ОДНОРОДНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ ИЗ МИКРО-ЭВМ
(Вычислительные системы)
1983 год Выпуск 97

УДК 681.324:681.142.4

ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА МНОГОМАШИННОМ КОМПЛЕКСЕ С НЕОДНОРОДНЫМИ СВЯЗЯМИ МЕЖДУ МАШИНАМИ

Ю.И. Колосова

В настоящее время предложено значительное число параллельных алгоритмов, записываемых на том или ином языке параллельного программирования. Отработана методика расширения установившихся языков последовательного программирования средствами явного задания параллелизма. Однако практическая реализация таких алгоритмов на многомашинных вычислительных комплексах (МВК) затруднена ввиду отсутствия серийно выпускаемых интерфейсных модулей, ориентированных на эффективную реализацию параллельных вычислений, особенно при решении больших и сложных задач, представленных параллельными программами. Под параллельной программой понимается организованная совокупность последовательных программ (ветвей), одновременно выполняющихся на отдельных машинах МВК и вступающих между собой во взаимодействия в соответствии с требованиями параллельного алгоритма [1]. Параллельная программа может занимать все или часть машин (подсистему) вычислительного комплекса.

С другой стороны, существующие и вновь разрабатываемые МВК, объединяющие машины на базе серийно выпускаемых средств межмашинной связи, например, на базе дуплексных регистров, можно использовать для отладки и отработки определенных параллельных алгоритмов, предварительно расширив средства программирования объединяемых машин программными модулями, организующими взаимодействия с ориентацией на конкретные связи. Более того, такая модификация расширяет функциональные возможности существующих комплексов, позволяя решать качественно новые задачи, а также отрабатывать методику параллельного программирования для будущих МВК.

В данной работе предлагается подход к организации параллельных вычислений на сосредоточенных МВК, структурную схему которых можно представить в виде дерева. Каждая из машин может быть связана с k соседними, причем по аппаратурным свойствам связи могут быть различными. Любая из связей может обеспечивать обмен только между парой соединенных ею машин.

Идентификация межмашинных связей

На рис.I представлена структурная схема МВК из семи машин. Вершины дерева соответствуют отдельным вычислительным машинам комплекса, а ребра - связям между ними. Машины идентифицируются номерами, а связи - буквами. Каждая из машин i (i = 1,2,...,L , где L - число машин МВК) связана с k_i сосед - ними (смежными), причем связи могут быть неоднородными.

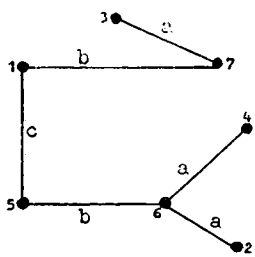

Рис. I

Для организации параллельных вычислений на подобных МВК диспетчер каждой из машин должен иметь два вида информации о связях. Во первых, об особенностях интер - фейсов с каждой из своих смежных машин и, во вторых, о маршрутах между любыми парами машин комплекса.

Информация первого вида содержится в таблице связей, каждая строка j (j = 1,2,...,k_i) которой соответствует интерфейсу данной машины с одной из своих смежных и представляет собой двойку ⟨M(j),S(j)⟩. Здесь M(j) - относительный номер j-й смежной маши - ны, S(j) - параметр, конкретизирующий связь. Каждой из связей соответствует определенный программный драйвер, учитывающий ее аппаратурные возможности и выполняющий функции приема или передачи информации при активации данной связи. Над таблицей задается операция поиска M(j) и выбора связи S(j).

Информация второго вида содержится в таблице маршрутов, каж - дый столбец которой (инициатор) j (j = 1,2,...,L) соответствует началу, а каждая строка (адресат) i (i = 1,2,...,L) - концу марш - рута. Каждый элемент таблицы указывает номер машины (транзит - T(i,j)), смежной с инициатором, через которую проходит маршрут от инициатора к адресату. Над таблицей задается операция поиска транзита T(i,j). Такой подход аналогичен представлению информации в алгоритмах фиксированной маршрутизации в вычислительных сетях [2].

Построение таблиц связей (рис.2) и маршрутов для каждой из машин комплекса может быть возложено либо на генератор операционной системы МВК, либо на некоторую автономную программу, выполняющую функции планировщика и централизованный запуск параллельной программы на счет.

M(j)	S(j)
7	a
5	c

M(j)	S(j)
6	b
1	c

M(j)	S(j)
4	a
5	b
2	a

M(j)	S(j)
1	b
3	a

Рис. 2

При создании таблицы связей для каждой машины i (i = 1,2,... ...,L) должно быть указано: общее число k_i машин, смежных с данной, и их относительные номера вместе с соответствующими параметрами, характеризующими связи на физическом уровне. Другими словами, по каждой из связей должна быть привязка к конкретному драйверу. На рис.2 представлен один из возможных вариантов таблиц связей для машин I,5,6,7, объединенных в комплекс согласно рис.I.Для машин 2,3 и 4 таблица связей содержит по одной строке.

