МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1979 год Выпуск 78

УДК 681.3.001:519.68:681.32

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ С ЛОКАЛЬНЫМИ ВЗАИМОДЕЙСТВИЯМИ

А.И.Мишин, С.Г.Садухин

Рассматривается вычислительная система (ВС) с локальными информационными и локальными управляющими взаимодействиями элементов, работа которых согласуется не с помощью команды обобщенного условного перехода [1], а посредством локальных операторов, т.е. когда каждый элемент системы взаимодействует с соседними элементами по мере их готовности к взаимодействию.

Это рассмотрение может представлять не только теоретический, но и практический интерес, например, когда параметры линии передачи влияют на временные характеристики элементов [2]. Аналогичная ситуация имеет место и для случая оптоэлектронных вычислительных систем, для которых уже при плотности компоновки элементов $\sim 10^4$ эл.см$^{-2}$ длина оптического канала связи сопоставима с линейными размерами элемента [3]. При этом максимальное значение произведения $\rho \nu \leq 10^{12}$ эл.гц.см$^{-2}$, где ν - тактовая частота работы элементов, определяемая плотностью их размещения ρ, размерностью вычислительной структуры и скоростью распространения сигнала.

Показано на примере ряда задач, что несмотря на то, что каждый элемент обладает задержкой при передаче сигнала на длительность такта, как и в случае клеточных автоматов (см.например, [4]), однако система может иметь теоретически неограниченную производительность, пропорциональную сложности задачи.

1°. Взаимодействие процессоров в системе рассмотрим на модели ВС (рис.1), содержащей запоминающие устройства $ЗУ_1, ЗУ_2, \ldots, ЗУ_n$, процессоры $П_1, П_2, \ldots, П_n$ и общее ЗУ, посредством которого осуществляется обмен информацией между процессорами.

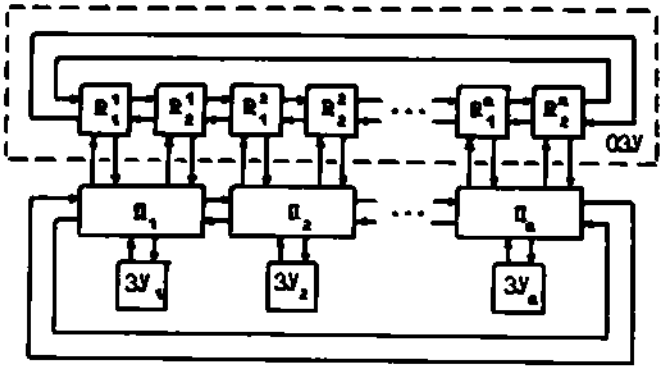

Рис. I

Если состояние i-го процессора ($i = \overline{1,n}$) в момент времени t обозначить через $\theta_i(t)$, то полную схему взаимодействия между параллельными процессами в общем случае можно задать в виде следующего отображения Φ: $\theta_i(t) \to (\theta_1(t+1), \theta_2(t+1), \ldots, \theta_n(t+1))$ ($i = \overline{1,n}$).

Максимальная производительность системы будет достигнута при условии отсутствия конфликтов при параллельном обращении всех процессоров к общему ЗУ, т.е. в том случае, когда схему взаимодействия в каждом такте (шаге) q можно представить в виде взаимно-однозначного отображения Ψ:

$$\theta_{i_q}(t+q-1) \to \theta_{(i_q+q) \bmod n}(t+q): \quad i_q = \overline{1,n}; \quad q = \overline{1,n}. \qquad (I)$$

Возможна другая процедура реализации полной схемы взаимодействия Φ, когда на каждом этапе вычислений все последующие слова $\theta_i(t+1)$ "зависят" только от одного исходного $\theta_i(t)$:

$$\theta_i(t+i-1) \to (\theta_1(t+i), \theta_2(t+i), \ldots, \theta_n(t+i)) \quad (i = \overline{1,n}). \qquad (2)$$

Такие "глобальные взаимодействия" процессоров снижают быстродействие системы, так как время реализации каждого такого взаимодействия для рассматриваемой модели равно $n\tau$, где τ – длительность такта.

