АЛГОРИТМИЧЕСКИЙ АНАЛИЗ СТРУКТУРНОЙ ИНФОРМАЦИИ
(Вычислительные системы)
1985 год Выпуск 112

УДК 681.32:519.68

НЕКОТОРЫЕ АСПЕКТЫ ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

С.Г.Седухин

Введение

Решение ряда задач, имеющих матричную формулировку, может быть осуществлено по неветвящимся алгоритмам, основанным на рекуррентных соотношениях. К таким алгоритмам относятся, например, матричное произведение, методы треугольного разложения и обращения матрицы, дискретное преобразование Фурье, многие методы цифровой фильтрации и т.п. Все эти алгоритмы, базирующиеся на матричных преобразованиях (операциях), характеризуются высокой степенью регулярности и массовостью однотипных вычислений, что лежит в основе их высокопараллельной реализации на однородных вычислительных структурах [1,2].

Однако многие современные алгоритмы, практически используемые для решения на ЭВМ различных матричных задач, не обладают столь высокой степенью регулярности вычислений, так как часто характеризуются наличием операций условного ветвления, смесью скалярных, векторных и матричных операций, чередующихся обработкой строк и столбцов преобразуемой матрицы и т.п. Все эти особенности в значительной мере усложняют параллельную реализацию таких нерегулярных алгоритмов и делают невозможным применение методов автоматического синтеза соответствующих вычислительных структур [3,4]. Проектирование вычислительных структур для этих алгоритмов основано пока на эвристических подходах.

В работе для вычислительных систем с локальными взаимодействиями [1] рассматриваются некоторые аспекты организации нерегулярных параллельных вычислений. В первой части работы для алгоритмов, использующих операцию транспонирования матрицы, приводится

эвристическое решение задачи по организации одновременной обработки строк и столбцов преобразуемой матрицы данных. Во второй части рассматривается параллельная интерпретация алгоритма Хаусхолдера, обладающего всеми перечисленными выше особенностями нерегулярных вычислений. Алгоритм Хаусхолдера используется при нахождении сингулярного разложения матрицы или SVD (singular-value decomposition), которое является мощным вычислительным средством анализа матриц и задач, связанных с матрицами, и которое имеет приложения во многих областях (см.,например, [5]). Важность нахождения сингулярного разложения стимулирует многочисленные исследования по параллельной реализации SVD [6-8]. Однако все эти исследования связаны с распараллеливанием методов, отличных от рассматриваемого ниже алгоритма Хаусхолдера и уступающих ему при реализации на ЭВМ с фиксированной длиной слова в смысле учета ошибок округления и численной устойчивости [5]. Преобразование (nxn)-матрицы по методу Хаусхолдера связано с выполнением $O(n^3)$ операций, последовательное выполнение которых на однопроцессорных ЭВМ требует соответственно $O(n^3)$ временных шагов. При достаточно большом размере исходной матрицы указанная временная сложность может привести к невозможности последовательного решения задачи на ЭВМ. В работе показывается, что параллельная реализация алгоритма Хаусхолдера на системе из n циклически связанных вычислителей (ЭВМ) требует $O(n^2)$ временных шагов.

Параллельная обработка строк и столбцов матрицы

Существенную сложность при организации решения на ЭВМ ряда матрично формулируемых задач представляют алгоритмы, использующие операцию транспонирования преобразуемой матрицы данных. Указанная сложность заключается в несоответствии классической линейной организации памяти ЭВМ двумерной структуре матрицы. Это несоответствие приводит к неодинаковой эффективности доступа процессора ЭВМ к последовательным элементам строк или столбцов матрицы, хранимой в линейной памяти. Непосредственное выполнение на однопроцессорной ЭВМ операции транспонирования (n x n)-матрицы требует перезаписи в памяти $O(n^2)$ элементов и,следовательно, $O(n^2)$ единиц времени. Многие матричные алгоритмы основаны на многократном использовании операции транспонирования, непосредственное выполнение которой на ЭВМ часто делает невозможным решение всей задачи. Так, например, в [9] указывается, что решение на ЭВМ ряда

практически важных задач математической физики становится возможным только при условии эффективного проведения операции транспонирования преобразуемой матрицы коэффициентов. Отметим, что в существующих векторных ЭВМ Cray-1 и Cyber 205 одинаковая эффективность доступа процессора к строкам и столбцам матрицы данных обеспечивается соответствующей организацией хранения элементов в расслоенной памяти (interleaved memory) ,работающей в режиме перекрытия циклов ее модулей [I0]. Расслоенная память позволяет эффективно осуществлять последовательный доступ к строкам и столбцам матрицы без явного проведения операции транспонирования.