При создании таблицы маршрутов можно использовать два подхода соответственно для простых и сложных структур МВК. Для простых структур по каждой из машин сразу указываются транзиты ко всем ее адресатам. Для сложных структур МВК построение таблицы маршрутов возлагается на программу, работающую в два этапа. На первом этапе по аналогии с [3] структура вычислительной установки представляется в форме связного списка на основе информации о всех ребрах дерева, характеризуемых инцидентными вершинами,на втором этапе выполняется модифицированный алгоритм поиска в глубину [3], который по связному списку находит все транзиты к адресатам по каждой из машин МВК.

Программа построения таблицы маршрутов

```
      PROGRAM TMAR
      DIMENSION IADJ(400),NEXT(400),IYIS(200),ITAB(20,20),IST(200)
1700  FORMAT(//,8X,"СВЯЗНЫЙ СПИСОК",/,"ЧИСЛО ВЕРШИН=",I2,2X,
     *"ЧИСЛО  РЕБЕР",I2,/16(2H--),/6X,"I",6X,"ADJ",6X,"NEXT",/16
```

```
     *(2H--),/)
 1800 FORMAT(5X,I2,6X,I2,8X,I2)
 2600 FORMAT(//,8X,"ТАБЛИЦА МАРШРУТОВ",/17(2H--),/,2X,"ИНИЦИА
     *ТОР  1  2  3  4  5  6  7  ",/,2X,"АДРЕСАТ ",/,17(2H--))
 2800 FORMAT((4X,I2,7X,7(I2,X)))
C             ЧТЕНИЕ ЧИСЛА ВЕРШИН И ЧИСЛА РЕБЕР

    7 READ(5,*)IP,IQ
      I=IP+1
      IWORD=IP+2*IQ
      DO 10 J=1,400
      ITAB(J)=0
      IADJ(J)=0
   10 NEXT(J)=0
C             ЧТЕНИЕ РЕБЕР И ОРГАНИЗАЦИЯ СВЯЗНОГО СПИСКА
      DO 100 J=1,IQ
      READ(5,*) M,N
      NEXT(I)=NEXT(M)
      IADJ(I)=N
      NEXT(M)=I
      I=I+1
      NEXT(I)=NEXT(N)
      IADJ(I)=M
      NEXT(N)=I
      I=I+1
  100 CONTINUE
      WRITE(7,1900)(IP,IQ)
      WRITE(7,1800)(I,IADJ(I),NEXT(I),I=1,IWORD)
C             ОРГАНИЗАЦИЯ ТАБЛИЦЫ МАРШРУТОВ
  110 DO 500 II=1,IP
      IY=II
      DO 120 J=1,200
      IYIS(J)=0
  120 IST(J)=0
      ISX=IY
      IT=1
      ITR=0
      ITAB(IY,II)=IY
      DO 400 I=1,IP
```

```
      IYIS (IY)=I
      K=IY
  200 K=NEXT(K)
      IF(K)300,210,300
  210 IF(ITR)220,500,220
  220 IY=IST(ITR)
      K=IST(ITR-1)
      ITR=ITR-2
      GO TO 200
  300 IW=IADJ(K)
      IF(ISX-IY)320,310,320
  310 MC=IW
  320 IF(IYIS(IW))200,330,200
  330 ITAB(IW,II)=MC
      IT=IT+1
      IF(IP-IT)500,500,340
  340 IST(ITR+1)=K
      IST(ITR+2)=IY
      ITR=ITR+2
      IY=IW
  400 CONTINUE
      GO TO 600
  500 CONTINUE
      WRITE(7,2600)
      WRITE(7,2800)(I,(ITAB(I,J),J=1,IP),I=1,IP)
  600 END
```

На рис.3 приведены результаты выполнения программы построения таблицы маршрутов. По некоторому произвольному перечислению ребер (а) структуры МВК из рис.I получены связный список (б) и таблица маршрутов (в).

Обратим внимание, что параллельная программа может занимать или весь МВК, или же его подсистему.В последнем случае предусматривается создание на уровне операционной системы так называемых рабочих таблиц связей и маршрутов, получаемых из основных в соответствии с перенумерацией машин в подсистеме.

Далее отметим, что в общем случае МВК может иметь и более сложную топологию по сравнению с рассматриваемой. Тогда построение таблицы маршрутов может быть основано на методах поиска кратчайших путей, например, с помощью алгоритма Дейкстры [3].

