ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1973 год Выпуск 57

УДК 681.142.2

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ НА ОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Н.Н.Миренков

В работах по программированию для однородных вычислительных систем (ОВС) [1-2] можно выделить два основных взаимно дополняющих подхода:

- непосредственное написание параллельных программ на основе параллельных (р-) алгоритмов,
- автоматическое распараллеливание программ, реализующих последовательные алгоритмы.

В настоящее время в нашей стране реализован первый подход [3-6]. Трансляторы с автоматическим распараллеливанием еще не вышли из стадии разработки. Появление таких трансляторов, однако, не исключит первого подхода, который будет применяться, как правило, для сложных задач, критичных к качеству распараллеливания.

Опыт математической эксплуатации системы "Минск-222" [7-8] показал, что написание р-программы, когда известен р-алгоритм решения задачи, аналогично написанию программы, основанной на последовательном алгоритме.

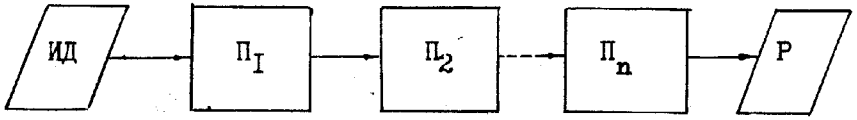
В работе содержатся описание методики крупноблочного распараллеливания [2, 8-12] и краткий обзор алгоритмов, которые могут непосредственно использоваться при программировании для ОВС.

Значительные усилия при создании р-алгоритмов затрачиваются, как правило, на нахождение оптимальных р-алгоритмов, которые эффективно используют объединенную мощность процессоров и суммарный объем памяти системы. При этом нередко удается получить ускорение счета по сравнению с одной машиной в $\alpha \cdot \ell$ раз, где $\alpha > 1$, ℓ – число машин. Дополнительный выигрыш (в α раз) достигается

на основе учета особенностей алгоритма и связан с решением трудноформализуемых задач: изменением [13], и развертыванием циклов, переупаковкой массивов, использованием таблиц [14] и др. [15]. Рассмотрение методов оптимизации выходит за рамки данной работы.

I. Методика крупноблочного распараллеливания

I.I. П о н я т и е р - а л г о р и т м а . При решении задач на электронной вычислительной машине используются обычно последовательные алгоритмы, условная схема которых имеет вид:



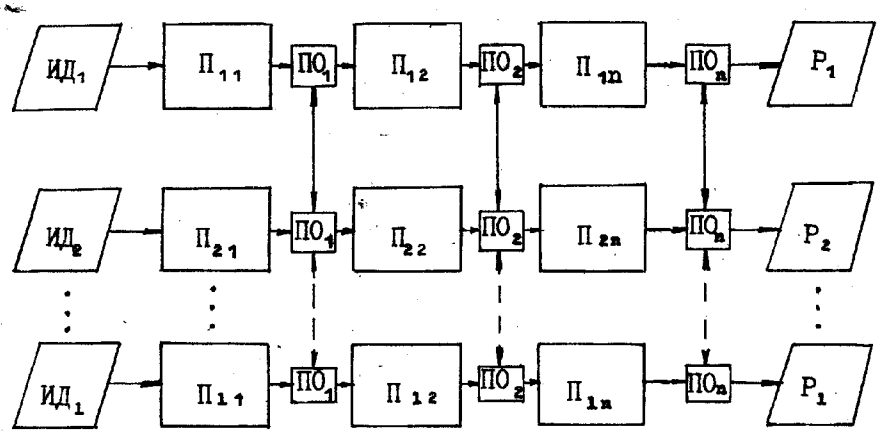
ИД – исходные данные, $П_i$ $(i=1,2,\dots,n)$ – процедуры, Р – результаты.

Последовательность, задающая порядок выполнения процедур, а так же операций внутри них, часто произвольна. Это объясняется тем, что при необходимости выполнить большие вычисления появляется огромное количество операций, с которых можно было бы начать работу машины. При написании такого алгоритма принимается произвольное решение о последовательности, в которой нужно выполнить операции.

Указанная возможность позволяет при решении задач на ОВС использовать р-алгоритмы, условная схема которых представлена на рис. I.

Анализ задач показал, что на практике, как правило, используются более простые схемы р-алгоритмов, представляющие собой совокупность идентичных параллельных ветвей, в которых для любого $j\in\{1,2,\dots,n\}$ $П_{ij}=П_j$, $i=1,2,\dots,l$.

Разработка параллельной программы для такого р-алгоритма сводится собственно к разработке последовательной программы для одной ветви. Возникает вопрос, как разработать сам параллельный алгоритм?



$ПО_j$ – процедуры обменных взаимодействий*), $j\in\{1,2,\dots,n\}$.

Рис. I

Из схемы р-алгоритма видно, что должны быть

I) поделены между ветвями исходные данные,

2) разработан алгоритм ветви, содержащий процедуры обменных взаимодействий.

I.2. Р а с п р е д е л е н и е д а н н ы х м е ж д у в е т в я м и . От того, как разделены массивы данных между ветвями, непосредственно зависит частота и объемы взаимодействий ветвей. Для уменьшения взаимодействий предлагается опираться на принцип о д н о р о д н о г о распределения информации, при котором :

– равны объемы распределяемых частей массивов;

– нумерация распределяемых частей массивов соответствует нумерации ветвей;

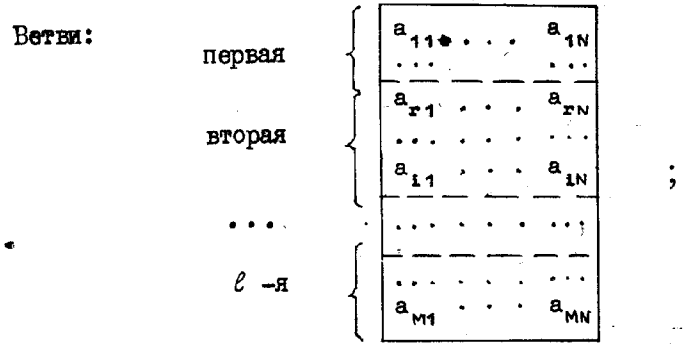
– массивы разделяются на части параллельными прямыми (многомерные массивы – параллельными плоскостями);

– массивы или их части дублируются одинаковым образом.

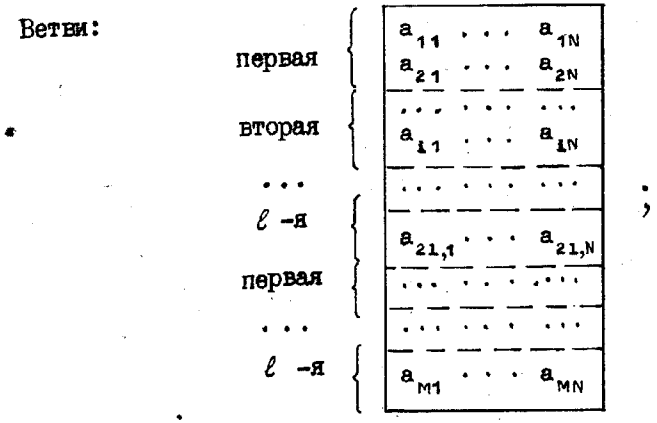
*) Поскольку отдельные процедуры, множества исходных данных и результатов могут быть пустыми, то приведенная схема задает фактически любой р-алгоритм.

Примеры. Для наглядности проанализируем двумерные массивы $A[1:M,1:N]$. К ним применимы следующие основные способы однородного распределения по ветвям:

I) горизонтальные полосы (ГП – распределение)



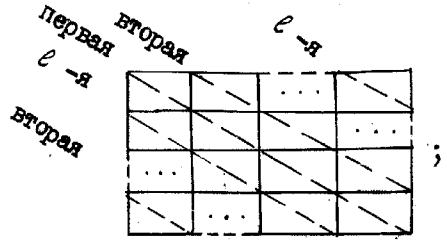
2) циклические горизонтальные полосы (ЦГП – распределение)



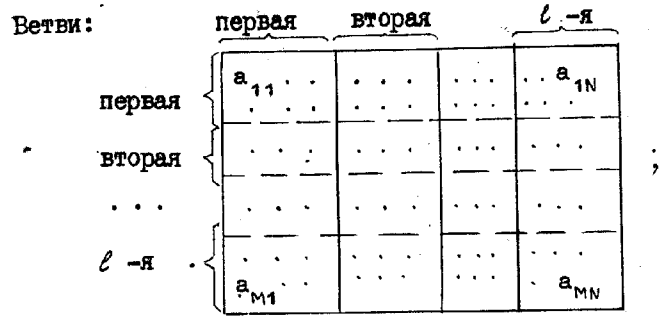
3) вертикальные полосы (ВП – распределение);

4) циклические вертикальные полосы (ЦВП – распределение);

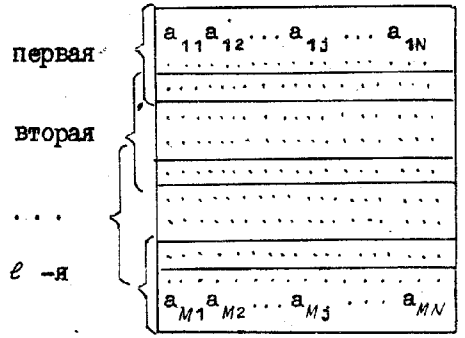
5) скошенные полосы (С – распределение)



6) горизонтальные и вертикальный полосы с дублированием (ГВД-распределение)



7) горизонтальные полосы с частичным дублированием (ГПД-распределение)



I.3. Алгоритм параллельной ветви. При разработке р-алгоритма главное внимание уделяется выявлению циклов и распределению связанных с ним массивов. При этом используются две основные схемы распараллеливания.

