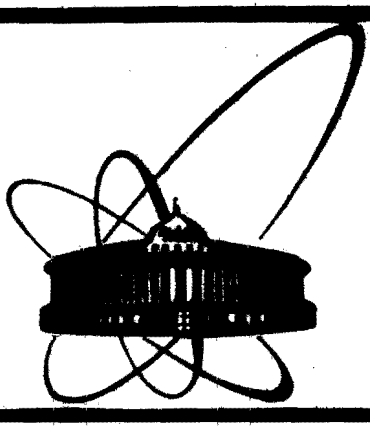


сообщения
объединенного
института
ядерных
исследований
дубна

Р11-87-900

Л.Грегушова

МЕТОДИКА РАСПАРАЛЛЕЛИВАНИЯ ЦИКЛОВ ДЛЯ МУЛЬТИКОНВЕЙЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

1987

ВВЕДЕНИЕ

В настоящее время, когда физический характер элементов ЭВМ препятствует дальнейшему увеличению их быстродействия, ведутся исследования с целью улучшения организации их работы. Большой интерес представляет создание проблемно-ориентированных мультипроцессорных систем высокой производительности. Широкое внедрение мини- и микроЭВМ позволяет создавать самые разнообразные вычислительные системы с архитектурой, определенной конкретным применением.

Существует класс больших задач управления сложными экспериментальными установками по изучению физических объектов, допускающих большую степень распараллеливания. Периодический характер решения этих задач в реальном масштабе времени показывает необходимость реализации конвейерно-параллельной обработки [1,2]. В работах [3,4] определена общая идеология построения мультиконвейерных вычислительных систем (МКВС) на уровне процессоров, по обеспечению высокого темпа обработки информации. Для организации работы такой вычислительной системы необходимо регуляризовать (тактировать) процесс решения заданного класса задач [5].

Неотъемлемой частью идеологии развития МКВС является создание методики проблемной ориентации, предусматривающей регуляризацию заданного класса программ. Однако в общую технологию построения МКВС плохо вкладываются циклические участки программ. При наличии семейства вложенных и итерационных циклов в конвейере может нарушаться его балансировка и производительность [6]. Цель настоящей работы заключается в определении методов распараллеливания циклических участков с учетом особенностей МКВС.

2. ОПРЕДЕЛЕНИЕ КЛАССА МКВС

Вычислительный процесс Φ в первом приближении определяется последовательностью фаз (программных модулей) $\Phi_1, \Phi_2, \ldots, \Phi_m$ и некоторым множеством процессоров $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_n\}$. Для реализации каждого $\Phi_i \in \{\Phi_i\}_{i=1}^{m}$ выделяется группа процессоров.

Каждому распределению по фазам поставим в соответствие вектор $\vec{B} = (b_1, b_2, \ldots, b_n)$, где

$$b_j = \begin{cases} i, & \text{если } \Pi_j \text{ выделен для фазы } \Phi_i \\ \emptyset, & \text{иначе, } j = \overline{1,n},\ i = \overline{1,m}. \end{cases}$$

Тогда всевозможные векторы $\vec{B}$, удовлетворяющие сказанному выше, определяют класс МКВС, построенный из процессоров $П_1, П_2, ..., П_n$ и вычисляющий Ф.

Возможный вид МКВС представлен на рис. 1, где фазы $Ф_i$ $(i = \overline{1,m})$ вычислительного процесса Ф изображены уже в распараллеленном виде, и процессоры $П_j$ $(j = \overline{1,n})$ объединены системной шиной СШ передачи данных. Отсюда следует, что $m \le n$, и если $m = n$, получаем простой конвейер. Пусть $t_{\phi_i} = \max\limits_j \{ t_j \mid b_j = i;\ j = \overline{1,n} \}$ — максимальное время работы процессоров, вычисляющих фазу $Ф_i$, $i = \overline{1,m}$. Обозначим $\tau_{вх}$ заданный темп поступления входной информации в конвейер (такт работы МКВС). Повышение эффективности МКВС будем связывать с условием $\sum\limits_{i=1}^{m} | \tau_{вх} - t_{\phi_i} | \to \min$, причем для $\forall i = \overline{1,m}$ выполнено $t_{\phi_i} \le \tau_{вх}$. Тогда время решения пакета L задач равно

$$T_{конв} = m \cdot \tau_{вх} + (L - 1) \cdot \tau_{вх},$$

и среднее время простоя одного процессора

$$\bar{T}_{прост} = \sum_{j=1}^{n} (\tau_{вх} - t_j) / n.$$

Основным показателем эффективности МКВС будем считать коэффициент увеличения темпа обработки, который можно определить следующим образом:

