа плохообусловленных СЛАУ большого размера, построенные на основе метода базисных матриц. Подаются данные вычислительных экспериментов по сравнению характеристик параллельных алгоритмов метода базисных матриц с алгоритмами, реализованными в пакете Scalapack.

Ключевые слова: *анализ СЛАУ, параллельные алгоритмы, метод базисных матриц.*

Математические модели реальных процессов массопереноса, которые описываются уравнениями с частными производными, характеризуются плохой обусловленностью, а зачастую и большой размерностью. Такое свойство матрицы ограничений (после нахождения дискретного аналога непрерывной модели) коррелирует в задачах геогидродинамики со значением коэффициента конвективной диффузии [1]. При использовании большинства методов решения систем линейных алгебраических уравнений (СЛАУ) большого размера свойство плохой обусловленности существенно влияет на характеристики получаемого решения (точность, величину невязок) [2—6]. Подобные проблемы кроме как в задачах математического моделирования физических процессов, возникают и при решении некоторых задач экономического моделирования.

В настоящее время при решении задач большого размера весьма активно используется методология параллельных вычислений [2—3, 9] для которых основными показателями качества являются [3]: ускорение (speedup) — сокращение времени решения задачи; эффективность (efficiency) и другие, такие как стоимость (cost) и масштабируемость (scalability).

Критерий эффективности можно рассматривать как оценку распараллеливания конкретного алгоритма, в то время, как другой важный показатель состоит в построении оценок максимально возможного ускорения процесса решения задачи конкретного типа (оценка эффективности распараллеливания решения задачи).

Ускорение, получаемое при использовании параллельного алгоритма для p процессов по сравнению с последовательным вариантом выполнения вычислений, определяется величиной

© В. А. Богаенко, В. И. Кудин, 2009

$$s_p(n) = T_1(n) / T_p(n),$$

где $T_1(n)$ — время выполнения алгоритма при использовании одного процессора (последовательный алгоритм), n — размерность задачи (количество входной информации).

Эффективность использования параллельных алгоритмов при решении задачи определяется соотношением

$$E_p(n) = T_1(n)/(pT_p(n)) = S_p(n)/p.$$

Величина эффективности определяет среднюю долю времени выполнения алгоритма, когда процессоры реально задействованы для решения задачи.

Существует ряд разнообразных принципов построения и максимизации эффективности параллельных вычислений. Следует отметить, что повышение качества параллельных вычислений по одному из показателей (ускорение или эффективность) могут привести к ухудшению характеристик алгоритма по другому показателю. Разработка метода параллельных вычислений предполагает выбор некоторого непротиворечивого варианта с учетом желаемых показателей ускорения и эффективности.

Целью настоящей работы является: построение эффективной параллельной вычислительной схемы метода базисных матриц (МБМ) [6]; разработка программного обеспечения для проведения вычислительного эксперимента на типовых моделях; проверка достоверности вычислений [7], сравнение основных параметров (ускорение, эффективность) алгоритма с реализованными в пакете Scalapack [8] алгоритмами LU и SVD декомпозиции [5, 7].

В частности, к задаче обращения квадратной (или прямоугольной) матрицы общего вида сводится решение краевых задач для эллиптических и параболических уравнений с постоянными коэффициентами методом функций Грина и граничных элементов. Модели матриц ограничений СЛАУ таких задач характеризуются большим размером и плохой обусловленность и имеют вид

$$Au = C, \qquad (1)$$

где матрица A (со строками $a_1, a_2, \ldots, a_m$, $a_j = (a_{j1}, a_{j2}, \ldots, a_{jm})$, $j = \overline{1,m}$) — квадратная матрица размерности $(m \times m)$, в которой $u = (u_1, u_2, .., u_m)^T$ и вектор ограничений $C = (c_1, c_2, ..., c_m)^T$ имеют размерность m.

В основу МБМ положена идея базисной матрицы. Базисные матрицы в ходе итераций изменяются последовательным замещением строк некоторой вспомогательной базисной матрицы (вспомогательной СЛАУ) строками ограничений (1).

Введем в рассмотрение векторы $a_{i1}, a_{i2}, \dots, a_{im}$ — нормали ограничений (в дальнейшем будем называть строками), $a_j u^T \le c_j$, $j \in J_б$, которые образуют матрицу $A_б$, где $J_б = \{i_1, i_2, \dots, i_m\}$ — индексы ограничений.