Параллельная реализация алгоритмов, многократно использующих операцию транспонирования матрицы, требует осуществления чередующегося одновременного доступа к элементам строк и столбцов матрицы, т.е. преобразование должно попеременно применяться сначала к последовательности элементов всех строк (столбцов), а затем к последовательности элементов всех столбцов (строк) матрицы. Заметим, что время непосредственного выполнения операции транспонирования матрицы может существенно превысить время параллельного осуществления основных вычислений и свести на нет выигрыш от параллельной реализации алгоритма. В существующих параллельных вычислительных системах одновременный доступ процессоров к строкам и столбцам матрицы осуществляется по-разному. Так, чередующаяся параллельная обработка строк и столбцов матрицы эффективно осуществляется в так называемой ортогональной памяти, входящей в состав вычислительных комплексов STARAN и OMEN-60 [II]. В вычислительной системе BSP эта же задача решается специальным коммутатором, который под общим управлением обеспечивает бесконфликтный доступ всех шестнадцати процессоров системы либо к столбцам,либо к строкам, либо к диагоналям матрицы данных, записанной в расслоенной общей памяти [I2]. Решению этой задачи посвящены исследования в области реконфигурируемых сетевых архитектур таких, как TRAC [I3].

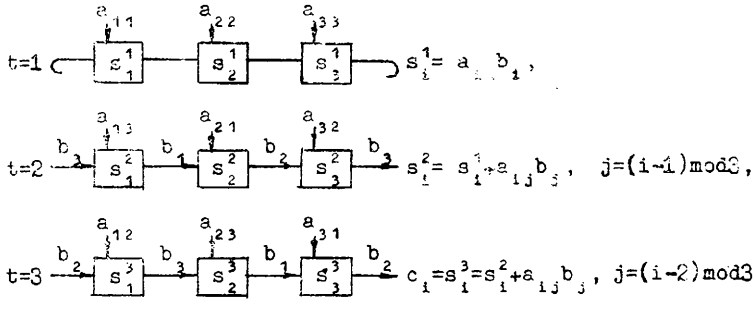
Следует заметить,что упомянутые выше вычислительные системы, относящиеся к классу SIMD-систем, не выполняют в явном виде операцию транспонирования матрицы, а используют для этой цели соответствующий коммутатор, который обеспечивает одновременный доступ вычислителей (процессоров) системы к элементам строк или столбцов матрицы, хранимой в общей памяти. Увеличение числа и/или скорости вычислителей в системе приводит к соответствующему росту сложности коммутатора, который превращается при этом в одно из наиболее узких мест архитектуры SIMD-систем.

Для систем с локальными взаимодействиями вычислителей [I], относящихся к классу MIMD-систем, решение задачи по обеспечению одинаковой эффективности чередующейся одновременной обработки строк и столбцов преобразуемой матрицы осложняется по причинам отсутствия в системе общедоступной памяти, централизованного (общего) управления и ограниченной только близкодействием сети связи вычислителей. Для указанных систем предлагаемое ниже решение рассматриваемой задачи основывается на совместном использовании двух различных, но эквивалентных по результату способов параллельно-поточной реализации произведения матрицы на вектор. Известно, что если необходимо выполнить умножение $(n \times n)$-матрицы $A = [a_{ij}]$ на вектор $b = [b_i]$, т.е. найти вектор $c = [c_i]$, где $c_i = \sum_{j=1}^{n} a_{ij} b_j$, то последовательное осуществление вычислений требует $O(n^2)$ временных шагов, каждый из которых связан с выполнением операций умножения и сложения. Параллельная реализация алгоритма возможна на системе из n циклически связанных вычислителей двумя различными способами.

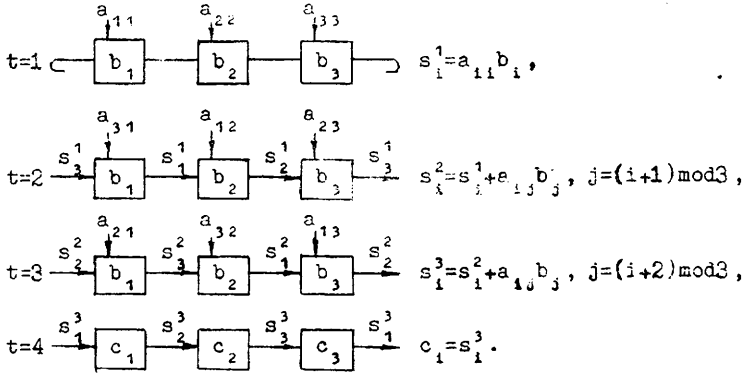
При первом способе в исходном состоянии i-я строка матрицы A и i-й элемент вектора b хранятся в локальной памяти соответствующего i-го вычислителя системы $(i \in [1,n])$. Осуществляя на каждом временном шаге согласованный циклический обмен элементами вектора b и независимо выполняя требуемые вычисления, через n шагов получим результирующий вектор c, поэлементно распределенный по вычислителям системы (рис.I). Для простоты считаем, что все вычислители начинают работать одновременно на временном шаге $t = I$. Отметим, что данный способ применялся ранее при параллельно-поточной интерпретации ряда численных алгоритмов, использующих операцию умножения матрицы на вектор [I].