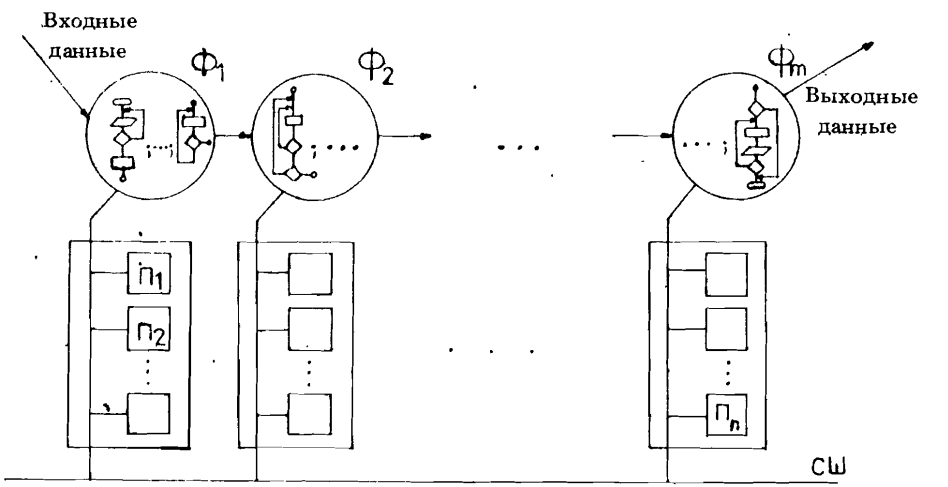


Рис. 1. Функциональная структура МКВС.

$K_{темпа} = \tau_{вх} / \tau_x$, где $\tau_x = \max \{ t_{\phi_i} \mid i = \overline{1,m} \}$.

Достоинствами класса МКВС являются:

— высокий темп обработки информации в реальном масштабе времени;

— простая организация вычислительного процесса большого объема, иначе вынуждающего реализацию на мощной ЭВМ;

— возможность параллельной обработки и обменов информацией;

— обеспечение высокой производительности;

— относительно невысокая стоимость в широком диапазоне мощностей.

3. ЗАДАЧИ ПРОБЛЕМНОЙ ОРИЕНТАЦИИ МКВС

Дано:

$G = (\vec{V}, \vec{U})$ — информационный граф входной программы,

$\tau_{вх}$ — такт поступления входных данных для МКВС,

n — число процессоров, где $\vec{V}$ — множество вершин (программных модулей) и $\vec{U}$ — множество связей между ними.

Допущения:

1. Не учитывается зависимость объема памяти $\vec{M}$ от исходных данных;

2. Затраты времени на обмен информации: $\tau_{об.} \ll \tau_{вх}$.

Задача проблемной ориентации ВС сводится к нахождению такого отображения $\delta : G \to \{G_i\}$, $G_i \subset G$, $i = \overline{1,n}$, чтобы

$$\tau_x \to \min, \quad \sum_{\forall j \in G_i} t_j \le \tau_{вх}; \quad \sum_{i=1}^{n} \nu_i \le M, \quad \bigcup_{i=1}^{n} G_i = G,$$

где ν_i — объем памяти для G_i.

Основными этапами решения являются:

1. Построение граф-схемы $G = (\vec{V}, \vec{U})$ программы (сегментация входной программы) с оценкой вершин $\vec{V}$ и дуг $\vec{U}$ (дуги сначала только управляющие).

2. Описание $Ф(\vec{V})$ каждой вершины графа G на функционально-структурном языке, выявляющем их информационную зависимость.

3. Преобразование и распараллеливание структуры G и получение функциональной граф-схемы G', отображающей выбранный метод решения с новыми оценками вершин (исключение обратных управляющих связей).

4. Сегментация графа G' с учетом $\tau_{вх}$ и получение мультиконвейерной структуры (дуги уже только информационные).

5. Определение необходимого обмена информацией между сегментами.

6. Запись сегментов на языке конвейерной системы, представляющем каждый сегмент в виде программного модуля с включенными операторами обмена информацией между ними.

Последовательность перечисленных этапов показана на рис. 2 и образует методику проблемной ориентации МКВС. Алгоритм методики имеет итерационный характер и работает в диалоговом режиме. Уже при первой итерации после этапов 1, 2 получаем первое приближение к будущей структуре МКВС. Пример конвейеризации простой программы без циклов (рис. 3а) приведен на рис. 3б и 3в, где прямоугольники

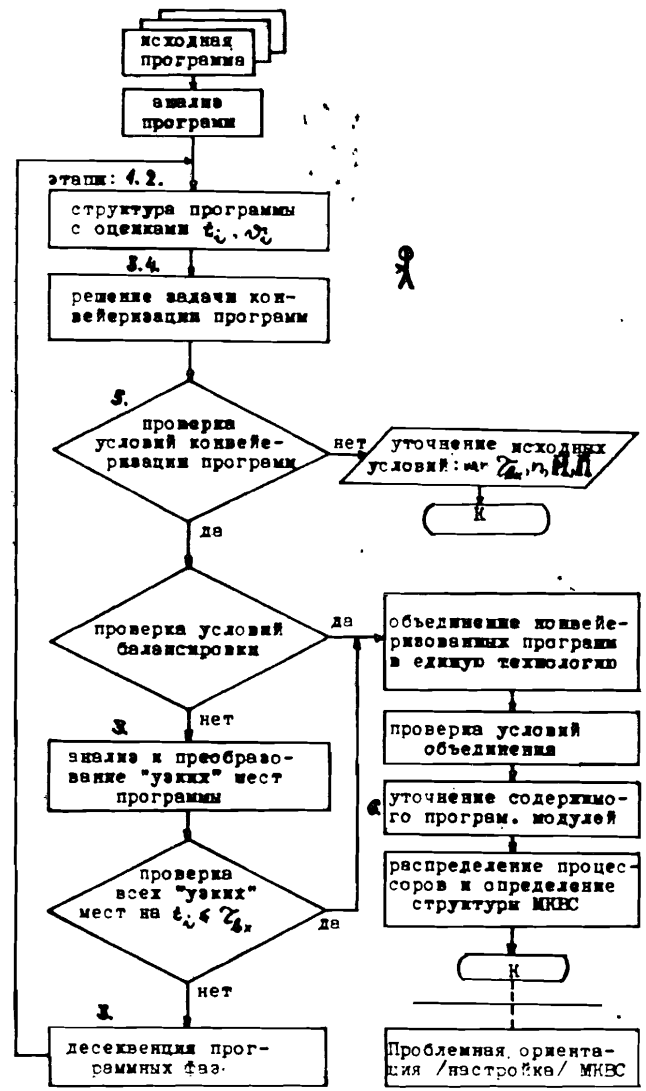


Рис. 2. Методика проблемной ориентации МКВС.

