УДК 68J.142.2(075.8)

Г.И. ШПАКОВСКИЙ, А.Е. ВЕРХОТУРОВ

АЛГОРИТМ ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ СЛАУ МЕТОДОМ ГАУССА - ЗЕЙДЕЛЯ*

This work concerns the parallel Gauss - Zeidel method. The peculiarity of this method is consecutive calculation; therefore, the effective organization of parallel computing for this method could be synthesized to other calculus of approximations. The parallel method is based on the principle of preliminary calculations, allowing minimizing the influence of consecutive calculation of the roots.

Локальные сети на базе персональных ЭВМ без дополнительных материальных вложений нетрудно превратить в кластеры для параллельных вычислений [1]. Время решения системы линейных алгебраических уравнений (СЛАУ) и ряда других задач на таких кластерах может быть уменьшено во много раз. При этом кластеры, в частности вузовские, могут быть объединены в сеть для распределенных вычислений, что позволит многократно увеличить степень распараллеливания вычислений.

" Авторы статьи - сотрудники кафедры информатики.

При решении СЛАУ большого размера, в том числе при моделировании микроэлектронных структур [2] для сокращения количества итераций, используется метод Гаусса - Зейделя, который состоит в следующем. Пусть задана СЛАУ

$$\mathbf{Ax = b},$$

где A - матрица коэффициентов, x - искомый вектор-столбец решения, b - вектор-столбец свободных членов. Обычно для выполнения итерационных вычислений исходную систему уравнений приводят к виду

$$x_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m + b_1$$
$$x_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m + b_2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_m = a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mm}x_m + b_m.$$

Метод Гаусса - Зейделя заключается в том, что очередная компонента вектора приближения вычисляется с учетом ранее полученных значений компонент:

$$x_i^{k+1} = \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i}^{m} a_{ij}x_j^{k} + b_i,$$

где m - размерность матрицы коэффициентов $\left(a_{ij}\right)_{1,1}^{m,m}$, $к$ - номер итерации, $\mathbf{x_k} = (x_1^k, x_2^k, \ldots, x_m^k)$ - вектор приближения, определяемый на k-м шаге итерации.

Дополнительное ускорение вычислений по методу Гаусса - Зейделя достигается за счет распараллеливания этих вычислений. Большой объем материала по параллельному решению СЛАУ, в том числе и методом Гаусса - Зейделя, представлен в [3]. Из последних исследований наибольший интерес представляет работа [4], в которой процесс параллельного выполнения одной итерации выглядит следующим образом:

- процессоры закрепляются за столбцами матрицы коэффициентов, элементы вектора приближения вычисляются последовательно по строкам;
- в каждой строке все процессоры одновременно вычисляют произведения для отведенных им столбцов строки. Эти произведения затем суммируются через коммуникационную систему кластера.

При выполнении последней операции (главной в алгоритме) удельный вес обмена значительно возрастает, что и определяет низкую эффективность вычислений. В частности, авторы приводят график, на котором ускорение для нескольких тысяч процессоров не превышает двух раз.

Ниже предлагается эффективный алгоритм распараллеливания метода Гаусса - Зейделя на основе *упреждающих вычислений*. Оценка распараллеливания производится для одной итерации, поэтому вопросы приведения матриц к виду, удобному для выполнения итераций, вопросы сходимости, выбора начального приближения, оценки достигнутой точности далее не рассматриваются.

Сначала изучим ситуацию, когда $n = m$, где и - число процессоров (рис. 1 *а*). В правой по отношению к диагонали части матрицы (П) хранятся коэффициенты для вычислений с компонентами вектора приближения текущей итерации, а в левой (Л) - с новыми компонентами. Рассмотрим начальный момент, когда в строке i-1 вычисляется компонента x_{i-1}^{k+1}. Пусть заранее вычислены суммы:

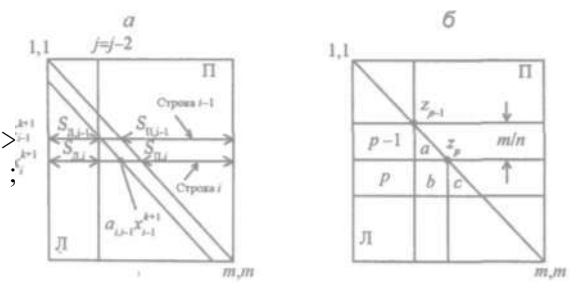

Рис. 1. Организация параллельных вычислений СЛАУ методом Гаусса – Зейделя: *а*: $n = m$; *б*: $n < m$

$S_{Л,i-1} = \sum_{j=1}^{i-2} a_{i-1,j} x_j^{k+1}$ и $S_{П,i-1} = \sum_{j=i-1}^{m} a_{i-1,j} x_j^k + b_{i-1}$, тогда $x_{i-1}^{k+1} = S_{Л,i-1} + S_{П,i-1}$. Полученное значение этой компоненты вектора приближения через коммуникационную систему рассылается всем процессорам выше и ниже процессора строки i-1.

