УДК 004.4'422

А.А. Чемерис, Ю.А. Горунова, А.Д. Смирнов
ИПМЭ им. Г.Е.Пухова НАН Украины
ipme@ipme.kiev.ua

Об одном методе распараллеливания тесногнездовых циклов

Предлагается модифицированный метод аффинных преобразований распараллеливания циклических участков программ. Модификация состоит в обработке групп операторов циклов, которые не имеют между собой зависимостей по данным. Предлагаемый метод позволяет формализовать процесс распараллеливания и увеличить качество распараллеливаниия

Распараллеливание, тесногнездовые циклы, аффинные преобразования

Введение

Применение многопроцессорных компьютеров, как с общей памятью, так и распределённых систем, определяет для вновь создаваемых программных средств особые требования, обеспечивающие надежную и экономичную реализацию алгоритма при решении прикладных задач. Частично реализация этих требований сводится к применению алгоритмов и программ трансформации циклов, так как на выполнение циклических операций требуется наибольшее количество времени.

Наиболее эффективным в настоящее время есть использование аффинных преобразований для автоматизации распараллеливания циклических участков программ [1]. Здесь при анализе и преобразовании используется три вида пространств – пространство данных, пространство итераций и пространство процессоров. Итерации в цикле с глубиною вложенности n моделируется как n-мерный многогранник, границы которого определяются пределами цикла в коде программы. Аффинные функции отображают каждую итерацию на массив ячеек памяти, к которым она обращается. Для определения двух итераций, которые обращаются к одной и той же ячейке памяти, применяют методы целочисленного линейного программирования. Решением есть определение соответствия пространства итераций и пространства процессоров.

Итерации в цикле глубиной вложенности n представляются как n-мерный многогранник, границы которого определяются границами переменных цикла. Аффинные функции отображают каждую итерацию на массив ячеек памяти, к которым она обращается. Для определения существования двух итераций, которые обращаются к одной и той же ячейке памяти, используются методы целочисленного линейного программирования [2].

Авторами предлагается модификация метода аффинных преобразований для тесногнездовых циклов. Его преимуществом, по сравнению с методом аффинных преобразований, является то, что метод направлен, прежде всего, на формирование системы ограничений разбиения пространства итераций с возможностью трансформации циклов программ без ограничений количества зависимостей и уровня вложенности циклов.

Модифицированный метод аффинных преобразований

Для объяснения данного метода рассмотрим пример, приведенный на рис. 1.

```
for(i = 3; i    100; i++){

  for(j = 2; j     100; j++){

          a(i,j) = b(i,j   1);                    //s1

          b(i,j) = a(i   1,j);                    //s2

          c(i,j) = c(i,j) + c(i   2,j   1);       //s3

          }
}
```

Рисунок 1- Пример тесногнездового цикла

Операции s1 і s2 зависимы одна от другой и могут быть трансформированы при помощи метода, приведенного в [1], но для этого, прежде всего, требуется дополнительно расщепить цикл на два. Второй цикл при этом будет состоять из команды s3. Тогда становится возможным использование метода аффинных преобразований сначала для первого цикла, а потом для второго. После того, как аффинные разбиения для обоих циклов будут найдены, то целесообразно провести операцию объединения циклов. Как видно, для выполнения преобразования такого простого примера, требуется выполнить достаточно много

дополнительных вспомогательных шагов.

Рассмотрим возможность трансформации цикла на рис. 1 без дополнительных операций. Прежде всего, на основе ограничений переменных циклов сформируем неравенства $L_j \le i_j \le U_j$, где $j=\overline{1,n}$, n - вложенность цикла. Уравнения трансформации формируются исходя из условия существования зависимостей, которые определяются для каждой команды в цикле [3].

Далее рассматриваем случай, когда зависимости в цикле общем виде представлены как

$$I' = I + d, \qquad (1)$$

где d - вектор зависимостей.

Аффинное разбиение в пространстве процессоров запишем в виде $P = C \cdot I + S$, или

$$\begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix} \begin{pmatrix} i_1 \\ \vdots \\ i_m \end{pmatrix} + \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \qquad (2)$$

где P – вектор-столбец процессоров; I – итерационный вектор; C – неизвестная матрица коэффициентов; S – вектор-столбец констант.

