В Ы Ч И С Л И Т Е Л Ь Н Ы Е С И С Т Е М Ы

Сборник трудов
1967г. Института математики СО АН СССР Выпуск 24

РАСПАРАЛЛЕЛИВАНИЕ ПО ЦИКЛАМ

Ю.Г. Косарев

Изложенная в работах [I–4] методика решения задач на вычислительных системах исходит из представления вычислительного процесса в виде к р у п н ы х и по возможности о д н о р о д н ы х б л о к о в, которые могут параллельно выполняться в различных устройствах.

Исследование большого числа задач (часть которых описана в [3]) показало, что для вычислительных систем принцип крупноблочного распараллеливания по сравнению с пооперационным или мелкоблочным [5] обладает рядом преимуществ.

I. В о з м о ж н о с т ь ю р а с п а р а л л е л и в а н и я н а б о л ь ш о е ч и с л о в е т в е й, превышающее десятки, сотни, а нередко и тысячи, при сохранении в ы с о к о й э ф ф е к т и в н о с т и ВС. В то время как при пооперационном распараллеливании эффективность заметно убывает уже при увеличении числа ветвей до [illegible]IC.

2. О д н о р о д н о с т ь ю в е т в е й (обычно все ветви при крупноблочном распараллеливании одинаковы или почти одинаковы), позволяющей использовать для них одну и ту же программу.

3. П р о с т о т о й н а с т р о й к и п р о г р а м м на заданное число машин в системе. Это осуществляется следующим образом [I] .

Находится эффективная схема алгоритма Sg с наибольшим

(или достаточно большим) числом ветвей ℓ_g. Для построения схемы S_ℓ с числом ветвей $\ell < \ell_g$ ветви схемы S_g объединяются в группы по $\frac{\ell_g + \lambda}{\ell}$ штук. Здесь $0 \le \lambda < \ell$ - число, дополняющее ℓ_g до кратного ℓ. Обычно $\ell << \ell_g$, что и обеспечивает достаточную однородность распределения ветвей схемы S_g между ветвями схемы S_ℓ.

Заметим попутно, что этим же способом сравнительно просто решается задача распределения частей вычислительного процесса между различными по производительности машинами, или устройствами [6].

4. Равномерностью загрузки параллельно работающих устройств, которая обеспечивается, как правило, самим алгоритмом либо (что требуется далеко не во всех задачах) специальным блоком программы, который в отдельные моменты времени перераспределяет группы операций между машинами (см., например, решение задачи Коши для линейных дифференциальных уравнений в частных производных гиперболического типа [3]).

В конструктивном отношении это преимущество означает, что нет необходимости иметь в системе особую машину-директор [7] или другое какое-либо устройство, управляющее ходом вычислений, так как любая машина (или группа машин) может время от времени выполнять эту функцию.

Описанная в работе [1] методика составления схем параллельных алгоритмов на основе граф-схем связей между операторами позволяет устанавливать предельно возможные границы распараллеливания данного алгоритма. С её помощью удалось исследовать некоторые свойства параллельных алгоритмов, в частности зависимость между числом ветвей и длиной параллельного алгоритма (характеристическая функция), и указать способы нахождения эффективных схем параллельных алгоритмов.

Для вычислительной системы "Минск-222" достаточно распараллелить процесс решения на сравнительно небольшое число ветвей. Это позволяет ограничиться наиболее простыми способами распараллеливания.

В данной работе рассматривается один из таких способов, основывающийся на циклической структуре алгоритма, который будем далее называть распараллеливанием по циклам. Этот способ оказывается применимым также и для построения схем программ параллельных алгоритмов (схем р- программ). Распараллеливание

по циклам так же, как и описанная ранее методика [1], исходит из разделения процесса решения на крупные блоки.

§ 1. Некоторые понятия и определения

Не претендуя на строгость, укажем содержательный смысл понятий, большинство из которых было введено в [1].

Пусть некоторый алгоритм представлен в виде последовательности операторов $A_1 A_2 \dots A_S$ из некоторого набора A. На операторы, входящие в набор, наложено ограничение только по сложности. Сложность оператора определяется сложностью схемы, реализующей его за время, не превышающее заданное. Каждый оператор не более чем двухместный.

Будем полагать, что в этот набор включены и предикаты и что любому оператору могут быть приписаны символы, определяющие порядок выполнения операторов в зависимости от значения предикатов (например, правая или левая полускобки, стрелки и т.п.).