Предлагаемый в данной работе второй способ параллельного произведения матрицы на вектор использует исходное распределение матрицы A по столбцам, т.е. j-й столбец матрицы хранится в памяти j-го вычислителя $(j \in [1,n])$ системы. Элементы вектора b распределены при этом аналогично первому способу. Осуществляя теперь на каждом шаге циклический обмен формируемыми в вычислителях результатами частичных вычислений s, также через n временных шагов получим искомый вектор c (рис.I,б). Дополнительно $(n+1)$-й шаг (сдвиг данных без обработки) нужен только для того, чтобы соот-

а) Ab=c, матрица A распределена по строкам (i=1,2,3)



б) Ab=c, матрица A распределена по столбцам (i=1,2,3.



в) $A^T b = d$, матрица A распределена по строкам (i=1,2,3)

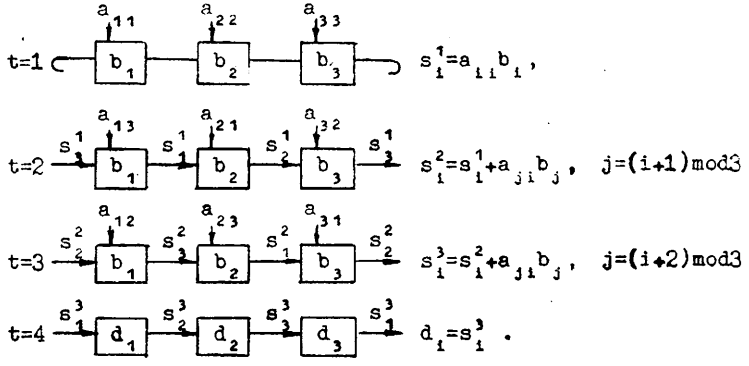


Рис.I. Различные способы умножения матрицы на вектор.

ветствующие элементы вектора c оказались в тех вычислителях, с которых начиналось их определение. Таким образом, видно, что второй способ основан на обмене текущими результатами вычислений, а не элементами вектора b.

Рассмотренные способы организации параллельно-поточных вычислений могут быть совместно использованы для чередующейся одновременной обработки строк и столбцов матрицы данных. Так, если матрица A распределена между вычислителями по строкам, то последовательность вычислений, связанная с нахождением сначала вектора $c = Ab$, а далее вектора $d = A^T b$, где A^T – транспонированная матрица A, может быть эффективно осуществлена на циклической системе из n вычислителей за $O(n)$ временных шагов без явного проведения операции транспонирования матрицы A (рис.I,а,в). Отметим, что циклический порядок поступления в каждый вычислитель элементов вектора b или текущих результатов s, задает циклический порядок выборки вычислителем из своей локальной памяти соответствующих элементов строк матрицы A (рис.I). Следовательно, для хранения элементов матрицы в вычислителе достаточно использовать память с последовательным доступом.

Нетрудно показать теперь, что нахождение первого собственного значения $(n \times n)$-матрицы A методом скалярных произведений:

$$\lambda_1 \approx \frac{(b_k, b_k^T)}{(b_{k-1}, b_k^T)} = \frac{(A^k \cdot b_0, A^{Tk} \cdot b_0)}{(A^{k-1} \cdot b_0, A^{Tk} \cdot b_0)} ,$$

где $k = I,2,\ldots$ – номер итерации, а b_0 – начальный вектор, можно эффективно осуществить на циклической системе из n вычислителей за время $O(n^2 k)$. Для сравнения напомним, что последовательная реализация данного метода требует времени $O(n^3 k)$.

Современные алгоритмы, практически применяемые для решения различных матричных задач, характеризуются целым рядом особенностей (включая рассмотренную выше операцию транспонирования), которые в совокупности усложняют эффективную параллельную реализацию алгоритмов. Типичным представителем таких алгоритмов является алгоритм двусторонних ортогональных отражений Хаусхолдера, практически используемый при нахождении сингулярного разложения матрицы.

Параллельно-поточная интерпретация алгоритма Хаусхолдера

Сингулярным разложением действительной $(n \times n)$-матрицы $A = [a_{ij}]$ называется всякая ее факторизация вида $A = U\Sigma V^T$, где U и V – ортогональные $(n \times n)$-матрицы, а Σ – диагональная $(n \times n)$-матрица, у которой $\sigma_{ij} = 0$ при $i \neq j$ и $\sigma_{ii} = \sigma_i \geq 0$. Величины σ_i называются сингулярными числами матрицы A, а столбцы матриц U и V – левыми и правыми сингулярными векторами. Важность нахождения SVD заключается в том, что практически все сведения для решения системы линейных уравнений с матрицей A связаны с множеством сингулярных чисел этой матрицы [I4].