Сказанное относительно взаимодействия процессоров в системе относится и к взаимодействию адресов ячеек внутри каждого ЗУ. Например, если $ЗУ_i$ построено по принципу ЗУ с произвольной выбор-

кой, т.е. с использованием глобальных связей, то его быстродействие минимально.

Учитывая, что между блоками $П_i$ и $ЗУ_i$ и между блоками $П_i$ и общего ЗУ существуют как информационные, так и управляющие взаимодействия, а также то, что каждое взаимодействие может быть осуществлено либо по схеме глобальных, либо локальных взаимодействий, можно заключить, что существуют шестнадцать типов ВС, причем только система с локальными информационными и локальными управляющими взаимодействиями как между процессорами, так и между процессорами и памятями имеет теоретически неограниченную производительность, пропорциональную сложности задачи (предполагается, что решение задачи осуществляется по схеме локальных взаимодействий).

Оценим производительность системы из n процессоров. Один процессор с быстродействием $T_п$ (сек/бит) решает задачу с объемом вычислений W (бит) ($W = f(V)$, V (бит) – объем исходных данных) за время $\max\{T_п, T'_{зу}\} W$, где $T'_{зу}$ – длительность цикла ЗУ' емкостью V (бит). При решении задач итерационными методами на n-процессорной системе каждый процессор обрабатывает количество информации Wn^{-1} (бит) за время $\max\{T_п, T''_{зу}\} Wn^{-1}$, где $T''_{зу}$ – длительность цикла ЗУ'' емкостью Vn^{-1} (бит). Кроме этого, системе требуется время $T_к Wn^{-1}$ (где $T_к$ (сек/бит) – задержка канала взаимодействия) на взаимодействие процессоров. В этом случае производительность системы равна

$$S_n = \frac{\max\{T_п, T'_{зу}\}W}{\max\{T_п, T''_{зу}\}Wn^{-1} + T_к Wn^{-1}} = \frac{\max\{T_п, T'_{зу}\}n}{\max\{T_п, T''_{зу}\} + T_к}.$$

Из этого выражения следует, что если функции ЗУ' и ЗУ'' выполняются устройствами с локальными взаимодействиями элементов памяти, то при $n \to \infty$ величина S_n может неограниченно увеличиваться только при "локальных взаимодействиях" процессоров, в противном случае $T_к = f(n)$ (например, для кольцевой системы $T_к = n\tau_э$, где $\tau_э$ – задержка элементарной логической схемы) и S_n стремится к постоянной величине, равной $\max\{T_п, T'_{зу}\}\tau_э^{-1}$.

Параллельный обмен информацией между процессорами посредством локальных взаимодействий (I) может быть выполнен с помощью циклического ЗУ, содержащего n основных и n вспомогательных регистров (модулей памяти) R_1^i и R_2^i (см. рис. I). При обращении процессоров к такому общему ЗУ каждый процессор $П_i$ считывает информацию с соответствующего регистра $R_1^i(R_2^i)$ и записывает в регистр $R_2^i(R_1^i)$.

В синхронном режиме работы все процессы производят "чтение" и "запись" одновременно, после чего информация с регистров $R_1^i(R_2^i)$ переписывается в регистры $R_2^{(i-1) \bmod n}(R_1^{(i+1) \bmod n})$, а при асинхронной работе процессоры обмениваются информацией по сигналам готовности - каждая пара процессоров взаимодействует по принципу "запрос-ответ".

2^0. В качестве примера локальных и глобальных взаимодействий процессов рассмотрим вычисления на системе из n процессоров, позволяющей одновременно производить n операций, рекуррентного выражения

$$X_k = A_k X_{k-1} + G_k , \qquad (3)$$

где $A_k = [a_{ij}^k]$ - матрица размера $n \times n$; $X_k = [x_j^k]$, $G_k = [g_i^k]$ - матрицы размера $n \times 1$. Возможные способы вычисления этого рекуррентного выражения могут различаться только вычислением произведения матриц $A_k X_{k-1}$, которое может быть осуществлено двумя способами.