Целью аффинного разбиения является поиск максимально возможного количества независимых разделов, удовлетворяющих ограничениям разбиения пространства, т.е поиск такой матрицы C и вектора s, чтобы ранг матрицы C был максимальным [1].

Назовем группой зависимых операций такой набор операндов, каждый из которого обращается как минимум к одному общему элементу и, как минимум, одно из этих обращений есть записью в память. Такое группирование операций определяет, что между группами зависимых операций нет зависимостей по данным. Так, в примере на рис. 1 есть две группы зависимых операций - к первой относятся операции s1 и s2, а ко второй - s3.

Пусть в некотором цикле есть G групп зависимых операций. Определим ограничение разбиения пространства для k-й группы зависимых векторов, $1 \le k \le G$ Пусть эта группа векторов имеет L зависимых операций.

Учитывая уравнение (2), система ограничений для множества групп цикла принимает вид

$$\begin{cases} C_k^1 \cdot I^1 + c_k^1 = C_k^2 \cdot I^2 + c_k^2 \\ \dots \\ C_k^1 \cdot I^1 + c_k^1 = C_k^L \cdot I^L + c_k^L \end{cases} \qquad (2)$$

и формализует условия выполнения зависимых операций в цикле одним и тем же самым процессором.

С учетом того, что для k-й группы зависимых операций

$$\begin{cases} I^2 = I^1 + d^2 \\ ..... \\ I^L = I^1 + d^L \end{cases} \qquad (4)$$

систему (3) приводим к виду (5):

$$\begin{cases} (C_k^1 - c_k^2)I + c_k^1 - C_k^2 d^2 - c_k^2 = 0 \\ \dots \\ (C_k^1 - c_k^L)I + c_k^1 - C_k^L d^L - c_k^L = 0 \end{cases} \qquad (5)$$

или

$$\begin{cases} C_k^1 = C_k^2 = \dots = C_k^L \\ c_k^1 - C_k^2 d^2 - c_k^2 = 0 \\ \dots \\ c_k^1 - C_k^L d^L - c_k^L = 0 \end{cases} \qquad (6)$$

Если $C_k^1 = C_k^2 = \dots = C_k^L$, тогда ограничения разбиения пространства для k-й группы зависимых векторов сформируются в виде системы неравенств

$$\begin{cases} I_{\min} \le I^1 \le I_{\max} \\ I_{\min} \le I^2 \le I_{\max} \\ \dots \\ I_{\min} \le I^L \le I_{\max} \\ I^1 = I^2 + d^2 = \dots = I^L + d^L \\ C_k I^1 + c_k^1 = C_k I^2 + c_k^2 = \dots = C_k I^L + c_k^L \end{cases} \qquad (7)$$

Решением системы является набор коэффициентов $C_L = C_k^1$ и констант $c_k^1,\dots,c_k^L$ которые определяют систему ограничений разбиения пространства процессоров. Целевой функцией есть ранг матрицы неизвестных C, значения которого требуется максимизировать для поиска максимально возможного количества независимых разбиений.

Существование в цикле нескольких групп зависимых операций может привести к единичному рангу матрицы C из-за одинаковых значений для зависимостей в одной группе зависимых операций. Для исключения такой ситуации сформируем матрицу C^*, каждая строка которой будет соответствовать коэффициентам от каждой из групп зависимых операций: $rang(C^*)=max$.

Система (7), включая целевую функцию, относится к классу оптимизационных задач целочисленного программирования.

Программная реализация

(3)

Для реализации модифицированного метода аффинных преобразований программных циклов предлагается использовать уже существующие программные пакеты, которые отдельно друг от друга реализуют определенные

задачи, необходимые в данном методе. Блок-схема алгоритма приведена на рис. 2. В алгоритме можно выделить четыре основных этапа: 1) определение и анализ зависимостей в цикле; 2) составление системы неравенств и оптимизационной задачи на ее основе; 3) поиск аффинного разбиения и 4) генерация кода трансформированного цикла.

Для определения зависимостей в цикле предлагается использование программы Petit, входящей в программный пакет, разработанный по проекту Omega [4]. Для решения оптимизационной задачи выбран пакет GAMS, обладающий мощными решателями задач целочисленного линейного программирования [5]. Генерацию кода модифицированного цикла выполняем программой Omega Calculator, разработанной также по проекту Omega.