Порядок выполнения операторов будем предполагать таким, что после данного оператора A_i может выполняться оператор $A_j, j>i$, т.е. в схеме записи отсутствуют петли.

Введем понятие зависимости между операторами. Будем говорить, что оператор A_j **непосредственно информационно зависит** от оператора A_i, если оператор A_j выполняется над результатом оператора A_i; оператор A_i **непосредственно управляет** операторами A_j и A_k, если результат оператора A_i определяет, какой из этих двух операторов должен выполняться непосредственно после оператора A_i.

Совокупность операторов, не зависящих друг от друга ни по информации, ни по управлению, называется **р-оператором**.

Число операторов, входящих в данный р-оператор, называется его **высотой**.

Последовательность р-операторов называется **р-кортежем**, если каждый оператор, входящий в любой из р-операторов, может зависеть только от исходных данных и от результатов операторов, входящих в предшествующие р-операторы.

Р-кортеж характеризуется **длиной** h -числом образующих его р-операций, шириной L - наибольшей высотой его

р-операций и числом пустых операций, которыми нужно дополнить его р-операции до высоты L. Под пустой операцией понимается операция, результат которой безразличен при выполнении данного алгоритма.

Алгоритм, представленный в виде р-кортежа, будем называть **параллельным алгоритмом**, или **р-алгоритмом**. Один и тот же р-алгоритм может быть представлен различными р-кортежами, множество которых будем называть **семейством** р-кортежей данного р-алгоритма.

Р-алгоритм можно трактовать как понятие более широкое, чем последовательный алгоритм. Последовательному алгоритму соответствует р-кортеж с шириной $L = 1$. Такой р-кортеж, называемый **вырожденным**, имеет наибольшую длину и в то же время наименьшее общее число операций $\mathcal{L}$ в данном семействе. $\mathcal{L}$ называется **объемом** р-алгоритма.

Коэффициентом эффективности δ р-кортежа будем называть отношение объема р-алгоритма $\mathcal{L}$ к общему числу операций данного р-кортежа.

Наибольший интерес представляют прямоугольные р-кортежи, все р-операции у которых одной высоты. Такие р-кортежи могут быть записаны в виде матрицы

$$\begin{pmatrix} \Omega_{11} & \Omega_{12} & \dots & \Omega_{1h} \\ \Omega_{21} & \Omega_{22} & \dots & \Omega_{2h} \\ \dots & \dots & \dots & \dots \\ \Omega_{L1} & \Omega_{L2} & \dots & \Omega_{Lh} \end{pmatrix} \qquad (1)$$

Некоторые операторы Ω_{ij} могут быть пустыми. Строки матрицы (I) представляют собой последовательности операторов, которые могут зависеть только от предшествующих операторов (а также от исходных данных и операций из других строк). Такие строки мы будем называть **ветвями** р-кортежа.

Из независимости операций в р-операции следует, что существует большое число вариантов распределений операций между ветвями. Каждый вариант распределения операций между ветвями назовем **покрытием** данного р-кортежа.

Каждой ветви припишем часть исходной информации. При этом одни и те же коды могут быть отнесены к нескольким ветвям. Отношение суммы кодов, приписанных к ветвям, ко всей исходной информации назовем **коэффициентом избыточности** $\varkappa$ распределения начальной информации.

Рассмотрим две ветви k_i и k_j некоторого покрытия. Будем

говорить, что k_i не зависит от k_j, если ни одна операция, входящая в k_i, непосредственно не зависит ни от одной операции или кода исходной информации ветви k_j.

Мерой зависимости k_i от k_j служит сумма c_{ij} числа операций и числа кодов исходной информации ветви k_j, от которых непосредственно зависят операции ветви k_i. Для всего покрытия имеем матрицу (c_{ij}) L-го порядка.

Величина

$$C = \sum_{i=1}^{L} \sum_{j=1}^{L} c_{ij} \qquad (2)$$

называется связностью покрытия, а матрица (c_{ij}) – матрицей связности.

Будем говорить, что схема р-алгоритма построена, если определены 1) р-кортеж; 2) покрытие р-кортежа; 3) распределение исходной информации между ветвями покрытия; 4) матрица связности.

Качество схемы р-алгоритма при заданном числе ветвей L определяется коэффициентом эффективности δ, связностью C и коэффициентом избыточности $\varkappa$. Эти три характеристики зависят друг от друга. Можно в известных пределах уменьшить величину связности C, увеличив коэффициент избыточности $\varkappa$ или введя дополнительные операции, повторяющие результат, полученный в других ветвях (т.е. уменьшив коэффициент эффективности δ).