При первом способе вычисления выражения (3) (глобальные взаимодействия) в одном такте вычисляются n произведений $a_{ij}^k x_j^{k-1}$ для одного и того же значения $j \in \overline{1,n}$. Очевидно, что вычисление $A_k X_{k-1}$ для всех $j = \overline{1,n}$ осуществится за n шагов, а вычисления X_k осуществляются за $(n+1)$ шагов.

При втором способе вычисления (3) (локальные взаимодействия) в одном такте вычисляются произведения $a_{ij}^k x_j^k$ для различных значений $j = \overline{1,n}$. Матричное произведение $A_k X_{k-1}$ будет найдено, как и в первом случае, за n шагов, а X_k - за $(n+1)$ шагов.

Первый способ вычисления основан на глобальном взаимодействии вычислительных процессов (2), при котором каждый процессор обрабатывает один и тот же операнд, формируемый тем процессором, который в данном такте является ведущим.

Длительность такта (шага) T определяется временем $T_П$ обработки операнда на процессоре и временем $T_{ЗУ}$ выборки операнда, зависящего от емкости памяти V и способа организации вычислений. Принимая во внимание, что $T = \max(T_П, T_{ЗУ})$, а $T_{ЗУ}$ для ЗУ с произвольной выборкой равно $c_1 \sqrt[p]{V}\, \tau_э$ (где c_1 - константа, не зависящая от V, p - размерность ЗУ, а $\tau_э$ - задержка элементарной схемы), можно заключить, что при больших V длительность такта при глобальных взаимодействиях элементов памяти определяется не временем $T_П$, а временем $T_{ЗУ}$. Учитывая также, что при решении задач линейной алгебры $V = c_2 n^2$ (где n - порядок матрицы, а c_2 - констан

та, не зависящая от n), можно заключить, что для ВС с глобальными взаимодействиями процессора (процессоров) с элементами двумерной памяти существует такое число n^*, начиная с которого длительность такта пропорциональна n.

При локальных взаимодействиях длительность такта не зависит от числа процессоров в системе, так как каждый процессор на каждом такте обрабатывает операнд, отличный от других. Причем эти операнды могут быть занумерованы (упорядочены) так, что в каждом такте каждый процессор будет обрабатывать операнд, номер которого отличается на единицу от номера предыдущего операнда.

3°. Взаимодействие элемента кольцевой ВС (которая может быть получена путем настройки - программной или технологической, - например, вычислительной структуры (среды)) с соседними элементами можно задать функцией $\Omega(\gamma_i, \gamma_j, \gamma_n)$, значение которой, а также значение ее аргументов принадлежат алфавиту $\Gamma = \{\gamma_q\}$ $(q = \overline{1,\mu})$. Равенство $\Omega(\gamma_i, \gamma_j, \gamma_n) = \gamma_n$ означает: если элемент, находясь в состоянии γ_j, воспринимает от левого соседа γ_i, а от правого γ_n, то в следующий момент времени $t = [0, \infty)$ он переходит в состояние γ_n. Такое равенство можно записать в виде неймановой команды $\gamma_i\gamma_j\gamma_n \to \gamma_n$ [5].

Элемент может находиться в одном из следующих активных состояний: γ_1 - "прием информации от левого соседа", γ_2 - "передача информации правому соседу", γ_3 - "прием информации от правого соседа" и γ_4 - "передача информации левому соседу". Кроме этого, имеются два настроечных состояния γ_5 и γ_6 для указания граничных элементов ВС и два состояния γ_7 и γ_8 для задания требуемой системы команд.

Функционирование элемента, принадлежащего некоторой связной системе (линейной или циклической) и не являющегося граничным (это указывается в процессе настройки системы), задается командами: 1) $\gamma_2\gamma_1\gamma_{1,2} \to \gamma_2$; 2) $\gamma_{1,2}\gamma_2\gamma_1 \to \gamma_1$; 3) $\gamma_{3,4}\gamma_3\gamma_4 \to \gamma_4$; 4) $\gamma_3\gamma_4\gamma_{3,4} \to \gamma_3$, где $\gamma_{1,2}$ ($\gamma_{3,4}$) означает, что элемент находится в одном из состояний: γ_1 или γ_2 (γ_3 или γ_4).