Определение 1. Квадратную матрицу $A_б$, составленную из m линейно независимых строк некоторой вспомогательной СЛАУ, будем называть искусственной базисной, а решение u_0 соответствующей ей системы уравнений $A_б u = C^0$, где $C^0 = (c_{i_1}, c_{i_2}, \dots, c_{i_m})^T$ — искусственным базисным решением.

Определение 2. Две базисных матрицы, отличающиеся одной строкой, будем называть смежными.

Пусть e_{ri} — элементы матрицы $A_б^{-1}$, обратной к $A_б$, $e_k = (A_б^{-1})_k$ — k-ый столбец обратной матрицы, $u_0 = (u_{01}, u_{02}, \dots, u_{0m})^T$ — базисное решение, $\alpha_r = (\alpha_{r1}, \alpha_{r2}, \dots, \alpha_{rm})$ — вектор разложения нормали ограничения $a_r u \le c_r$ по строкам базисной матрицы $A_б$, $\Delta_r = a_r u_0 - c_r$ — невязка r-го ограничения в u_0. Для строки a_l (нормаль ограничения $a_l u \le c_l, l \notin J_б$), $\alpha_l = (\alpha_{l1}, \alpha_{l2}, \dots, \alpha_{lm})$ — вектор разложения строки a_l по строкам базисной матрицы $A_б$, находится из соотношения $a_l = \alpha_l A_б$.

Между коэффициентами разложения нормалей ограничений (1) по строкам искусственной базисной матрицы, элементами обратных матриц, базисными решениями, невязками ограничений (1) в двух смежных базисных матрицах при замене k-ой строки в базисной матрице $A_б$ строкой a_l имеют место связывающие соотношения [9]:

$$\bar{\alpha}_{rk} = \frac{\alpha_{rk}}{\alpha_{lk}}, \quad \bar{\alpha}_{ri} = \alpha_{ri} - \frac{\alpha_{rk}}{\alpha_{lk}}\alpha_{li}, \quad r = \overline{1,m}; \quad i = \overline{1,m}; \quad i \ne k; \tag{2}$$

$$\bar{e}_{rk} = \frac{e_{rk}}{\alpha_{lk}}, \quad \bar{e}_{ri} = e_{ri} - \frac{e_{rk}}{\alpha_{lk}}\alpha_{li}, \quad r = \overline{1,m}; \quad i = \overline{1,m}; \quad i \ne k; \tag{3}$$

$$\bar{u}_{0j} = u_{0j} - \frac{e_{jk}}{\alpha_{1k}}\Delta_l, \quad r = \overline{1,m}; \tag{4}$$

$$\bar{\Delta}_k = -\frac{\Delta_l}{\alpha_{lk}}, \quad \bar{\Delta}_r = \Delta_r - \frac{\alpha_{rk}}{\alpha_{lk}}\Delta_l, \quad r = \overline{1,n}; \quad r \ne k; \tag{5}$$

причем условием невырожденности базисной матрицы при замещении строкой a_l k-ой строки базисной матрицы $A_б$ является выполнение $\alpha_{lk} \ne 0$.

Для существования единственного решения (1) необходимо и достаточно, чтобы $\alpha_{lk}^{(i)} \neq 0, \quad i = \overline{1,m}$, где $\alpha_{lk}^{(i)}$ ведущие элементы итерации замещения строк базисной матрицы (2) нормалями ограничений (1).

Матрица А системы (1) невырождена, если $\alpha_{lk}^{(i)} \neq 0, \quad i = \overline{1,m}$.

Нетрудно убедиться, что в векторном виде формулы (3) можно представить как

$$\bar{e}_k = \frac{e_k}{\alpha_{lk}}, \; \bar{e}_i = e_i - \frac{e_k}{\alpha_{lk}}\alpha_{li} \quad i = \overline{1,m}; \; i \neq k;$$

Последовательный алгоритм. Ниже приведены основные стадии алгоритмической схемы нахождения величины машинного ранга, базисной матрицы и решения системы (1) на основе известных свойств тривиальной СЛАУ той же размерности:

– проводим симплексные итерации по замещению строк тривиальной базисной матрицы с известными элементами метода строками ограничений системы (1), согласно соотношениям (2)—(5) [9];

