© 2012 г. Р.Б. ШТЕЙНБЕРГ, к.ф-м.н
(факультет математики механики и компьютерных наук Южного федерального университета, г. Ростов-на-Дону)

ПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ РАСЧЕТА ТЕМПА ПОДАЧИ ДАННЫХ

В статье рассматривается отображение программных циклов на архитектуру, допускающую настройку конвейера (перестраиваемый конвейер). Предлагается полиномиальный алгоритм расчета темпа подачи данных с помощью графа вычислений. Темп подачи данных является важнейшей характеристикой для настройки конвейерных вычислений.

POLYNOMIAL ALGORITHM FOR ITERATION INITIALIZATION INTERVAL CALCULATION / R.B. Steinberg (MMCS SFEDU, 8a Milchakova-st, Rostov-on-Don, Russia, e-mail: romanofficial@yandex.ru). Mapping of program loops into reconfigurable pipeline architecture is considered in this paper. Computation graph is used to calculate iteration initialization interval and polynomial algorithm is suggested. Iteration initialization interval is the most important characteristic for pipelining.

1. Введение

В современном мире, мы все чаще встречаем компьютеры, которые на практике должны показывать хорошую эффективность на задачах разного типа. Борьба за такую эффективность идет разными путями: увеличение количества ядер, нетрадиционные архитектуры, добавление спецвычислителя в помощь ЦПУ, перестраиваемость архитектуры и т.д. Конвейер является очень эффективным средством для выполнения больших вычислений, но не всегда удается отобразить программу на уже настроенный конвейер, что приводит к потерям эффективности. Архитектуры, допускающие перестраиваемые конвейеры, позволяют расширить класс программ отображаемых на конвейер, тем самым решают указанную выше проблему.

Для организации конвейерных вычислений важно определить такую характеристику конвейера как темп подачи данных (интервал инициализации вычислений, iteration initialization interval). Время работы конвейера линейно зависит от темпа подачи данных. При использовании программируемых конвейеров, например на ПЛИС, эту характеристику очень желательно определять автоматически.

В данной статье рассматривается архитектура, допускающая перестраиваемый конвейер.

2. Задача вычисления темпа подачи данных

Алгоритмы вычисления темпа подачи данных для заданного программного цикла разрабатывались различными авторами [1, 2, 3]. В литературе упоминаются случаи, ко-

гда эта задача является np-полной [1]. В связи с этим, авторы часто прибегают к рассмотрению приближенных алгоритмов (modulo scheduling).

Рассмотрим графовую модель конвейера – граф вычислений. Граф вычислений состоит из вершин, представляющих собой операции, и дуг, соответствующих передаче данных (результат одной операции передается как аргумент другой операции). Этот граф можно построить по тексту программы. Обычно этот граф строится для одномерных циклов, но есть обобщения на многомерные циклы.

Формула вычисления темпа подачи данных имеет вид:

$$iii = \max_{c} \left| d(c) / p(c) \right|,$$

где максимум берется по всем c — циклам графа вычислений, $d(c)$ — сумма задержек операций по вершинам цикла c, $p(c)$ — сумма расстояний зависимости по дугам зависимости входящим в цикл с, а символом $\lceil a \rceil$ обозначено округление с избытком числа a.

Таким образом, задачу вычисления этой характеристики конвейера можно свести к перебору всех циклов на графе. Доказано, что достаточно перебирать не все циклы на графе, а только элементарные [4, 5]. Но даже в этом случае, перебор остается экспоненциальным.

Обратим внимание, на тот факт, что множество значений $d(x)$ задержек операций конечно и определяется архитектурой вычислителя. Например, если у нас архитектура допускает четыре арифметических операции (+,–,*,/), то у нас всего четыре возможных значения для задержки операции в произвольной вершине, а если сложение и вычитание, как обычно бывает, выполняются за одно и то же время, то остается 3 различных значения. Таким образом, можно считать, что множество различных значений $d(c)$ также конечно для заданной архитектуры вычислителя. Нужно отметить, что этот факт лежит в основе доказательства того, что алгоритм, который будет приведен ниже является полиномиальным.

3. Полиномиальный алгоритм вычисления темпа подачи данных

3.1. Случай графа вычислений, содержащего только одну обратную дугу

Сначала сформулируем две вспомогательные теоремы.