Отметим, что после приема информации элемент выполняет (если требуется) ее обработку, а затем переходит в состояние γ_2 или γ_4.

Взаимодействие элемента с внешней средой осуществляется с помощью команд $\gamma_p\gamma_p\gamma_p \to \gamma_p$, $p = 1,2,3,4$.

Возможность изменения направления передачи и приема информации увеличивает логическую гибкость ВС, так как позволяет естест-

венным образом реализовать вычислительные процессы, основанные на изменении направления хода процесса (см.,например,прямой и обратный ходы при решении системы уравнений методом Гаусса).

4^0. Из особенностей работы рассматриваемой ВС непосредственно следует,что наибольшая эффективность ее применения может быть достигнута при решении задач, которые могут быть вычислены путем массовых локальных действий, т.е. параллельной обработкой информации в окрестности каждого элемента. Примерами таких задач могут служить задачи, связанные с обработкой и распознаванием изображений [6], с решением систем дифференциальных уравнений в частных производных [7], решением систем линейных уравнений, выполнением матричных операций и т.п. всех тех задач, при решении которых необходимо выполнение большого числа однотипных параллельных вычислений. Решение задачи на ВС представляется системой параллельных идентичных локально взаимодействующих процессов, учитывающих особенности функционирования элементов ВС. При записи алгоритмов на языке АЛГОЛ командам 2/4 поставлены в соответствие процедуры sendr(x)/sendl(x) передачи операнда x правому/левому соседям, а командам I/3 - процедуры receivel(x)/receiver(x) приема операнда x от левого/правого соседей.

Многие задачи сводятся к решению систем линейных алгебраических уравнений, основными методами решения которых являются итерационные и точные. Итерационный метод основан на вычислении рекуррентных соотношений. Основным точным методом решения системы линейных уравнений является метод последовательного исключения неизвестных.

Учитывая также, что задачи интерполирования, решения дифференциальных уравнений, а также n-мерных задач математической физики основаны на решении систем линейных алгебраических уравнений (например, методы стрельбы и прогонки являются частным случаем метода исключения неизвестных для $(2m + 1)$-диагональной матрицы при $m << n$, где n - порядок матрицы [8]), можно заключить, что распараллеливание решения этих задач основано на распараллеливании решения систем линейных уравнений.

Поскольку основу вычислительных методов составляют матричные операции (преобразование Фурье, например, сводятся к умножению матриц [9]), вначале рассмотрим параллельную работу элементов при выполнении умножения матриц, а затем работу элементов при решении системы линейных алгебраических уравнений.

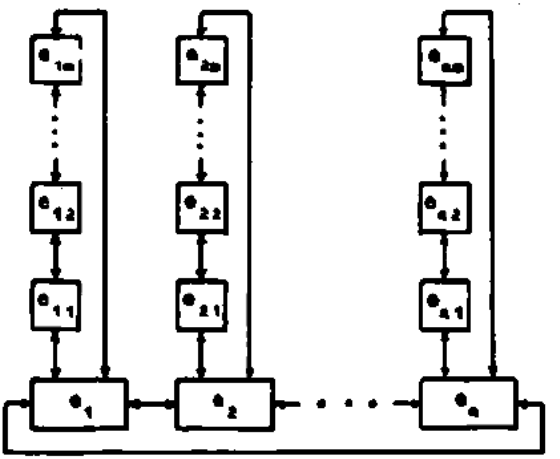

Рис. 2

Структура ВС для решения задач линейной алгебры, системы обыкновенных дифференциальных уравнений, задач математической физики, а также информационно-логических задач (рис.2) может быть получена, например, путем настройки вычислительной структуры (среды) [10, 11]. Рассматриваемая ВС содержит n кольцевых запоминающих систем, образованных элементами $e_{ij}(i = \overline{1,n};\ j = \overline{1,m})$, и одну кольцевую систему из элементов (процессоров) $e_k(k = \overline{1,n})$, посредством которой осуществляется взаимодействие между запоминающими системами. Процессор e_k с соответствующей запоминающей системой образует вычислительную машину, называемую в дальнейшем "вычислителем".