– проверяем выполнение условий невырожденности на каждой итерации;

– находим соответствующие элементы метода: вектора разложения по строкам базисных матриц ограничений (1), обратную базисную матрицу, невязки, искусственные базисные решения $u_0^{(r)}$, где k — номер итерации;

– контролируем количество итераций k замещения строк вспомогательной системы строками основной системы (1) для которых выполняются условия невырожденности. Если количество итераций, для которых $\alpha_{kk}^{(k)} \neq 0$, равно m, находим единственное решение из соотношения: $A_{\delta}^{-1} \cdot C = u^0$. В противном случае при выполнении условия $k < m$ для СЛАУ (1) не выполняется условие единственности решения. Модель требует дальнейшего анализа разрешимости задачи.

Не трудно убедиться, что вычислительный процесс пересчета обратной матрицы представляет собой проведение двух стадий матричных операций:

– первая, деление ведущего *k*-го столбца $e_k = (A_{\delta}^{-1})_k$ на значение ведущего элемента $\alpha_{kk}^{(k)} \neq 0$ ($e_k^{k+1} = e_k^k / \alpha_{kk}$);

– вторая, вычитание на *k*-й итерации от *i*-го столбца обратной матрицы ($i \neq k$, $i \in I$) нового ведущего столбца умноженного на α_{ki} $e_i^{k+1} = e_i^k - e_k^{k+1} \times \alpha_{ki}$.

Такая особенность обосновывает построение параллельных вычислений распределением по независимым процессам столбцов обратной матрицы (естественный параллелизм алгоритма).

Параллельный алгоритм с одномерным распределением данных.

Рассмотрим вариант параллельного алгоритма метода базисных матриц в допущении, что обратная матрица A^{-1} вычисляется так, чтобы она была распределена на вычислительной системе блоками столбцов A_i^{-1}, $i \in P = \{1,2,...N\}$, где $A^{-1} = \left(A_1^{-1}, A_2^{-1}, ..., A_p^{-1}\right)$, $J = \bigcup_{r=1}^{P} J_r$, J_r —подмножество индексов разбиения J.

Алгоритм, который работает по схеме с одним управляющим процессом-менеджером, опишем следующим образом:

Менеджер	**Серверы**
Вычисление J_r, $r \in P$	Инициализация — распределение информации по процессам
Итерации $l = \overline{1,m}$	
Выбор l, рассылка a_l по процессам	Прием a_l, вычисление $\alpha_i^l = a_l A_i^{-1}$, $i \in P$, $\alpha_l = (\alpha_1^l, \alpha_2^l, ..., \alpha_p^l)$
Прием e_l^{l+1} от одного из серверов	Вычисление $e_l^{l+1} = e_l^l / \alpha_{ll}$ на сервере i для которого $l \in J_i$
Рассылка e_l^{l+1} по серверам	Вычисление $e_j^{l+1} = e_j^l - e_l^{l+1}\alpha_{lj}$, $j \neq l, j \in J_r$, $r \in P \pm$
Сбор результатов $l = m+1$	
Выбор C, разбиение $C = (C_1, C_2, ..., C_p)$ рассылка по процессам	Прием C_r, $r \in P$ по процессам, вычисление $u_0^r = A_r^{-1} C_r$, $r \in P$
Прием u_0^r, вычисление $u^0 = \sum_{r \in P} u_0^r$	

Алгоритм имеет следующие свойства:

1) Отсутствует обмен данными между серверами;

2) Операции обмена — широковещательные передачи данных (broadcast);

3) Обратная матрица хранится по столбцам, в то время как исходная матрица — по строкам;

4) Выделение отдельного процесса-менеджера связано только с необходимостью инициализации системы и рассылки-сбора данных.

Время выполнения последовательного алгоритма для задачи размерностью n можно оценить как:

$$T^0(n) = \sum_{i=1}^{n} \big(2ni + n)\big)t_c = (n^3 + 2n^2)t_c, \tag{6}$$

где t_c — время выполнения операции умножения.