Связный список

Число вершин = 7
Число ребер = 6

4 6	I		
7 3	I	0	I8
5 6	2	0	I6
7 I	3	0	II
2 6	4	0	8
I 5	5	0	I9
	6	0	I7
а)	7	0	I4
	8	6	0
	9	4	0
	I0	3	0
	II	7	0
	I2	6	0
	I3	5	9
	I4	I	I0
	I5	7	0
	I6	6	0
	I7	2	I3
	I8	5	I5
	I9	I	I2

б)

Таблица маршрутов

Адресат	Инициатор						
	I	2	3	4	5	6	7
I	I	6	7	6	I	5	I
2	5	2	7	6	6	2	I
3	7	6	3	6	I	5	3
4	5	6	7	4	6	4	I
5	5	6	7	6	5	5	I
6	5	6	7	6	6	6	I
7	7	6	7	6	I	5	7

в)

Рис. 3

Организация взаимодействий ветвей

Использование таблиц связей и маршрутов в подпрограммах, реализующих взаимодействия между ветвями параллельной программы в соответствии с семантикой языковых средств их задания, позволяет унифицировать организацию обмена информацией на МВК с неоднородными связями между машинами. Обратим внимание, что речь идет не об эффективности, а о возможности реализации параллельной программы на конкретной вычислительной установке.

Рассмотрим алгоритмы функционирования подпрограмм, реализующих взаимодействия, задаваемые в рамках идеологии однородных вычислительных систем [I]. Номера ветвей соответствуют номерам ма-

шин, на которых они выполняются. При обращении к подпрограмме, например, на ФОРТРАНе с помощью оператора CALL , указывается ее имя, а в скобках перечисляются параметры, имеющие следующий смысл:

N2 - спецификатор передаваемого массива, отличающий операторы с одинаковыми именами, т.е. операторы, задающие одно и то же взаимодействие, должны во всех ветвях иметь одинаковое значение N2;

N1 - номер передающей (или принимающей) ветви;

NS - тип передаваемого массива, указывающий число ячеек, занимаемых одним элементом, например, для массива целых чисел NS = = 1, для вещественных - NS = 2;

N - число передаваемых элементов;

A - идентификатор передаваемого массива;

B - идентификатор принимающего массива.

Остальные параметры характеризуются по ходу их появления при описаниях алгоритмов. При выполнении подпрограмм используется присутствующая в каждой машине информация о порядковом относительном номере IS соответствующей ветви, о числе k смежных с ней ветвей (IS = 1,2,...,L; $k = k_1, k_2, \ldots, k_L$) и о числе L всех ветвей параллельной программы. Такая информация сообщается на уровне операционной системы при загрузке программы.

При описании будем использовать соглашения, принятые в [3].

ТРАНСЛЯЦИОННЫЙ ОБМЕН: BEXC(N2,NS,N1,N,A,B)

Algorithm BEXC. Передать из ветви IS = N1 во все остальные ветви NS·N кодов массива A в массив B.

Шаг I. [Данная ветвь - передающая или принимающая?] if IS = = N1 then шаг 2 else шаг 5 fi.

Шаг 2. [Передача всем смежным] for I ← 1 to k do throuh шаг 4 od; and ВЫХОД из подпрограммы.

Шаг 3. ПОИСК по таблице связей номера I-й смежной ветви M(I) и ВЫБОР связи S(I).

Шаг 4. ПЕРЕДАЧА ветви с номером M(I) по связи S(I) NS·N кодов из массива A.

Шаг 5. [Поиск транзита (W) от данной ветви к адресату N1] set j ← N1; and W ← T (j, IS).

Шаг 6. [Выявление смежной ветви с номером, равным значению транзита W и определение соответствующей связи] for I 1 to k do шаг 7 od .

Шаг 7. if M(I) = W then ВЫБОР связи S(I); шаг 8 fi.

Шаг 8. ПРИЕМ NS·N кодов в массив B от ветви W по связи S(I).

Шаг 9. [Передача NS·N принятых кодов в смежные ветви, отличные от транзита] for I ← 1 to k do шаг I0 od; and ВЫХОД из подпрограммы.

Шаг I0. if M(I) ≠ W then ВЫБОР связи S(I); and ПЕРЕДАЧА ветви M(I) по связи S(I) NS·N кодов из массива B fi .

