ВОПРОСЫ ТЕОРИИ И ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1977 год Выпуск 70

УДК 681.142.2

РАСПАРАЛЛЕЛИВАНИЕ ПО ЦИКЛАМ НА НЕБОЛЬШОЕ ЧИСЛО ВЕТВЕЙ

В.А. Вальковский

1. П о с т а н о в к а з а д а ч и. Исследования и эксперименты по статическому распараллеливанию программ позволяют выделить среди работ в этой области четыре основных направления, требующих существенно различных подходов и затрат при реализации.

1. П о с т р о е н и е п а к е т о в п а р а л л е л ь н ы х п р и к л а д н ы х п р о г р а м м. Это направление возникло из предпосылки, что большую часть вычислений программы занимают обращения к различным стандартным средствам. Поэтому целесообразны разработка параллельных вычислительных методов и тщательное ручное написание сбалансированных параллельных программ, ориентированных на данное вычислительное средство. Такие программы могут быть включены в пакет наряду с обычными последовательными программами и вызываться для высокоприоритетных задач. Число параллельных ветвей в программе может быть параметром, значение которого подставляется программистом или устанавливается операционной системой. Направление ручного распараллеливания широко представлено в работах ряда зарубежных и советских авторов (см., например, [1-2]).

2. Р а с п а р а л л е л и в а н и е м е т о д о м я р у с н о-п а р а л л е л ь н ы х ф о р м [3]. Этот вид распараллеливания связан с преобразованием макроструктуры программ и требует глубокого и тщательного анализа информационно-логических связей алгоритма, а также некоторых предварительных преобразований типа переименования переменных. Автоматическое распараллеливание методом ярусно-параллельных форм рекомендуется производить внутри оптимизирующего транслятора, извлекающего из программы много структурной информации. Это распараллеливание может оказаться эффективным для неразветвленных больших программ. Примером реализации, идейно связанным с методом ярусно-параллельных форм, является экспериментальный распараллеливатель для вычислительной системы МИНИМАКС [4].

3. М и к р о р а с п а р а л л е л и в а н и е. В отличие от других, это распараллеливание производится не по различным процессорам, а по аппаратуре одного и того же процессора. Характерным примером микрораспараллеливания служит распараллеливание выражений с ориентацией на их конвейерную реализацию [5]. Как правило, микрораспараллеливание требует локального анализа, сравнительно удобно реализуется аппаратно в динамическом режиме, однако выигрыш во времени от него ожидается небольшой.

4. Р а с п а р а л л е л и в а н и е п о ц и к л а м. В последние годы появился ряд работ по распараллеливанию циклических выражений [1,6-8]. Такое распараллеливание при небольших затратах на анализ локального характера дает наибольший эффект, так как основное время счета обычно приходится на циклы.

В настоящей работе описываются методы распараллеливания по циклам в предположении, что реализующая вычислительная система обладает небольшим числом процессоров (порядка десяти), причем в типичной ситуации каждая задача вычисляется на одном процессоре, а захват более одного процессора происходит только для высокоприоритетных задач. Ожидается, что так будут работать первое время многопроцессорные машины четвертого поколения. Таким образом, не ставится задача максимального распараллеливания, а считается достаточным распараллеливание на несколько больших слабосвязанных процессов. При анализе возможности распараллеливания циклов, как правило, в пространстве значений переменных для всех итераций выделяются гиперплоскости, на которых вычисления заведомо могут быть выбраны не только на основе отдельных гиперплоскостей, но и на основе данных из разных гиперплоскостей. Эти данные, быть может, не представляют удобную для описания поверхность, но связывают определенную часть внутренней параллельности. Например, если следовать теории гиперплоскостей в случае одиночного цикла, то либо каждая гиперплоскость вырождается в точку, т.е. цикл может быть реализован только последовательно, либо гиперплоскость "совпадает" с прямой, и все итерации могут быть реализованы параллельно. Но ясно, что цикл может быть вычислен, вообще говоря, и другими способами, например, кортежами по ρ итераций (ρ-кортежами) или несколькими независимыми ветвями (частично зависимые циклы [1]). В работе рас-

смотрены вопросы, связанные с этими режимами реализации. Предлагаемая методология может быть полезна при построении автоматических распараллеливателей, а также при разработке параллельных прикладных программ.