Целью рассматриваемого алгоритма SVD является нахождение ортогональных матриц U и V таких, чтобы матрица $\Sigma = U^T A V$ была диагональной. Матрицы U и V получаются как произведения ортогональных матриц, называемых хаусхолдеровыми отражениями. Алгоритм состоит из двух этапов. Первый этап заключается в приведении матрицы A к двухдиагональной форме, т.е. матрице, у которой ненулевыми могут быть лишь элементы диагонали и первой наддиагонали. Второй этап – это итерационный процесс, в котором наддиагональные элементы уменьшаются до пренебрежимо малой величины, что дает искомую диагональную матрицу. Так как определяющим с точки зрения числа расходуемых операций является первый этап, ниже рассматривается только его параллельная интерпретация.

Отражения Хаусхолдера основаны на построении двух конечных последовательностей [I5]:

$$U^{(k)} = I_n - 2x^{(k)} \cdot x^{(k)T} \qquad (k = 1, 2, \ldots, n-1);$$

$$V^{(k)} = I_n - 2y^{(k)} \cdot y^{(k)T} \qquad (k = 1, 2, \ldots, n-2),$$

где I_n – единичная $(n \times n)$-матрица, а $x^{(k)T}x^{(k)} = y^{(k)T}y^{(k)} = 1$. Последовательности приводят исходную $(n \times n)$-матрицу $A=[a_{ij}]$ к верхней двухдиагональной форме следующего вида:

$$U^{(n-1)} \ldots U^{(1)} A V^{(1)} \ldots V^{(n-2)} = \begin{bmatrix} q_1 & e_2 & & & \\ & q_2 & e_3 & & \\ & & q_3 & e_4 & \\ & & & \ddots & \\ & & & & e_n \\ & & & & q_n \end{bmatrix}.$$

Если положить $A^{(1)} = A$ и считать, что справедливы соотношения

$$A^{(k+1/2)} = U^{(k)}A^{(k)} \qquad (k = 1,2,\ldots, n-1);$$

$$A^{(k+1)} = A^{(k+1/2)}V^{(k)} \qquad (k = 1,2,\ldots, n-2),$$

то матрицы $U^{(k)}$ находятся так, что $a_{ik}^{(k+1/2)}=0$ $(k+1 \le i \le n)$, а матрицы $V^{(k)}$ находятся так, что $a_{kj}^{(k+1)}= 0$ $(k+2 \le j \le n)$. Теоретические основания отражений Хаусхолдера и рекомендации по использо - ванию метода подробно изложены в [5,I4].

Программа, описывающая преобразование Хаусхолдера и рекомендуемая в [I6] для практического использования на однопроцессорной ЭВМ (вычислителе), представлена на рис.2. Нашей целью является рассмотрение параллельной интерпретации данных преобразований на регулярной сети вычислителей. Для удобства дальнейшего изложения мы разбили программу на части и ввели дополнительную индексацию некоторых переменных. Контроль за точностью вычислений осуществ - ляется во второй и шестой частях программы с помощью проверки соответствующих результатов с величиной $tol = \beta / \varepsilon_0$, где β является наименьшим положительным числом, представимым в вычислителе (ЭВМ), а ε_0 - машинная точность [I4].

Прежде всего отметим, что каждый i-й этап преобразований (i= = I,2,..., n-1), выполняемый внутри цикла по параметру i ,характеризуется наличием смеси скалярных (типа e[i]:=g),векторных (части I, 5 и 8 программы), матричных (части 4 и 9) операций, а также операций условного ветвления (части 2 и 6). При определении операционной сложности алгоритма будем считать, что скалярные операции имеют нулевой вес, а векторные и матричные операции - вес, пропорциональный размеру обрабатываемых векторов и матриц соответственно. При этих условиях общее число операций, расходуемых методом, определится матричными преобразованиями, осуществляемыми участками 4 и 9 приведенной программы, т.е.

