АРХИТЕКТУРА И МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1984 год Выпуск 104

УДК 681.32:519.68

ПАРАЛЛЕЛЬНАЯ ИНТЕРПРЕТАЦИЯ ОДНОЙ ЗАДАЧИ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Л.А. Седухина

С развитием технологии сверхбольших интегральных схем, позволяющих создать большие параллельные системы, становится актуальной задача распараллеливания последовательных алгоритмов. Для многих задач линейной алгебры, сортировки, обработки сигналов разработаны параллельные алгоритмы, которые реализуются на вычислительных структурах, отвечающих требованиям интегрального исполнения: процессоры соединены регулярной, локальной и по возможности планарной сетью связи, широко используется конвейеризация и параллелизм при обработке данных [1,2].

Существуют формальные процедуры синтеза вычислительных структур [3], непосредственно реализующих заданный алгоритм, однако в большинстве случаев выявление соответствующей вычислительной структуры получается неформальным анализом алгоритма.

В работе рассматривается параллельная интерпретация задачи динамического программирования на сети регулярно связанных вычислителей. Показывается, что задача динамического программирования - задача распределения ресурса, характеризующаяся n переменными и m точками дискретизации ресурса, решается на системе из n либо m линейно связанных вычислителей за время $O(m^2)$ либо $O(mn)$ соответственно, а на системе из $(m \times n)$ ортогонально связанных вычислителей - за время $O(m+n)$. Последовательное решение задачи требует времени $O(nm^2)$.

Рассмотрим задачу нелинейного программирования [4]:

$$R_n(x_1, x_2, \ldots, x_n) = \max \sum_{i=1}^{n} g_i(x_i), \qquad (1)$$

где $g_i(0) = 0$, $g_i(x_i) \geq 0$ по всей области $S_n(x) = \{(x_1, \ldots, x_n) / \sum_{i=1}^{n} x_i = x,\ x_i \geq 0\}$. Эту задачу можно понимать как задачу распределения, в которой x – ограниченный источник сырья, x_i – количество сырья, распределяемое i-му способу обработки, $g_i(x_i)$ – доход от i-го способа обработки x_i единиц сырья. Требуется определить оптимальную стратегию распределения, при которой общий доход (I) будет максимальным.

Табличная техника динамического программирования состоит в пошаговом построении последовательности функций состояния $f_1(x)$, $f_2(x)$, ..., $f_n(x)$ с помощью рекуррентных соотношений $f_k(x) = \max_{S_k(x)} [R_k(x_1, \ldots, x_n)]$ и последовательности функций управления $\bar{x}_1(x), \ldots, \bar{x}_n(x)$. Это прямой ход. На обратном ходе отыскиваются оптимальные значения $x_n^* = \bar{x}_n(x)$, $x_{n-1}^* = \bar{x}_{n-1}(x - x_n^*)$, $x_{n-2}^* = \bar{x}_{n-2}(x - x_n^* - x_{n-1}^*)$ и т.д.

Для задачи (I) начальные условия определяются из:

$$f_0(x) = 0,\ f_1(x) = \max_{0 \leq x_1 \leq x} g_1(x),\ f_k(x) = \max_{0 \leq x_k \leq x} [g_k(x_k) + f_{k-1}(x - x_k)].$$

Интервал $[x_0, x]$ разбивается на m участков с шагом Δ, в этом случае функцию состояния на шаге k можно представить в зависимости от номера участка j:

$$f_k(x_0 + j\Delta) = \max_{0 \leq i \leq j} [g_k(x_0 + i\Delta) + f_{k-1}(x_0 + (j-i)\Delta)].$$

Обозначим для удобства выражение в квадратных скобках

$$F_k(i, j) = g_k(x_0 + i\Delta) + f_{k-1}(x_0 + (j-i)\Delta),\ f_k(x_0 + j\Delta) = f_k(j),$$

тогда функция состояния на шаге k определится значениями

$$f_k(j) = \max_{0 \leq i \leq j} F_k(i, j), \qquad (2)$$

где $j \in \{0, m-1\}$.