2. Структура зависимости между итерациями. Вначале рассмотрим один частный случай цикла, условно представив его в следующем виде:

$$DO\ I = 0, \Gamma \qquad \overbrace{X[\alpha * I + b]\ X[\alpha * I + \beta]}^{\text{тело цикла}} \ . \qquad (I)$$

Заголовок цикла имеет фортраноподобное написание. Тело цикла может иметь сложную структуру, в частности содержать циклы и операторы перехода, не выводящие за пределы тела цикла. Оператор присваивает выполнить телу цикла $\Gamma + 1$ раз при $I = 0, \ldots, \Gamma$. Символы $X[\alpha * I + b]$ и $X[\alpha * I + \beta]$ обозначают два вхождения некоторой переменной с индексом в тело, которые могут повлиять друг на друга в процессе параллельного вычисления итерации цикла. Такое влияние возможно, если хотя бы одно из вхождений переменной является выходным для тела цикла, т.е. стоит в левой части оператора присваивания. Все коррелирующие между собой вхождения переменной с индексом находятся в теле цикла специальной процедурой; мы же пока ограничимся случаем, когда существуют два вхождения, связывающих итерации цикла.

Задача о выявлении информационных связей между итерациями ставится так: найти все пары целых чисел (I, K) такие, что а) $\Gamma \geq I,\ K \geq 0$; б) $I \neq K$; в) $\alpha I + b = \alpha K + \beta$. При выполнении этих соотношений I-я и K-я итерации будут связаны через ячейку $X[\alpha I + b]$ и должны выполняться в заданном программистом порядке. Предположим, что α отлично от нуля. Из "в" имеем

$$K = \frac{\alpha I + b - \beta}{\alpha} = \frac{\alpha}{\alpha} I + \frac{b - \beta}{\alpha} = c I + \alpha ,$$

где $c = \frac{\alpha}{\alpha}$, $\alpha = \frac{b - \beta}{\alpha}$. Таким образом, зависимость между итерациями задается уравнением прямой, а целочисленные точки квадрата $[0, \Gamma]$, $[0, \Gamma]$, через которые она проходит, кроме точек на диагонали, являются искомыми парами итераций. При этом если $K > I$, т.е. точка лежит выше диагонали $K = I$, то имеет место зависимость между итерациями в прямую сторону: последующая итерация зависит от результата предыдущей или "затирает" результат предыдущей. Если же $K < I$, то имеется обратная связь; в этом случае итерация I

не может выполниться позднее итерации K, так как возможен незапланированный программистом обмен информацией через ячейку $X[\alpha I + b]$. Отношение зависимости между итерациями удобно задавать списком пар итераций, находящихся в зависимости

$$(I_1, K_1), (I_2, K_2), \ldots, (I_n, K_n);\quad 0 \leq I_j,\quad K_j \leq \Gamma. \qquad (2)$$

Полагаем, что $\forall j\ \ k(I_j < K_j \wedge I_j \geq I_k \leftrightarrow j \geq k)$ (иначе мы можем поменять их местами). Для нахождения полной зависимости между итерациями необходимо взять транзитивное замыкание отношения зависимости. Нетрудно показать, по индукции, что при целых c и α последовательность итераций, зависимых от итерации I, задается формулой

$$K_n(I) = I \cdot c^n + \frac{\alpha (c^n - 1)}{c - 1} \ . \qquad (3)$$

Формула (3) может быть применена для оценки степени параллельности цикла [7].

Рассмотрим более общий случай – цикл вида

$$DO\ I = H, \Gamma, Ш \qquad \overbrace{X[\alpha * I + b]\quad X[\alpha * I + \beta]}^{\text{тело цикла}}, \qquad (4)$$

где Н – начальная граница, Г – конечная граница, Ш – шаг. Сделаем замену: $I' = \frac{I - H}{Ш}$. Очевидно, что когда I изменяется от Н до Г с шагом Ш, I' изменяется от 0 до $\left[\frac{\Gamma - H}{Ш}\right]$ с шагом 1. Таким образом, выполнение цикла (4) сводится к выполнению цикла (I):

$$DO\ I' = 0, \left[\frac{\Gamma - H}{Ш}\right] \qquad \overbrace{X[\alpha Ш * I' + \alpha H + b]\ X[\alpha Ш * I' + \alpha H + \beta]}^{\text{тело цикла}} .$$

В наиболее общем случае, когда тело цикла имеет несколько вхождений выходных и входных переменных, зависимость между итерациями дается прямыми, построенными для каждой пары одноименных переменных, одна из которых – выходная переменная для цикла. Последовательность (2) в этом случае есть объединение последовательностей для отдельных прямых. Наконец, в случае нескольких индексов, удобное, хотя и более сильное, отношение зависимости может быть построено так. Для одноименной пары вхождений переменных выбирается первая позиция, в которой индексные выражения имеют рассмотренный вид линейных форм от параметра цикла, и для этой позиции производится анализ. Таким образом, неявно делается предположение, что в других позициях значения всегда совпадают. Если искомой позиции нет, то считается, что переменные тотально зависимы и цикл не распараллеливается.

3. Построение параллельных алгоритмов. Имея структуру зависимости между итерациями, заданную несколькими прямыми или списком пар, можно строить конкретные параллельные алгоритмы, определяющие распределение итераций по ветвям. Рассмотрим три типа таких алгоритмов.