Время выполнения параллельного алгоритма на системе с N процессоров для задачи размерности n можно оценить как:

$$T_N(n) = \sum_{i=1}^{n} \left(\frac{n(i+1)}{N} + i + \frac{i(n-1)}{N} \right) t_c + (2k+1)n^2 t_s + 3nt_l =$$

$$= \left(\frac{n^3 + \frac{3}{2}n^2 - \frac{1}{2}n}{N} + \frac{1}{2}n^2 + \frac{1}{2}n \right) t_c + (2k+1)n^2 t_s + 3nt_l, \tag{7}$$

где t_s — время передачи одного элемента матрицы, t_l — время выполнения вспомогательных операций при пересылке блока данных, время пересылки блока данных размерностью s элементов равно $t(s) = st_s + t_l$, а k — коэффициент времени работы broadcast операции: $b(s) = kst_s + t_l$.

В случае, когда R_0=rnd(0,1) или R_0=A, время работы возрастает (из-за невозможности проводить оптимизации, которые учитывают нулевые элементы матриц) и составляет:

$$T^1(n) = (2n^3 + n^2)t_c, \tag{8}$$

$$T_N^1(n) = \left(\frac{2n^3}{N} + n^2 \right) t_c + (2k+1)n^2 t_s + 3nt_l. \tag{9}$$

При применении процедуры выбора ведущего столбца, алгоритм становится более медленным:

$$T^2(n) = (3n^3 + n^2)t_c, \tag{10}$$

$$T_N^2(n) = \left(\frac{3n^3}{N} + n^2 \right) t_c + (2k+1)n^2 t_s + 3nt_l, \tag{11}$$

$$T_{exch}^1(n) = (2k+1)n^2 t_s + 3nt_l. \tag{12}$$

Оценим оптимум по времени выполнения для последнего алгоритма. Будем рассматривать случаи, когда среда обмена данными поддерживает (или не поддерживает) выполнение broadcast операций на аппаратном уровне. Временем, которое тратится на вспомогательные операции при обмене данными, пренебрегаем.

В первом случае k=1, наилучшее время достигается при $N \to \infty$ и равняется

$$\lim_{N\to\infty,k=1} T_N^2(n) = n^2\left(t_c + 3t_s\right) \tag{13}$$

Во втором случае $k = \log_l N$, а наилучшее время достигается при $N_o^2 = \frac{3}{2} n \frac{t_c}{t_s} \ln l$, где l — параметр среды передачи данных. Вычислительная сложность алгоритма при этом имеет порядок $O(n^2 \ln n)$, а минимальное время работы равняется

$$T_{N_o^2}^2\left(n, k = \log_l N\right) = \left(\left(2\frac{t_s}{t_c \ln l} + 1t_c\right) + \right.$$

$$\left. + \left(2\log_l\left(\frac{3}{2}\frac{t_c}{t_s}\ln l\right) + 1\right)t_s\right)n^2 + 2\frac{\ln n}{\ln l}n^2 t_s. \tag{14}$$

Параллельные алгоритмы с двухмерным распределением данных.

Рассмотрим алгоритм, когда на каждом узле по красно-чёрной схеме сохраняется N подматриц $A_{ij}^{-1}, i = 1..N, j = 1..N$ обратной матрицы размерностью $\left(\frac{n}{N}, \frac{n}{N}\right)$ в предположении, что $\frac{n}{N}$ целое.

При такой схеме, на процесс-менеджер возлагаются исключительно задачи рассылки и сбора данных, при этом, как и в предыдущем алгоритме, процесс-менеджер рассылает данные текущего столбца всем узлам на каждой итерации.

Алгоритм работы процессов-вычислителей на одной итерации опишем следующим образом:

1) Прием $a_l = \left(a_{l1,\ldots,}a_{lN}\right)$, $\dim a_{lj} = \left(1, \frac{n}{N}\right)$; Вычисление $\tilde{\alpha}_i^l = a_{lj} A_{ij}^{-1}$; Редукция и рассылка (allreduce) $\alpha_i^l = \sum_{i=1}^{N} \tilde{\alpha}_i^l$;

2) Вычисление $e_{li}^{l+1} = e_{li}^l / \alpha_{ll}$, $e_l^{l+1} = \left(e_{l1}^{l+1},\ldots,e_{lN}^{l+1}\right)$; Проведение обменов: процесс i рассылает всем другим процессам e_{li}^{l+1} и принимает от всех других процессов e_{lj}^{l+1}. В результате, на каждом узле сохраняется полный вектор e_l^{l+1};

3) Вычисление $e_{ji}^{l+1} = e_{ji}^l - e_{li}^{l+1}\alpha_{lj}, j \neq l, i \in P$.