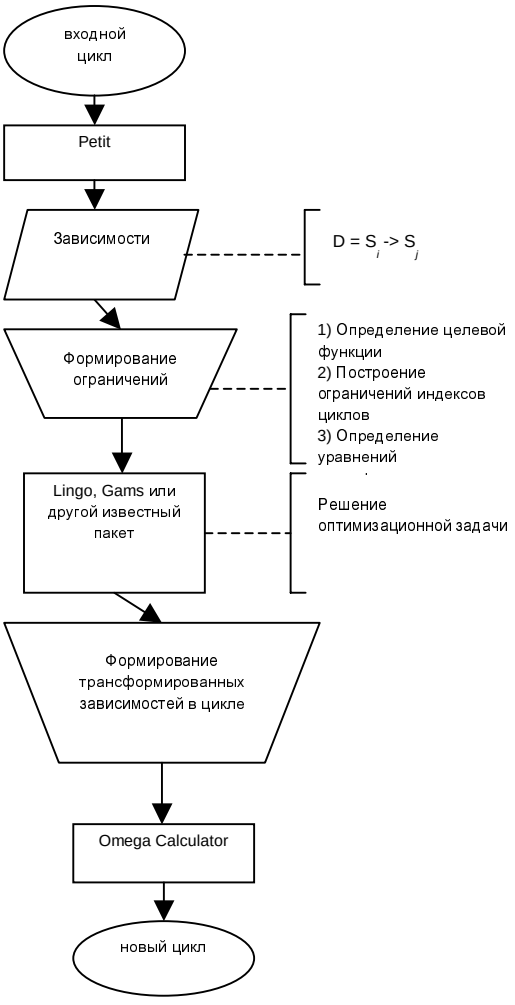

Рисунок 2 - Блок-схема алгоритма распараллеливания циклов

Пример. Рассмотрим теперь применение метода на конкретном примере. Пусть дан цикл (рис. 3).

Применим программный пакет Petit и сформируем матрицу векторов зависимостей

$$D = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}.$$

```
integer i,j,a(100,100),b(100,100),c(100,100)
do i = 3, 100, 1 {
        do j = 2, 100, 1 {
                a(i,j) = b(i,j-1)              // s1
                b(i,j) = a(i-1,j) + b(i-1,j-1) // s2
                c(i,j) = c(i,j) + c(i-2,j-1)   // s3
                }
}
```

Рисунок 3 - Пример цикла

В данном цикле имеется 2 группы зависимых операций. В первую входят операции s1 и s2 (с зависимостями d_1, d_2, d_3), а во вторую - s3 (с зависимостью d_4).

Нами разработан шаблон программы GAMS, куда подставляя собственные параметры и решая оптимизационную задачу, получаем аффинное разбиение пространства. Если p – идентификатор процессора, то каждая (i,j)-я операция s1, s2, s3 должна быть выполнена соответственно процессорами p_1, p_2 и p_3, причем

$$p_1 = c_{11}i + c_{12}j + s_{11} = j + 1, \quad 3 \le p_1 \le 101,$$

$$p_2 = c_{11}i + c_{12}j + s_{12} = j, \quad 2 \le p_2 \le 100,$$

$$p_3 = c_{21}i + c_{22}j + s_{21} = i + 2, \quad 5 \le p_3 \le 102.$$

Код программы, генерирующий новый, трансформированный, цикл представлен на рис. 4.

```
T10:={[p,i,j] -> [j+1,i,j]};
T20:={[p,i,j] -> [j,i,j]};
T30:={[p,i,j] -> [i+2,i,j]};
IS10 := {[p,i,j]:  3 <= i <= 100 && 3 <= j <= 100 &&
4<=p<=101 };
IS20 := {[p,i,j]:  3 <= i <= 100 && 3 <= j <= 100 &&
3<=p<=100};
IS30 := {[p,i,j]:  3 <= i <= 100 && 3 <= j <= 100 &&
5<=p<=102};
codegen  T10:IS10,  T20:IS20, T30:IS30;
```

Рисунок 4 - Код программы для генерации трансформированного цикла

Здесь выражения T10, T20, T30 – расписание параллелизма или, в терминах OMEGA CALCULATOR – кортеж зависимостей, а IS10.. IS30 – набор подмножеств из пространства процессоров. Строки после фразы «codegen» назначают выполнение операций различным множествам процессоров.