Введем два следующих р-оператора, в явном виде учитывающих зависимости между ветвями р-кортежа: р-оператор обмена и р-оператор обобщенного условного перехода (ОУП) (см.[4]).

С помощью р-оператора обмена можно передавать коды из данной ветви во все остальные. Р-операторы ОУП позволяют выполнять операторы, которые по управлению непосредственно зависят от операторов, находящихся в других ветвях.

В соответствии с матрицей связности включим эти два типа р-операторов в схему р-алгоритма. Связность покрытия может характеризоваться числом р-операторов обмена и ОУП. Увеличение общего числа операций р-кортежа из-за включения р-операторов обмена и ОУП уменьшит коэффициент эффективности δ. Тем самым коэффициент δ будет учитывать и величину связности.

Коэффициент избыточности $\varkappa$ при реализации р-алгоритма характеризует объем памяти, необходимый для хранения исходной

информации. Обычно важно не столько сведение этого объема к минимуму, сколько то, чтобы он не превышал некоторой величины, определяемой как задачей, так и техническими параметрами конкретного вычислительного устройства.

Таким образом, выбор оптимальной схемы р-алгоритма сводится к нахождению варианта с достаточно большим коэффициентом эффективности δ при заданном ограничении величины коэффициента избыточности $\varkappa$.

Условимся называть схему р-алгоритма э ф ф е к т и в н о й, если

$$\delta \geq 1 - \varepsilon ,$$

где $\varepsilon \geq 0$ может изменяться от задачи к задаче. Для большинства рассмотренных задач ε = 0,01 ÷ 0,10.

Рассмотрим р-кортеж K с шириной L . Его прямоугольной l - р а з в е р т к о й назовем р-кортеж с шириной l , образованный путем замены каждой р-операции р-кортежа K последовательностью р-операций высотой l каждая.

Если р-кортеж K прямоугольный и его ширина L кратна l , то коэффициент эффективности р-кортежа, являющегося его l -разверткой, не меньше чем у р-кортежа K .

§ 2. Схема р-алгоритма

Исследование задач, частично изложенных в [4] , показывает, что построение эффективных схем р-алгоритмов может оказаться более простым, если имеется подходящая схема эквивалентного последовательного алгоритма.

I°. Для построения схемы р-алгоритма исходный последовательный алгоритм удобно представить в форме, явно обнаруживающей циклы. Для этой цели можно, например, рекомендовать операторную форму, в которой все линии, соответствующие циклам, расположены под строчкой, а все остальные - над нею. Операторам приписываются индексы циклов, от которых они зависят. Последовательности операторов с одинаковыми индексами заменяются, где это удобно, обобщенными операторами. Нумерация операторов ведется внутри группы однородных операторов, что упрощает преобразование схем. Такую запись будем называть нормальной операторной формой представления алгоритма.

П р и м е р I. Решение системы линейных уравнений методом последовательных приближений [8].

Решение этой задачи сводится к вычислению новых значений переменных

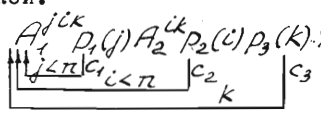
$$x_i^{(k)} = g_i + \sum_{j=1}^{n} b_{ij} x_j^{(k-1)}, \quad i = 1, \ldots, n, \tag{3}$$

пока не будет достигнута нужная точность

$$\left| x_i^{(k)} - x_i^{(k-1)} \right| < \varepsilon \tag{4}$$

для всех x_i .

Операторная схема алгоритма может быть записана следующей строчкой.



(5)

Здесь оператор A_1^{ijk} суммирует парные произведения в выражении (3).

Оператор A_2^{ik} вычисляет $\left| x_i^{(k)} - x_i^{(k-1)} \right|$.

Операторы $p_1(j)$ и $p_2(i)$ устанавливают конец повторения циклов по j и по i .

Оператор $p_3(k)$ повторяет цикл по k при выполнении условия (4). $Я$ - оператор конца.