I. Умножение матриц B x A. Пусть все n "вычислителей" соединены в цепочку и каждый i-й "вычислитель" $(i = \overline{1,n})$ содержит соответствующий i-й столбец матрицы A размером m x n. Все "вычислители" принимают соответствующие значения b_{jk} матрицы B размером r x m, поступающие от соответствующих соседних элементов (на вход первого "вычислителя" значения b_{jk} подаются извне), транслируют эти значения соседним "вычислителям", определяют произведения $b_{jk}a_{ki}$, к которым прибавляются произведения $b_{jk}a_{ki}$, вычисленные на предыдущем шаге. Результирующее значение $c_{ji} = \sum_{k=1}^{m} b_{jk}a_{ki}$, вычисленное соответствующим i-м "вычислителем", выдается на (m+i+1)-м шаге во внешнюю среду. Если на вход первого "вычислителя" значения b_{jk} поступают с периодом τ, то значения c_{ji} выдаются "вычислителями" с этим же периодом τ (первоначальная задержка для i-го "вычислителя" составляет

(n+1)τ). Алгоритм работы i-го "вычислителя" можно записать в следующем виде:*)

```
multi:begin integer k; real s; real array a[1:m];
 loopi:begin s:=0;
          for k:=1 step 1 until m do
          begin receivel(b[k]);s:=s+a[k]·b[k];
            if i≠n  then sendr(b[k])
          end;
          output(s); go to loopi
       end;
    end  multi
```

2. М е т о д и т е р а ц и и. Общая схема вычисления корней системы линейных уравнений методом итерации может быть представлена циклическим графом, содержащим (n+1) вершин (n вершинам сопоставлены n вычислительных процессов, а одной вершине - следящий за приближением процесс).

Каждый i-й "вычислитель" содержит коэффициенты $b_{i1}, b_{i2}, \ldots, b_{in}$, элемент g_i и программу работы. Предполагается, что система уравнений приведена к виду $X^{(k)} = G + BX^{(k-1)}$, где $G=[g_i]$ - столбцевая матрица, $B=[b_{ij}]$ - матрица размером n x n, k - номер итерации.

Процесс решения системы уравнений методом итерации аналогичен процессу умножения матриц, за исключением того, что на (n+1)-м шаге каждый i-й "вычислитель" выдает результирующее значение $x_i^{(n)}$ не во внешнюю среду, а соседнему элементу. Вывод результата вычисления осуществляется следящим процессом, выдающим значение $X^{(n)}$ во внешнюю среду при выполнении условия $|X^{(n)} - X^{(n-1)}| < \varepsilon$, $\varepsilon > 0$, либо условия k = l, где l - заданное число итераций.

Программа работы i-го "вычислителя" имеет вид:

```
processi: begin real s,g[i]; integer p,q,j; real array b[1:n];
          real procedure modn(z); real z;
          s:=0; q:=0;
          loopi: s:=g[i] + b[i] x a;
                 for j:=1 step 1 until n-q do
```

*) Здесь и далее i является константой, т.е. ее описание не требуется, а m и n получают положительные значения.

```
                begin p:=if q=1 then modn(i-j) else j;
                receive1(x[p]); s:=s + b[p]*x[p];
                sendr(x[p]);
              end;
              q:=1; sendr(s); go to loop1;
      end;
```

comment процедура modn(s) вычисляет значение числа s по модулю n.

Программа работы следящего процессора имеет вид:

```
control:begin integer k,j,i; real array y[1:n]; real s,eps;
          for j:=1 step 1 until n do
        begin receiver(x[j]); y[j]:=x[j]
        end; k:=0;
    count:k:=k+1;  s:=0;
        for j:=n step-1 until 1 do
        begin receiver(x[j]); if abs(y[j]-x[j])< x[j]then
        s:=x[j]; y[j]:=x[j]; sendl(x[j])
      end;
      go to if s< eps then stop else if k< 1 then count
        else stop;
  stop:output(y);
 end control
```

На рис.3 показана зависимость числа активных "вычислителей" h при n = 10 от времени (номера шага) t. Можно выделить два режима функционирования системы: переходный (h = f(t)) и стационарный (h = const). Переходный режим системы имеет место до (3n-1) такта, причем на шаге n величина h достигает максимального значения, равного n, а на шаге 2n - минимального, равного n/2 . Начиная с (3n-1) такта, система входит в стационарный режим.