Теперь рассмотрим последовательность операций в процессоре строки *i,* где уже сформировано значение суммы $S_{Л,i} = \sum_{j=1}^{i-2} a_{i,j} x_j^{k+1}$. Такие же суммы уже вычислены и всеми низлежащими процессорами для своих коэффициентов. Далее в строке *i* необходимо выполнить две следующие операции: $S_{Л,i} = S_{Л,i} + a_{i,i-1} x_{i-1}^{k+1}$ и $x_i^{k+1} = S_{Л,i} + S_{П,i}$. Эти операции на уровне строки / требуют времени $t = t_*, + t_+ + t_+,$ где t_* и t_+ - время операции умножения и сложения соответственно. Все операции на уровнях $i + 1$, $i + 2$ и так далее до уровня m будут повторяться, поэтому полное время T_m параллельного решения СЛАУ методом Гаусса - Зейделя без учета обменов для m процессоров будет $T_m = mt$.

Как отмечалось, полученное в строке *i* значение x_i^{k+1} посылается также и в процессоры, лежащие выше строки *i.* Эти процессоры уже освободились от вычисления компонент x_i^{k+1}, где $l < i,$ и используются для вычисления в каждой *l-й* строке сумм $\sum_{j=1}^{i} a_{l,j} x_j^{k+1}$. Поэтому после вычисления последней компоненты вектора приближения x_m^{k+1} в матрице будут получены все суммы строк для следующей *(k+2)-й* итерации.

Приведенная ситуация для $n = m$ использовалась для пояснения принципа распараллеливания. На практике обычно $n < m$ или даже $n << m$. Рассмотрим процесс параллельных вычислений для двух смежных процессоров p - 1 и p (рис. 1 *б),* каждый из которых вычисляет m/n компонент вектора приближения.

При входе в зону процессора p - 1 (точка z_{p-1}) для каждой его строки, как и прежде (см. рис. 1 *а),* на основе вычисленных ранее компонент вектора приближения x_i^{k+1} сформированы суммы для левой и правой частей строк, за исключением треугольника *а.* Таким образом, для получения m/n компонент в зоне процессора p - 1 необходимо произвести вычисления в треугольнике *а.* Эти вычисленные компоненты через коммуникационную систему будут переданы во все процессоры, находящиеся выше и ниже процессора p - 1, в том числе и в процессор $p,$ который вычислит очередные m/n компонент вектора приближения. Назовем шагом вычислений все операции, отнесенные к одному процессору. Следовательно, сумма затрат времени на всех n шагах и составит время решения СЛАУ T_n-$nt_ш$.

Пусть время одного шага t_m состоит из времени перехода от точки z_{p-1} до точки z_p. Чтобы процессор p мог начать вычисления в точке $z_p,$ должны быть выполнены следующие действия.

- Ожидание при вычислении первой части компонент треугольника *а:*

$$t_{ож} = \left(\frac{m}{ne}\right)^2 \frac{1}{2} t_\rangle$$

где $t = t_*+t_+, e = 1, \ldots, m/n$ - доля компонент вектора приближения, вычисляемых в процессоре p - 1 до первой посылки в другие процессоры. Если $e = 1$, передача производится после вычисления всех компонент процессором p - 1; в случае $e = m/n$ - после вычисления каждой компоненты.

• Далее выполняется *e* тактов. Каждый такт состоит из двух фаз: передача *e*-й части компонент вектора приближения во все *n* процессоров (время t_{e1}); вычисление очередной e-й части компонент вектора приближения в треугольнике *a* и строк в прямоугольнике *b* (t_{e2}). Тогда

$$t_e = t_{e1} + t_{e2} = \left(t_л + \frac{vm}{ne} t_{сл}\right) + \left(\frac{m}{ne}\right)^2 t$$

где t_e - время одного такта обработки, $t_л$ - латентность (задержка) коммуникатора, $t_{сл}$ - время передачи одного слова (8 байтов) коммуникационной системой, v - коэффициент, зависящий от реализации операции *bcast* в коммуникаторе. Если устройство коммуникатора позволяет производить передачу всем процессорам одновременно, тогда $v=1$; в случае передачи по двоичному дереву $v= \log_2 n$; если передача производится поочередно каждому процессору, то $v=n$ [5]. Время te_2 определяется временем вычисления части прямоугольника *b,* поскольку оно превосходит время вычисления соответствующей части треугольника с.