Оценим операционную сложность алгоритма. В качестве критерия оценки алгоритмов рассматривают порядок роста необходимых для решения задачи времени и емкости памяти при увеличении входных данных [5]. Мерой количества входных данных задачи распределения, называемой размером задачи, являются параметры задачи: n – число переменных и m – число участков.

Если вес операции нахождения значения $F_k(i,j)$ при определенных значениях i,j принять равным единице, то определение $f_k(j)$ при одном значении j потребует $(j+1)$ единичных операций. На построение функции состояния для всех участков потребуется $\sum_{j=0}^{m-1}(j+1) = \frac{m(m+1)}{2}$ операций. Таким образом, операционная сложность прямого хода при определении n функций состояния определится величиной $\frac{nm(m+1)}{2} = O(nm^2)$.

Обратный ход требует при определении x_i^*, $i \in \{1,n\}$, $O(n)$ операций выборки из памяти, в которой хранятся таблицы $\bar{x}_i(x)$.

Очевидно, что основную долю затрат составляет прямой ход. Рассмотрим его параллельную интерпретацию.

Вычисление значений функций $F_k(i,j)$, $k \in \{1,n\}$, на всех участках $j \in \{0,m-1\}$ для значений i, удовлетворяющих условию $l = 1+j-i=\text{const}$, можно представить графом информационной зависимости $G_l = \{F_k(i,j), v\}$. Вершинами графа являются операторы, вычисляющие значения функций $F_k(i,j)$, а дуги v показывают связь функций по обработке общих данных или использование последующими операторами результатов предыдущих операторов. Так как $0 \le j \le m-1$, $0 \le i \le j$, то $1 \le l \le m$ и вычисление всех значений $F_k(i,j)$, необходимых для определения $f_k(j)$ по формуле (2) на прямом ходе, можно представить множеством графов $G = \{G_l : 1 \le l \le m\}$, показанных на рис.I для случая $m = 4$, $n = 5$.

Горизонтальные дуги в графе G_l показывают, что операторы $F_k(0,l-1)$, $k \in \{1,n\}$, используют результат соседнего оператора слева $f_{k-1}(l-1)$. Вертикальные дуги в графе G_l связывают в цепочку операторы $F_k(j,j+l-1)$, $j \in \{0,m-1\}$, для фиксированного k использованием общих операндов $g_k(l-1)$ и $f_{k-1}(l-1)$. Так как $f_0(l-1) = 0$ по условию задачи, то оператор $F_1(0,l-1)$ в графе G_l, $l \in \{1\ m\}$, является начальным, независимым от других операторов.

Видно, что операторы $F_k(i,j)$ в графе G_l, входящие в подмножество $W_d = \{F_k(i,j): (k \ne j-l) = d = \text{const}, 1 \le k \le n, 0 \le j \le m-1\}$, т.е. отстоящие на одинаковом расстоянии $d = 1,2,\ldots, d_{max}^l$ от начального оператора $F_1(0, l-1)$, являются взаимно независимыми и могут выполняться одновременно. На рис.I подмножества W_d выделены пунктирными линиями, каждая линия помечена расстоянием d. Для