Однородная схема. В последовательном алгоритме выявляются один или несколько независимых циклов, покрывающих операторы с основным временем счета. Эти циклы связаны с основными массивами задачи и обычно известны пользователю. По этим циклам и проводится распараллеливание, то есть выбирается способ распределения массивов и вставляются операторы обменных взаимодействий.

Для примера рассмотрим решение системы линейных уравнений $X = BX + G$ методом последовательных приближений. Вычисления сводятся к следующим формулам:

$$x_i^{(k)} = g_i + \sum_{j=1}^{n} b_{ij} x_j^{(k-1)}, \quad i = 1, 2, \ldots, n, \qquad (1)$$

пока не будет достигнута нужная точность

$$\left| x_i^{(k)} - x_i^{(k-1)} \right| < \varepsilon, \qquad i = 1, 2, \ldots, n, \qquad (2)$$

Операторная схема последовательного алгоритма может быть записана следующим образом:

$$A_1^{jik}\underbrace{{}_{j<n}\ P_1(j)}_{c_1}\ A_2^{ik}\underbrace{{}_{i<n}\ P_2(i)}_{c_2}\ \underbrace{P_3(k)}_{c_3}\ \mathcal{Я}, \qquad (3)$$

где оператор A_1^{jik} суммирует парные произведения в формуле (I); A_2^{ik} вычисляет $|x_i^{(k)} - x_i^{(k-1)}|$; $P_1(j)$ и $P_2(i)$ устанавливают конец повторения циклов по j и i; $P_3(k)$ повторяет цикл по k при невыполнении условия (2); $\mathcal{Я}$ – оператор конца.

Из трех циклов схемы (3) наиболее интересен для наших целей c_2. Он независимый (вычисления для $i = i_1$ не зависят от вычислений для $i = i_2$; $i_1, i_2 = 1, 2, \ldots, n$; $i_1 \neq i_2$) и покрывает операторы с основным временем счета. Применим к массивам B, X и G ГП-распределение, тогда для реализации цикла c_2 каждой параллельной ветви нужно выполнить по r повторений, а затем обменяться с другими ветвями частями вектора X нового приближения ($r = \left[\frac{n}{\ell}\right] + 1$ для первых m ветвей, где $0 \leq m \leq \ell - 1$ – остаток от деления n на ℓ; для всех остальных $r = \left[\frac{n}{\ell}\right]$).

Операторная схема параллельной ветви будет следующей:

$$A_1^{jik}\underbrace{{}_{j<n}\ P_1(j)}_{c_1}\ A_2^{ik}\underbrace{{}_{i<r}\ P_2(i)}_{c_2}\ \dot{O}_k\ \underbrace{\dot{P}_{(k)}}_{c_3}\ \mathcal{Я},$$

где оператор $\dot{O}_k$ выполняет обмены частями вектора X между ветвями, $\dot{P}_{(k)}$ – осуществляет переход к новой итерации, если хотя бы в одной из ветвей не выполнено условие (2). Остальные операторы те же, что и в схеме (3).

Однородная схема позволяет получать р-программы в виде совокупности идентичных ветвей. Поскольку одновременное выполнение этих ветвей не требует обязательного хранения соответствующей программы целиком в каждой машине, то достаточно иметь программу одной ветви, распределенной по системе, и предоставлять одновременно всем машинам только ту часть (сегмент), которая реализуется в данный момент.

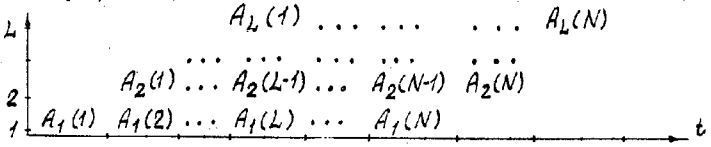
К о н в е й е р н а я с х е м а. Последовательный алгоритм может быть распараллелен по схеме "конвейер", выполненной над зависимыми циклами, если те допускают следующее представление:

$$\underbrace{A_1(i)\ A_2(i) \ldots A_L(i)}_{i<N} P,$$

где оператор $A_j(i)$ ($j = 1, 2, \ldots, L$) может функционально зависеть только от $A_{j_1}(i-k)$ ($j_1 < j$, $k \geq 0$).

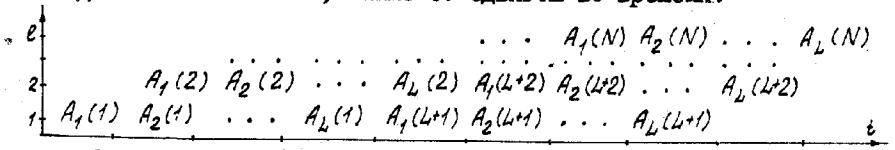
Возможны две основные схемы:

– к о н в е й е р н а я н е о д н о р о д н а я, когда каждая ветвь выполняет один из операторов $A_j(i)$ со сдвигом во времени (t):



Данная схема позволяет хранить в каждой машине только часть операторов и соответствующих им массивов. Ее выгодно применять, когда удается добиться хорошего согласования времени выполнения разнородных ветвей;

– к о н в е й е р н а я о д н о р о д н а я, когда ветвь выполняет всю последовательность операторов $A_j(i)$, в общем случае для нескольких i, также со сдвигом во времени:



Эта схема, так же как и просто однородная, позволяет получать хорошее согласование времени выполнения ветвей, однако каждая ветвь должна иметь всю последовательность операторов $A_j(i)$ и соответствующие массивы (или их части). Конвейерная однородная схема выгодна, когда все $A_j(i)$ одинаковы, то есть имеется два цикла: по i и j.

Возможность распараллеливания по схеме "конвейер" эквивалентна возможности преобразования двух зависимых циклов по i и j к зависимому циклу по n и независимому по m (рис. 2).

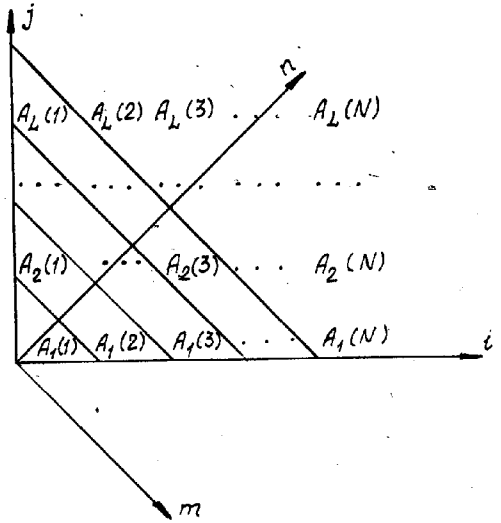


Рис. 2

Одно из важнейших свойств однородных схем – возможность построения универсальных р-алгоритмов [2], настраивающихся на число ветвей как на параметр и сохраняющих при этом высокую эффективность. Универсальные программы, построенные на основе таких алгоритмов, позволяют продолжать счет при выходе некоторых машин из строя, обладают возможностью варьирования времени решения задач, обеспечивают совместимость программных средств при наращивании числа машин в системе.

I.4. Процедуры обменных взаимодействий. Все многообразие взаимодействий между ветвями р-алгоритма может быть сведено к шести основным процедурам [II]:

– трансляционный обмен (ТО) – информация передается из одной ветви во все остальные: ТО (N, A, B), где N – номер передающей ветви, A и B – идентификаторы передаваемого и принимающего массивов, соответственно;

– трансляционно-циклический обмен (ТЦО) – реализует ТО в цикле по ветвям: ТЦО (A, B);

– обмен сдвигом (ОС) – информация передается из ветви i в ветвь $i+1$ (сдвиг по возрастанию номеров ветвей), либо в ветвь $i-1$ (сдвиг по убыванию номеров ветвей): ОС (Z_1, Z_2, A, B), где $Z_1 = 0$ при сдвиге ($i \to i+1$), $Z_1 = 1$ при сдвиге ($i \to i-1$); $Z_2 = 0$ при циклическом сдвиге ($\ell \to 1$ или $1 \to \ell$), иначе $Z_2 = 1$;

– дифференцированный обмен (ДО) – информация передается из одной ветви в некоторые выделенные: ДО ($N, A, B, N_1, N_2, \ldots, N_k$), где $N_1, N_2, \ldots, N_k$ – номера принимающих ветвей;

– коллекторный обмен (КО) – информация собирается из всех ветвей в одну выделенную: КО (N, A, B);

– обобщенный условный переход (ОУП) – изменяет естественный порядок реализации операторов во всех ветвях при выполнении обобщенного условия: ОУП (F, M), где F – идентификатор переменной, участвующей в выработке обобщенного признака. Например, если значения F отрицательны во всех ветвях, то управление передается на метку M. Этот же оператор синхронизирует ветви.