Проследим работу данного алгоритма на МВК из семи машин, структура которого представлена на рис.I. Все ветви выполняются параллельно, но в общем случае асинхронно. Предполагаем, что все семь ветвей некоторой параллельной программы при своем выполнении обязательно выйдут на оператор ВЕХС. Это является одной из характеристик правильности параллельной программы [4] и отрабатывается в процессе отладки [5]. Ввиду асинхронности выполнения появление оператора ВЕХС в ветвях, как правило, не одновременно. Будем считать, для определенности, что передающей ветвью является ветвь IS = 5, т.е. N1 = 5. Пусть таблицы связей имеют вид, представленный на рис.2. Ветвь N1 = 5 имеет две смежные ветви, т.е. $k = k_5 = 2$. После выполнения шага I эта ветвь переходит на шаг 2.При первой итерации цикла задается операция над первой строкой таблицы связей, т.е. M(1) = 6 и S(1) = b . При выполнении шага 4 ветвь 5 должна передать информацию в ветвь 6 по связи b . Если в данный момент ветвь 6 выполняет шаг 8, то передача произойдет. В противном случае передача задерживается до перехода ветви 6 в состояние выполнения шага 8, т.е. при обмене ветви синхронизируются. После передачи информации ветви 6 выполняется очередная итерация цикла (шаги 2,3,4), т.е. при I = 2 M(2) = 1, S(2) = c и передача будет теперь в ветвь 1. Далее выполнение цикла заканчивается и ветвь 5 выходит из подпрограммы.

Ветви IS = I,2,3,4,6,7 имеют соответственно $k_1 = 2$, $k_2 = 1$, $k_3 = 1$, $k_4 = 1$, $k_6 = 3$, $k_7 = 2$ количество смежных ветвей. После выполнения шага I они переходят на шаг 5, как правило, не одновременно. Каждая из ветвей определяет транзит W для взаимодействия с N1 и соответствующую связь S.Значения транзитов и связей,полученных при выполнении шагов 5,6,7 в каждой из указанных ветвей дана в табл. I.

Таблица I

	Относительный номер					
	I	2	3	4	6	7
W	5	6	7	6	5	I
S	c	a	a	a	b	b

Далее ветви выполняют шаг 8, т.е. должны принять информацию от транзита W. Так как ветвь 5 вначале передает информацию ветви 6, то последняя и будет первой, принявшей информацию. Далее ветвь 6 выполняет шаги 9 и I0, передавая принятую информацию ветви 4, а затем ветви 2, в соответствии с таблицей связей (рис.2, машина 6).

На рис.4 показана волна движения информации от ветви 5 при выполнении оператора BEXC. Распространение волны можно предста - вить по уровням, объединяющим пары ветвей, которые могут обмени - ваться независимо. Выбор смежной ветви для передачи информации ба- зируется на очередности смежных ветвей в таблице связей (рис.2).

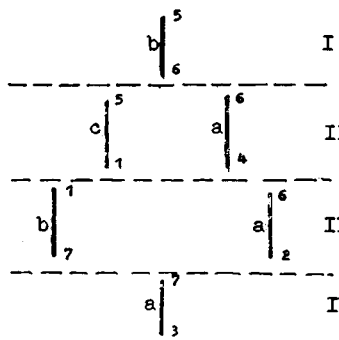

Рис. 4

Проанализируем временную слож - ность алгоритма BEXC . Будем считать, что число передаваемых кодов NS · N есть некоторая константа c , меньшая объема оперативной памяти. Тогда ал- горитм имеет сложность O(L). Такая оценка относится к случаю, когда каж- дый уровень волны движения информа - ции от передающей ветви содержит лишь одну пару обменивающихся ветвей. От- сюда можно сделать вывод, что (при одинаковых порядках характеристик скоростей межмашинных интерфейсов) таблицу связей в каждой машине следует строить в соответствии с убыванием количества машин, связываемых смежными машинами с дан - ной.

ТРАНСЛЯЦИОННО-ЦИКЛИЧЕСКИЙ ОБМЕН: CBEXC(N2, NS, N, A, B).

Algorithm CBEXC. Передать из всех ветвей в каждую ветвь IS (IS = 1,2,...,L) массив A. После выполнения данного алгоритма в каждой ветви в массиве B будут находится N принятых элементов. Из каждой ветви принимается по γ элементов массива A, где $\gamma = [N/L]+1$, если относительный номер ветви меньше или равен остатку от деле - ния N на L, иначе, $\gamma = [N/L]$ - целая часть от N/L . Элементы из первой ветви расположены с первого элемента массива B , элементы из второй - начиная с (B + γ·NS) и т.д. Выполнение данного алгорит- ма сводится к выполнению алгоритма BEXC в цикле по всем ветвям, причем каждая из передающих, кроме того, пересылает свой массив A в соответствующую часть массива B.

Шаг 0. [Инициализация] set GAM ← [N/L]; and REST ← (N - GAM·L).

Шаг I. [Выполнение алгоритма BEXC в цикле по всем ветвям] for I ← 1 to L do through шаг 5 od; and ВЫХОД из подпрограммы.

Шаг 2. [Определение числа передаваемых элементов]if I ≤ REST then set E ← (GAM + 1) else set E ← GAM fi.