$$p(n) = \sum_{i=1}^{n-1} [4(n-i)^2 + O(n)] = 4n^3/3 + O(n^2).$$

При нахождении данной величины мы полагали также, что все получаемые на каждом i-м этапе значения $s \ge tol$ (части 2 и 6). Последнее приводит к обязательному выполнению на каждом этапе всех матрич -

```
    begin integer i,j,k,l,n; real s,g,tol,f,h;  real array
      a[1:n,1:n], e[1:n], q[1:n];
      g:=0; for i:=1 step 1 until n-1 do
1:  begin  e[i]:=g; s:=0; l:=i+1;
    for  j:=1 step 1 until n do s:=s+a[j,i]↑2;
2:  if s<tol then g:=0 else
3:  begin f:=a[i,i]; g:= if f<0 then sqrt(s) else -sqrt(s);
    h:=f× g-s; a[i,i]:=f-g;
4:  for j:=l step 1 until n do
    begin s_j:=0; for k:=i step 1 until n do s_j:=s_j+a[k,i]× a[k,j];
          f_j:=s_j/h; for k:=i step 1 until n do a[k,j]:=a[k,i]× f_j+
          + a[k,j]
    end j end s;
5:  q[i]:=g; s_i:=0
    for j:=l step 1 until n do s_i:=s_i+a[i,j]↑2;
6:  if s_i<tol then g:=0 else
7:  begin f:=a[i,i+1]; g:= if f<0 then sqrt(s_i) else -sqrt(s_i);
          h:=f× g-s_i; a[i,i+1]:=f-g;
8:  for j:=l step 1 until n do e[j]:=a[i,j]/h;
9:  for j:=l step 1 until n do
    begin s_j:=0; for k:=l step 1 until n do s_j:=s_j+a[i,k]× a[j,k];
    for k:=l step 1 until n do a[j,k]:=a[j,k]× s_j× e[k]
    end j end s_i;
    end
10: e[n]:=a[n-1,n]; q[n]:=a[n,n]
    end
```

Рис. 2

ных операций программы. Если теперь за единицу времени принять длительность исполнения операции накопления типа $s:=s+a_i \times b_i$, число которых равно величине $p(n)$, то время последовательной реализации метода на ЭВМ с одним потоком команд и одним потоком данных будет равно $T_1(n) = 4n^3/3 + O(n^2)$.

Параллельно-поточная реализация рассматриваемого алгоритма осложняется по следующим причинам.

I. Матричные операции (части 4 и 9 программы), которые являются претендентами на параллельную реализацию, перемежаются со

скалярными и векторными операциями (например,5 и 7 части), выполняющихся последовательно. При этом если для магистральных векторных ЭВМ наличие в программе скалярных операций приводит к снижению эффективности выполнения векторных операций [17], то для параллельных вычислительных систем наличие не только скалярных, но и отдельных векторных операций будет уменьшать эффективность параллельного осуществления матричных операций.

2. Присутствие в программе операций условного ветвления делает заранее непредсказуемым поведение процесса вычисления. Так, например, выполнение операции ветвления в части 2 программы может исключить реализацию не только последовательности скалярных операций (часть 3), но и осуществление на данном этапе матричной операции по левостороннему отражению (часть 4). Для параллельной реализации алгоритма это означает непредсказуемое заранее исключение массовых одновременных вычислений.

3. Левосторонние и правосторонние отражения требуют чередования обработки строк и столбцов преобразуемой матрицы данных на каждом этапе вычислений. Так, левостороннее отражение матрицы (часть 4) связано с нахождением на i-м этапе суммы попарных произведений элементов i-го столбца с соответствующими элементами всех j-х столбцов матрицы данных ($i+1 \le j \le n$). Дополнительно i-й столбец требуется далее для последующего переопределения элементов матрицы. Правостороннее отражение (часть 9) требует аналогичных действий с элементами строк преобразуемой матрицы. Как показано выше, эффективная параллельно-поточная реализация такой чередующейся обработки строк и столбцов матрицы возможна в циклически связанной вычислительной системе (рис.5,а). Однако нижеследующая причина требует дополнительных структурных характеристик вычислительной системы.