Пример 2. Обращение матриц методом пополнения [8]. Эта задача сводится к вычислению выражения:

$$\alpha_{ij}^{(k)} = \alpha_{ij}^{(k-1)} - \frac{e_j^{(k)}}{1 + e_k^{(k)}} \alpha_{ik}^{(k-1)}, \tag{6}$$

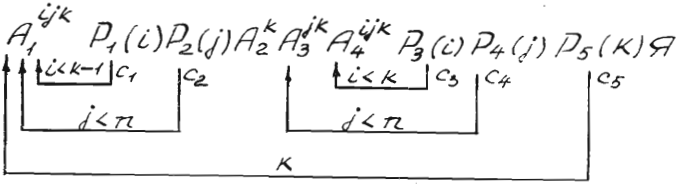
где

$$e_j^{(k)} = z_{kj} + \sum_{i=1}^{k-1} \omega_{ki} \alpha_{ij}^{(k-1)}, \tag{7}$$

$$z_{kj} = \begin{cases} 0 & \text{при } j < k, \\ \omega_{kj} & \text{при } j \geq k, \end{cases} \tag{8}$$

$k = 1, \ldots, n;\ i = 1, \ldots, k;\ j = 1, \ldots, n,$

ω_{ij} - элементы исходной матрицы, из которой вычтена единичная матрица; $\alpha_{ij}^{(0)}$ - элементы единичной матрицы.



(9)

Операторы A_1^{ijk}, A_2^k, A_3^{jk}, A_4^{ijk} вычисляют соответственно

$$e_j^{(k)},\ \frac{1}{1+e_k^{(k)}},\ \frac{e_j^{(k)}}{1+e_k^{(k)}},\ \alpha_{ij}^{(k)}.$$

2^0. Рассмотрим некоторые свойства циклов. Будем различать циклы зависимые, независимые и частично зависимые.

Цикл называется з а в и с и м ы м, если при каждом повторении (начиная со второго), используется результат, полученный при предыдущем повторении цикла, и - н е з а в и с и м ы м, если результат каждого повторения цикла не зависит от результатов, получаемых при остальных повторениях цикла. Цикл называется ч а с т и ч н о з а в и с и м ы м, если результаты при некоторых (но не при всех) повторениях цикла не зависят друг от друга.

Примеры I и 2 содержат все три типа циклов. В примере I цикл c_3 - зависимый, цикл c_2 - независимый. Цикл c_1 -частично зависимый, в нем можно независимо друг от друга выполнить $n/2$ повторений цикла (суммирование пар произведений двух чисел). Аналогично в примере 2 цикл c_5 - зависимый, циклы c_2, c_3, c_4 - независимые, цикл c_1 - частично зависимый.

Циклы будем называть п е р е с е к а ю щ и м и с я, если имеется хотя бы один оператор, входящий в оба цикла. В примере I все циклы пересекающиеся. В примере 2 циклы c_1 и c_2 пересекающиеся, c_1 и c_3 - непересекающиеся.

Если все операторы, входящие в цикл c_i, входят и в цикл c_j, то цикл c_i будем называть в н у т р е н н и м, а цикл c_j - в н е ш н и м.

Будем говорить, что множество взаимно непересекающихся циклов $c_1,\ldots,c_m$ п о к р ы в а е т с х е м у данного алгоритма, если каждый оператор, входит в один и только один из циклов $c_1,\ldots,c_m$.

Рассмотрим участок схемы последовательного алгоритма, входящий в цикл c, повторяющийся n раз. Будем называть $\mathcal{C}$ - о т о б р а ж е н и е м этого участка участок схемы р-алгоритма, образованный следующим образом.

1) Каждый оператор Ω исходного участка заменяется р-оператором $\tilde{\Omega}$, состоящим из ℓ одинаковых компонент Ω

2) Число повторений r цикла C заменяется на

$$z = \frac{r+\lambda}{\ell}, \qquad (10)$$

где $0 \le \lambda < \ell$ – число повторений цикла C с пустыми операторами. При этом в λ ветвях будет по одному пустому повторению цикла.

3) Если цикл частично зависимый, то в схему р-алгоритма могут вводиться р-операторы обмена, а если между ветвями имеется зависимость по управлению, то и – р-операторы ОУП.

Число операторов обмена и ОУП зависит от характера зависимости и распределения повторений цикла по ветвям.

4) Если участок входит только в зависимые циклы или не входит ни в один из циклов, то считается, что он входит в независимый цикл с $r=1$ (вырожденный независимый цикл). Тогда $\lambda=\ell-1$ и во всех ветвях, кроме одной, будут выполняться пустые операторы.