Шаг 3. ПЕРЕХОД к выполнению алгоритма BEXC(N2, NS, I, E,A,B).

Шаг 4. [Пересылка переданного массива A в передающей ветви] if I = IS then ПЕРЕСЫЛКА массива A в массив B fi .

Шаг 5. [Подготовка адреса массива B для приема очередной порции информации] set B ← (B + E·NS).

Сложность алгоритма CBEXC есть $O(L^2)$, так как передающими должны быть все ветви параллельной программы .

ДИФФЕРЕНЦИРОВАННЫЙ ОБМЕН: DEXC(N2, NS,N1,N,A,B,Z1,Z2,...,ZC)

Algorithm DEXC. Передать из ветви IS = N1 в ветви Z1,Z2,...,ZC NS·N кодов массива A в массив B. Если Zi (i=1,2,...,C) равно N1, то переслать в ветви N1 массив A в массив B. Пусть BUF- память для запоминания транзитов; CT - счетчик транзитов в памяти BUF.

Шаг 0. [Инициализация] set I ← 0; CT ← 0; and BUF ← 0.

Шаг I. [Данная ветвь - передающая?] if IS=N1 thenшаг 2; else шаг I2 fi.

Шаг 2. [Проверка: все ли ZI (I=1,2,..., C) исчерпаны] if I = C then ВЫХОД из подпрограммы fi.

Шаг 3. set I ← (I+1) .

Шаг 4. [Проверка: является ли очередная ветвь ZI передающей] if ZI = N1 then ПЕРЕСЫЛКА массива A в массив B;and шаг 2 fi.

Шаг 5. [Поиск транзита (W) для передачи из N1 в ZI] set j←ZI; and W ← T(j, N1).

Шаг 6. [Проверка: был ли уже задействован найденный транзит W. Если CT = 0, то обмен не проводился ни разу] if CT = 0 then шаг 9 fi.

Шаг 7. [Сравнение имеющихся в буфере транзитов с найденным транзитом W. Если в буфере уже есть транзит, равный W, то повторной передачи в соответствующую смежную ветвь не производится] for p ← 1 to CT do шаг 8 od .

Шаг 8. if BUF(p)=W then шаг 2 fi .

Шаг 9. ПОИСК по таблице связей смежной ветви, равной транзиту W, и ВЫБОР соответствующей связи S.

Шаг I0. ПЕРЕДАЧА ветви W по связи S NS·N кодов массива A .

Шаг II. set BUF(CT+1) ← W; CT=CT+1; and шаг 2.

Шаг I2. [Проверка: все ли ZI (I=1,2,...,C) исчерпаны] if I = C then ВЫХОД из подпрограммы fi.

Шаг I3. set I ← (I+1) .

Шаг I4. [Проверка: является ли очередная ветвь ZI передающей. Если да, то обмен не производится] if ZI = N1 then шаг I2 fi .

Шаг I5. [Инициализация поиска первого транзита в маршруте от N1 к ZI] set j ← ZI; and INI ← N1.

Шаг I6. [Поиск транзита от очередного инициатора (INI) к адресату ZI] set W ← T(j,INI).

Шаг I7. [Равен ли номер данной ветви найденному адресату W?] if IS = W then шаг 20 fi.

Шаг I8. [Если найденный транзит W равен ZI, то через данную ветвь маршрут от N1 к ZI не проходит] if W = ZI then шаг I2 fi.

Шаг I9. [Инициализация поиска очередного транзита в маршруте от N1 к ZI] set INI ← W; and шаг I6.

Шаг 20. [Работа с буфером, аналогичная шагам 6, 7, 8] if CT=0 then шаг 23 fi .

Шаг 2I. for p ← 1 to CT do шаг 22 od .

Шаг 22. if BUF(p) = W then шаг 25 fi.

Шаг 23. ПОИСК по таблице связей смежной ветви, равной очередному инициатору INI и ВЫБОР соответствующей связи S.

Шаг 24. ПРИЕМ от ветви INI по связи S NS·N кодов в массив B; set BUF(CT+1) ← W; and CT ← (CT+1).

Шаг 25.[Если найденный транзит W равен ветви ZI, то информация от N1 к ZI передана, переход к выбору очередной ZI] if W = ZI then шаг I2 fi.

Шаг 26. [Если найденный транзит отличен от ZI , то инициализация поиска очередного транзита для передачи информации ZI] set INI ← W .

Шаг 27. [Поиск очередного транзита от ветви, принявшей информацию, к ветви ZI] set W ← T(j, INI).

Шаг 28.[Работа с буфером, аналогичная шагам 6-8 или же 20-22] if CT = 0 then шаг 3I fi .

Шаг 29. for p ← 1 to CT do шаг 30 od .