Время выполнения такого параллельного алгоритма на системе с N процессоров можно оценить как:

$$T_N(n) = \left(\frac{n^3 + \frac{3}{2}n^2 - \frac{1}{2}n}{N} + \frac{1}{2}n^2 + \frac{1}{2}n \right) t_c + (k + r + 1)n^2 t_s + 3nt_l, \quad (15)$$

где r — коэффициент времени работы allreduce операции.

Естественно, что время, которое затрачивается на вычисления в случае рассматриваемого алгоритма, не зависит от схемы распределения данных. Исследуя алгоритм с двумерным распределением с точки зрения операций обмена, приходим к выводу, что единое отличие от одномерного варианта — это замена одной из операций типа broadcast на операцию типа редукция-рассылка. Последняя имеет большую сложность, которая зависит от реализации MPI и аппаратного обеспечения. Например, для системы quadrics interconnect [10] время выполнения операции broadcast имеет порядок O(n), а время выполнения операции allreduce — O(n log N).

Полученные теоретические оценки дают возможность сделать вывод, что применение к рассматриваемому алгоритму красно-черной двумерной схемы распределения данных не улучшает его характеристики.

Использование блочной двумерной схемы распределения, когда на каждом узле хранится одна подматрица результирующей матрицы, также существенно не улучшает алгоритм в связи со следующими факторами:

– объем рассылаемых данных при рассылке строки исходной матрицы не изменяется, тем не менее, количество вызовов функции broadcast увеличивается;

– необходимо проводить операции вида allreduce при вычислении результата умножения вектора на матрицу. В отличие от алгоритма с красно-черным распределением, выполнение этих операций распараллеливается;

– операция рассылки измененного ведущего столбца распараллеливается.

Учитывая вышеописанное, время, которое тратится на обмены в этом алгоритме можно оценить как

$$T_{exch}(N,n) = \left(k\left(1 + \frac{1}{\sqrt{N}}\right) + \frac{r}{\sqrt{N}} \right) n^2 t_s + \left(2 + \sqrt{N}\right) nt_l .$$

По сравнению с одномерным алгоритмом, количество обменов при $N \to \infty$ уменьшается на $(k+1)n^2$, однако количество операций обмена увеличивается на $\sqrt{N}n$.

При $n \to \infty$ и при условии, что операция рассылки осуществляется параллельно на аппаратном уровне (k=1), ускорение от использования двумерного алгоритма составит

$$e = \lim_{n\to\infty} \frac{T_{exch} - T^1_{exch}}{T^1_{exch}} = -\frac{1}{3}\frac{2\sqrt{N} - \ln N - 1}{\sqrt{N}},$$

$$\lim_{N\to\infty} e = -\frac{2}{3}.$$

Идентичную оценку получаем и в случае, когда $k = r = \ln N$.

Время, которое тратится на обмены при двумерном блочном распределении данных, может быть максимально на 66% меньшим, чем при использовании одномерного алгоритма.

Тестирование параллельных алгоритмов. Тестирование алгоритмов проводилось на кластере СКІТ-3. Измерялось время обращения прямоугольных матриц общего вида большой размерности. Проводилось сравнение времени работы параллельной реализации МБМ с одномерным распределением данных и параллельных алгоритмов LU и SVD декомпозиции, реализованных в пакете Scalapack.

Для вычислительного эксперимента были выбраны модели систем линейных алгебраических уравнений со 100% заполнением матрицы ограничений. Генерировались системы полного ранга с матрицей вида

$$A = \begin{cases} a_i = (\mathrm{rnd}(1.0), ..., \mathrm{rnd}(1.0)), \; i = 0 \\ a_i = a_{i-1} + \dfrac{\alpha}{n}(\mathrm{rnd}(1.0), ..., \mathrm{rnd}(1.0)), \; i > 0, \end{cases}$$

где $\| a_i - a_j \|^2 < \alpha$, а $\mathrm{rnd}(1.0)$ — случайное число в диапазоне $[0,1]$.