Пусть имеется схема S последовательного алгоритма, которую покрывают циклы $C_1, \ldots, C_m$. Построим ℓ-отображение для каждого из участков, входящих в циклы $C_1, \ldots, C_m$. Каждой из ℓ ветвей припишем часть исходной информации и введем, где это необходимо, р-операторы обмена и ОУП с тем, чтобы каждый участок был обеспечен необходимой для его выполнения информацией (в том числе и управляющей). Полученную таким способом схему S^{ℓ} р-алгоритма будем называть ℓ – о т о б р а ж е н и е м схемы S.

Нетрудно видеть, что любой цикл, не принадлежащий множеству $C_1, \ldots, C_m$, будет повторяться в схеме S^{ℓ} столько же раз, сколько и в схеме S.

3^0. Припишем каждому оператору в е с, определяемый числом и длительностью некоторых стандартных операторов, из которых он состоит, и средним числом его повторений в процессе реализации алгоритма. Для вычислительной системы "Минск-222" [9] вес оператора можно приравнять, например, среднему времени, необходимому для его реализации, в микросекундах, умноженному на среднее число его повторений в ходе вычислений.

Удобно также пользоваться понятием о т н о с и т е л ь н ы й в е с оператора – отношение веса оператора к сумме весов всех операторов данного алгоритма.

Под в е с о м ц и к л а будем понимать сумму относи-

тельных весов операторов, входящих в данный цикл. Циклы, вес которых превышает заданную величину γ, назовем о с н о в н ы м и. Далее, для определенности будем считать, что $\gamma = 0,1$. В конкретных примерах основные циклы обычно выявляются прямо по схеме алгоритма без каких-либо сложных расчетов. В примере I все три цикла-основные при $n > 10$. В примере 2 основными будут циклы c_3, c_4. Циклы c_1 и c_2 могут быть основными только при $n < 10$.

4°. При ℓ - отображении схемы последовательного алгоритма общий суммарный вес операторов возрастает. Это происходит вследствие введения дополнительного числа λ_s повторений цикла из пустых операций (из-за некратности n_s - числа повторения цикла c_s, $\ell(s=1,\dots,m)$), а также добавления р-операторов обмена и обобщенного условного перехода.

Условимся называть ℓ - отображение эффективным, если увеличение суммарного веса операторов не превышает заданной величины ε, $1 > \varepsilon \geq 0$ (для определенности будем считать $\varepsilon = 0,01$).

5°. Нетрудно видеть справедливость следующего утверждения.

Если для каждого оператора с большим относительным весом в данной совокупности циклов $c_1,\dots,c_m$ имеется хотя бы один охватывающий его независимый цикл c_s с числом повторений $n_s \gg \ell$, то данной совокупности циклов будет соответствовать эффективное ℓ -отображение.

Действительно, в этом случае число пустых повторений цикла $\lambda_s < \ell \ll n_s$. Дополнительные р-операторы обмена и ОУП не входят в цикл c_s, так как в независимом цикле нет надобности в обмене информацией (в том числе и управляющей). Поэтому относительный вес этих р-операторов будет мал.

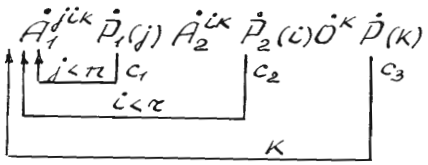
ℓ - отображение может также быть эффективным, если число вводимых р-операторов обмена и ОУП мало.

Рассмотрим примеры.

В примере I схема алгоритма (5) имеет три цикла: частично зависимый цикл c_1, независимый c_2 и зависимый c_3. Относительный вес операторов, входящих в цикл c_2, более чем в n^2 раз превышает вес остальных операторов ($P_3(\kappa)$ и Ψ), поэтому естественно выбрать для ℓ -отображения цикл c_2 и вырожденный цикл, включающий в себя операторы $P_3(\kappa)$ и Ψ.

Каждой ветви придадим τ строк исходной матрицы (b_{ij}) и τ соответствующих свободных членов g_i, где $\tau = (n+\lambda)/\ell$.