Шаг 30.[Если в буфере найдется транзит, равный W , то повторной передачи не производится]if BUF(p) = W then шаг I2 fi .

Шаг 3I. ПОИСК по таблице связей смежной ветви, равной транзиту W и ВЫБОР соответствующей связи S.

Шаг 32. ПЕРЕДАЧА в ветвь W по связи S NS·N кодов из массива B; set BUF(CT + 1) ← W; CT ← (CT+1)and шаг I2.

Таблица 2

Ветви						
I	2	3	4	5	6	7
0	0	0	0	0	0	0
I	I	I	I	I	I	I
2	I2	I2	I2	I2	I2	I2
3	I3	I3	I3	I3	I3	I3
4	I4	I4	I4	I4	I4	I4
5	I5	I5	I5	I5	I5	I5
6	I6	I6	I6	I6	I6	I6
9	I7	I7	I7	I7	I7	I7
(1) [I0]	I8	I8	I8	20	I8	I8
II	I9	I9	I9	23	I9	I9
2	I6	I6	I6	(1) [24]	I6	I6
3	I7	I7	I7	25	I7	I7
4	I8	I8	I8	26	20	I8
5	I9	I9	I9	27	23	I9
6	I6	I6	I6	28	(2) [24]	I6
7	I7	I7	I7	29	25	I7
8	20	I8	I8	30	26	I8
2	23	I2	I2	3I	27	I2
ВЫХОД	(3) [24]	ВЫХОД	ВЫХОД	(2) [32]	28	ВЫХОД
	25			I2	29	
	I2			I3	30	
	I3			I4	3I	
	I4			I5	(3) [32]	
	I5			I6	I2	
	I6			I7	I3	
	I7			20	I4	
	I8			2I	I5	
	I2			22	I6	
	ВЫХОД			25	I7	
				I2	I8	
				ВЫХОД	I2	
					ВЫХОД	

Проследим работу данного алгоритма на уже рассмотренном МВК из семи машин при условии, что передающей является первая ветвь (N1 = 1), а принимающими - вторая и пятая (Z1 = 2, Z2 = 5). В табл.2 перечислены выполняемые ими шаги в порядке следования по каждой ветви. Номера шагов, выполняющих обмены, обведены квадратиками, рядом с которыми в кружочке стоит номер обмена по порядку. Одинаковые номера в кружочках соответствуют обменивающим ветвям. Всего произойдет три обмена: первая → пятой, пятая → шестой, шестая → второй. Таким образом, в данном примере шестая ветвь выполняет роль транзита, а ветви третья, четвертая и седьмая никакого участия в обменах не принимают.

Сложность алгоритма DEXC равна O(L) , так как в предельном случае функционирование алгоритма DEXC аналогично функционированию алгоритма BEXC .

КОЛЛЕКТОРНЫЙ ОБМЕН: CEXC (N2, NS, N1, N, A, B, N3)

Algorithm CEXC. Передать из всех ветвей в одну ветвь N1 массив A. После выполнения данного алгоритма в ветви N1 в массиве B будут находиться N принятых элементов. Из каждой ветви принимаются по γ элементов массива A, где $\gamma = [N/L]+1$, если относительный номер ветви меньше или равен остатку от деления N на L ,иначе $\gamma = [N/L]$. Элементы от первой ветви расположены с первого элемента массива B , элементы от второй - начиная с (B+γ·NS) и т.д.Элементы массива A ветви N1 пересылаются в соответствующее место массива B , при N3 ≠ 0 , при N3 = 0 заполнения этой части массива B не происходит. Выполнение данного алгоритма сводится к выполнению алгоритма DEXC в цикле по всем ветвям.

Шаг 0. [Инициализация] set GAM ← [N/L]; and REST ← (N - GAM·L).

Шаг I. [Выполнение алгоритма DEXC в цикле по всем ветвям]for I ← 1 to L do through шаг 5 od;and ВЫХОД из подпрограммы.

Шаг 2. [Определение числа передаваемых элементов] if I ≤ REST then set E ← (GAM + 1) else set E ← GAM fi .

Шаг 3. [Если N3 ≠ 0 , то алгоритм DEXC будет выполняться и для ветви N1 , иначе шаг 4 пропускается] if I = N1 then if N3 ≠ 0 then шаг 4 else шаг 5 fi fi.

Шаг 4. ПЕРЕХОД к выполнению алгоритма DEXC(N2,NS,I,E,A,B,N1).

Шаг 5. [Подготовка адреса массива B для приема очередной порции информации] set B ← (B + E·NS).

Сложность алгоритма CEXC есть $O(L^2)$, так как все ветви должны выполнить алгоритм DEXC (эта оценка несколько завышена, так

как не при каждом обращении к алгоритму DEXC выполняется его предельный случай).