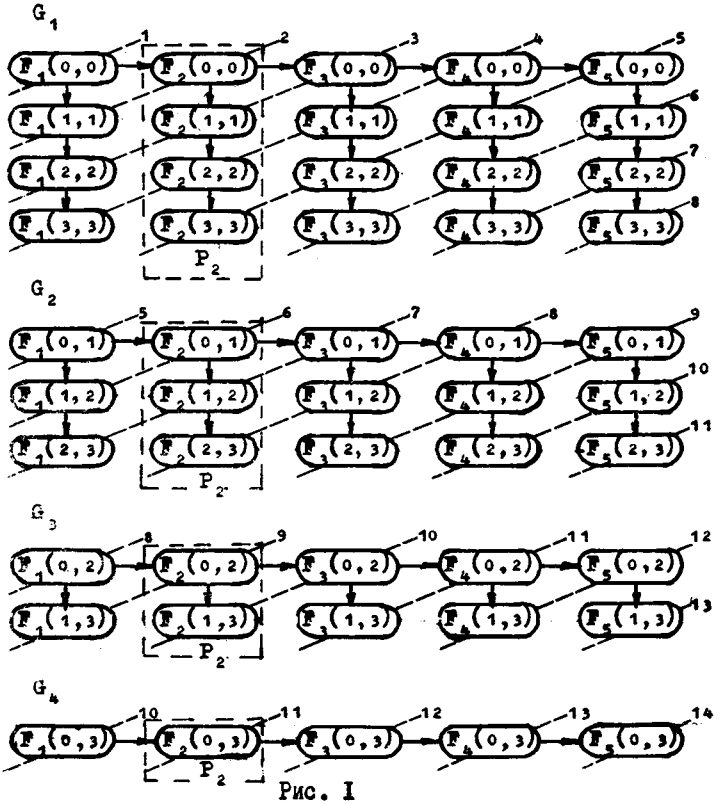

Рис. I

графа G_1 максимальное расстояние определяется величиной $d^1_{max} = n+m-1-1$.

Полученное множество вычислительных графов G можно отобразить на сеть из n линейно связанных вычислителей, показанную на рис.2, так, что в памяти вычислителя P_k будут храниться или значения функции дохода $g_k(x_k)$ для всех m дискретных точек в интервале $[x_0, x]$, или программа, по которой можно определить значения $g_k(x_k)$ для любой точки в интервале. Назначим вершины графов $G_1\{F_k(i\ j)\colon 1-1 \le j \le m-1,\ i = j - 1+1\}$ (вертикальные цепочки операторов) для выполнения вычислителю P_k (как показано на рис.I). В

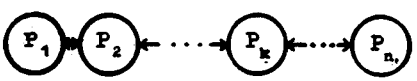

Рис. 2

этом случае горизонтальные дуги в графе G_1 будут соответствовать передаче данных между вычислителями, а вертикальные дуги - временной последовательности реализации операторов в одном вычислителе.

Если принять время выполнения обобщенной операции: прием операнда, определение значения $F_k(i, j)$, передача операнда соседнему вычислителю, равным единице, то временная сложность t вычисления G_1 будет изменяться в интервале $t_1 \le t \le t_1 + d^1_{max}$, где t_1 - время вычисления начального оператора в графе G_1. Одна обобщенная операция соответствует переходу от одной вершины в графе к другой.

Удобно считать, что вычисления, представленные графом G_1, связаны с распространением в сети волнового фронта вычислений l, которому соответствует на каждом временном шаге d выполнение подмножества операторов W_d в графе G_1. Крайний левый вычислитель (рис.2) в моменты времени $t_1 = 1, m+1, \ldots, m(l-1) - \frac{l(l+1)}{2}+2l, \ldots$ $\ldots, \frac{m(m+1)}{2}$ инициирует волну вычислений l, посылая результат вычисления соседу (и тем самым активизируя его), который, в свою очередь, активизирует своего соседа и волна распространяется далее до конца цепочки вычислителей. С каждым фронтом l в вычислителе P_k осуществляется вычисление значения функции состояния в точке $j = l-1$ и нахождение значений $F_k(i,j)$ для точек $l \le j \le m-1$, $i = j-l+1$. Ясно (см.рис.I), что при таком распределении операторов по вычислителям подмножества W_d взаимно независимых операторов будут выполняться на разных вычислителях сети одновременно.