$\lambda (0 \le \lambda < \ell)$ – пустых строк матрицы (b_{ij}) и свободных членов распределены между λ -последними по номеру ветвями по одной на каждую ветвь. В результате получим следующую схему р-алгоритма:



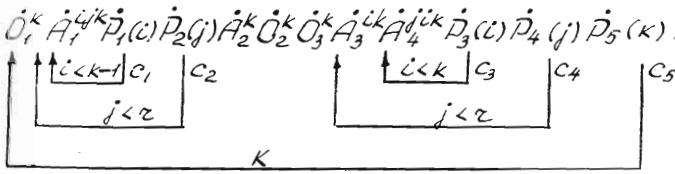
(II)

В этой схеме р-операторы $\dot{A}_1, \dot{P}_1(j), \dot{A}_2^{ik}, \dot{P}_2(i)$ и $\dot{Я}$ состоят из ℓ одинаковых составляющих, совпадающих с одноименными операторами в схеме (5). Р-оператор $\dot{O}^k$ выполняет обмен между ветвями, полученными значениями x_i^k ; р-оператор обобщенного условного перехода $\dot{P}(k)$ осуществляет переход к новой итерации, если хотя бы в одной из ветвей не выполнено условие (4).

Нетрудно видеть, что при $n \gg \ell$ данная схема будет эффективной.

В примере 2 два независимых непересекающихся цикла c_2 и c_4 в совокупности покрывают все операторы с большим весом. Веса остальных операторов более, чем в $n^2/2$ раз меньше весов операторов A_1^{ijk} и A_4^{ijk} .

Построим ℓ -отображение по циклам c_2 и c_4 .При этом каждой ветви припишем τ столбцов матрицы (α_{ij}) и τ строк матрицы (ω_{ij}). В результате получим следующую схему р-алгоритма



(I2)

где р-операторы обмена $\dot{O}_1^k, \dot{O}_2^k, \dot{O}_3^k$ передают во все ветви строку $\omega_{k1}, \ldots, \omega_{kn}, \frac{1}{1+e_k^{(k)}}$, столбец $\alpha_{1k}^{(k-1)}, \ldots, \alpha_{kk}^{(k-1)}$.

Все остальные р-операторы состоят из ℓ одинаковых операторов, совпадающих с одноименными операторами из схемы (9). При этом у р-оператора $\dot{A}_2^{ik}$ будет $\ell-1$ пустых составляющих.

Нетрудно видеть, что данная схема будет эффективной при $n \gg \ell$.

6°. Рассмотрим теперь случай, когда один или несколько

из выбранных циклов – частично зависимые. При распараллеливании по таким циклам приходится учитывать характер зависимости.

Ограничимся примером частично зависимого цикла – суммированием парных произведений. Такие циклы встречались в примерах I (c_1) и 2 (c_1).

Пусть число повторений этого цикла n и число ветвей ℓ р-алгоритма связаны зависимостью (I0). Распределим между ветвями по τ повторений цикла. Каждой ветви припишем соответствующие исходные данные, то есть проделаем то же самое, что и с независимым циклом. В отличие от последнего после повторения τ циклов потребуется сложить ℓ частичных сумм, полученных в каждой из ветвей. Для этого передадим все частичные суммы в одну из ветвей (или во все) и там их сложим (за $\ell-1$ операцию сложения). Если получающееся при этом число пустых операций велико, то его можно несколько уменьшить, выполняя сложение попарно в $\ell/2$ ветвях, потом в $\ell/4$ ветвях и т.д. Всего будет $\log_2 \ell$ шагов. В этом случае уменьшаются также затраты и на обмен.

7°. В заключение можно рекомендовать следующий порядок составления схемы р-алгоритмов.

I. Приводим схему исходного последовательного алгоритма к виду, явно обнаруживающему имеющиеся циклы (например, к нормальной операторной форме).

2. Оцениваем относительные веса операторов.

3. Ищем совокупности независимых или частично зависимых непересекающихся циклов, включающие в себя операторы с большими относительными весами.

4. Сопоставляем возможные варианты выбора циклов. Для этого:

а) оцениваем относительный вес циклов с пустыми операциями для каждой совокупности;

б) распределяем исходную информацию между ветвями и оцениваем относительный вес операций обмена и обобщенного условного перехода.

5. Строим ℓ -отображение для выбранного варианта покрытия схемы циклами.

Если полученная схема р-алгоритма окажется неэффективной, то рассматриваем другие варианты схемы исходного алгоритма. Эти варианты обычно получаются путем изменения порядка выполнения операторов.

8^0. Рассмотрим круг задач, для которых может быть пригодна методика распараллеливания по циклам.