ОБМЕН СДВИГОМ: MEXC(N2, NS, x, N, A, B)

Algorithm MEXC. Передать информацию из каждой ветви в ветвь с относительным номером, на 1 бо́льшем номера передающей (при x = = 0) или на 1 ме́ньшем (при x = 1). После выполнения данного алгоритма при x=0 массив A из ветви i окажется в ветви (i+1) (i = = 1,2,...,L-1). Из последней L-й ветви массив A окажется в массиве B первой ветви. При x=1 массив A из ветви i окажется в массиве B ветви (i-1) (i = 2,3,..., L). Из первой ветви массив A окажется в массиве B L-й ветви. Выполнение данного алгоритма сводится к выполнению алгоритма DEXC в цикле по каждой паре ветвей. Так как в рассматриваемой структуре МВК ветви с указанными номерами могут быть в общем случае не смежными, то для организации транзитных обменов вводится вспомогательный массив M размером NS·N. Пересылка принятой информации из вспомогательного массива в массив B производится только для ветви-адресата.

Шаг I. if x = 0 then шаг 2 else шаг 7 fi .

Шаг 2. [Выполнение алгоритма DEXC в цикле для каждой пары ветвей при x = 0] for I ← 1 to L do through шаг 6 od; and ВЫХОД из подпрограммы.

Шаг 3. if I = L then set j ← 1 ;and шаг 5 fi.

Шаг 4. set j ← (I+1).

Шаг 5. ПЕРЕХОД к выполнению алгоритма DEXC(N2,NS,I,N,A,M,j).

Шаг 6. [Если данная ветвь равна j , т.е. адресату, то переслать NS·N кодов из вспомогательного массива M в массив B] if IS = j then ПЕРЕСЫЛКА NS·N кодов из M в B fi .

Шаг 7. [Выполнение алгоритма DEXC в цикле по каждой паре ветвей при x = 1] for I ← 1 to L do through шаг II od; and ВЫХОД из подпрограммы.

Шаг 8. if I = 1 then set j ← L; and шаг I0 fi .

Шаг 9. set j ← (I-1).

Шаг I0. ПЕРЕХОД к выполнению алгоритма DEXC(N2,NS,I,N,A,M,j).

Шаг II. if IS = j then ПЕРЕСЛАТЬ NS·N кодов из M в B fi.

Сложность алгоритма MEXC есть $O(L^2)$,так как необходимые пары ветвей могут быть сколь угодно далеко удалены друг от друга.

ОБОБЩЕННЫЙ УСЛОВНЫЙ ПЕРЕХОД: GCP(N2, F, M).

Algorithm GCP. Меняет порядок реализации операторов в ветвях в зависимости от значения обобщенного условия. Переменная F

участвует в выработке такого значения: если значения F отрицательны во всех ветвях, то управление передается на метку M в каждой из ветвей, иначе все ветви выполняют очередной оператор. Выполнение данного алгоритма сводится к выполнению алгоритма BEXC в цикле по всем ветвям. Каждая из них передает один элемент F в массив B остальных ветвей и в свой тоже.

Шаг I. [Выполнение передачи значения F во все ветви] for I←1 to L do through шаг 4 od;and ВЫХОД из подпрограммы на метку M .

Шаг 2. set B ← F .

Шаг 3. ПЕРЕХОД к выполнению алгоритма BEXC(N2,NS,I,1,F,B).

Шаг 4. [Если значение B ≥ 0, то выход из цикла] if B ≥ 0 then шаг 5 fi .

Шаг 5. [Если хотя бы одно значение B = F положительно,то будет во всех ветвях выполняться очередной оператор] ВЫХОД из подпрограммы на выполнение очередного оператора.

Сложность алгоритма GCP есть $O(L^2)$.

Использование предложенных алгоритмов в системах математического обеспечения ЭВМ, объединенных в вычислительный комплекс,позволяет пользователю оставаться в рамках языков высокого уровня и не заботиться о конкретных связях между машинами. Сведение рассмотренных алгоритмов к двум основным BEXC и DEXC не является принципиальным.Возможны и другие варианты, но их рассмотрение выходит за рамки данной работы.

Реализация

Предложенный подход к организации параллельных вычислений реализован на многомашинном вычислительном комплексе МИНИМАКС-3, состоящем из двух мини-машин "М-6000" и одной мини-машины СМ-I. Мини-машины "М-6000" соединены между собой специальными модулями межмашинной связи [I], обеспечивающими обмен информацией между этими машинами. Кроме того, одна из них соединена с мини-машиной СМ-I дуплексными регистрами АСВТ-М. На рис.5 приведена структура данного комплекса МИНИМАКС-3.