Теорема 1. Всегда существует элементарный цикл, на котором достигается максимум выражения для iii.

Теорема 2. Если все дуги классифицировать в соответствии с построением остовного дерева методом обхода в глубину, то любой цикл в ориентированном связном графе содержит хотя бы одну обратную дугу.

Доказательства этих теорем можно найти в [4, 5].

Рассмотрим случай графа вычислений, который является деревом и у него есть только одна обратная дуга $u = (s, r)$, ведущая в корень дерева. Тогда по теореме 2 все циклы будут содержать эту дугу, но рассматривать будем не все, а только элементарные по теореме 1.

Будем ставить в соответствие каждой вершине x этого графа множество характеристик путей, начинающихся в r и заканчивающихся в x. Обозначим множество характеристик вершины x через $C(x)$. Каждая характеристика будет представлять собой пару (d, p). Первый элемент характеристики d – это сумма задержек операций по некоторому пути ведущему из корня дерева в данную вершину, а второе значение p – это сумма рас-

стояний зависимости по тому же пути. Для корня, таким образом, множество характеристик будет состоять из одной пары (d_0, 0), где d_0 – задержка операции в корне дерева.

Таким образом, если найти множество всех характеристик для вершины s, то можно найти $\max\limits_{(a,b)\in C(s)} \left| a/\left(b+p(u)\right) \right|$ эта величина, очевидно, и будет равна iii. Для достижения этой цели необходимо найти множества $C(x)$ для каждой из вершин на графе вычислений. Это можно сделать аналогично алгоритму Дейкстры или волновому алгоритму (см. [6]) поиска кратчайшего пути на графе.

Если для каждой вершины мы будем хранить, все множество характеристик $C(x)$, то мощность $C(x)$ будет расти экспоненциально, что приведет к тому, что поиск максимума упомянутого выше будет иметь экспоненциальную оценку сложности. Поэтому будем хранить не все характеристики, а только те которые «претендуют» на то, чтобы максимум достигался на соответствующем пути. Чтобы определить какие характеристики следует хранить, введем отношение частичного порядка <<.

Определим частичный порядок << на множестве пар натуральных чисел следующим образом: $(a_1, b_1) << (a_2, b_2)$, если ($a_1 \le a_2$ и $b_1 > b_2$) или ($a_1 < a_2$ и $b_1 \ge b_2$). Очевидно, что для таких пар чисел верно, что $\frac{a_1}{b_1} < \frac{a_2}{b_2}$.

Обратим внимание, что если $(a_1, b_1) << (a_2, b_2)$. Тогда для любых неотрицательных целых чисел c и d, верно что $(a_1+c, b_1+d) << (a_2+c, b_2+d)$ и, следовательно, $\frac{a_1+c}{b_1+d} < \frac{a_2+c}{b_2+d}$.

Вернемся к множеству характеристик $C(x)$. Будем хранить, только те характеристики которые не сравнимы в смысле введенного выше частичного порядка. Теперь вспомним, что суммы расстояний зависимостей принимают конечное множество значений, а, следовательно, и мощность множества $C(x)$ будет полиномиально зависеть от количества вершин на графе. Таким образом, поиск максимума $\max\limits_{(a,b)\in C(s)} \left| a/\left(b+p(u)\right) \right|$ будет являться задачей полиномиальной сложности.

Рассмотрение случая произвольного графа вычислений сводится к рассмотрению описанного случая.

3.2. Случай произвольного графа вычислений

Алгоритм, описываемый в данном разделе может быть использован для составления расписания работы конвейера.

На входе алгоритма граф вычислений $G = G(X, U)$, без дуг инициализации, функция $d(x)$, ставящая в соответствие каждой вершине x графа G задержку операции в этой вершине, и функция $p(u)$, ставящая в соответствие каждой дуге u, порожденной истинной информационной зависимостью, расстояние зависимости для этой дуги. Пусть также $d(x)$ принимает ограниченное количество различных значений, не зависящее от $|X|$.

На выходе алгоритма число iii.

В процессе работы алгоритма, строится множество характеристик $C(x)$, которое представляет собой множество пар неотрицательных целых чисел.