I. Сопоставление увеличения производительности ЭВМ с удлинением программ показывает, что за последние I0 лет производительность возросла на несколько порядков, а средняя длина программ увеличилась всего в несколько раз. Это говорит о том, что имеется тенденция увеличения числа повторений циклов.

2. Наиболее длинные из известных программ содержат 50 000 - I00 000 команд (трансляторы, мониторы). Длина программ с большим временем счета редко превышает I0 000 команд. У задач, для которых предназначается ВС "Минск-222",число выполняемых операций должно быть порядка $I0^8$ и более, т.е. каждая команда повторяется в процессе счета в среднем не менее $I0^4$ раз.

3. Если предположить, что все циклы в программах зависимые, то должны существовать длинные цепочки вычислений, каждое из которых выполняется над результатом предыдущей операции. Но с увеличением длины таких цепочек падает точность вычислений. При числе выполняемых операций порядка $I0^8$ и более длина таких цепочек будет на много больше предельно допустимой, определяемой точностью вычислений.

Отсюда можно сделать вывод, что в задачах с большим счетом должны быть независимые или, по крайней мере, частично зависимые циклы. В противном случае задача на ЭВМ не реализуется.

Указанные выше соображения позволяют утверждать,что круг задач, для которых применима методика распараллеливания по циклам, довольно широк. Это подтверждается также рассмотрением конкретных классов задач [3].

§ 3. Схема программы для вычислительной системы

Схемы р-алгоритма, рассмотренные в предыдущем параграфе, носят общий характер и в основном отражают особенности метода решения конкретной задачи. Для построения схем программ, учитывающих особенности реализации данной схемы р-алгоритма на конкретной вычислительной системе, в схему р-алгоритма нужно ввести две группы р-операторов, отражающих свойства как элементарной машины системы (систему команд, размеры оперативной памяти, обмен с внешними устройствами,преобразования мас-

сивов кодов и т.п.), так и системы в целом (принцип взаимодействия машин, особенности употребления команд системы и т.п.).

Учет свойств машины неспецифичен для вычислительной системы и мало чем отличается от того, который делается при разработке схемы программы для одной ЭВМ. Это позволяет при составлении схемы программы р-алгоритма исходить не из схемы р-алгоритма, а из схемы программы для ЭВМ, которая нередко бывает известна. Рассмотрим это подробнее.

Схема программы представляет собой более подробную и конкретизированную схему алгоритма, поэтому мы можем преобразовывать её формально так же, как это было описано в предыдущем параграфе, а именно: записать в нормальном операторном виде; оценить относительные веса операторов; выбрать совокупность циклов,по которым будет вестись распараллеливание; построить ℓ - отображение; распределить исходную информацию между ветвями р-алгоритма; ввести в схему р-операторы системы (обмена, условного и безусловного обобщенных переходов и настройки).

Полученная таким путем схема еще не будет окончательной, так как в исходной схеме программы (в отличие от схемы алгоритма) могут быть не вошедшие в покрытие циклы, число повторений которых зависит от ℓ . Эти циклы связаны с обменом информацией между оперативной и вспомогательными памятями,вводом и выводом информации. При равномерном распределении информации между машинами системы и коэффициенте избыточности $\varkappa = 1$ остается только уменьшить число повторений соответствующих циклов в ℓ раз, либо в ℓ раз сократить размеры вводимых массивов. Это будет соответствовать случаю, когда все машины принимают участие в вводе и выводе информации. При коэффициенте избыточности $\varkappa > 1$ число повторений циклов или объем вводимых массивов уменьшается в $\ell/\varkappa_k$ раз, где $\varkappa_k$ - коэффициент избыточности k -го массива информации. Дублируется обычно только информация, хранящаяся в оперативной памяти,так как вместо того, чтобы хранить одну и ту же информацию во вспомогательных памятях различных машин, достаточно содержать её в одной из машин и передавать в остальные с помощью команд обмена, скорость выполнения которых значительно выше скорости обмена со вспомогательными памятями. Учет дублирования информации сводится в этом случае к введению р-операторов обмена.

Таким образом, схема программы для системы может быть получена из схемы программы для одной машины путем сравнительно простых преобразований.

Возникает вопрос: в каких случаях полученная схема будет эффективной?

При рассмотрении схем р-алгоритма были указаны две причины увеличения числа дополнительных операций при распараллеливании:

I) неравномерность распределения числа повторений циклов между ветвями, в результате чего появляются циклы с пустыми операциями и

2) введение р-операторов обмена и обобщенного условного перехода. Для схем программ могут быть и другие причины изменения коэффициента эффективности.