3. М е т о д Г а у с с а (метод исключения неизвестных). Процесс решения системы линейных уравнений AX=B , где A=[$a_{i,j}$]- матрица порядка n, а B = [$a_{i,n+1}$] и X = [x_i]- столбцевые матрицы, заключается в приведении исходной системы

$$\sum_{j=1}^{n} a_{i,j}x_j = a_{i,n+1}, \quad i = \overline{1,n},$$

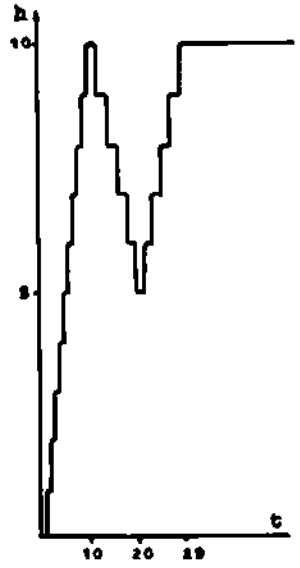

Рис. 3

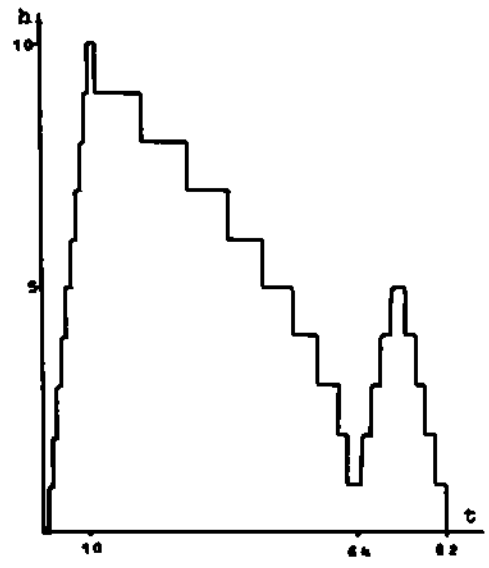

Рис. 4

к треугольному виду

$$x_i + \sum_{j=i+1}^{n} a_{ij}^{i} x_j = a_{i,n+1}^{i}, \quad i = \overline{1,n},$$

путем деления i-го уравнения на a_{ii}^{i-1}, и вычитания i-го уравнения, умноженного на a_{ik}^{i-1}, из уравнений с номерами $k=i+1,\ldots,n$. Решение осуществляется на вычислительной системе из n элементов. Алгоритм работы i-го "вычислителя" имеет вид:

```
gauss1:begin integer j,z; real array a[1:n+1], b[i+1:n+1];
       real s; z:=1;
       loopi:begin if (i-z)=0
             then begin for j:=i+1 step 1 until n+1 do
             begin b[j]:=a[j]/a[i]; if i≠n then sendr(b[j])
             end; s:=0;
               for j:=0 step 1 until n-i-1 do
             begin receiver(x[n-j])→ s:=s+b[n-j]*x[n-j];
              if i≠1 then sendl(x[n-j]) else output(x[n-j])
             end; x[i]:=b[n+1]-s;
```

```
            if i<>1 then sendr(x[i]) else output(x[i])
          end;
    else begin for j:=z+1 step 1 until n+1 do
         begin receive1(b[j]); a[j]:=a[j]-a[z]*b[j]; if i<>n then
               sendr(b[j])
         end; z:=z+1; go to loop1
         end;
    end;
end gauss
```

На рис. 4 показана зависимость числа активных "вычислителей" от времени при n = 10. Видно, что через n шагов активны все n "вычислителей", а затем число активных "вычислителей" уменьшается вплоть до одного на шаге $n(n+1)/2 + n-1$, характеризующем окончание прямого хода вычислений. Далее число активных "вычислителей" в системе увеличивается до $n/2$, а затем уменьшается до одного "вычислителя", который на шаге $n(n+7)/2-3$ заканчивает вычисление.