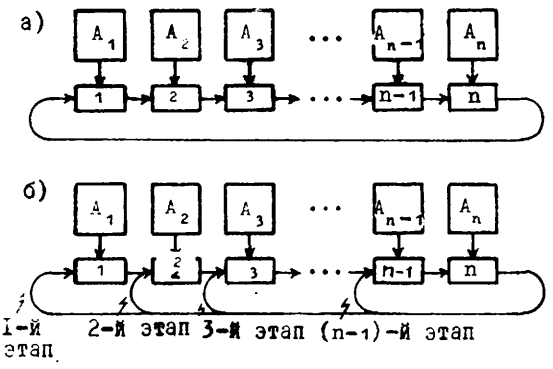


Рис. 3

4. Понижение на каждом этапе вычислений порядка преобразуемой матрицы требует при параллельной реализации алгоритма поэтапного исключения из циклической системы соответствующих вычислителей. Действительно, на первом этапе, т.е. при i = I, обрабатывается вся исходная (n x n)-матрица, на втором этапе (i = 2) приводится уже (n-1) x (n-1)-матрица, а первые столбец и строка в дальнейшей обработке не участвуют и т.д., вплоть до последнего этапа вычислений (i = n-1) , на котором преобразуется только (2x2)-матрица. Если теперь исходная матрица распределена между вычислителями циклической системы по столбцам, т.е. каждый j-й столбец хранится в локальной памяти соответствующего j-го вычислителя ($j \in [1,n]$),то исключение на i-м этапе преобразований i-й строки и i-го столбца матрицы требует исключения из системы вычислителя с номером j = i. Для этого по окончании каждого этапа необходимо осуществлять реконфигурацию структуры связи вычислителей (рис.3,б). Мы, однако, будем рассматривать организацию параллельно-поточных вычислений на циклической системе без реконфигурации структуры связи (рис.3,а). При этом исключаемый из системы вычислитель переходит в состояние, при котором он в дальнейшем будет осуществлять только транзитные передачи принимаемых данных. Другими словами, мы фактически моделируем поведение одной структуры (рис.3,б) в другой (рис.3,а). Ниже будет получена оценка растяжения времени реализации алгоритма при таком моделировании.

Итак, пусть исходная (n x n)-матрица $A = [a_{ij}]$ распределена между вычислителями циклической системы по столбцам.Следует отметить, что так как процесс преобразований данных характеризуется определенной симметрией, то допустимо начальное распределение исходной матрицы по строкам. На каждом i-м этапе обработку начинает вычислитель с номером j=i (вычислители с номерами $1 \le j < i$ находятся в транзитном состоянии), который последовательно выполняет первую, вторую и, возможно, третью и четвертую части программы. При этом часть 4 одновременно и согласованно реализуется в магистральном режиме цепочкой из (n-i+1)-го вычислителя,один из которых, с номером j=i (ведущий), последовательно передает (n-i+1) элементов i-го столбца матрицы всем последующим (ведомым) вычислителям с номерами $i+1 \le j \le n$. Требуемое в части 4 последующее переопределение (n-i+1)x(n-i) элементов матрицы выполняется в системе за счет повторной передачи вычислителем с номером j=i хранимых в его локальной памяти элементов i-го столбца.Возможная бу-

феризация принимаемых ранее элементов, исключающая повторную передачу, будет требовать в каждом вычислителе системы дополнительной памяти размером O(n) слов. Процесс параллельно-поточного преобразования (n x n)-матрицы для случая n = 4 на циклической системе из n вычислителей и соответствующие комментарии приведены в таблице, показывающей динамику локальных взаимодействий вычислителей на каждом временном шаге.

После окончания повторной передачи i-го столбца матрицы, характеризующей для i-го вычислителя завершение части 4 программы, либо после выполнения операции условного ветвления во второй части, исключающей при $s < tol$ реализацию частей 3 и 4, i-й этап обработки продолжает соседний вычислитель с номером $j = i+1$, а вычислитель с номером $j=i$ переходит в транзитное состояние. Новый ведущий вычислитель с номером $j = i+1$, подобно своему предшественнику, последовательно выполняет части 5, 6 и, возможно (при $s_i \geq tol$), части 7, 8, 9 программы. При этом векторные операции (части 5 и 8) выполняются последовательно на всех вычислителях системы либо путем накопления передаваемого результата s_i для части 5, либо путем "протяжки" общего для части 8 операнда h, найденного в $(i+1)$-м вычислителе системы. Матричная операция (часть 9), связанная с приведением элементов строк матрицы, реализуется одновременно всеми вычислителями путем согласованного обмена формируемыми в каждом вычислителе суммами s_j (см.таблицу).

Окончив i-й этап преобразований вычислитель с номером $j = i+1$ начинает реализацию $(i+1)$-го этапа, который заканчивает уже вычислитель с номером $j = i+2$ и т.д. вплоть до последнего $(i = n-1)$ этапа, на котором вычислитель с номером $j=n$ выполняет заключительную десятую часть программы. Ясно, что искомые элементы e_j и q_j получаемой двухдиагональной матрицы будут храниться в локальной памяти соответствующего j-го вычислителя, т.е. система подготовлена для дальнейших преобразований, связанных с нахождением SVD.

Нетрудно показать, что время параллельно-поточной реализации i-го этапа вычислений на циклической системе из n вычислителей оценивается величиной $T_n^{(i)}(n) = 3(2n-i)+7$. Тогда на осуществление всех преобразований расходуется время