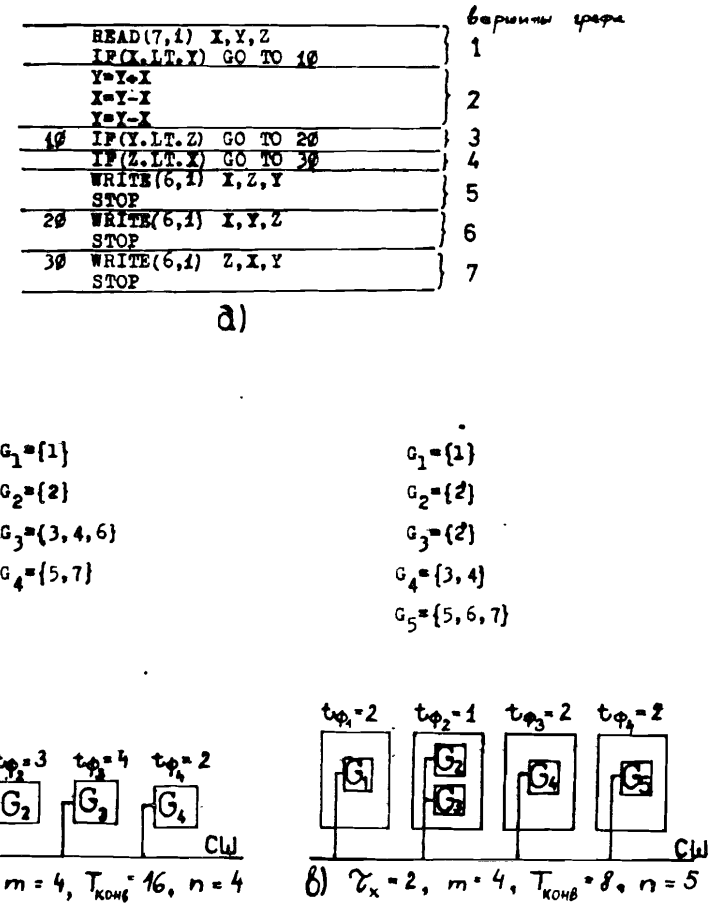


Рис. 3. Пример конвейеризации программы: а) программа без циклов; б) первое приближение структуры МКВС; в) структура МКВС после распараллеливания и преобразования фаз 2-4.

обозначают процессоры, а G_i — программные модули с временем выполнения t_{ϕ_i}. Из рис. 3 видно, что структура МКВС зависит от содержимого программных модулей. За счет распределенной памяти оказалось возможным распараллелить вторую фазу, в которой меняется содержимое переменных X,Y на два модуля 2' (модуль G_2 с оператором X = Y и модуль G_3 с оператором Y = X), реализуемых параллельно.

Если в программе находятся циклические участки, возникает задача их преобразования с целью балансировки МКВС. Основой служит распараллеливание циклов.

4. ПОСТАНОВКА ЗАДАЧИ РАСПАРАЛЛЕЛИВАНИЯ ЦИКЛА

Определим две основных задачи распараллеливания цикла.

I задача

Дано:
```
DO α i1 = H1, r1   }
DO α i2 = H2, r2   }   заголовок цикла
  ⋮                }
DO α in = Hn, rn   }
T(i1, i2, ..., in) }   тело цикла
α  CONTINUE
```

Существует $I = \{(i_1, i_2, \ldots, i_n) \mid H_k \le i_k \le r_k, k = \overline{1, n}\}$ — пространство итераций. Необходимо разбить его на такие подмножества, чтобы все итерации каждого из них можно было выполнять параллельно при соблюдении информационных связей [7]. Это интерциклическое распараллеливание цикла и самое важное ограничение на тело цикла предполагают отсутствие оператора GO TO, передающего управление из тела цикла.

II задача ставится при допущении, что тело цикла задано последовательностью

$T_1(i_1, i_2, \ldots, i_n)$
$T_2(i_1, i_2, \ldots, i_n)$
$\vdots$
$T_q(i_1, i_2, \ldots, i_n)$, которая не содержит условных операторов [16].

Задача (интрациклического) распараллеливания тела цикла сводится к:

а) декомпозиции — нахождению сильно связанных компонент в графе с вершинами $\vec{V} = \{T_1, T_2, \ldots T_q\}$;

б) распараллеливанию (десеквенции) линейного участка, полученного декомпозицией.

Так как нашей целью распараллеливания цикла является сокращение времени его реализации в пределах такта МКВС с учетом обмена нужной информации, возникают проблемы:

1) языкового задания параллельного цикла;

2) включения операций передачи и приема информации, определяющих синхронизацию;

3) восстановления данных после параллельного выполнения отдельных итераций цикла.