4. И н ф о р м а ц и о н н о - л о г и ч е с к а я з а д а ч а. Одной из важных задач, встречающихся при решении численными методами, является информационно-поисковая задача, включающая элементы комбинаторики. Например, при решении системы линейных уравнений методом главных элементов требуется перед исключением каждого неизвестного найти максимальный элемент в соответствующей матрице, т.е. каждый этап вычисления требует решения информационно-поисковой задачи и задачи исключения неизвестного. При этом поиск максимальных элементов осуществляется для матриц порядков $n, n-1, \ldots, 1$, а общий объем вычислений (сравнений) составляет $\sum_{k=1}^{n} k^2$, что соизмеримо с объемом вычислений по исключению неизвестных методом Гаусса, при котором за главный элемент всегда выбирается левый верхний элемент соответствующей матрицы.

Рассмотрим простую задачу поиска максимального (минимального) элемента в матрице $A=[a_{i,j}]$ $(i,j = \overline{1,n})$ на кольцевой системе из n "вычислителей". Пусть каждый i-й "вычислитель" содержит i-ю строку исходной матрицы A и программу работы. Каждый i-й "вычислитель" независимо от других осуществляет поиск максимального элемента в своей строке, расходуя на это n тактов. После определения

максимальных элементов в строках матрицы ищутся максимальные элементы среди найденных. Для этого каждый i-й "вычислитель", вычислив свой максимальный элемент, переходит в режим передачи информации (i+1) "вычислителю" и приема информации от (i-1) "вычислителя". Каждый "вычислитель" сравнивает свой максимальный элемент с принятым, запоминает максимальный из этих элементов и транслирует принятый элемент соседнему (i+1) "вычислителю". Время поиска максимального (минимального) элемента в исходной матрице A порядка n с учетом времени распространения сигнала пуска системы составляет 3n тактов.

Отметим, что при интерполировании функции многочленом степени n задача интерполирования сводится либо к решению системы линейных алгебраических уравнений, либо к вычислению многочлена по схеме Горнера (распараллеливание вычисления линейного рекуррентного соотношения первого порядка (схема Горнера) рассмотрено в [I2,I3]).

Таким образом, многие задачи можно свести к вычислению либо рекуррентных соотношений первого порядка, либо рекуррентных соотношений n-го порядка, либо к решению информационно-логической задачи, либо к задаче, включающей комбинацию указанных задач. При этом управление параллельными процессами осуществляется самой схемой вычисления, т.е. нет необходимости в использовании других представлений параллельных процессов.

Из алгоритмов решения задач на одном "вычислителе" и коллективе "вычислителей" можно определить число алгоритмических шагов, требуемых для решения задач. Алгоритмический шаг складывается из приема операнда, его обработки и передачи операнда соседнему процессору.

Характеристики рассматриваемой кольцевой ВС из n "вычислителей" при решении ряда задач: I) преобразование Фурье, 2) умножение матриц, 3) информационно-поисковая задача; решение линейных уравнений 4) методом Гаусса, 5) методом главных элементов, 6) методом простой итерации, 7) методом Зейделя; определение собственных значений матрицы 8) методом Данилевского, 9) методом Крылова, I0) решение системы обыкновенных дифференциальных уравнений - представлены в таблице*).

В нижеследующей таблице:

T_1 - время (число алгоритмических шагов) выполнения операции (задачи) на одном процессоре;

---

*) Номера задач соответствуют строкам таблицы.