Большинство взаимодействий между ветвями приходится на первые три процедуры. Они эффективны при выполнении на ОВС. КО применяется сравнительно редко и в основном для объединения результатов перед выдачей. Потребность в ДО возникает, когда нужен срочный доступ к информации, находящейся в других машинах. КО и ДО менее эффективны при реализации (особенно для ОВС магистрального типа), поэтому их использование желательно сводить к минимуму.

I.5. Эффективность р-алгоритма. При разработке р-алгоритма выделяются две основные причины увеличения числа операций по сравнению с последовательным алгоритмом:

– введение процедур обменных взаимодействий;

– появление пустых операций, связанных с простоями при запуске (завершении) конвейерных схем и с неравномерностью распределения программных блоков или числа повторений циклов между ветвями.

Эффективность р-алгоритма определяется величиной

$$S = \frac{t_{cr}}{t_{cr} + t_{o\delta} + t_{np}}, \qquad (4)$$

где $t_{сч}$ – чистое время счета на ОВС, $t_{об}$ – время обменов между машинами, $t_{пр}$ – время простоев.

Удобно (4) представить в виде

$$S = k_{сч}(\ell)/\ell \,, \quad \text{где} \quad k_{сч}(\ell) = \frac{t_{сч}\cdot\ell}{t_{сч}+t_{об}+t_{пр}}$$

Аналогично вводится

$$k_{доп}(\ell) = \frac{(t_{об}+t_{пр})\,\ell}{t_{сч}+t_{об}+t_{пр}}$$

Если условно разделить операции счета и дополнительные расходы (обмены и простои), то $k_{сч}$ – это число машин, занятых только счетом, а $k_{доп} = \ell - k_{сч}$ – только обменами и простоями.

По зависимости $k_{сч}$ от ℓ можно судить о возможностях распараллеливания алгоритма. Обычно существует две области распараллеливания: эффективная (I), где $k_{доп} \ll \ell$; неэффективная (2), где доля $k_{доп}$ резко растет и достигает при $\ell = \ell_{пр}$ наибольшей величины (рис. 3).

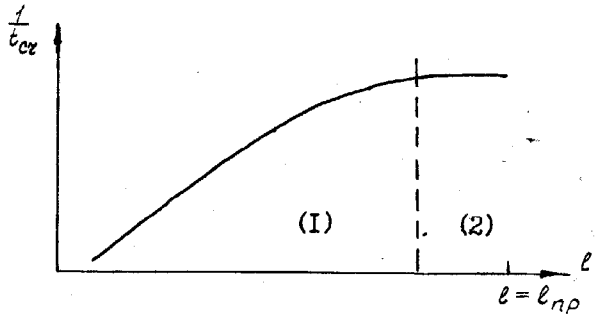


Рис. 3

Дальнейшее увеличение ℓ уже не уменьшает времени счета.

Для определения качества распараллеливания алгоритмов могут использоваться и другие коэффициенты эффективности, например,

$$\delta = \frac{t_{об}+t_{пр}}{t_{сч}} \quad \text{или} \quad \alpha = \frac{t_{эм}}{\ell\cdot t_{сч}} \,,$$

где $t_{эм}$ есть время счета на одной машине.

Помимо $t_{об}$ и $t_{пр}$ на эффективность р-программ влияет зависимость времени выполнения основных арифметических операций

от вида операнд. Объединенные в систему машины будут заканчивать работу в различное время, даже если они выполняют одну и ту же программу. Для того чтобы простои машин, связанные с десинхронизацией, были меньше ε , достаточно выполнения следующего условия [I0]:

$$\tau \geqslant \frac{2(\ell-1)}{\varepsilon^2}\,\frac{\sum_{k=1}^{s} \tau_k\, \mathcal{D}\tau_k}{\left(\sum_{k=1}^{s} \tau_k\, M\tau_k\right)^2} \,,$$

где $\mathcal{D}\tau_k$ – дисперсия продолжительности операции k -го вида; s – число операций, время выполнения которых зависит от операнд; $M\tau_k$ – математическое ожидание τ_k ; τ_k – случайная величина, характеризующая потери на синхронизацию при выполнении команд k -го типа; τ_k – число операций k -го вида, τ – общее число выполненных операций.

Для каждой конкретной машины величины $\mathcal{D}\tau_k$ и $M\tau_k$ могут быть заранее установлены экспериментально.

Как показал опыт, при использовании методики распараллеливания по циклам затраты на обменные операции, простои и десинхронизацию машин не превышают нескольких процентов от общего объема вычислений.

2. Примеры параллельных алгоритмов

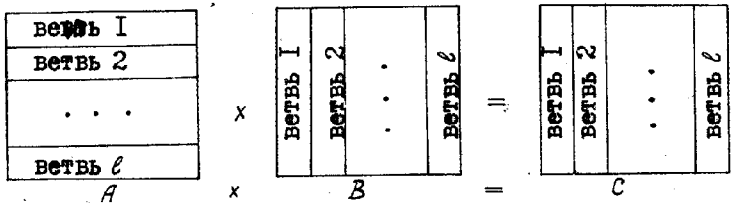
На практике для многих задач часто существует несколько равноценных последовательных алгоритмов. Поэтому, как правило, удается из них выбрать такой, который легко распараллеливается. Покажем это на примерах, акцентируя внимание на способе распределения массивов, типе процедур обменных взаимодействий и величине дополнительных расходов.

2.I. Задачи линейной алгебры.

2.I.I. Умножение матриц. Пусть имеются матрицы больших размеров $A[1{:}N, 1{:}M]$, $B[1{:}M, 1{:}J]$, произведение которых необходимо вычислить на ОВС. Элементы результирующей матрицы $C = A\times B$ вычисляются по формуле $c_{ij} = \sum_{k=1}^{M} a_{ik}\, b_{kj}$. Введем функцию $DEL(N)$, вычисляющую значение $\left[\frac{N}{\ell}\right]+1$ для ветвей с относитель-

ными номерами, не превышающими остаток от деления N на ℓ, и значение $\left[\frac{N}{\ell}\right]$ для остальных ветвей. Пусть $DEL_\alpha(N)$ – значение $DEL(N)$ в машине α.

Применим к матрице A ГП-распределение, а к матрице B ВП-распределение и, кроме того, заведем в каждой ветви рабочий массив $W=max\{N,M,\mathcal{J}\}$, который может содержать любую строку или столбец.



Опишем взаимодействие ветвей параллельной программы.

Сначала первая ветвь передает первую строку матрицы A для других ветвей. После чего все ветви параллельно вычисляют элементы первой строки матрицы C. Затем то же повторяется со следующими строками, содержащимися в первой ветви. Когда первая ветвь перешлет все свои строки, пересылками начинает заниматься вторая ветвь и т.д. до ℓ.

Результирующая матрица C получается ВП-распределенной. Если пересылать не строки матрицы A, а столбцы матрицы B, то C получится ГП-распределенной. Отсюда видно, что если для дальнейшего использования подходит любое из этих распределений, то лучше пересылать матрицу с меньшим объемом.

Блок-схема ветви представлена на рис. 4.

Оценим долю времени, которое идет на обмен между элементарными машинами (ЭМ) системы. Пересылка строки из M элементов обеспечивает выполнение в каждой ЭМ $\tau \cdot M$ умножений и $\tau(M-1)$ сложений ($\tau = DEL(\mathcal{J})$). При большом M можно положить, что на один принятый ЭМ код приходится по τ умножений и сложений. Пусть t_n – время пересылки одного кода, t_y и t_c – время выполнения операций умножения и сложения соответственно, тогда отношение времени обменных операций к времени "чистых вычислений" составит величину $\delta = t_n[(t_y + t_c)\mathcal{J}/\ell]^{-1}$ (рис. 5).

Максимальное значение δ, равное k, будет при $\mathcal{J}=\ell$. Для ОВС "Минск-222" оно составляет:

50 мксек [(360 + 220) мксек]$^{-I}$ = 0,08

При $\mathcal{J}/\ell = 8$ $\delta \approx 1\%$.