Согласно [8], сложность алгоритмов (не учитывая операции рассылки входных данных и сбора результатов), реализованных в Scalapack можно оценить как

$$T(N,n) = \frac{C_f n^3}{N} t_c + \frac{C_v n^2}{\sqrt{N}} t_s + \frac{C_m n}{NB} t_c,$$

где C_f, C_v, C_m — коэффициенты, которые зависят от конкретного алгоритма, а NB — размер блока минимальной единицы вычислений.

Сравнивая теоретические оценки быстродействия можно прийти к выводу, что реализованные в Scalapack алгоритмы имеют асимптотически лучшие характеристики, чем параллельные алгоритмы метода базисных матриц. Это подтверждается и данными вычислительных экспериментов. Тем не менее, при определенном количестве задействованных процессоров, алгоритмы МБМ имеют характеристики сопоставимые, или даже лучшие, чем Scalapack.

Из полученных значений ускорения (рис. 1) можно сделать вывод, что при $N<48$ алгоритмы имеют сравнимую эффективность, тем не менее, при увеличение количества задействованных процессоров, эффективность алгоритмов МБМ начинает спадать.

Из рис. 2 видно, что алгоритм МБМ с выбором ведущего элемента при $N<32$ имеет лучшее быстродействие, чем SVD, а алгоритм без выбора ведущего элемента хоть и более медленный, чем алгоритм LU декомпозиции, но работает в среднем втрое быстрее, чем SVD.