$$T_n(n) = \sum_{i=1}^{n-1} T_n^{(i)}(n) = 4{,}5n^2 + O(n) .$$

Таблица

№ шага	Вычислители P_1	P_2	P_3	P_4	Проводимые действия
1	i=1				Выполнение первым вычислителем части I программы
2					
3	1				
4					
5	2,3				Выполнение частей 2 и 3
6	a_{11}				Начало передачи столбца a_{11}
7	a_{21}	a_{11}			$s_2^1=a_{11}a_{12}$;
8	a_{31}	a_{21}	a_{11}		$s_2^2=s_2^1+a_{21}a_{22}$; $s_3^1=a_{11}a_{13}$;
9	a_{41}	a_{31}	a_{21}	a_{11}	$s_2^3=s_2^2+a_{31}a_{32}$; $s_3^2=s_3^1+a_{21}a_{23}$; $s_4^1=a_{11}a_{14}$;
10	a_{11}	a_{41}	a_{31}	a_{21}	$s_2^4=s_2^3+a_{41}a_{42}$; $s_3^3=s_3^2+a_{31}a_{33}$; $s_4^2=s_4^1+a_{21}a_{24}$;
11	a_{21}	a_{11}	a_{41}	a_{31}	Вычисление a_{12} $s_3^4=s_3^3+a_{41}a_{43}$; $s_4^3=s_4^2+a_{31}a_{34}$;
12	a_{31}	a_{21}	a_{11}	a_{41}	a_{22} Вычисление a_{13} $s_4^4=s_4^3+a_{41}a_{44}$;
13	a_{41}	a_{31}	a_{21}	a_{11}	a_{32} a_{23} Вычисление a_{14}
14		a_{41}	a_{31}	a_{21}	a_{42} a_{33} a_{24}
15		s_1^1	a_{41}	a_{31}	$s_1^1=a_{12}^2$ a_{43} a_{34}
16			s_1^2	a_{41}	$s_1^2=s_1^1+a_{13}^2$; a_{44}
17				s_1^3	$s_1^3=s_1^2+a_{14}^2$;
18	▶s_1^3◀				Транзитная передача s_1^3
19		s_1^3			
20		6,7			Выполнение частей 6 и 7
21		h			$e_2=a_{12}/h$;
22		s_2^1	h		$s_2^1=a_{22}a_{12}$; $e_3=a_{13}/h$;
23		s_3^1	s_2^2	h	$s_3^1=a_{32}a_{12}$; $s_2^2=s_2^1+a_{23}a_{13}$; $e_4=a_{14}/h$;
24		s_4^1	s_3^2	s_2^3	$s_4^1=a_{42}a_{12}$; $s_3^2=s_3^1+a_{33}a_{13}$; $s_2^3=s_2^2+a_{24}a_{14}$;
25	▶s_2◀		s_4^2	s_3^3	$s_4^2=s_4^1+a_{43}a_{13}$; $s_3^3=s_3^2+a_{34}a_{14}$;
26	▶s_3◀	s_2		s_4^3	Вычисление a_{22} $s_4^3=s_4^2+a_{44}a_{14}$;
27	▶s_4◀	s_3	s_2		a_{32} Вычисление a_{23}
28		s_4	s_3	s_2	a_{42} a_{33} Вычисление a_{24}

Продолжение таблицы

29		i=2	s_4	s_3	Второй вычислитель начинает новый этап преобразований a_{43} a_{34}
30				s_4	a_{44}
31		1			
32					
33		2,3			Выполнение частей 2 и 3
34		a_{22}			Начало передачи столбца a_{12}
35		a_{32}	a_{22}		$s_3^1=a_{22}a_{23}$;
36		a_{42}	a_{32}	a_{22}	$s_3^2=s_3^1+a_{32}a_{33}$; $s_4^1=a_{22}a_{24}$;
37		a_{22}	a_{42}	a_{32}	$s_3^3=s_3^2+a_{42}a_{43}$; $s_4^2=s_4^1+a_{32}a_{34}$;
38		a_{32}	a_{22}	a_{42}	Вычисление a_{23} $s_4^3=s_4^2+a_{42}a_{44}$;
39		a_{42}	a_{32}	a_{22}	a_{33} Вычисление a_{24}
40			a_{42}	a_{32}	a_{43} a_{34}
	...	...	...	...	... ...

Обеспечиваемая при этом степень совмещения операций или ускорение процесса вычисления, составит величину $S_n = T_1(n)/T_n(n) = O(n)$, а степень использования вычислителей в системе или эффективность параллельных вычислений, определится величиной $E_n = S_n/n = O(1)$, которая не зависит от логического размера решаемой задачи (порядка исходной матрицы).

Так как на каждом i-м этапе преобразований наличие транзитных вычислителей приводит к растяжению времени реализации этапа на величину $\tau_i = 2i$, то общее время вычислений увеличится на величину

$$T_\tau(n) = \sum_{i=1}^{n-1} \tau_i = n(n-1),$$

которая соизмерима с временем параллельного осуществления алгоритма и учтена в полученной оценке $T_n(n)$. Найденные временные затраты $T_\tau(n)$ отсутствуют в системе с реконфигурируемой на каждом этапе вычисления структурой связи вычислителей (рис.3,б). Однако реконфигурируемость связи также требует определенных временных за-

трат, которые увеличивают возможное время параллельно-поточной реализации алгоритма. Следует отметить, что полученная оценка $T_n(n)$ учитывает также временные затраты, связанные с инерционностью заполнения магистрали, образованной цепочкой вычислителей системы.