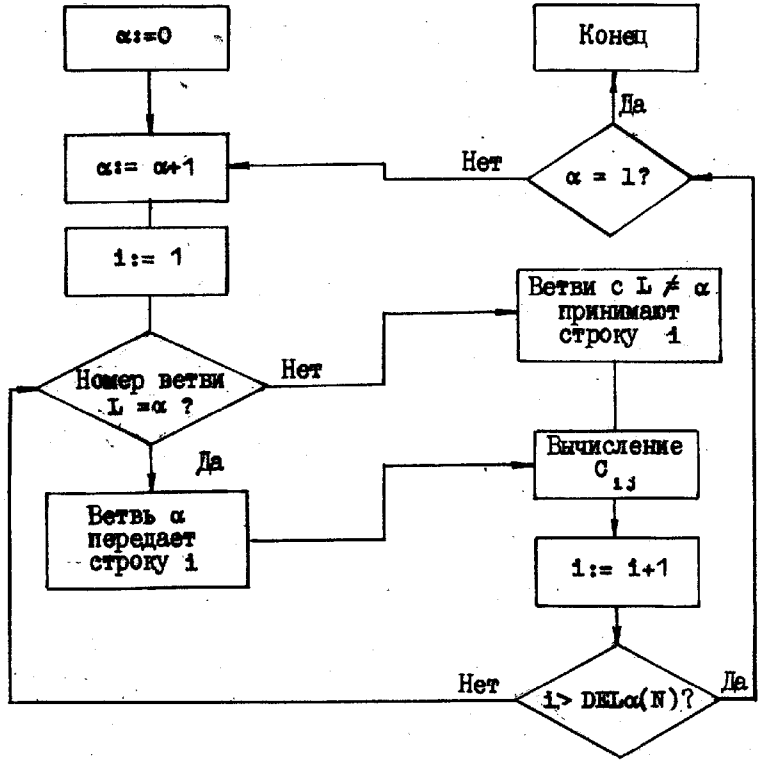


Рис. 4

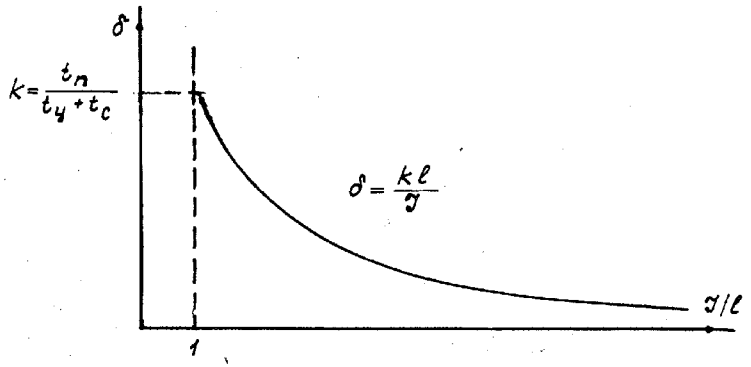


Рис. 5

Кроме приведенного распределения массивов подходят и другие, например, ГП-распределение для обеих матриц. В этом случае столбцы матрицы B разрезаны, поэтому ветви поочередно пересылают содержащиеся в них части очередного столбца. Как только столбец оказывается задублированным, во всех машинах вычисляются элементы столбца j матрицы C. Доля накладных расходов аналогична доле расходов в предыдущем случае.

Обмены между ЭМ при решении рассмотренной задачи являются упорядоченными, сравнительно редкими и сводятся к двум схемам – трансляционной и трансляционно-циклической.

2.I.2. О б р а щ е н и е м а т р и ц. Рассмотрим обращение матрицы $A=\{\alpha_{kj}\}$ методом пополнения. Его суть состоит в том, что A^{-1} вычисляется за n шагов по рекуррентной формуле [9,I6]:

$$\alpha_j^{(k)} = \alpha_j^{(k-1)} - \frac{\sum_{i=1}^{n} \alpha_{ki}\,\alpha_{ij}^{(k-1)}}{1+\sum_{i=1}^{n} \alpha_{ki}\,\alpha_{ik}^{(k-1)}}\,\alpha_k^{(k-1)}, \qquad (5)$$

где $\alpha_j^{(k)}$ – столбец j матрицы, которая получается на шаге k; $\alpha_j^{(0)}$ – столбец j единичной матрицы.

На каждом шаге, чтобы вычислить столбец j матрицы нового приближения, нужно иметь столбцы j и k матрицы $(k-1)$ приближения и строку k матрицы A. Возможны следующие разделения массивов по ЭМ: ГП-распределение для матрицы $\{\alpha_{ij}\}$, НП-распределение для матриц $\alpha_{ij}^{(k)}$. На каждом шаге k ЭМ, имеющая строку k матрицы $\{\alpha_{ij}\}$ и столбец k матрицы $\{\alpha_{ij}^{k-1}\}$, посылает их во все ЭМ. Знаменатель формулы (5) вычисляется параллельно. Каждая ЭМ вычисляет часть $\sum_{i=1}^{n} \alpha_{ki}\,\alpha_{ik}^{(k-1)}$ и пересылает результаты с использованием трансляционно-циклической схемы в другие ЭМ. Затем выполняются основные вычисления.

Дополнительные расходы в этой задаче составят $(2n+\ell)$ операций обмена и $(\ell-1)$ операций сложения (последние связаны с тем, что только в $(\ell-1)$ машинах дублируется сложение частичных сумм при вычислении знаменателя формулы (5)).

$$\delta \approx \frac{(2n+\ell)t_n + (\ell-1)t_c}{\frac{n}{\ell}(n+1)(t_y+t_c)} < \frac{2t_n}{\frac{n}{\ell}(t_y+t_c)} - \frac{t_c+t_n}{(\frac{n}{\ell})^2(t_y+t_c)};$$

при $n/\ell = 8$ для ОВС "Минск-222" $\delta \approx 0{,}025$.

При обращении матриц, как и при умножении, используются трансляционная и трансляционно-циклическая схемы обмена.

2.I.3. Р е ш е н и е с и с т е м ы у р а в н е н и й м е т о д о м Г а у с с а. Метод основан на последовательном исключении неизвестных. Рассмотрим одну из возможных вычислительных схем [I6]. Пусть дана система уравнений:

$$\begin{aligned}
&\alpha_{11}x_1 + \alpha_{12}x_2 + \ldots + \alpha_{1n}x_n = f_1,\\
&\alpha_{21}x_1 + \alpha_{22}x_2 + \ldots + \alpha_{2n}x_n = f_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&\alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots + \alpha_{nn}x_n = f_n.
\end{aligned}$$

Исключим x_1. Для этого уравнение i делим на α_{i1}, а затем вычитаем первое уравнение из всех остальных. Система принимает вид:

$$\begin{aligned}
&x_1 + b_{12}^1 x_2 + \ldots + b_{1n}^1 x_n = F_1^1,\\
&b_{22}^1 x_2 + \ldots + b_{2n}^1 x_n = F_2^1,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&b_{n2}^1 + \ldots + b_{nn}^1 x_n = F_n^1,
\end{aligned}$$

где $b_{ij}^1 = \frac{\alpha_{ij}}{\alpha_{i1}}$; $F_i^1 = \frac{f_i}{\alpha_{i1}}$; $i = 1,2,\ldots,n$; $j = 2,3,\ldots,n$.

Затем исключим x_2, для чего с уравнениями $2 \div n$ преобразованной системы повторим предыдущую процедуру и т.д., пока система не станет вида:

$$\begin{aligned}
&x_1 + b_{12}^1 x_2 + \ldots + b_{1n}^1 x_n = F_1^1\\
&x_2 + \ldots + b_{2n}^2 x_n = F_2^2\\
&\ldots\ldots\ldots\ldots\ldots\\
&x_n = F_n^n,
\end{aligned}$$

которую вычисляем, начиная с x_n.

Процесс д е л е н и я строк на коэффициенты перед x_i независим и не требует информации от других ЭМ*). Процесс в ы ч и т а н и я осуществляется пересылкой нужной строки во все машины.

---

*) Для простоты предполагается, что элементы матрицы в нужных местах отличны от нуля.

Использование ГП- или ВП-распределений для данного метода ведет к значительным простоям машин после исключения некоторого числа неизвестных. Простои существенно уменьшаются применением ЦГП-распределения с шириной полосы, равной единице.

При $n/\ell > 10$

$$\delta \approx \frac{3\ell}{4n} + \frac{3\ell\, t_n}{2n(t_q + t_b)} ,$$

где t_q и t_b – время выполнения операций деления и вычитания, соответственно. Первое слагаемое учитывает простои отдельных ЭМ, второе – затраты на обмены. Простои ЭМ возникают в основном из-за различия в числе обрабатываемых строк при исключении x_i .

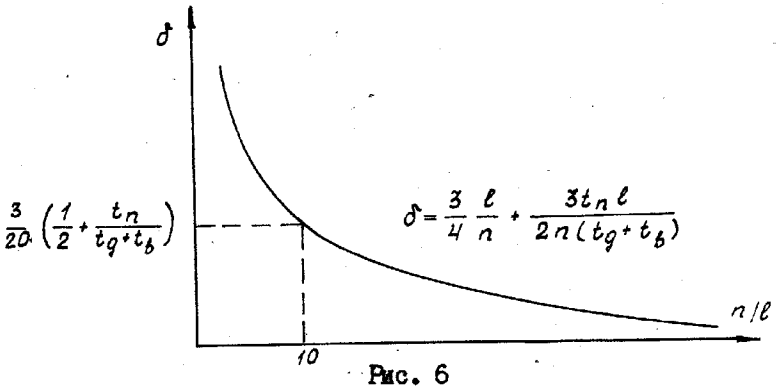


Рис. 6

Обычно $t_n \ll (t_q + t_b)$, поэтому для $\delta \le 0{,}05$ необходимо $\frac{n}{\ell} \geqslant 15$.