У существующих ЭВМ время выполнения основных арифметических операций, как правило, зависит от вида операнд. Объединенные в систему эти ЭВМ будут заканчивать работу в различное время, даже если они выполняют одну и ту же программу. Из-за этого перед выполнением некоторых р-операторов системы может потребоваться введение синхронизирующих команд.

Способ оценки простоев машин, возникающих по указанной причине, дается в работе [IO]. Там же показано, что путем увеличения числа операций, выполняемых между двумя синхронизациями, можно свести простои машин к пренебрежимо малой величине. Для системы "Минск-222" это удалось сделать для всех рассмотренных задач надлежащей расстановкой р-операторов обмена и ОУП.

Другой величиной простоев может быть неравномерность затрат времени у машин системы на обращения к вспомогательным памятям и другим внешним устройствам. Уменьшение коэффициента эффективности и по этой причине во всех рассмотренных задачах удалось свести практически к нулю путем соответствующего распределения информации между машинами.

Кроме причин, уменьшающих коэффициент эффективности, имеются и факторы, его увеличивающие:

I. Вследствие того, что суммарный объем оперативной памяти системы позволяет хранить значительно большую информацию, чем в одной машине, и тем самым уменьшить или даже свести к нулю время обращения к вспомогательным памятям, коэффициент эффективности системы возрастает, например для задач линейной алгебры и линейного программирования в I,5-2 раза.

2. Коэффициент эффективности системы может увеличиваться и благодаря тому, что в некоторых задачах (например, при упорядочении больших массивов информации) время решения опреде-

ляется объемом оперативной памяти и количеством вспомогательных памятей.

3. Увеличение объема оперативных и вспомогательных памятей позволяет для некоторых задач запоминать промежуточные данные, неоднократно используемые в процессе счета, лиоо насчитывать и хранить специальные таблицы и тем самым сокращать время счета. Так,при моделировании взаимодействия электронов с веществом методом Монте-Карло, решении системы обыкновенных уравнений методом Рунге-Кутта с правыми частями определенного вида и некоторых других задачах запоминание таблиц сложных функций от одного и двух переменных позволяет сократить время счета на системе "Минск-222" в десятки и сотни раз.Так как эти таблицы не размещаются в оперативной памяти одной машины, а ввод их с магнитной ленты занимает большое время, то коэффициент эффективности системы при решении этих задач значительно больше I.

В заключение отметим, что описанный метод распараллеливания по циклам позволяет сравнительно просто составлять схемы р-алгоритмов и р-программ вручную, а также может служить основой для автоматизации этого процесса.

Литература

I. Ю.Г. Косарев. О методике решения задач на универсальных вычислительных системах.- Вычислительные системы,Новосибирск, Изд-во "Наука", Сибирское отделение, I965, вып. I7, стр. 6I-99.

2. Э.В. Евреинов, Ю.Г. Косарев. Матричный р-язык для описания параллельных алгоритмов. Там же,стр. I00-I05.

3. Э.В. Евреинов, Ю.Г. Косарев. О решении задач на универсальных вычислительных системах. Там же, стр.I06-I64.

4. Э.В. Евреинов, Ю.Г. Косарев. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, Изд-во "Наука", Сиб. отд., I966 г.

5. Ю.С. Шварц. Автоматический процесс упорядочивания модулей и его приложение к параллельному программирова-

нию. - Кибернетический сборник, Изд-во "Мир", 1964г., № 9, стр. 240-269.
6. Д.А. Поспелов. О структуре вычислительной машины с несколькими арифметическими устройствами. - Труды МЭИ, 1962, вып. 41, стр. 57-60.
. Д.А. Поспелов. Об одном способе построения вычислительной системы. - Доклад на семинаре по теории автоматов, Киев, 1964, стр. 3-24.
8. А.К. Фадеев и В.Н. Фадеева. Вычислительные методы линейной алгебры. М., Физматгиз, 1960.
9. Э.В. Евреинов, Г.П. Лопато. Универсальная вычислительная система "Минск-222". - Вычислительные системы, Новосибирск, Изд-во "Наука", Сиб. отд., 1966г., вып. 23, стр. 13-20.
10. Ю.Г. Косарев, С.В. Нагаев. О потерях времени на синхронизацию в вычислительной системе "Минск-222". - Данный сборник, стр. 21-39

Поступила в редакцию
7.X. 1966 г.