Ввод исходной $(n \times n)$-матрицы в систему может быть осуществлен за время $O(n^2)$, которое соизмеримо с временем параллельного выполнения основных преобразований. В качестве локальной памяти вычислителя достаточно использовать память с последовательным доступом, так как произвольный доступ к элементам каждого столбца не требуется (см.таблицу). Если необходимы матрицы U и V, то они могут вычисляться одновременно с нахождением матрицы U^TAV, но за счет увеличения локальной памяти в каждом вычислителе системы. При этом время вычисления будет по-прежнему $O(n^2)$.

Итак, в рамках систем с локальным взаимодействием вычислителей были рассмотрены некоторые особенности параллельной обработки матричных данных, осуществляемой по нерегулярным алгоритмам. Для алгоритмов, использующих операцию транспонирования матрицы, предложен способ чередующейся одновременной обработки строк и столбцов матрицы. Показано, что параллельная интерпретация метода Хаусхолдера, являющегося при нахождении SVD наилучшим в смысле численной устойчивости и считавшимся наихудшим в смысле его распараллеливания [I8], может быть эффективно осуществлена на циклической системе из n вычислителей (ЭВМ) за время $O(n^2)$. Последовательная реализация метода на однопроцессорной ЭВМ требует времени $O(n^3)$.

Литература

I. МИШИН А.И., СЕДУХИН С.Г. Вычислительные системы и параллельные вычисления с локальными взаимодействиями. -В кн.: Математическое обеспечение вычислительных систем (Вычислительные системы, вып. 78). Новосибирск, 1979, с.90-104.

2. KUNG H.T., LEISERSON C.E. Algorithms for VLSI Processor Arrays.- In: Introduction to VLSI Systems. Addison-Wesley, Reading, Mass.,1980,p.271-292.

3. MOLDOVAN D.J. On the Design of Algorithms for VLSI Systolic Arrays.-Proc.of the IEEE,1983,v.71,N 1,p.113-120.

4. QUINTON P. The Systematic design of Systolic Arrays. - IRISA Internal Report N 216. Institut National de Recherche en Informatique et en Automatique, France,1983. - 35 p.

5. ФОРСАЙТ Дж., МАЛЬКОЛЬМ М., МОУЛЕР К. Машинные методы математических вычислений. -М.: Мир, 1980. - 279 с.

6. LUK F.T. Computing the singular value decomposition on the ILLIAC IV.-ACM Trans.Math.Softw.,1980,N 6,p.524-539.

7. HELLER D.E., IPSEN J.C.F. Systolic networks for orthogonal equivalence transformations and their applications. - Proc. 1982. Conf.on Advanced Research in VLSI, MIT,1982,p.113-122.

8. BRENT R.P., LUK F.T. A Systolic Architecture for the Singular Value Decomposition.- Technical Report. The Australian National Univ.,Departament of Computer Science,TR-CS-82-09,1982.

9. САФРОНОВ И.Д. Оценка параметров вычислительной машины, предназначенной для решения задач механики сплошной среды.-Б сб.: Численные методы механики сплошной среды. Новосибирск,1975, т.6, № 3, с.98-147.

10. KOZDROWICKI E.W., THEIS D.I. Second Generation of Vector Supercomputers.- Computer,1980,N 11,p.71-83.

11. Мультипроцессорные системы и параллельные вычисления /Под ред. Ф.Г.Энслоу. -М.: Мир, 1976. - 383 с.

12. KUCK D.I., STOKES R.A. The Burroughs Scientific Processor (BSP).-IEEE Trans.on Computers,1982,v.C-31,N 5,p.363-376.

13. KAPUR R.N., BROWNE J.C. Block Tridiagonal System Solution on Reconfigurable Array Computers.- Proc.of Int.Conf. of Parallel Processing,1981,p.92-99.

14. ГОДУНОВ С.К. Решение систем линейных уравнений. -Новосибирск: Наука, 1980. - 177 с.

15. GOLUB G.H., REINSCH C. Singular Value Decomposition and Least Squares Solutions.-Mumerische Math.,1970,N 14,p.403-420.

16. УИЛКИНСОН Дж., РАЙНШ К. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. -М.: Машиностроение, 1976. - 390 с.

17. AMDAHL G. Validity of the Single-Processor Approach to Achieving Large-Scale Computing Requirements.- Comp.Des., 1976, v.6,N 12,p.39-40.

18. TIDEN E., LISPER B., SCHREIBER R. Systolic Arrays.- Report TRITA-NA-8315. The Royal Institute of Technology, Stockholm, Sweden,1983.- 45 p.

Поступила в ред.-изд.отд.
24 сентября 1985 года