Вторая половина задачи (непосредственное вычисление компонент вектора X), как и первая, требует только трансляционной схемы обмена. Вычисленная компонента x_n передается всем машинам, каждая из которых умножает на нее коэффициенты α_{in} и пересчитывает компоненты вектора F . В результате в одной из машин оказывается вычисленной x_{n-1} . Она пересылается во все машины, умножается на соответствующие $\alpha_{i,n-1}$, после чего пересчитываются компоненты вектора F , находится x_{n-2} и т.д.

Рассмотренный метод применим и для вычисления определителей.

2.I.4. Р е ш е н и е с и с т е м ы у р а в н е н и й м е т о д о м З е й д е л я . По Зейделю, в отличие от метода последовательных итераций вычисленное очередное приближение для компоненты вектора сразу же используется при отыскании следующей компоненты.

Для уравнения вида $AX = B$ вычисления ведутся по формуле [I6]:

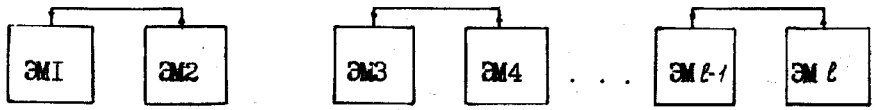
$$x_i^{(k+1)} = -\frac{1}{\alpha_{ii}}\left[\sum_{j=1}^{i-1} \alpha_{ij}\, x_j^{(k+1)} + \sum_{j=i+1}^{n} \alpha_{ij}\, x_j^{k} - b_i\right].$$

Для нахождения $x_i^{(k+1)}$ нужно иметь значения части компонент вектора старого приближения и всех компонент нового, а также соответствующие им столбцы матрицы A , что достигается ГП-распределением для векторов X и B и ВП-распределением для матрицы A .

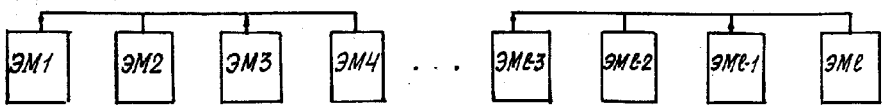
Перед основными вычислениями строка i $(i = 1,2,\ldots,n)$ матрицы A и компонента i вектора B делятся на α_{ii} , предварительно пересланных в каждую ЭМ. Для вычисления компоненты i вектора $x^{(k)}$ ЭМ умножают содержащиеся в них части векторов $x^{(k-1)}$ и $x^{(k)}$ на соответствующие части строки матрицы A и складывают результаты. Частичные суммы, полученные в каждой машине, пересылаются в ЭМ, содержащую $x_i^{(k-1)}$. Сложение частичных сумм определяет компоненту $x_i^{(k)}$, которая замещает $x_i^{(k-1)}$.

Коэффициент эффективности: $\delta \approx \frac{\ell^2 (t_n + t_c)}{n(t_y + t_c)}$

Величина δ уменьшится, если частичные суммы складывать параллельно. Для этого система разбивается на подсистемы по две ЭМ. При этом четные ЭМ передают значение суммы предшествующей нечетной, которая вычисляет новую частичную сумму.



Затем система разбивается на подсистемы по 4 ЭМ для вычисления очередных частичных сумм.



и т.д., пока не получится окончательная сумма.

Общее число таких этапов равно $\log_2(\ell+z)$ ($z=2^{k+1}-\ell$, если $2^k<\ell<2^{k+1}$, иначе $z=0$), а время каждого этапа – $(t_c+t_n+t_н)$, где $t_н$ – время операции "настройка системы". Соответственно

$$\delta \approx \frac{\ell \log_2(\ell+z)}{n(t_y+t_c)}\left(t_n+t_н+\frac{3}{4}t_c\right);$$

где $\frac{3}{4}$ – коэффициент при t_c – учитывает среднее число простаивающих ЭМ.

При $\ell=10$ и $n/\ell=20$ для ОВС "Минск-222" $\delta \approx 0{,}07$.

Здесь приведена непосредственная реализация метода Зейделя*) на ОВС. Можно, однако, использовать и его модификацию [18], ориентированную на параллельный счет. Алгоритм р-ветви в этом случае состоит в следующем.

Для массивов A и B используется ГП-распределение, X дублируется. Каждая ЭМ начинает и применяет метод Зейделя для своей полосы, то есть по значениям $X_i^{(k-1)}$ первая ЭМ вычисляет значение $X_1^{(k)}$, вторая – $X_{\frac{n}{\ell}+1}^{(k)}$ и т.д., ℓ-я – $X_{\frac{n}{\ell}(\ell-1)+1}^{(k)}$.

Вычисленные компоненты $X^{(k)}$ передаются в другие машины, и каждая из них вычисляет следующую компоненту $X^{(k)}$. Например, первая ЭМ вычисляет $X_2^{(k)}$, а вторая – $X_{\frac{n}{\ell}+2}^{(k)}$, используя $X_1^{(k)}$, $X_{\frac{n}{\ell}+1}^{(k)}$, $X_{2\frac{n}{\ell}+1}^{(k)}$, . . . , $X_{(\ell-1)\frac{n}{\ell}+1}^{(k)}$ и необходимые компоненты вектора $X^{(k-1)}$.

При таком подходе на ℓ операций обмена приходится $(n-1)$ операций умножения и сложения: $\delta \approx \frac{\ell \cdot t_n}{(n-1)(t_y+t_c)}$. При $n=10$ для ОВС "Минск-222" $\delta < 0{,}01$. Взаимодействия между ЭМ свелись в первом алгоритме к коллекторному обмену и параллельным вычислениям при сложении ℓ чисел, находящихся в разных ЭМ, во втором – к трансляционно-циклическому обмену.

*) Решение системы уравнений методом Зейделя-Самарского см. в [17].

2.I.5. Вычисления корня алгебраического или трансцендентного уравнения $f(x)=0$. Требуется на интервале (a,b) найти корень при условии $m<f'(x)<M$; $f''(x)>0$.

Для построения алгоритма параллельной ветви (рис.7) используется метод хорд*), заключающийся в следующем. Кривая $y=f(x)$ заменяется хордой $P(\alpha)Q(b)$, определяется точка $x_1=\alpha+\frac{(b-\alpha)f(\alpha)}{f(b)-f(\alpha)}$, после чего проводится хорда $P(x_1)Q(b)$, определяется x_2 и т.д.

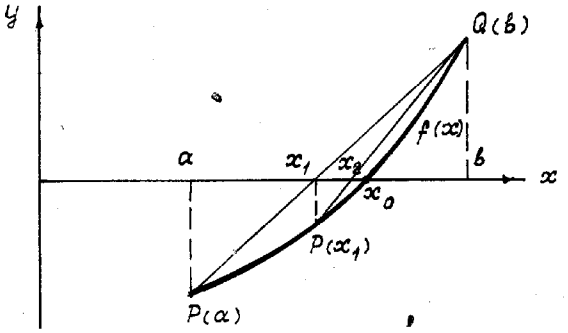


Рис. 7

Для реализации задачи на системе из ℓ машин (α,b) разбивается на ℓ подынтервалов, и в середине каждого вычисляется $f(x)$. Это определяет подынтервал (x_1,x_2) длиной $\frac{b-\alpha}{\ell}$, содержащий корень x_0. Точки x_1 и x_2 – соседние; в них значения функции имеют разные знаки (рис. 8).

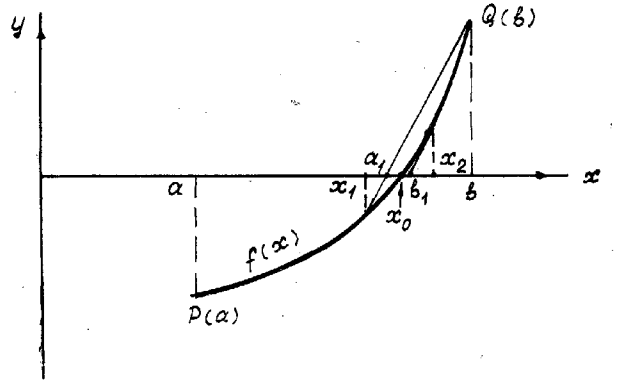


Рис. 8

*) Аналогичный результат получается и при использовании метода Ньютона [19].

После интервала (x_1, x_2) рассматривается интервал (a_1, b_1), где

$$a_1 = x_1 + \frac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)}; \qquad b_1 = x_2 - \frac{f(x_2)}{M}. \qquad (6)$$

Интервал (a_1, b_1) разбивается на l частей, повторяется описанная процедура, получается интервал (a_2, b_2) и т.д., пока длина интервала не станет меньше заданной величины.