С учетом введенных уточнений предлагается методика распараллеливания циклов МКВС, которая базируется на известных методах распараллеливания циклов для задач I и II.

5. АНАЛИЗ МЕТОДОВ РАСПАРАЛЛЕЛИВАНИЯ ЦИКЛОВ

За последние двадцать лет резко возрос интерес к распараллеливанию циклов как к важному источнику параллелизма в вычислительном процессе. Методы распараллеливания линейных участков программ для циклов неприменимы. Разработан ряд методов, ориентированных на различные структуры вычислителей и накладываемых на циклы более или менее жесткие ограничения [7–9]. Самым важным ограничением является требование, чтобы все индексные выражения переменных тела цикла имели определенную форму линейной функции параметров цикла. Иначе задача соответствия индексной переменной одному и тому же элементу массива является NP-полной задачей целочисленного программирования. Таким образом каждый из методов ограничивает класс распараллеливаемых циклов. Это указывает на необходимость создания методики распараллеливания циклов с учетом особенностей МКВС, так как при нераспараллеливаемости цикла одним методом надо найти способ его преобразования и вложения в структуру МКВС.

Отметим методы, представляющие для нас интерес: метод гиперплоскостей (МГ), метод координат (МК), метод параллелепипедов (МПа), метод линейных преобразований (МЛП) и метод пирамид (МПи). Названия методов соответствуют геометрической интерпретации подмножеств пространства итераций, которые могут выполняться параллельно. Для МГ [13] такие подмножества являются гиперплоскостями пространства итераций с определенной последовательностью асинхронного выполнения параллельных итераций по новой координатной оси, в общем случае отличающейся от координатных осей пространства итераций. Для МК [13] осью параллельного выполнения гиперплоскостей является всегда одна из координатных осей пространства итераций, что требует синхронизации выполнения параллельных итераций, а иногда и преобразования тела цикла. МПа [10] ищет максимальные прямоугольные n-мерные параллелепипеды, на которые "режет" пространство итераций. Размер параллелепипеда определяет количество параллельно работающих процессоров. МЛП [14, 15] находит подмножества пространства итераций для общей линейной функции от параметров цикла, чем расширяет ограничения при практической реализации МПа, который является естественным обобщением МГ. МПи [11, 12] предлагает параллельные пирамиды, содержащие все информационно зависимые друг от друга итерации, для каждой результирующей итерации. Полученные автономные ветви, не обменивающиеся информацией, могут дублировать вычисления.

Анализ возможностей применения этих методов для МКВС показал следующие ограничения и допущения, которые определили место расположения метода в общей методике распараллеливания циклов:

— метод предназначен для матричных и векторных процессоров (касается методов МГ, МК, МЛП);

— методу нужна поитерационная синхронизация (МПа, МК);

— метод учитывает распределение памяти (МПи);

— полученные параллельные ветви не будут одинаковой длины (МПи, МГ, МЛП);

— необходимость включения в транслятор распознавателя языковой формы задания параллельного цикла (МГ, МЛП, МПи);

— трудности при включении в распараллеленный цикл операторов обмена информацией (МГ, МК, МЛП);

— довольно сложный математический аппарат для определения параллельного цикла (МГ, МК, МЛП, МПа — без ограничения на линейность индексных выражений).

6. МОДИФИКАЦИЯ МЕТОДОВ РАСПАРАЛЛЕЛИВАНИЯ ЦИКЛОВ

В основу положена модификация метода параллелепипедов (ММПа). Она предназначена для простых циклов, которые не поддаются распараллеливанию МПа (т.е. существует не менее одной итерации, информационно зависимой от предыдущей). Для каждой пары одноименных индексных переменных, из которых одна является выходной, предлагается применить алгоритм построения таблицы зависимых итераций, определяющий расстояние зависимости p и список зависимых пар итераций.

Пусть x — индексная переменная и $a * i + b$ — линейная форма индексного выражения, зависящая от параметра цикла i (a, b — целочисленные константы или переменные). Входными данными предлагаемого алгоритма являются a, b, a1, b1 — коэффициенты индексов одноименных переменных x, H, r — границы цикла и h_p — ширина распараллеливания (ограничение на число процессоров). Начиная с расстояния $p = 1$ (по $p \le h_p$), находятся такие целочисленные пары итераций (k, ℓ), что или $k = \frac{d - p}{1 - c}$ и тогда $\ell = k + p$, или $k = \frac{d + p}{1 - c}$ и $\ell = k - p$.

Коэффициенты c, d определены следующим способом: $c = \frac{a}{a1}$ и $d = \frac{b - b1}{a1}$, но если $d < 0$, то $d = \frac{b1 - b}{a}$, $c = \frac{a1}{a}$ и логическая переменная bool = TRUE. Пары итераций (k, ℓ) могут быть связаны информационной или конкуренционной зависимостью.