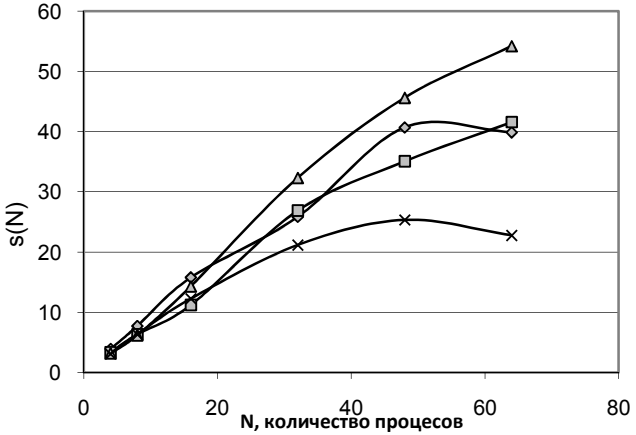

Рис.1. *Ускорение S(n) параллельных алгоритмов обращения матриц (n=8000)*

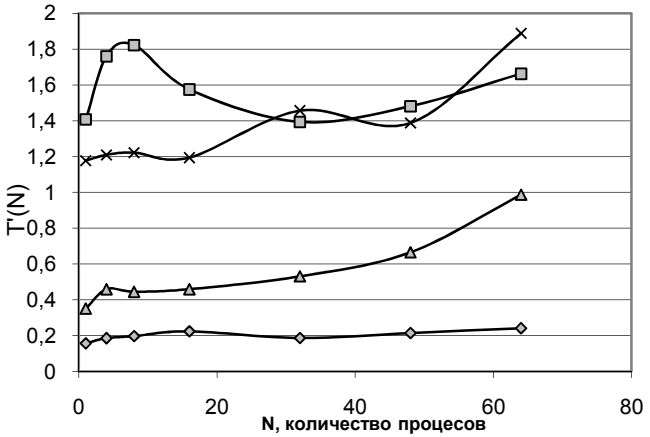

Рис.2. *Относительное время работы параллельных алгоритмов*

$$T'(N) = 10^8 N \frac{T(N)}{n^3}.$$

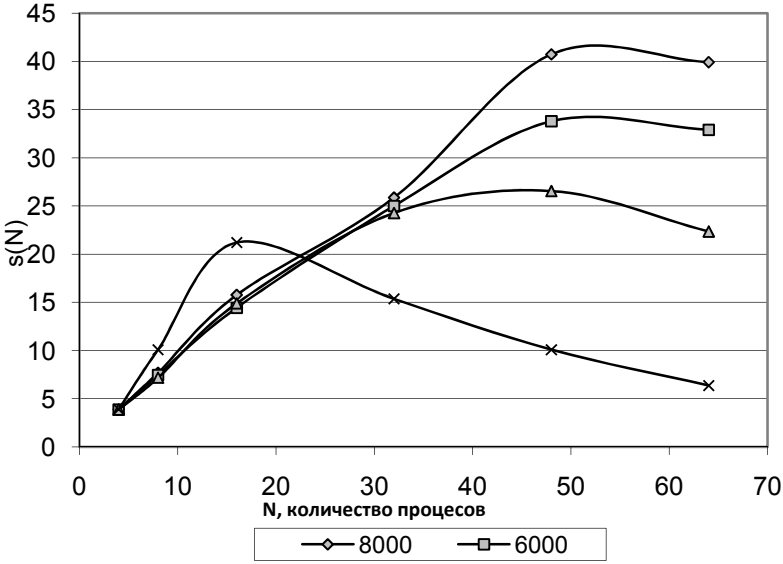

Рис.3. *Ускорение параллельных алгоритмов МБМ*

Из данных относительно ускорения параллельных алгоритмов МБМ (рис. 3) видно, что при увеличение размерности матрицы наибольшее ускорение достигается при $N \approx 48$.

В целом, протестированные реализации параллельных алгоритмов метода базисных матриц демонстрируют высокое быстродействие, сопоставимое с другими алгоритмами обращения матриц общего вида, реализованных в пакете Scalapack, однако из-за наличия операций глобального обмена, эффективность этих алгоритмов ниже.

Несмотря на это, рассмотренные алгоритмы могут применяться при анализе СЛАУ на небольших кластерах, с количеством процессоров порядка 50. На таких кластерных системах, для задач нахождения общего решения прямоугольных СЛАУ или решения плохо обусловленных СЛАУ, параллельные алгоритмы метода базисных матриц являются более быстрыми, чем алгоритмы, использующие SVD разложение.

Следует также отметить, что математический аппарат анализа и решения СЛАУ является основополагающим при исследовании, в частности, и экономических процессов, описываемых моделью Леонтьева [10] (МЛ). Одной из особенностей МЛ является то, что она включает в себе математические проблемы анализа систем линейных алгебраических уравнений с квадратной невырожденной матрицей ограничений, линейных алгебраических неравенств, а также задач линейного программирования. Проведения качественно анализа при построении моде-

ли предопределяет включение и количественного анализа непротиворечивости ее структурных элементов, обусловленности. Последовательные и параллельные схемы МБМ направлены на исследование именно таких проблем, показывая при этом высокую свою эффективность.

Список использованной литературы:

1. Бомба А. Я. Нелінійні математичні моделі процесів геогідродинаміки / А. Я. Бомба, В. М. Булавацький, В. В. Скопецький. — Київ : Наукова думка, 2007. — 272 с.
2. Воеводин В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. — Санкт-Петербург : БХВ-Петербург, 2002. — 608 с.
3. Гергель В. П. Теория и практика параллельных вычислений : учебное пособие / В. П. Гергель. — М. : Интернет-Университет Информационных Технологий, 2007. — 423 с.
4. Химич А. М. Параллельные алгоритмы решения задач вычислительной математики / А. М. Химич, И. Н. Молчанов и др. — Киев : Наукова думка, 2008. — 248 с.
5. Деммель Дж. Вычислительная линейная алгебра. Теория и приложение / Дж. Деммель. — М. : Мир, 2001. — 430 с.
6. Кудін В. І. Метод штучних базисних матриць // Доповіді НАН України, 9, — 2007. — С. 30—34.
7. Беклемишев Д. В. Дополнительные главы линейной алгебры / Д. В. Беклемишев — М. : Наука. 1983. — 335 с.
8. Blackford L. S. ScaLAPACK user's guide / L. S. Blackford, J. Choi, A. Cleary, E. D'Azeuedo, J. Demmel, I. Dhillon, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. C. Whaley. // SIAM, 1997. — 325 p.
9. Moody A. Programming for Optimal MPI Performance on LC's Linux / A. Moody // Quardics clusters https://computing.llnl.gov/mpi/moody.pdf
10. Леонтьев В. В. Межотраслевой анализ воздействия структуры экономики на окружающую среду / В. В. Леонтьев // Экономика и математические методы. — 1972. — Т. VII. — Вып. 3. — С. 370—400.

Parallel algorithms for ill-conditioned linear algebraic systems analysis, built upon basis matrix method (BMM), have been considered in the paper. Numerical experiments results on comparison of BMM parallel algorithms characteristics with characteristics of algorithms from Scalapack have been given.

Keywords: *linear algebraic systems analysis, parallel algorithms, basis matrix method.*

Отримано: 31.08.2009