Временная сложность прямого хода алгоритма на сети из n вычислителей будет определяться временем инициации последней m-й волны вычислений и максимальным расстоянием ее распространения по сети, т.е. составит величину

$$T_n(n,m) = t_m + d^m_{max} = \frac{m(m+1)}{2} + (n-1) = m^2/2 + O(m+n) .$$

Покажем, что такая линейная вычислительная сеть выполняет алгоритм правильно. Для этого достаточно показать, что все переменные, необходимые для вычисления математических выражений в вычислителе P_k, в нужный момент времени подаются на вход соответствующего вычислителя.

Отметим, что значение функции состояния $f_k(j)$ определяется в P_k на фронте $j+1$, который проходит через P_k в момент времени $t_{j+1} + k$.

Известно, что для определения $f_k(j)$ необходимо иметь значения $f_{k-1}(0), f_{k-1}(1), \ldots, f_{k-1}(j)$ и $g_k(0), g_k(1), \ldots, g_k(j)$. Значения $g_k(0), g_k(1), \ldots, g_k(j)$ находятся в памяти вычислителя k.

Так как волновые фронты вычислений не пересекаются, а идут один за другим, то через P_k последовательно проходят фронты с номерами $1, 2, \ldots, j+1$. Каждый фронт l для P_k связан с определением значения $f_k(l-1)$, приемом от P_{k-1} значения $f_k(l-1)$ и передачей P_{k+1} значения $f_k(l-1)$ в момент времени $t_l + k$. Таким образом, при прохождении фронтов $l = 1, 2, \ldots, j+1$ через P_k пройдут также значения $f_{k-1}(0), f_{k-1}(1), \ldots, f_{k-1}(j)$ в моменты времени $t_1+k, t_2+k, \ldots, t_{j+1}+k$, т.е. все необходимые значения для вычисления $f_k(j)$ к временному шагу $t_{j+1} + k$.

При реализации задачи распределения на системе с $p < n$ вычислителями в каждый вычислитель назначается n/p функций дохода и вес единичного временного шага увеличивается в n/p раз. Временная сложность прямого хода в этом случае оценится величиной

$$T_p(n,m) = (p-1) + \frac{m(m+1)}{2} \cdot \frac{n}{p} = O\left(\frac{nm^2}{p}\right).$$

Ввод исходных данных (функций дохода, заданных в табличной форме) можно осуществить, как показано на рис.3, совмещая его с распространением первого фронта вычислений.

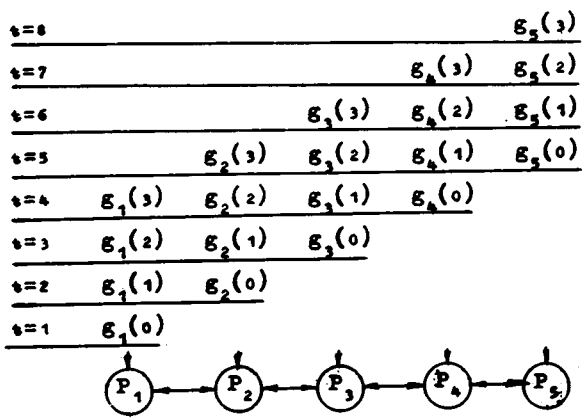

Рис.3

Это же множество графов G можно отобразить на сеть из m линейно связанных вычислителей так, что каждый вычислитель будет выполнять горизонтальные цепочки операторов из графов G_l, $l \in \{1, m\}$. В этом

случае каждая новая волна вычислений инициируется в моменты времени $t_i = 1, n+1, \ldots, (l-1)n, \ldots, (m-1)n+1$ вычислителями с номерами $P_1, P_2, \ldots, P_i, \ldots, P_m$ соответственно. Временная сложность определяется аналогично временем инициации последней m-й волны и максимальным расстоянием ее распространения $T_m(m,n) = t_m + d^m_{max} = (m-1)n+1 + (n-1) = O(mn)$.