Пусть In(k), Out(k) — входные и выходные наборы переменных k-той итерации соответственно. Тогда, если $Out(k) \cap In(\ell) \neq \emptyset$ и $k < \ell$, то пара итераций (k, ℓ) зависит информационно; если $\ell < k$, то пара итераций зависит конкуренционно. В вычислительной системе с общей памятью оба вида зависимости препятствуют распараллеливанию. Распределенная память МКВС позволяет конкуренционно-зависимые итерации простых циклов вычислять параллельно. Конкуренционная зависимость итераций вложенных циклов превращается в информационную, и для параллельного выполнения таких итераций необходимо обеспечить обмен информацией. Оба вида зависимостей формируются в таблицу зависимых итераций. Эта таблица состоит из двух

частей. Длину каждой части обозначим соответственно переменными J_1, J_2. Если $d = \frac{b - b1}{a1}$, то есть bool = FALSE, то J_1 строк списка представляют информационно-зависимые пары итераций (т.е. те, которые представляют для нас интерес) и J_2 строк — конкуренционно-зависимые. Если $d = \frac{b1 - b}{a}$, то J_2 информационно-зависимых пар итераций, расположенных после J_1 строк конкуренционно-зависимых пар. Таким образом, алгоритм строит $J_1 + J_2$ строк ($J_1, J_2 \geq 0$) таблицы зависимостей со столбцами P, K, L, в которых занесены расстояние зависимости p и зависимая пара (k, ℓ) итераций цикла. Алгоритм представлен в таблице.

Таблица

Алгоритм построения таблицы зависимостей итераций простого цикла

Начало "построения таблицы зависимых итераций"

1. Ввод коэффициентов a, b, a1, b1 одноименных переменных, границ цикла H, r и ширины распараллеливания h_p;
2. Строка таблицы j := 1;
3. Расстояние зависимости p := 1;
4. $c := \frac{a}{a1}$; $d := \frac{b - b1}{a1}$; bool := FALSE;
5. Если $d \geq 0$, то к п.6, иначе — начало $d := \frac{b1 - b}{a}$; $c := \frac{a1}{a}$; bool := TRUE конец;
6. $k := \frac{d - p}{1 - c}$; $\ell := k + p$;
7. Если (k, ℓ) являются зависимой парой (т.е. $k = [k]$ и $H \leq k, \ell < r$, где [k] — целое значение, не превосходящее k), то начало запомним p, k, ℓ в списках P_j, K_j, L_j соответственно; j := j + 1 конец;
8. p := p + 1;
9. Если $p \leq h_p$, то к п.6, иначе — к п.10;
10. $J_1 := j - 1$ (количество элементов первого списка);
11. p := 1;
12. $k := \frac{d + p}{1 - c}$; $\ell := k - p$;
13. Если (k, ℓ) является зависимой парой, то начало запомним p, k, ℓ в P_j, K_j, L_j соответственно; j := j + 1 конец;
14. p := p + 1;
15. Если $p \leq h_p$, то к п.12, иначе — к п.16;
16. $J_2 := j - 1$ (количество элементов второго списка);

Продолжение таблицы

17. Если bool , список имеет J_2 информационно-зависимых пар итераций, иначе — в списке J_1 информационно зависимых пар итераций;
18. Вывод всех P_i, K_i, L_i для $i = 1, 2, ..., J_1 + J_2$ конец.

На основе предложенного алгоритма нахождения зависимостей разработана методика распараллеливания циклов, позволяющая решить задачи I и II. ММПа решает задачу I и затем проектирует прямо фазы МКВС. Исходный цикл расщепляется на такие две части, из которых первая не распараллеливается (или слабо распараллеливается), а вторая может быть реализована параллельно. Критерием выбора структуры МКВС и определением количества итераций первой фазы служит требование расщеплять цикл на такие фазы, время вычисления которых не превосходит $\tau_{вх}$. Алгоритм выбора структуры МКВС для заданного простого (не вложенного) цикла имеет следующие этапы:

1. Построение таблицы зависимостей итераций, расстояние которых не превосходит заданное h_p.
2. Определение числа итераций первой фазы с учетом требования балансировки фаз.
3. Распараллеливание первой фазы, если это допустимо.
4. Распараллеливание второй фазы расщепленного цикла с минимальным значением p для всех зависимых пар индексных переменных.
5. Уточнение содержимого программных модулей с включением операторов обмена информацией.

Данный подход обеспечивает более высокую степень параллельной обработки цикла и лучшую загрузку ресурсов МКВС. Приведем пример модификации метода параллелепипедов.

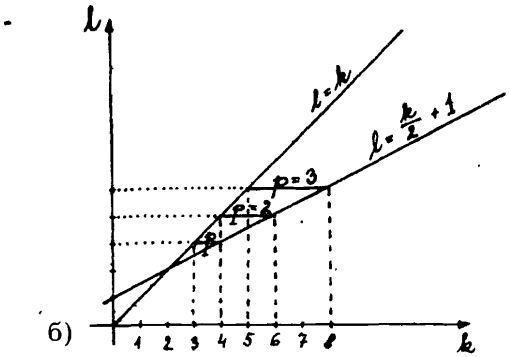
Пусть задан цикл

```
      DO   1   I = 1, 15
                                                    (1)
1     X(2 * I) = X(I + 2),
```

который не распараллеливается методом МПа. Входными данными для ММПа будут a = 2, b = 0, a1 = 1, b1 = 2, H = 1, r = 15 и пусть $h_p = 8$. Тогда $c = \frac{1}{2}$, d = 1, bool = TRUE , $J_1 = 0$ и $J_2 = 6$. Таблица информационно-зависимых итераций цикла дана на рис. 4а. Графическое нахождение зависимости /10/ показано на рис. 4б. На рис. 5 даны возможные структуры МКВС, полученные методом ММПа. Если $T_{посл} = 15$ единиц условного времени, то t_{ϕ_1} , t_{ϕ_2} представляют время выполнения первой и второй формируемой фазы соответственно. Выбор оптимальной структуры зависит от $\tau_{вх}$.