Можно видеть, что параллельный счет оправдан, когда объем вычислений значения $f(x)$ намного превосходит объем вычислений границ интервала a_i, b_i по формулам (6).

2.2. Задачи линейного программирования

2.2.I. Решение общей задачи линейного программирования симплексным методом. Требуется найти $\min \sum_{j=1}^{n} c_j x_j$ при следующих ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i;$$

$$x_j \geq 0; \quad j = 1,2,\dots,n; \quad i = 1,2,\dots,m;$$

где c_j, a_{ij}, b_i – неотрицательные константы, $m < n$.

Не вдаваясь в тонкости алгоритма, укажем только на то, что основные вычисления сводятся к умножениям вектора на матрицу и матрицы на матрицу, что уже рассмотрено в предыдущих примерах. Подробнее смотри [2], где приведен также р-алгоритм решения транспортной задачи.

2.3. Обыкновенные дифференциальные уравнения

2.3.I. Задача Коши. Решение системы дифференциальных уравнений:

$$\frac{dy_i}{dt} = f_i(y_1, y_2, \dots, y_n)$$

при начальных условиях:

$$y_i(t_0, y_1, y_2, \dots, y_n) = g_i(y_1, y_2, \dots, y_n); \quad i = 1,2,\dots,n;$$

согласно методу Рунге-Кутта выполняется последовательными шагами. Новые значения функций на $(s+1)$-м шаге вычисляются по следующим формулам:

$$k_{1i}^s = h f_i(y_1^s, y_2^s, \dots, y_n^s);$$
$$k_{ji}^s = h f_i(y_1^s + k_{j-1,1}^s, y_2^s + k_{j-1,2}^s, \dots, y_n^s + k_{j-1,n}^s);$$
$$i = 1,2,\dots,n; \quad j = 2,3,4; \quad h \text{ – константа},$$
$$\Delta^s y_i = \frac{1}{3}(k_{1i}^s + 2k_{2i}^s + k_{4i}^s); \quad y_i^{s+1} = y_i^s + \Delta y_i^s$$

Нетрудно видеть, что каждая последующая формула использует результаты предыдущей и счет по ним на каждом шаге ведется независимо от i. Поэтому для всех массивов подходит ГП-распределение. Взаимодействия между ветвями происходят один раз на все вычисления шага и составляют n обменных операций, что пренебрежимо мало по сравнению с общим объемом вычислений [2]

2.4. Дифференциальные уравнения в частных производных.

2.4.I. Задача Дирихле. В прямоугольной области $0 \leq x \leq a$, $0 \leq y \leq b$ требуется найти решение уравнения

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = g(x, y)$$

при заданных значениях функции u на границе.

Явная разностная схема:

$$u_{jk}^{s+1} = \frac{1}{4}(u_{j-1,k}^s + u_{j+1,k}^s + u_{j,k-1}^s + u_{j,k+1}^s - h^2 g_{jk}),$$

где u_{jk}, g_{jk} – значения функций u, g в точке (j,k) разностной сетки, ведет к использованию для массива u_{jk} ГПД-распределения (рис.9, N, M – число точек разностной сетки соответственно по x и y). Следовательно, каждой ЭМ, помимо строк полосы, которую она обрабатывает, предоставляются крайние строки соседних полос. В этом случае ЭМ могут независимо вычислять значения u_{jk} нового приближения. Передача крайних строк полос своим соседям выпол –

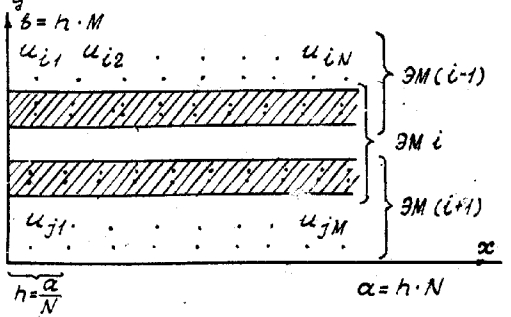
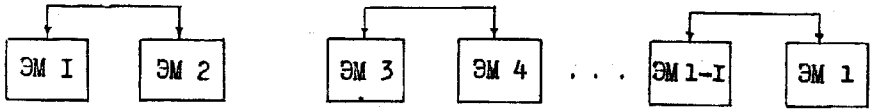
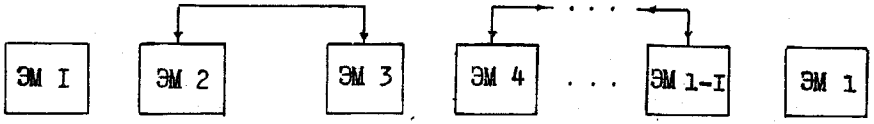


Рис. 9.

няется за два этапа. На первом – система делится на подсистемы по две ЭМ:



В каждой подсистеме машины обмениваются нужными строками. Затем реализуется новое разбиение:



Оставшиеся пары обмениваются нужными строками и весь процесс взаимодействия завершается.

Общее время выполнения этапов $\alpha = 4 \cdot t_n \cdot N$. Величина коэффициента эффективности

$$\delta \approx \frac{2 t_n}{\frac{N}{\ell}(2t_c + t_y)}$$

Неявная разностная схема [21]. Итерационный процесс задается формулой

$$(E+\frac{\tau}{2}\Lambda_1)(E+\frac{\tau}{2}\Lambda_2)u^{i+1} = (E-\frac{\tau}{2}\Lambda_1)(E-\frac{\tau}{2}\Lambda_2)u^{i} + \tau g ,$$

которая сводится к последовательности 4-х уравнений:

$$u^{i+1/4} = (E - \frac{\tau}{2}\Lambda_2)u^{i} , \qquad (7)$$

$$u^{i+2/4} = (E - \frac{\tau}{2}\Lambda_1)u^{i+1/4} + \tau g , \qquad (8)$$

$$(E + \frac{\tau}{2}\Lambda_1)u^{i+3/4} = u^{i+2/4} , \qquad (9)$$

$$(E + \frac{\tau}{2}\Lambda_2)u^{i+1} = u^{i+3/4} , \qquad (10)$$

где Λ_1 , Λ_2 – разностные аналоги операторов $\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}$; τ – положительная константа.

Явные (первые два) уравнения решаются, как и в предыдущем случае, неявные приводятся к уравнениям следующего вида:

$$u^{i+3/4}_{j+1,k} - b_{jk} u^{i+3/4}_{j,k} + c_{jk} u^{i+3/4}_{j-1,k} = -F_{jk} ,$$
$$u^{i+1}_{j,k+1} - b'_{jk} u^{i+1}_{jk} + c'_{jk} u^{i+1}_{jk-1} = -f_{jk} , \qquad (11)$$

где $b_{jk}, c_{jk}, F_{jk}, f_{jk}, b'_{jk}, c'_{jk}$ – известные функции.

Уравнения (II) решаются методом прогонок, при этом сначала вычисляются величины:

$\beta_{j+1} = \frac{c_{j+1,k}}{b_{jk} - \beta_j}$; $z_{j+1} = \beta_{j+1}(z_j + F_{jk})$ (прямая прогонка), а затем $u_{jk} = (\beta_{j+1} u_{j-1,k} - z_{j+1}) / c_{j+1,k}$ (обратная прогонка).

Для массивов можно применить ГП-распределение и C-распределение.

При ГП – р а с п р е д е л е н и и уравнения 7,8 решаются, как явные, уравнения 9 – независящими от y прогонками по x, а уравнения I0 – прогонками по y с помощью конвейерной схемы (рис.I0), где

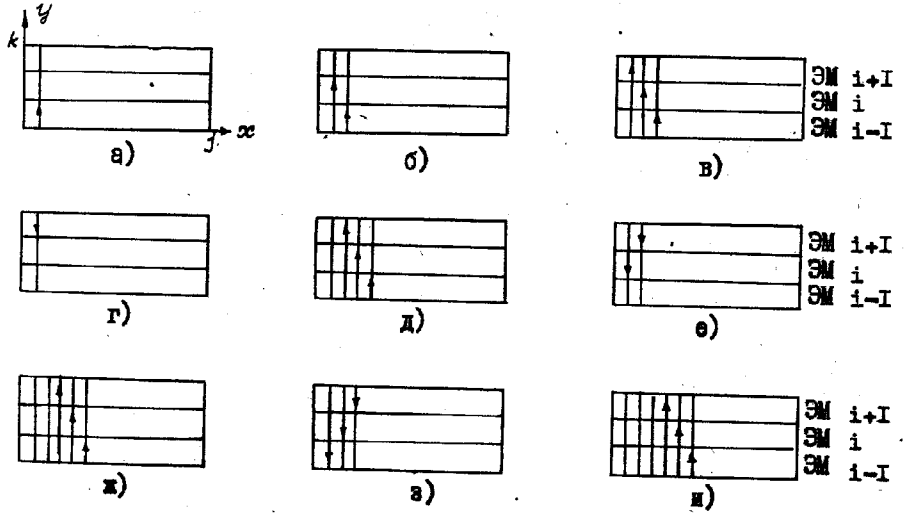


Рис. I0

а) ЭМ $(i-1)$ вычисляет прогоночные коэффициенты, машины i и $(i+1)$ простаивают;