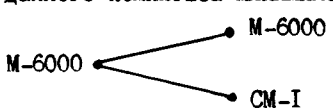

Рис. 5

Таблицы связей и маршрутов строятся автономной программой "Монитор запуска" [6], осуществляющей централизованное управление запуском параллельных программ на счет с дисплея

одной из машин комплекса. Монитор дает возможность: выбрать структуру и состав подсистемы, на которой будет выполняться параллельная программа, запустить на счет на выбранной подсистеме параллельную программу. Подсистема может состоять из трех, двух и одной машины комплекса. В последнем случае на подсистеме может выполняться либо параллельная программа, настроенная на число ветвей, равное единице, либо некоторая последовательная программа. Управление выбором структуры и состава подсистемы, а также запуском на счет ведется с помощью директив, вводимых с дисплея ВИДЕОТОН-340, установленном на одной из машин комплекса и осуществляющим функции централизованного пульта.

Параллельные программы записываются на ОВС-языках [1]. Системные программы, реализующие функции взаимодействия, разработаны в соответствии с описанными в предыдущем разделе алгоритмами с ориентацией на обработку таблиц связей и маршрутов для унификации использования различных связей. Обмен информацией между машинами осуществляется драйвером межмашинных связей и драйвером дуплексных регистров.

Заметим, что предложенные алгоритмы и их реализация соответствуют операторам ОВС-языков, задающим групповые взаимодействия между ветвями параллельной программы [1]. Это связано с предположением, что связи в рассмотренной древовидной структуре МВК могут выполнять только функции приема-передачи между парами машин комплекса. Алгоритмы, соответствующие операторам индивидуальных взаимодействий, требуют, как минимум, аппаратурной реализации в межмашинной связи признака "запрос связи" и возможности программной проверки наличия этого признака. Тем не менее такие функции взаимодействий, как индивидуальное чтение или запись [1] можно задавать посредством группового оператора DEXC. В МИНИМАКС-3 индивидуальные взаимодействия между ветвями, расположенными в мини-машинах "М-6000", могут выполняться в том объеме, как они рассмотрены в ОВС-языках [1]. Это возможно благодаря модулю межмашинной связи и операционной системе МИНИМАКС [7].

Заключение

Использование МВК со стандартными сопряжениями между машинами для выполнения параллельных вычислений расширяет его функциональные возможности, способствует накоплению опыта и развитию на-

выков параллельного программирования еще до появления вычислительных комплексов со специальными интерфейсами, ориентированными на эффективную организацию параллелизма. В частности, комплекс МИНИМАКС-3 можно использовать не только как инструмент для исследования и разработки системного математического обеспечения (например, для новых вычислительных систем из мини- и микро-ЭВМ), но и как средство для прикладного математического обеспечения (например, для автоматизированных систем управления, базирующ[illegible] таких МВК [8]).

Литература

I. Программное обеспечение системы МИНИМАКС /Кербель В.Г., Колосова Ю.И., Крылов Э.Г., Корнеев В.Д., Миренков Н.Н. - Новосибирск, 1979. - 43 с. (Препринт/Институт математики СО АН СССР, ОВС-09).

2. Вычислительные сети и сетевые протоколы / Дэвис Д., Барбер Д., Прайс У., Соломонидес С. - М.: Мир, 1982. - 563 с.

3. ГУДМАН С., ХИДЕТНИЕМИ С. Введение в разработку и анализ алгоритмов. -М.: Мир, 1981. - 366 с.

4. КОЛОСОВА Ю.И. О приемлемости блок-схем параллельных программ. -В кн.: Математическое обеспечение вычислительных систем (Вычислительные системы, вып. 78). Новосибирск, 1979, с.70-77.

5. КОЛОСОВА Ю.И. Использование системы МИНИМАКС для отладки параллельных программ. -В кн.: Архитектура вычислительных систем с программируемой структурой (Вычислительные системы. вып. 82). Новосибирск, 1980, с. 55-66.

6. КОЛОСОВА Ю.И. Монитор запуска параллельных программ на вычислительной системе МИНИМАКС-3. Отчет ИМ СО АН СССР, 1981. - 31 с.

7. КОРНЕЕВ В.Д. Операционная система МИНИМАКС. -В кн.: Математическое обеспечение вычислительных систем (Вычислительные системы, вып. 78). Новосибирск, 1979, с. 3-30.

8. Система сбора, хранения и обработки данных районного звена на базе вычислительного комплекса МИНИМАКС /Горелик В.М., Кербель В.Г., Кербель Н.К. и др. -Тезисы докл. У Всесоюз. школы-семинара "Параллельное программирование и высокопроизводительные системы". Киев, Наукова думка, 1982, часть 4, с. 26-28.

Поступила в ред.-изд.отд.
13 декабря 1981 года