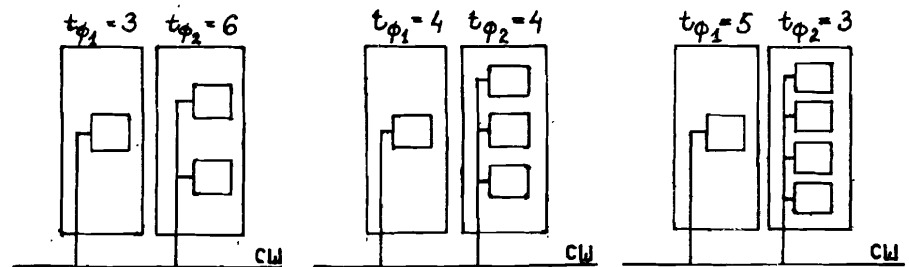
p	k	l
1	4	3
2	6	4
3	8	5
4	10	6
5	12	7
6	14	8

а)



б)

Рис. 4. а) Таблица зависимых пар итераций (k , ℓ); *б) Графический вид зависимости итераций цикла* (I) *с расстоянием зависимости* p.



а) б) в)

Рис. 5. Возможные структуры МКВС для цикла (I).

7. МЕТОДИКА РАСПАРАЛЛЕЛИВАНИЯ ЦИКЛОВ

Эта методика основана на нахождении зависимостей между итерациями цикла. Для решения задачи II необходимо знать информационные и конкуренционные зависимости между операторами тела цикла. Поэтому предлагается приведенный в таблице алгоритм нахождения зависимостей использовать в качестве подпрограммы, которую надо вызывать для каждой индексной позиции одноименных переменных (на языке фортран их максимально три). Зависимость итераций определяется сначала по всем индексным позициям, а затем по минимальному расстоянию всех зависимых пар переменных.

Необходимость обеспечения высокого темпа обработки на МКВС приводит к требованию такого распараллеливания циклических участков, которое гарантировало бы эффективную обработку исходной

программы. Поэтому была предложена методика распараллеливания циклических участков с учетом особенностей МКВС (распределенная память, обмен информацией и т.д.).

На рис. 6 представлена блок-схема разработанной методики, в которой используются следующие обозначения: n — степень вложенности цикла (для простого цикла n = 1); H_1, r_1, H_2, r_2, ... H_n, r_n — нижние и верхние границы цикла каждого уровня вложенности соответственно; rr — число пар связующих переменных в теле цикла, через которые могут возникнуть информационные и конкуренционные зависимости; q — число операторов в теле цикла; $\tau_{вх}$ — время такта конвейера, соизмеримое с частотой поступления входной информации на МКВС; h_p — верхняя граница распараллеливания; t' — время вычисления одной итерации; $t^{(1)}$, $t^{(2)}$ — свободные времена проектируемых фаз МКВС; a_{ij}, b_{ij}, $a1_{ij}$, $b1_{ij}$ — коэффициенты i-той позиции индексов j-той пары связующих переменных, находящихся в операторах $O1_j$, $O2_j$ соответственно, причем $i = \overline{1,n}$; $j = \overline{1,rr}$ и $O1_j$, $O2_j \in \{1, 2, ..., q\}$. Рассмотрим способ распараллеливания в отдельных моделях предлагаемой методики.

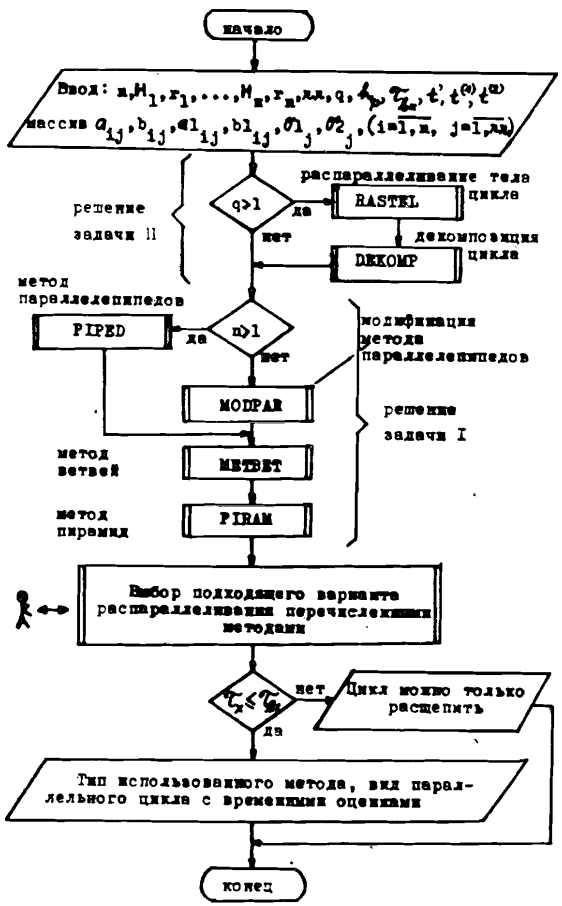


Рис. 6. Блок-схема методики распараллеливания циклических участков программ.