Конструирование алгоритма первого типа основано на том, что, начиная с первой итерации, последовательно набираются максимально возможные кортежи, имеющие, как правило, различную ширину. Таким образом, если некоторая итерация до ρ-й связана с $\rho+1$-й, то в первый кортеж войдут первая, вторая, ..., ρ-я итерация, во второй кортеж – $\rho+1$-я, ..., $\rho+q$-я, так что итерация $\rho+q+1$-я находится в зависимости с некоторой из $\rho+1$, ..., $\rho+q$. Формально если для последовательности (2) определить: $M(N)=\min\{K_j\}$, где $\min$ распространяется на такие j, что $I_j, K_j > N$, то номер последней итерации кортежа h вычисляется по формуле:

$$l_0 = 0; \quad l_h = M(l_{h-1}) - 1.$$

Построенный алгоритм имеет существенный недостаток – неравномерность, т.е. различную ширину кортежей и вытекающие отсюда трудности для программирования. Можно поставить задачу: найти такое максимальное ρ, чтобы алгоритм из ρ-кортежей, каждый шириной ρ, не имел ни в одном кортеже связанных итераций. Другими словами, требуется построить параллельный равномерный алгоритм максимальной ширины. Рассмотрим решение задачи, дающее нижнюю грань для ρ в случае, когда зависимость задается только одной прямой: $K = cI + \alpha$. Пусть (I_1, K_1), (I_2, K_2) – первые две пары связанных итераций. Тогда очевидно, что последующие пары задаются формулами:

$$I_n = I_1 + (I_2 - I_1)(n-1); \quad K_n = K_1 + (K_2 - K_1)(n-1).$$

Таким образом, требуется найти $\rho = \min |K_n - I_n|$, причем $0 \le K_n$, $I_n \le \Gamma$. Мы найдем минимум, не заботясь о выполнении последнего условия. Сделаем преобразование:

$$K_n - I_n = (K_1 - I_1) + ((K_2 - K_1) - (I_2 - I_1))(n-1) =$$

$$(K_1 - I_1) + ((K_2 - I_2) - (K_1 - I_1))(n-1) = u + (v - u)(n-1),$$

где $u = K_1 - I_1$, $v = K_2 - I_2$. Для нахождения u и v нужно найти I_1, K_1, I_2, K_2. Эти числа можно найти перебором в цикле вида:

$$I := 0; \quad K := cI + \alpha; \quad \underline{\text{пока}} \quad [K] \neq K \ \underline{\text{цикл}} \ I := I+1; \ K := cI + \alpha.$$

Величина $|u + (v-u)(n-1)|$ минимальна, когда минимальна $|u - (u-v)(n-1)|$, или, в предположении, что $u \neq v$, минимальна $\left|\frac{u}{u-v} - (n-1)\right|$. Если $\frac{u}{u-v} > 0$, то минимум достигается, когда $n-1$ ближе всего к величине $\frac{u}{u-v}$. В этом случае

$$\rho = \left| (u-v) \min \left\{ \frac{u}{u-v} - \left[\frac{u}{u-v}\right], \left[\frac{u}{u-v}\right] + 1 - \frac{u}{u-v} \right\} \right|,$$

а n равно либо $\left[\frac{u}{u-v}\right]+1$, либо $\left[\frac{u}{u-v}\right]+2$. Если $\frac{u}{u-v} < 0$ или $u = v$, то минимум достигается при $n = 1$ и равен $|u| = |K_1 - I_1|$. Первый случай соответствует прямой $K = cI + \alpha$, пересекающей диагональ $K = I$: $|K_n - I_n|$ минимально, когда K_n, I_n лежат ближе всего к точке пересечения прямой с диагональю. Второй случай имеет место, когда прямая не пересекает диагональ в первой четверти: расстояние между K_n и I_n только увеличивается с увеличением n и минимально для $n = 1$. Итак, решение в случае одной пары переменных получено. В случае нескольких пар достаточно сосчитать минимум для каждой из них и взять в качестве ширины наименьшую из полученных величин. Заметим, что метод годится и для циклов с нефиксированными границами.

Особенностью описанных выше алгоритмов является то, что они должны реализоваться кортежами и после выполнения каждого кортежа требуют синхронизации между ветвями. Таким образом, при вычислении система должна делать вынужденные прерывания. Следующим шагом оптимизации вычисления является постановка задачи об организации параллельных ветвей так, чтобы вся необходимая информация для каждой ветви вычислялась в ней самой. В этом случае можно обойтись без специальных прерываний и получить высокоэффективное вычисление. Рассмотрим задачу в случае одной связывающей прямой $K = cI + \alpha$ при целых c и α. Возможным решением является организация ветвей из зависимых цепочек в соответствии с формулой (3). Ветви будут иметь вид:

$$K_1(1), K_2(1), \ldots, K_{t_1}(1),$$
$$K_1(q_1), K_2(q_1), \ldots, K_{t_2}(q_1),$$
$$\cdots$$
$$K_1(q_m), K_2(q_m), \ldots, K_{t_{m+1}}(q_m),$$

где $K_{t_i}(\ell) \le \Gamma; q_i \neq K_j(q_n)$ для $n < i$. Некоторые из полученных ветвей могут быть последовательно склеены для образования более длинных автономных ветвей.

