АЛГОРИТМ ПОИСКА ПАРАЛЛЕЛЬНОСТИ С СИНХРОНИЗАЦИЯМИ ВО ВЛОЖЕННОСТЯХ ЦИКЛОВ

А.А. Новокрещенов

Нижегородский государственный технический университет им. Р.Е. Алексеева

Предлагается алгоритм поиска параллельности с синхронизациями во вложенностях циклов. Алгоритм может быть использован для получения параллельных программ, предназначенных для выполнения как на системах с общей памятью, так и на гетерогенных системах, в состав которых входит графический процессор. По результатам работы алгоритма параллельный код может быть сгенерирован существующими генераторами кода.

Введение

Актуальной, на сегодняшний день, является задача распараллеливающей трансляции. Это комплексная задача, состоящая из фаз синтаксического анализа, анализа зависимостей, поиска параллельности, генерации кода. Ядром процедуры распараллеливающей трансляции является фаза поиска параллельности, т.к. именно с ее помощью происходит преобразование последовательной программы в ее параллельную форму. В данной работе предлагается алгоритм поиска параллельности с синхронизациями во вложенностях циклов. На ряде тестовых примеров было подтверждено, что параллельные программы, полученные с помощью данного алгоритма, могут быть эффективно выполнены, как на системах с общей памятью, так и на гетерогенных системах, в состав которых входят графические процессоры.

1. Основные термины и обозначения

Пусть дана неидеальная вложенность циклов глубины d, с индексными переменными $x_1, x_2, ..., x_d$ и инструкциями присваивания $S_1, S_2, ..., S_n$. При этом границы изменения каждой индексной переменной x_i представляют собой аффинные выражения от индексных переменных $x_1, ..., x_{i-1}$ и/или символьных констант, а шаг изменения каждой индексной переменной равен +1. Условия в операторах ветвления, присутствующих в теле вложенности, и индексные выражения обращения к массивам в инструкциях $S_1, S_2, ..., S_n$, так же представляют собой аффинные выражения от индексных переменных охватывающих циклов и/или символьных констант [1]. Существенная часть вычислений, выполняемых в научных и инженерных приложениях, протекает во вложенностях циклов, удовлетворяющих данным условиям [2].

Срабатывание инструкции S_i при определенной комбинации индексных переменных называют экземпляром инструкции S_i [3]. Все множество комбинаций значений индексных переменных, при которых происходит срабатывание инструкции S_i, называют ее доменом – D_{S_i}. Если инструкцию S_i охватывают m циклов, то D_{S_i} может быть представлен в виде целочисленного параметризованного многогранника в Z^m от

$\vec{x}_{S_i} = \{x_1, x_2, ..., x_m\}$ и $\vec{n}_{S_i} = \{n_1, n_2, ..., n_k\}$. Вектор $\vec{n}_{S_i}$ называют вектором глобальных параметров циклов, охватывающих S_i [2].

Обращение к массиву, заключенное в инструкции S_i, может быть представлено в виде кортежа:

$$\left\langle A, f(\vec{x}_{S_i}), \delta \right\rangle, \quad (1)$$

где A – имя массива, к которому выполняется обращение, $f(\vec{x}_{S_i})$ – аффинная функция, устанавливающая соответствие между комбинацией значений индексных переменных $\vec{x}_{S_i}$ и элементом массива A, δ - переменная, принимающая значения истина или ложь, если происходит обращение на запись или чтение соответственно [4].

В качестве модели инструкции S_i может быть использован следующий кортеж:

$$\left\langle D_{S_i}, M_{S_i} \right\rangle, \quad (2)$$

где D_{S_i} – домен инструкции S_i, M_{S_i} – множество обращений, заключенных в теле S_i представленных в форме (1).

Модель входной вложенности циклов представляет собой множество кортежей вида (2), определяющее все инструкции данной вложенности.