Таблица

№	Операция	T_1	T_n	S_n	E_n
1	$y = \frac{1}{n} Px$	n^2	$2n$	$n/2$	$1/2$
2	$A\,B$	n^3	n^2	n	1
3	$(\min)\max(A)$	n^2	$3n$	$n/3$	$1/3$
4	$AX = B$	$(n^3+3n^2+2n-3)/3$	$(n^2+7n-6)/2$	$2n/3$	$2/3$
5	$AX = B$	$2n^3/3$	$4n^2$	$n/6$	$1/6$
6	$X = BX + G$	$kn(n+1)$	$k(n+1)$	n	1
7	$X = BX + G$	$kn(n+1)$	$2kn$	$(n+1)/2$	$1/2$
8	$\det(A-\lambda I)=0$	n^3+n^2-2n	$2n^2-3$	$(n+1)/2$	$1/2$
9	$\det(A-\lambda I)=0$	$n(n^2+n)$	n^2+n	n	1
10	$Y' = F(X,Y)$	$n^2(4n+1)$	$n(4n+1)$	n	1

T_n - время выполнения операции на n процессорах;

$S_n = T_1/T_n$ - ускорение процесса вычисления на n процессорах по сравнению с одним процессором;

$E_n = S_n/n$ - эффективность параллельного выполнения операции;

k - число итераций при выполнении задач 6 и 7.

Полученные показатели эффективности распараллеливания процесса решения задачи согласуются с оценками, приведенными в [5], где показано, что время при параллельном вычислении на автоматах Неймана не может быть меньше корня квадратного от времени при последовательном вычислении.

Хотя приведенные в таблице параметры характеризуют распараллеливание вычислительного процесса до уровня, пропорционального порядку матрицы, показатели эффективности распараллеливания (S_n, E_n) сохраняются и при решении задач на r процессорах (r=2,3,..., n), т.е. могут быть непосредственно получены из таблицы путем замены n на r.

Для задач с объемом вычислений порядка n^3 и более время ввода исходной информации (данных и программ) и время вывода результатов вычислений снижают эффективность рассматриваемой ВС не более чем в два раза, так как количество исходной информации пропорционально квадрату от порядка матрицы, а количество выводимой информации пропорционально порядку матрицы.

Литература

I. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. -Новосибирск: Наука, 1966. -308 с.

2. КИИЗ Р.У. Физические ограничения цифровых электронных схем. -Тр.ин-та инж. по электротехнике и радиоэлектронике. 1975, № 5, с.5-38.

3. МИШИН А.И., КОСЦОВ Э.Г. Особенности построения процессора изображений. -Однородные вычислительные системы и среды. Материалы IУ Всесоюзной конференции. Киев, 1975, с.154-156.

4. МИШИН А.И. Обработка и распознавание изображений с помощью клеточных автоматов. -Вычислительные системы. Вып.56. Новосибирск, 1973, с.136-149.

5. ТРАХТЕНБРОТ Б.А. Алгоритмы и вычислительные автоматы.-М.: Сов.радио, 1974. -200 с.

6. РОЗЕНФЕЛЬД А. Распознавание и обработка изображений. -М.: Мир, 1972. -230 с.

7. СОФРОНОВ И.Д. Оценка параметров вычислительной машины, предназначенной для решения задач механики сплошной среды.-Численные методы механики сплошной среды. Новосибирск, 1975, т. 6, № 3, с. 96-147.

8. БАХВАЛОВ Н.С. Численные методы. -М.: Наука, 1975. -632 с.

9. ФАДДЕЕВА В.Н., ФАДДЕЕВ Д.К. Параллельные вычисления в линейной алгебре. -Кибернетика, 1977, № 6, с.28-40.

10. МИШИН А.И. Вычислительная среда. Решение о выдаче авторского свидетельства по заявке № 2503997/18-24(088763) от 18.07.78 г.

11. МИШИН А.И. Ячейка однородной вычислительной среды. Решение о выдаче авторского свидетельства по заявке № 2716233/18-24 (016456) от 29.01.79 г.

12. HYAFIL L., KUNG H.T. The Complexity of Parallel Evaluation of Linear Recurrences.-J.ACM,1977,v.24, N 3,July,p.513-521.

13. KOGGE P.M., STONE H.S. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. - IEEE Trans.on Comput.,1973,v.C-22,N 8,August,p.786-792.

Поступила в ред.-изд.отд.
7 февраля 1979 года