б) ЭМ $(i-1)$ передает информацию ЭМ i , после чего ведет прогонку "вверх" для следующего значения x ;

в) ЭМ $(i-1)$ передает информацию ЭМ i , ЭМ i – ЭМ $(i+1)$, и все работают;

г) ЭМ $(i+1)$ ведет прогонку "вниз", машины i и $(i-1)$ простаивают;

д) все машины вычисляют коэффициенты для прогонки "вверх";

е) ЭМ i и $(i+1)$ ведут прогонку "вниз", машина $(i-1)$. простаивает;

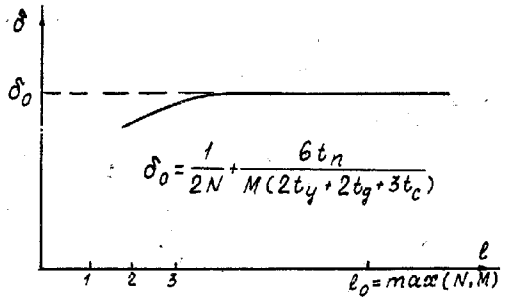


Рис. II

ж) аналогично д);

з) все ЭМ ведут прогонку "вниз", после чего выполняются по очереди случаи ж) и з) для соответствующих j.

Для запуска "конвейера вверх" (и для останова), при которых приблизительно половина машин простаивает, нужно время

$$t_1 \approx (\ell-1)\frac{M}{\ell}(2t_c + t_y + t_g).$$

Аналогичная картина для "конвейера вниз":

$$t_2 \approx (\ell-1)\frac{M}{\ell}(t_c + t_y + t_g).$$

Обмены между ЭМ представляют собой сдвиги данных "вверх" или "вниз" по машинам. Для конвейера "вверх" они требуют $4Nt_п$, а для конвейера "вниз" $2Nt_п$ единиц времени.

Величина коэффициента эффективности (рис. II) при $\ell >> 1$

$$\delta \approx \frac{(\ell-1)\frac{M}{\ell}(3t_c + 2t_g + 2t_y) + 12Nt_п}{2MN(2t_y + 2t_g + 3t_c)} \approx \frac{1}{2N} + \frac{6t_п}{M(2t_y + 2t_g + 3t_c)} = \delta_0;$$

при $N \approx M > 100$ для ОВС "Минск-222" $\delta \approx 0{,}01$.

К о н в е й е р н а я с х е м а применима при решении многих задач на ОВС. Цикл, который реализует эту схему, назван к о н в е й е р н ы м.

П р и С - р а с п р е д е л е н и и (рис.I2) массивы разделяются на более мелкие части, увеличивается протяженность границ раздела и, следовательно, общее число обменов. Для уравнений 7,8 накладные расходы возрастают вдвое (из-за обменов как через горизонтальные, так и через вертикальные границы разрезов).

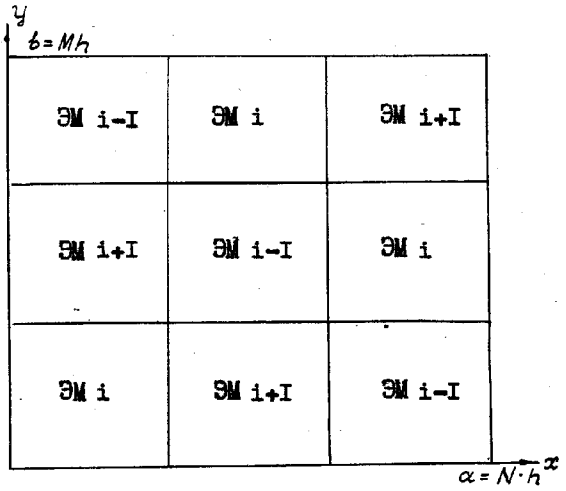


Рис. I2

Обмены при решении уравнений 9,I0 также увеличивают - ся, но зато нет простоев, свойственных конвейерной схеме. Все ЭМ начинают процесс вычислений одновременно, на границах раздела выполняют обмен сдвигом и затем обрабатывают точки до следующих границ раздела и т.д. (рис.I3).

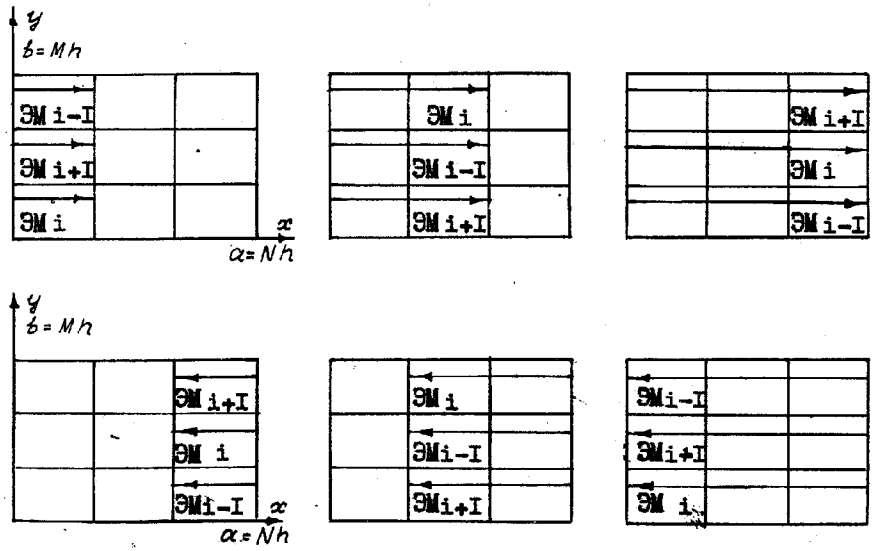


Рис. I3

Аналогичная картина, ориентированная только вдоль оси y, имеет место для уравнения I0.

Время на обменные операции для уравнений 9 (или I0) $\alpha = 6\frac{M}{\ell}(\ell-1)t_п$, где $(\ell-1)$ - число границ раздела; $4\frac{M}{\ell}t_п$ -

время обмена сдвигом при прогонке вперед, а $2\frac{M}{\ell}t_n$ – при прогонке назад. Отсюда

$$\delta \approx \frac{6\frac{M}{\ell}(\ell-1)t_n}{MN(2t_y+2t_g+3t_c)} \approx \frac{6t_n}{N(2t_y+2t_g+3t_c)}$$

Рассмотренные в данном параграфе методы применимы и для систем дифференциальных уравнений в частных производных [13,22,23].

2.5. Информационно-логические задачи

2.5.1. Перекрещивание массивов [24]. Пусть имеются два больших массива A и B, из которых A – массив некоторых сводок, B – массив документов, формируемых на основе этих сводок. Массивы находятся во внешней памяти, например на магнитных лентах (МЛ). Требуется обеспечить встречу в оперативной памяти (ОП) каждого элемента массива B с каждым элементом массива A.

Обычно массивы A и B разбивают на части соответственно A_j и B_j, $j=1,2,\ldots,n$; вводят часть B_1 в ОП и мимо неё прогоняют массив A, затем последовательно вводят B_2, B_3 и т.д., прогоняя мимо каждой по частям массив A (рис. 14). Для решения этой задачи на ОВС массивы A и B распределяются между ЭМ с помощью ГП- или ВП-распределений.

ЭМ i вводит части B_{i1} и A_{i1} в оперативную память и обеспечивает их встречу (при встрече, время которой, как правило, много меньше времени ввода, данные одного массива используются для преобразования другого).

Затем каждая ЭМ вводит часть A_{i2}, A_{i3} и т.д., используя информацию сама и предоставляя ее другим. После сформирования всех документов из частей B_{i1} в ОП вводятся части B_{i2} $(i=1,2,\ldots,\ell)$, и повторяется весь прогон для A.

Время решения этой задачи на системе из ℓ ЭМ (по отношению к времени решения на одной ЭМ) уменьшается:

– в ℓ раз – за счет параллельной работы над распределенными массивами;

– еще в α раз $(\alpha=\ell\div\frac{t_{мл}}{t_n})$ – за счет одновременного размещения ℓ частей массива B в ОП и замены обменов с МЛ обменами между оперативными памятями ЭМ по быстрым системным каналам.