Трансформированный цикл, сгенерированный при помощи OMEGA CALCULATOR после подстановки выражений s1, s2, s3 приведен на рис. 5.

```
do p = 3, 102
do i = 3, 100
 if (i .eq. (p-2)) then
   do j = 3, i
      c(p-2,j) = c(p-2,j) + c(p-4,j-1)
   enddo
 endif
 if ((p .ge. 4).and.(p .le. 101)) then
    a(i,p-1) = b(i,p-2)
 endif
 if ((i .eq. p-2) .and. (i.le.99)) then
   c(p-2,p-3) = c(p-2,p-3) + c(p-4,p-4)
 endif
 if (p .le. 100) then
   b(i,p) = a(i,p-1) + b(i-1,p-1)
 endif
 if ((i .eq. p-2) .and. (i .le. 98)) then
   c(p-2,p-2) = c(p-2,p-2) + c(p-4,p-3)
 endif
 if (i .eq. p-2) then
   do j = i+3, 100
     c(p-2,j) = c(p-2,j) + c(p-4,j-1)
   enddo
 endif
enddo
enddo
```

Рисунок 5 - Трансформированный цикл

Проверка правильности трансформации состоит в сравнении работы циклов до и после трансформации. Написав программу, выполняющую инструкции исходного и трансформированного цикла на одинаковых наборах входных данных, убеждаемся, что трансформированный цикл дает тот же результат, что и исходный вариант.

Заключение

Статья кратко описывает результаты, полученные в рамках работы над проектом разработки системы оптимизации циклических участков программ, предназначенных для работы на многопроцессорных вычислительных системах, как с общей, так и распределенной памятью. Авторами предложена модификация метода аффинных преобразований, позволяющая формализовать процесс распараллеливания циклов и расширить область применения, т.е. появляется возможность распараллелить те циклы, которые не могут быть оптимизированы другими методами.

Список литературы

1. Компиляторы: принципы, технологии и инструментарий: 2-е изд. / Ахо, Альфред В., Лам, Моника С, Сети, Рави, Ульман, Джеффри Д. – М.: И.Д. Вильямс, 2008. – Гл. 11.

2. Loop parallelization algorithms: from parallelism extraction to code generation: Research Report №97-17 / P.Boulet, A.Darte, G.Silber, F.Vivien. – France, June 1997.

3. Горунова Ю. Метод распараллеливания циклов на основе аффинных преобразований / Ю. Горунова, А. Чемерис // Моделирование-2010. Материалы международной научно-технической конференции, г. Киев, 12-14 мая 2010 г., ИПМЭ им. Г.Е. Пухова НАН Украины. – 2010 – С. 287-293.

4. Transitive Closure of Infinite Graphs and its Applications: Technical Report CS-TR-3457 / Wayne Kelly, William Pugh, Evan Rosser, Tatiana Shpeisman. – University of Maryland Institute for Advanced Computer Studies~Dept. of Computer Science, Univ. of Maryland, April 1994.

5. Центральноазиатская миссия Агентства США по международному развитию: руководство по GAMS (русская версия) / А. Брук, Д. Кендрик, А. Меераус, Р. Раман. – 1999 – 172 с.

Надійшла до редколегії 18.10.2011

О.А. ЧЕМЕРИС, Ю.О. ГОРУНОВА, О.Д. СМИРНОВ
ІПМЕ ім. Г.Є.Пухова НАН України

Про один метод розпаралелювання тісногніздових циклів

Пропонується модифікований метод афінних перетворень для розпаралелювання циклічних ділянок програм. Модифікація полягає в обробці груп операторів циклів, які не мають між собою залежностей за даними. Пропонований метод дозволяє формалізувати процес розпаралелювання та збільшити якість розпаралелювання.

Розпаралелювання, тісногніздові цикли, афінні перетворення

A. CHEMERIS, JU. GORUNOVA, A. SMIRNOV
Pukhov Institute for Modeling in Energy Engineering, NASc of Ukraine

A Method for Parallelizing Loop Nests

A modified method of affine transformations of cyclic sections in parallel programs is proposed. The modification consists in the processing of groups of operators, which do not have data dependencies. The proposed method allows formalizing the process of parallelization and increasing the quality of parallelization.

parallelization, tightly nested loops, affine transformations