Так как треугольники *c* и *a* равны, то для процессора *p* вычисления выполняются аналогично, поэтому время расчета одной итерации при $n > 1$ равно

$$T_n = n(t_{ож} + e \cdot t_e). \qquad (1)$$

Если средства кластера позволяют совмещать вычисления и обмен, то время T_n будет существенно меньше, чем определено выражением (1).

Величина T_n является наиболее объективной характеристикой оценки параллельного алгоритма, но часто используют параметр ускорения [1]:

$$R = T_1 / T_n = m^2 t / T_n, \qquad (2)$$

здесь T_1, T_n- время расчета одной и той же итерации для однопроцессорной и *n*-процессорной систем.

Формула (2) часто дает завышенные результаты для кластеров на ПЭВМ, поскольку в этом случае время T_1 из-за недостатка оперативной памяти может быть чрезмерно большим. Для кластеров типа СКИФ с большим объемом оперативной памяти это явление выражено намного слабее и показатель *R* более объективный.

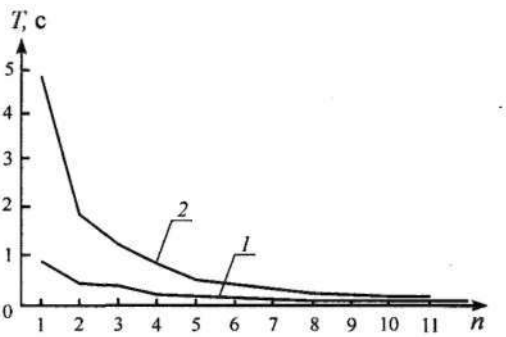

Рис. 2. Зависимость времени счета от числа процессоров:
1 – СКИФ, *2* – ПЭВМ

На рис. 2 представлены экспериментальные графики измерения времени: на кластере ПЭВМ (МП Celeron с частотой 1.7 GHz и коммуникатор Fast Ethernet со скоростью обмена 10 MB/s [5]) и кластере СКИФ-1000 (МП Opteron с частотой 2.4 GHz и коммуникатором Infiniband со скоростью обмена 830 MB/s [6]). Программы были написаны в стандарте MPI [1], матрица коэффициентов имела размер 4000 x 4000. Очевидным в обоих случаях является факт уменьшения времени счета при использовании параллельного алгоритма.

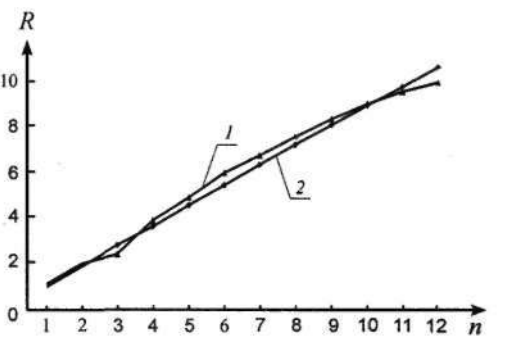

Рис. 3. Кластер СКИФ:
1 – расчет, *2* – эксперимент ($v = n$, $e = 2$)

На рис. 3 представлены характеристики ускорения для кластера СКИФ, полученные

расчетом по формуле (2) и по результатам эксперимента, подтверждающие возможность использования данной формулы (2) для оценочных расчетов.

Результаты экспериментов и расчета показывают, что ускорение для кластеров ПЭВМ и СКИФ К-1000 имеет примерно линейный характер с хорошим коэффициентом утилизации $R/n = 0{,}8 - 0{,}9$, который будет сохраняться при одновременном увеличении размера матрицы m и числа процессоров n.

Таким образом, разработанный алгоритм имеет высокую эффективность и может использоваться в практике. Эффективность достигнута распределением процессоров по строкам, что позволяет за счет резкого сокращения обменов при суммировании в одной строке выполнять не только упреждающие вычисления, но и одновременно готовить строки для новой итерации.

1. Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. Мн., 2002.

2. Мулярчик С. Г. Вычислительная электроника. Мн., 2003.

3. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем/ Пер. с англ. М., 1991.

4. Головашкин Д.Л., Горбунов О.Е. // Вестн. Самар. гос. техн. ун-та. Сер. Физ.-мат. науки. 2004. № 27.

5. Шпаковский Г.И., Верхотуров А.Е., Серикова Н.В. Организация работы на вычислительном кластере. Мн., 2004.

6. СКИФ // http://skif.bas-net.by.

Поступила в редакцию 21.02.06.

Геннадий Иванович Шпаковский - кандидат технических наук, доцент.

Алексей Евгеньевич Верхотуров - аспирант. Научный руководитель - доктор технических наук, профессор С.Г. Мулярчик.