Как уже было сказано ранее для вычисления iii не обязательно просматривать все циклы на графе вычислений, а достаточно лишь элементарные. Предположим теперь, что построено остовное дерево с помощью обхода в глубину и для всех дуг определен

их тип в соответствии с этим построением. Тогда каждый цикл будет содержать хотя бы одну обратную дугу (теорема 1). Следовательно, и любой элементарный цикл также будет содержать обратную дугу. Отсюда следует, что с помощью перебора обратных дуг можно построить нижнюю оценку *iii* для всех элементарных циклов содержащих данную обратную дугу.

Алгоритм 1 (полиномиальный).

1. К графу вычислений G добавить вспомогательную вершину и дуги, ведущие из этой вершины ко всем источникам. Полученный граф обозначим G'.
2. Построить для графа G' остовное дерево методом поиска в глубину, корнем которого будет являться единственный источник графа G'. Обозначим это дерево T. В процессе построения остовного дерева, найти множество всех обратных дуг U_{back}.
3. Для каждой дуги $u \in U_{\text{back}}$ найдем множество $D(u)$ всех вершин достижимых из конца дуги u в построенном выше остовном дереве T. Обозначим через s конец текущей дуги u, а начало через f.
4. Построим граф $T_{\text{u}} = T|_{\text{D(u)}}$. Этот граф является деревом. Корнем дерева T_{u}, очевидно, будет s, а начало дуги u также будет принадлежать этому дереву.
5. Построим правильную нумерацию для вершин дерева T_{u}.
6. Для корня дерева s определим множество характеристик $C(s) = \{(d(s), 0)\}$.
7. Будем обходить вершины T_{u} в порядке правильной нумерации, пропустив корень s. Для каждой вершины x дерева T_{u} будем строить множество характеристик $C(x)$. Каждая характеристика представлена парой неотрицательных чисел.
8. Положим $C(x) = \varnothing$.
9. Для текущей вершины x будем перебирать все вершины y такие, что существует дуга $v = (y, x)$.
10. Для каждой характеристики $(a, b) \in C(y)$ проверим, есть ли пара $(a', b') \in C(x)$, такая что пара (a', b') сравнима с парой $(a + d(x), b + p(v))$. Если пара (a', b') не сравнима с парой $(a + d(x), b + p(v))$, то добавим пару $(a + d(x), b + p(v))$ в множество характеристик $C(x)$. В противном случае проверим, если $(a', b') << (a + d(x), b + p(v))$, то заменим пару (a', b') на пару $(a + d(\text{x}), b + p(v))$.
11. Конец цикла по вершинам y.
12. Если $x = f$, то переходим к шагу 14.
13. Конец цикла по вершинам x.
14. Вычислим $iii(u) = \max\limits_{(a,b)\in C(f)} \dfrac{a}{b + p(u)}$.
15. Очистим все множества характеристик $C(x)$.
16. Конец цикла по дугам u.
17. Вычислим $iii = \max\limits_{u \in U_{back}} iii(u)$.
18. Конец алгоритма.

Замечание 1. В данном параграфе перебор всех путей между двумя заданными вершинами отсутствует, но при вычислении характеристик в каждой вершине учитываются все пути, которые могут привести в данную вершину.

На практике при построении графа вычислений функция $d(x)$ будет иметь конечное число различных значений, так как значения этой функции – это задержки операций, встроенных в вычислительное устройство, на которое будет отображаться программа.

СПИСОК ЛИТЕРАТУРЫ

1. *Французов Ю.А.* Обзор методов распараллеливания кода и программной конвейеризации // Программирование, 1992. № 3. С. 16-37.
2. *Сергиенко А.М.* VHDL для проектирования вычислительных устройств. – Киев.: ЧП «Корнейчук»; ООО «ТИД «ДС», 2003. 208 с.
3. *Вьюкова Н.И., Галатенко В.А., Самборский С.В.* Программная конвейеризация циклов методом планирования по модулю // Программирование. 2007. №6.
4. *Штейнберг Р.Б.* Использование решетчатых графов для исследования многоконвейерной модели вычислений // Известия вузов. Северо-Кавказский регион. Естественные науки. 2009. №2. С. 16-18.
5. *Штейнберг Р.Б.* Вычисление задержки между стартами конвейеров с учётом времени пересылки данных // Труды международной конференции «Параллельные вычисления и задачи управления» PACO-2006. Москва, 2-4 октября 2006 г. М: ИПУ РАН, 2006. С. 542-564.
6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МЦНМО, 2000. – 960 с., 263 ил. ISBN 5-900916-37-5.