1. RASTEL — данный модуль предназначен для распараллеливания тела цикла в смысле решения задачи II на q параллельных ветвей, без потребности осуществления обмена информацией между ветвями.

2. DEKOMP — в этом модуле решается задача II и связанная с ней декомпозиция цикла. В отличие от предыдущего модуля, здесь возможно проектирование параллельно-последовательных фаз МКВС на основе результатов декомпозиции, причем учитываются необходимые обмены информацией между ними.

3. PIPED — этот модуль распараллеливает вложенные циклы методом параллелепипедов. С использованием введенных выше обозначений основные шаги метода заключаются в следующем:

1) i: = 1;
2) j: = 1;
3) ввод коэффициентов a_{ij}, b_{ij}, $a1_{ij}$, $b1_{ij}$ для j-той пары связующих переменных;
4) вызов "Построение таблицы зависимых итераций";
5) определение минимальной информационной зависимости $p^j_{инф}$ между итерациями цикла для данного j;
6) определение обмена информацией;
7) j: = j + 1; если $j \le rr$, то переход к п.3;
8) $p_i := \min_j \{p^j_{инф} \mid j = 1, 2, ..., rr\}$;
9) i: = i + 1; если $i \le n$, то переход к п.2;
10) из $\{p_i\}_{i=1}^{n}$ определим размер параллелепипеда и уточним нужные обмены информацией;
11) конец.

Таким образом получаем параллелепипед, размером которого является $\prod_{i=1}^{n} p_i$, определяющее число параллельно работающих процессоров.

4. MODPAR — модуль, реализующий модификацию метода параллелепипедов для простых циклов, причем возможно проектирование фаз МКВС способом, описанным в разделе 6 (ММПа) настоящего сообщения.

5. METBET — данный модуль реализует идеи ММПа для вложенных циклов. На основе информации об информационных и конкуренционных связях в цикле осуществляется сбор итераций в параллельные ветви, отличающиеся лишь значением шага в заголовке цикла. Формирование ветвей происходит с учетом требуемых обменов информацией между ветвями, причем значение шага выбирается таким об-

разом, чтобы минимизировать число обменов между ветвями. Метод применим и для простых циклов.

6. PIRAM — здесь формируются параллельные ветви в виде пирамид, причем в одной ветви находятся все итерации, информационно зависимые друг от друга. Сборка в пирамиды осуществляется очень просто на основе таблицы зависимостей отдельных итераций. Таким образом устранены все обмены между ветвями.

Выбор подходящего варианта структуры МКВС происходит в модуле METRAS. Может оказаться, что цикл не распараллеливается ни одним из данных методов. В таком случае необходимо расщепление цикла на две (по крайней мере) последовательные фазы. При вложенных циклах надо расщеплять самый внешний цикл.

В методику не вошли методы МГ, МК, МЛП, которые без модификации нельзя применить для МКВС. При модификации этих методов необходимо решить две основные задачи. Во-первых, определить языковое задание распараллеленного цикла, учитывающее реальные возможности компонентов МКВС. Во-вторых, включить операции обмена информацией.

8. ОБЛАСТЬ ПРИМЕНЕНИЯ РАЗРАБОТАННОЙ МЕТОДИКИ

Предложенная методика позволяет существенно расширить класс задач, поддающихся реализации на МКВС. Однако часть характерных для теории параллельного программирования ограничений не снимается. Такими ограничениями являются отсутствие оператора условного перехода из тела цикла, требования индексности выходных переменных цикла и линейной формы индексных выражений.

В настоящее время методика реализована программно на языке Фортран в виде диалогового комплекса из восьми модулей, осуществляющих распараллеливание циклов по пп.1-6.

Методика находит применение при создании МКВС, ориентированных на решение задач управления режимами энергетических систем.

Автор выражает искреннюю благодарность И.И.Дзегеленку за общую постановку проблемы и оказанную в процессе работы конструктивную помощь, а также Е.П.Жидкову и Г.А.Ососкову за поддержку и внимание к работе на этапе ее завершения.