В случае зависимости, задаваемой несколькими прямыми, картина существенно меняется, так как итерация может одновременно зависеть от нескольких итераций и влиять на несколько итераций, и ее нужно одновременно вводить в несколько ветвей. Это делает вычисление избыточным, кроме того, ветви такого вида трудно задавать с помощью языковых средств. Автором еще не найдено удобных алгоритмов для комплектования таких ветвей или удобных средств для их динамического порождения.

4. Машинный эксперимент. В настоящее время делается попытка реализации некоторых из описанных алгоритмов. В качестве входного языка выбран фортраноподобный язык, содержащий операторы цикла, присваивания и условного перехода, но не имеющий подпрограмм. Реализуется алгоритм, который в каждой программе для каждого цикла вычисляет число ρ (из второго метода), и, если $\rho \ge 2$, распараллеливает этот цикл. Метод применяется, если анализируемый цикл удается интерпретировать в виде цикла, индексные выражения которого имеют вид из (I). Алгоритм пишется на языке ЯРМО (язык реализации машинно-ориентированный), реализованном на машине БЭСМ-6 [9]. В принципе, алгоритм распараллеливания может быть построен двумя способами: либо путем анализа циклов и генерирования параллельного кода непосредственно во время полного грамматического разбора при трансляции (язык ЯРМО имеет для этого удобные средства), либо путем выявления в телах циклов входных и выходных переменных, анализа и генерирования параллельной формы на уровне входного языка перед трансляцией. Транслятор при этом должен обладать возможностью обработки параллельных операторов. Разрабатываемый алгоритм устроен по второму принципу. При работе происходит движение по программе до первого заголовка цикла, запоминаются все данные о цикле, затем производится анализ тела цикла. Каждая переменная вместе с индексами заносится в таблицу входных или выходных переменных. Эту последовательность действий выполняет процедура АНАЛИЗ. Затем работает процедура СЧЕТ, которая для каждой одноименной пары переменных вычисляет первые две зависимые пары итераций, затем число ρ, в полном соответствии с описанным алгоритмом. Процедура СИНТЕЗ вырабатывает параллельную форму для каждого распараллеливаемого цикла. В качестве оператора

задания параллельности выбран оператор *DO PAR FOR I= 0, L*. Оператора задания обмена нет: предполагается, что обмен идет через общую память. В случае, если найденное ρ больше конечной границы, т.е. все итерации цикла можно выполнить параллельно, то процедура СИНТЕЗ организует ℓ ветвей, где ℓ – параметр. Предполагается, что аналогичный алгоритм будет реализован для языка ФОРТРАН.

В целом описанный в работе подход представляется достаточно интересным для первых опытов по автоматическому распараллеливанию на реальную машину.

Литература

I. КОСАРЕВ Ю.Г. Распараллеливание по циклам. –В кн.: Вычислительные системы. Вып. 24. Новосибирск, I967, с. 3–20.

2. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на ОВС. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, I973, с. 3–32.

3. ПОСПЕЛОВ Д.А. Введение в теорию вычислительных систем. М., "Сов.радио", I972.

4. КЕРБЕЛЬ В.Г., МИРЕНКОВ Н.Н. Распараллеливание алгоритмов специального вида. –В кн.: Вычислительные системы. Вып. 63. Новосибирск, I975, с. I49–I58.

5. RAMAMOORTHY C.V., PARK J.H., LI H.F. Compilation techniques for recognition of parallel processable tasks in arithmetic expressions. –"IEEE Trans.on Comp.", 1973, v.C-22 № 11.

6. LAMPORT L. The parallel execution of DO loops. – "Communs ACM", 1974, v.17, №2, p.83–93.

7. НУРИЕВ Р.М. Существенная десеквенция схем программ с массивами. Автореф. дис. Киев, I976.

8. ВАЛЬКОВСКИЙ В.А. Метод оценки внутренней параллельности алгоритма. –"Изв. АН СССР. Техническая кибернетика", I975, № 3, с. I45–I46.

9. ЧЕБЛАКОВ Б.Г. Представление структур данных в машинно-ориентированном языке высокого уровня. –В кн.: Труды Всесоюзного симпозиума по методам реализации новых алгоритмических языков. Ч. 2. Новосибирск, I975, с. I69–I78.

Поступила в ред.-изд.отд.
I6 июля I976 года