Если предположить, что каждое значение $F_k(i,j)$ графа G_i может определяться отдельным вычислителем, то множество графов G может быть интерпретировано сетью из $m \times n$ ортогонально связанных вычислителей (на рис.4 пунктирными линиями показаны дополнительные каналы, по которым можно осуществить ввод исходных данных одновременно с вычислениями по первому фронту). Горизонтальные и вертикальные дуги в графе G_i будут соответствовать передаче данных между вычислителями в сети. Волновые фронты образуются в пер-

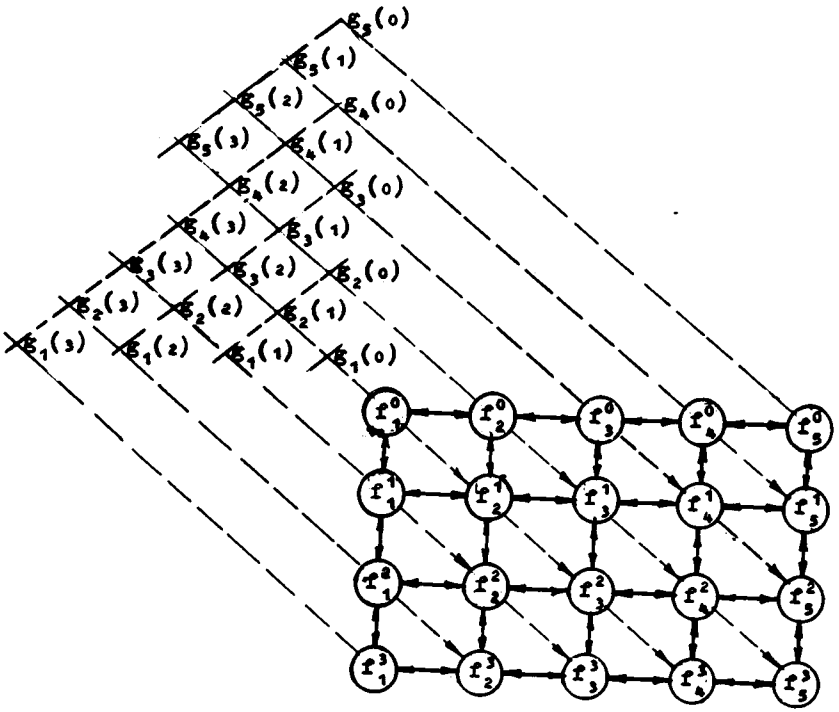

Рис.4

вом столбце сети вычислителей в моменты времени $t_1 = 1,3, \ldots$ $\ldots, 2^1-1, \ldots, 2m-1$.

В этом случае временная сложность определяется величиной $T_{nm}(m,n) = t_m + d^m_{max} = 2m-1+n-1 = O(m+n)$. Отметим, что последний фронт вычислений будет осуществляться на последней строчке в сети вычислителей, так как графы G_1 редуцируются (см.рис.1).

Таким образом, алгоритм решения задачи распределения ресурса допускает различные параллельные интерпретации как на сети из n либо m линейно связанных вычислителей, так и на сети из m x n ортогонально связанных вычислителей. Ускорение процесса вычисления в сравнении с последовательным выполнением алгоритма является пропорциональным числу вычислителей в сети.

Литература

1. TIDEN E., LISPER B., SCHREIBER R. Systolic arrays.-Report no TRITA-NA-8318, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, 1983.

2. МИШИН А.И., СЕДУХИН С.Г. Однородные вычислительные системы и параллельные вычисления. - АВТ, 1981, № 1, с.20-24.

3. MOLDOVAN D.I. On the Design of Algorithms for VLSI Systolic Arrays.- Proceedings of the IEEE,1983,v.71,N 1,p.113-120.

4. ХЕДЛИ Дж. Нелинейное и динамическое программирование.-М.: Мир,1967. - 506 с.

5. АХО А . ХОПКРОФТ Дж., УЛЬМАН Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979.- 536 с.

Поступила в ред.-изд. отд.
10 июля 1984 года