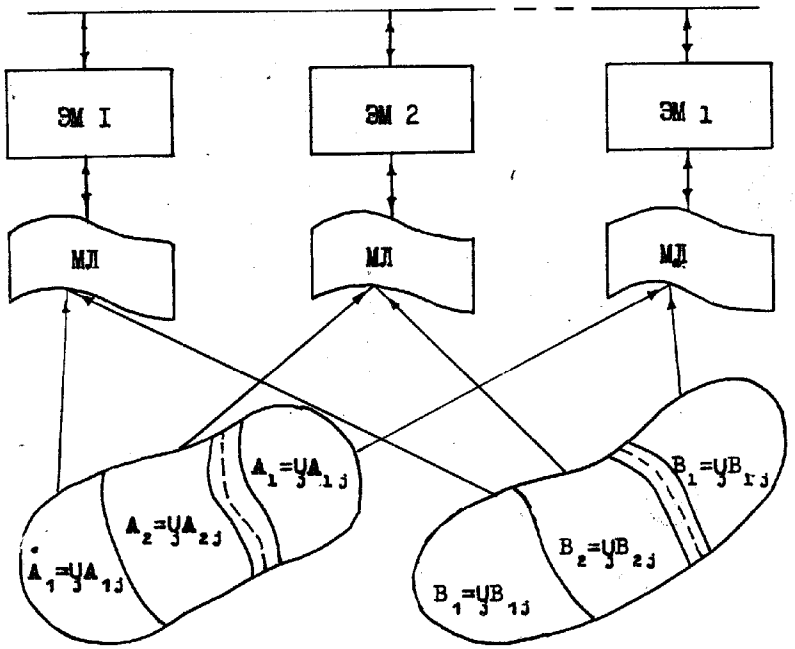


Рис. 14

Зависимость коэффициента α от ℓ иллюстрируется графиком, приведенным на рис. 15.

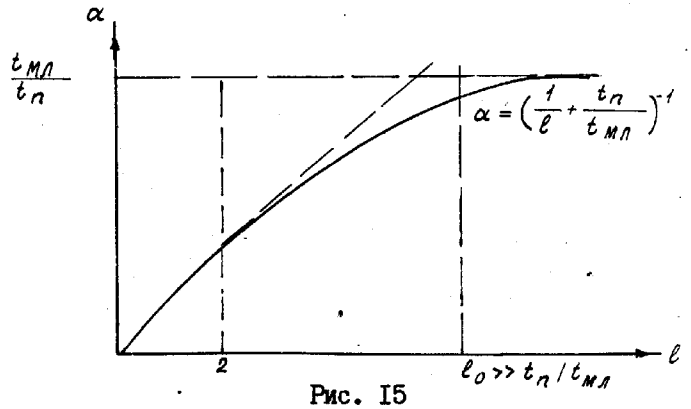


Рис. 15

2.5.2. Упорядочение массивов. На МЛ имеется большой массив C, элементы которого идентифицируются специальными признаками. Требуется упорядочить массив таким образом, чтобы элементы в нем располагались в порядке возрастания (убывания) значений признаков.

Реализация задачи на ОВС заключается в следующем. Массив C делится между ЭМ. Каждая ЭМ прогоняет свой подмассив через оперативную память и собирает статистику [25] о сортируемых элементах. На основе суммарной статистики, полученной из статистик каждой ЭМ, выбираются характеристики новых подмассивов, которые должны будут получаться в каждой ЭМ в результате упорядочения. То есть каждая ЭМ знает пределы значений признаков элементов, которые должны быть в её результирующем подмассиве, а также число элементов каждого признака.

Сортировка выполняется на базе перекрещивания массивов. Массив C играет в перекрещивании роль массива A, а вакантные места упорядоченного массива - роль B. Каждая ЭМ "вводит" первую часть вакантных мест своего подмассива, прогоняет мимо них исходный массив и, используя статистику, расставляет элементы на свои места. Упорядоченные части результирующих подмассивов записываются на МЛ. Затем "вводятся" новые порции вакантных мест, которым обеспечивается встреча со всеми элементами исходного массива, в результате чего заполняются очередные части выходных подмассивов и т.д. Эффективность реализации на ОВС перекрещивания массивов обеспечивает эффективность упорядочения массивов. Отметим, что для этих задач нужно использовать в качестве коэффициента эффективности величину

$$\alpha = \frac{t_{эм}}{\ell \cdot t_{овс}} = \left(\frac{1}{\ell} + \frac{t_п}{t_{мл}} \right)^{-1}$$

Обмены между ЭМ здесь не являются дополнительными расходами, так как заменяют собой обмены ЭМ с внешней памятью.

Если $\alpha > 1$, то имеем нелинейное ускорение счета. Если $t_п \geq t_{мл}$, что может быть, например, когда ввод/вывод имеет совмещение, а обмены между ЭМ осуществляются медленно и $\alpha < 1$, то имеем дополнительные расходы, характеризуемые величиной $D = 1 - \alpha$.

Приведенное рассмотрение различных классов задач показывает, что методика распараллеливания по циклам не вызывает больших затруднений и доступна фактически любому программисту средней квалификации. Она позволяет получать эффективные универсальные р-программы, обеспечивающие высокую загрузку процессоров, что дает ускорение счета примерно в ℓ раз. Почти во всех задачах существуют условия, при которых получается дополнительный выигрыш во времени, связанный с предоставлением каждой задаче всей оперативной памяти системы, что позволяет отказаться (полностью или частично) от использования внешней памяти. Этот выигрыш получается, по существу, автоматически, без специальных усилий со стороны программиста.

Литература

1. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, 1962.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука" СО, 1966.

3. ГОЛОВЯШКИНА Л.В., КОЛОСОВА Ю.И., КОСАРЕВ Ю.Г., МИРЕНКОВ Н.Н. Автоматизация программирования на основе существующих трансляторов. -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 63-69.

4. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., ЛЕВАГИНА Т.А., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Язык параллельных алгоритмов. -Настоящий сборник, с. 33-54.

5. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., ЛЕВАГИНА Т.А., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Транслятор с языка параллельных алгоритмов. -Настоящий сборник, с. 55-73.

6. КОНСТАНТИНОВ В.И., НУРИЕВ Р.М. Метаязык трансляции контекстно-свободных языков. Настоящий сборник, с. 74-83.

7. ЕВРЕИНОВ Э.В., ЛОПАТО Г.П. Универсальная вычислительная система "Минск-222". -"Вычислительные системы", Новосибирск, 1966, вып. 23, с. 13-20.

8. КОСАРЕВ Ю.Г. Опыт решения задач на системе "Минск-222". -"Труды I Всесоюзной конференции по вычислительным системам". Новосибирск, "Наука" СО, 1968, с. 70-74.

9. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. О решении задач на универсальных системах. -"Вычислительные системы". Новосибирск, 1965, вып. 17, с. 106-164.

10. КОСАРЕВ Ю.Г., НАГАЕВ С.В. О потерях времени на синхронизацию в однородных вычислительных системах. -"Вычислительные системы", Новосибирск, 1967, вып. 24, с. 21-40.

II. КОСАРЕВ Ю.Г. О схемах обмена между ветвями параллельных алгоритмов. -"Вычислительные системы", Новосибирск, 1972, вып.51, с. 70-75.

12. КОСАРЕВ Ю.Г. Распараллеливание по циклам. -"Вычислительные системы". Новосибирск, 1967, вып. 24, с. 3-20.

13. ВЕЛЕСЬКО А.А., КОСАРЕВ Ю.Г., КРЕВИЧ Р.В. Многогрупповой расчет двумерного реактора в диффузионном приближении. -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 15-21.

14. КОСАРЕВ Ю.Г. Примеры использования таблиц для сокращения времени счета. -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 46-54.

15. КОЛОСОВА Ю.И. Комплекс средств производства параллельных программ, -Настоящий сборник, с.98-114.

16. ФАДЕЕВ Д.К., ФАДЕЕВА В.Н. Вычислительные методы линейной алгебры. М. Физматгиз, 1963.

17. ЗАВЬЯЛОВ Ю.С. Экстремальное свойство бикубических многозвенников и задача сглаживания. -"Вычислительные системы", Новосибирск, 1970, вып. 42, с. 109-158.

18. МИРЕНКОВ Н.Н. К решению системы линейных уравнений на ВС. -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 8-11.

19. КАРЦЕВ М.А. Распараллеливание вычислительных алгоритмов итерационного типа. -Вопросы радиоэлектроники, серия ЭВТ, 1971, вып. 9, с. 36-39.

20. ПЕТРОВИЧ А.И., КОСАРЕВ Ю.Г. Исследование колебательных процессов автопоездов на системе "Минск-222". -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 12-14.

21. МАРЧУК Г.И. Численные методы прогноза погоды. М., Гидрометиздат, 1968.

22. МИРЕНКОВ Н.Н. Реализация продольно-поперечных прогонок на ВС "Минск-222". -"Вычислительные системы", Новосибирск, 1968, вып. 30, с. 26-33.

23. KUCK D. ILLIAC IV software and application programming. - "IEEE Trans.Comput.",1968,vol.17, N 8, p.758-770.

24. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г., УСТИНОВ В.А.Исследование рукописей древних майя, т.4, Новосибирск, "Наука", 1969.

25. КОНСТАНТИНОВ В.И., ЛЕВИН Б.И. Быстродействующий метод внутренней сортировки. -"Кибернетика", 1971, № 5, с. 33-34.

Поступила в ред.-изд.отд.
13 декабря 1972 года