ЛИТЕРАТУРА

1. *Agrawal D.P., Jain R. A pipelined pseudoparallel system architecture for real-time dynamic scene analysis. – IEEE Trans. on Comput. 1982, vol.31, No.10, p.952.*
2. *Валях Е. Последовательно-параллельные вычисления: Пер. с англ. – М.: Мир, 1985.*
3. *Дзегеленок И.И. Мультиконвейерные вычислительные системы на базе микро-ЭВМ (Под ред. А.К.Полякова).– М.: Изд.Моск.энергет.ин-та, 1985.*
4. *Дзегеленок И.И., Волков Е.Г., Брежнева О.А. Потоковая обработка больших задач на сетях ВЦ. – В кн.: VI Всесоюзная школа-семинар по вычислительным сетям. Изд.АН СССР и Винницкого политехнического института, Москва-Винница, 1981, ч.2, с.9.*
5. *Дзегеленок И.И. и др. Регуляризация взаимодействия микро-ЭВМ в вычислительной системе конвейерного типа. – В кн.: "Применение микропроцессоров в измерительной технике и управлении". М.: изд. МЭИ, 1987, с.39.*
6. *Дзегеленок И.И. и др. Режимы параллельной обработки информации в мультиконвейерных ВС. – В кн.: V Всесоюзная школа-семинар "Распараллеливание обработки информации". Тезисы докладов и сообщений. Львов, Изд. Физико-механического ин-та им.Г.В.Карпенко АН УССР, 1985, ч.3, с.247.*
7. *Элементы параллельного программирования. (Под. ред. В.Е.Котова) – М.: Радио и связь, 1983.*
8. *Параллельная обработка информации. Том I: Распараллеливание обработки информации. (Под ред.А.Н.Свенсона.) – Киев: Наук.думка, 1985.*
9. *Вальковский В.А., Котов В.Е. Автоматическое построение параллельных программ. Распараллеливание выражений и циклов. – Новосибирск, 1979, препринт ВЦ СО АН СССР: № 146.*
10. *Вальковский В.А. Параллельное выполнение циклов. Метод параллелепипедов. – Кибернетика, 1982, № 2, с.51.*
11. *Вальковский В.А. Параллельное выполнение циклов. Метод пирамид. – Там же, 1983, № 5, с.51.*
12. *Вальковский В.А. Распараллеливание циклов общего вида методом пирамид. Там же, 1985, № 4, с.16.*
13. *Lamport L. The Parallel Execution of DO Loops. – Commun. of the ACM, 1974, vol.17, No.2, p.83.*
14. *Бабичев А.В., Лебедев В.Г. Распараллеливание программных циклов. – Программирование, 1983, № 5, с.52.*
15. *Трахтенгерц Э.А. Влияние архитектуры и структуры многопроцессорных вычислительных машин на языки программирования и методы трансляции. – Автоматика и телемеханика, 1986, № 3, с.5.*
16. *Kuck D.J., Muraoka Y., Chen S.-C. On the number of operations simultaneously executable in FORTRAN-like programs and their resulting speedup. – IEEE Trans. on Comput.1972, vol.C-21, No.12, p.1293.*
17. *Padua D.A., Kuck D.J., Lawrie D.H. High-speed multiprocessors and compilation techniques. – IEEE Trans. on Comput., 1980, vol.C-29, No.9, p.763.*

Рукопись поступила в издательский отдел
23 декабря 1987 года.

НЕТ ЛИ ПРОБЕЛОВ В ВАШЕЙ БИБЛИОТЕКЕ?

Вы можете получить по почте перечисленные ниже книги, если они не были заказаны ранее.

Д7-83-644	Труды Международной школы-семинара по физике тяжелых ионов. Алушта, 1983.	6 р.55 к.
Д2,13-83-689	Труды рабочего совещания по проблемам излучения и детектирования гравитационных волн. Дубна, 1983.	2 р.00 к.
Д13-84-63	Труды XI Международного симпозиума по ядерной электронике. Братислава, Чехословакия, 1983.	4 р.50 к.
Д2-84-366	Труды 7 Международного совещания по проблемам квантовой теории поля. Алушта, 1984.	4 р.30 к.
Д1,2-84-599	Труды VII Международного семинара по проблемам физики высоких энергий. Дубна, 1984.	5 р.50 к.
Д10,11-84-818	Труды V Международного совещания по проблемам математического моделирования, программированию и математическим методам решения физических задач. Дубна, 1983.	3 р.50 к.
Д17-84-850	Труды III Международного симпозиума по избранным проблемам статистической механики. Дубна, 1984./2 тома/	7 р.75 к.
Д11-85-791	Труды Международного совещания по аналитическим вычислениям на ЭВМ и их применению в теоретической физике. Дубна. 1985.	4 р.00 к.
Д13-85-793	Труды XII Международного симпозиума по ядерной электронике. Дубна, 1985.	4 р.80 к.
Д4-85-851	Труды Международной школы по структуре ядра. Алушта, 1985.	3 р.75 к.
Д3,4,17-86-747	Труды V Международной школы по нейтронной физике. Алушта, 1986.	4 р.50 к.
	Труды IX Всесоюзного совещания по ускорителям заряженных частиц. Дубна, 1984. /2 тома/	13 р.50 к.
Д1,2-86-668	Труды VIII Международного семинара по проблемам физики высоких энергий. Дубна, 1986. /2 тома/	7 р.35 к.
Д9-87-105	Труды X Всесоюзного совещания по ускорителям заряженных частиц. Дубна, 1986. /2 тома/	13 р.45 к.
Д7-87-68	Труды Международной школы-семинара по физике тяжелых ионов. Дубна, 1986	7 р.10 к.
Д2-87-123	Труды Совещания "Ренормгруппа-86". Дубна, 1986	4 р.45 к.

Заказы на упомянутые книги могут быть направлены по адресу: 101000 Москва, Главпочтамт, п/я 79. Издательский отдел Объединенного института ядерных исследований.

Грегушова Л.
Р11-87-900

Методика распараллеливания циклов для мультиконвейерных вычислительных систем

Описывается методика распараллеливания циклических участков программ для мультиконвейерных вычислительных систем (МКВС), обеспечивающих высокий темп обработки информации. Методика основана на нахождении информационных и конкуренционных зависимостей всего цикла с учетом особенностей построения МКВС, в частности, распределенной памяти.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1987

Перевод автора

Gregušová L.
P11-87-900

A Methodology for Parallelizing Loops in Multipipelined Computer Systems

A methodology for parallelizing loop segments of programs for multipipelined computer systems (MPCS) enabling a high rate of information processing is described. It is based on finding the data dependencies and the competitive dependencies of the whole loop with respect to the features of the construction of MPCS, in particular, the distributed memory.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1987