Инструкция $\left\langle D_{S_j}, M_{S_j} \right\rangle$ зависит по данным от инструкции $\left\langle D_{S_i}, M_{S_i} \right\rangle$ в том случае, если существует пара обращений $\left\langle A, f_1(\vec{x}_{S_i}), \delta_1 \right\rangle \in M_{S_i}$ и $\left\langle A, f_2(\vec{x}_{S_j}), \delta_2 \right\rangle \in M_{S_j}$, для которой справедливо:

$$(\delta_1 \vee \delta_2) \wedge (\exists \vec{x}^*_{S_i} \in D_{S_i}, \exists \vec{x}^*_{S_j} \in D_{S_j} \mid \vec{x}^*_{S_i} \prec \vec{x}^*_{S_j}) \wedge f_1(\vec{x}^*_{S_i}) - f_2(\vec{x}^*_{S_j}) = \vec{0}\ , \quad (3)$$

где $\vec{x}^*_{S_i}$ и $\vec{x}^*_{S_j}$ – комбинации конкретных значений индексных переменных, символ «$\prec$» обозначает отношение «меньше» в лексикографическом смысле [4]. В данной работе в качестве аппроксимации множества пар используются вектора расстояний. Вектором расстояния $\vec{v}_{i,j}$ называется вектор, для которого справедливо условие:

$$\vec{x}^*_{S_j} = \vec{x}^*_{S_i} + \vec{v}_{i,j} \quad (4)$$

для пар $\left\{\vec{x}^*_{S_i}, \vec{x}^*_{S_j}\right\}$, удовлетворяющих (3) [5].

Многогранным сокращенным графом зависимостей (МСГЗ) называют граф, каждая вершина которого представляет собой инструкцию S_i вложенности циклов, дуга из S_i присутствует S_j в том случае, если инструкция S_j зависит по данным от S_i. При этом каждая такая дуга помечается многогранником D_{S_i,S_j}, аппроксимирующим множество векторов расстояния для данной зависимости [5].

2. Алгоритм поиска параллельности

Ключевая идея алгоритма заключается в отображении экземпляров инструкций исходной вложенности в Z^1. В данном случае каждую точки Z^1 следует интерпретировать, как логическую дату выполнения некоторого множества экземпляров инструкций. В одну точку Z^1 (на один момент времени) должны отображаться экземпляры инструкций, между которыми отсутствуют зависимости по данным [2].

Результатом работы алгоритма поиска параллельности является аффинное отображение Θ_{S_i}, построенное для каждой инструкции S_i исходной вложенности [2]:

$$\Theta_{S_i} = T\begin{pmatrix} \vec{x}_{S_i} \\ \vec{n}_{S_i} \\ 1 \end{pmatrix}. \qquad (5)$$

Функция Θ_{S_i} определяет момент времени выполнения для каждого экземпляра инструкции S_i.

Ниже представлен алгоритм поиска функций вида (5).

Вход: МСГЗ исходной вложенности циклов.

Выход: набор функций Θ_{S_i} для каждой инструкции S_i исходной вложенности.

Алгоритм: для каждой дуги МСГЗ, проведенной из вершины S_i в вершину S_j, выполнить.

1. Составить ограничение, не позволяющее зависимым экземплярам S_j выполняться раньше экземпляров S_i [2]:
   $$\Theta_{S_j}(\vec{x}_{S_j}) - \Theta_{S_i}(\vec{x}_{S_i}) - 1 \geq 0. \qquad (6)$$
2. Составить прообраз аффинной функции, положительной на всем многограннике зависимостей D_{S_i,S_j}, используя для этого аффинную формулировку леммы Фаркаша [6].
3. Установить соответствие между коэффициентами полученного прообраза и коэффициентами неравенства (6). Результатом данного шага является система линейных уравнений от искомых коэффициентов функций отображения Θ_{S_i} и Θ_{S_j}, а так же коэффициентов прообраза аффинной функции, полученного на шаге 2.
4. Получить систему уравнений и неравенств, объединив уравнения предыдущего шага и неравенства, требующие неотрицательности искомых коэффициентов и коэффициентов прообраза, полученного на шаге 2 (данные неравенства являются следствием использования аффинной формулировки леммы Фаркаша).
5. Найти лексикографически минимальное решение полученной системы уравнений и неравенств, установив предварительно порядок следования коэффициентов искомых уравнений Θ_{S_i} и Θ_{S_j}.

Следует отметить, что изменения порядка искомых коэффициентов может привести к различным формам параллелизма.

3. Пример

Предлагается рассмотреть результаты работы алгоритма для следующей исходной вложенности циклов:

```
for (i = 0; i <= N - 1; i++)
  for (j = 0; j <= N - 1; j++)
  {
     X[i][j] = X[i][j] * sqrt(Y[i - 1][j]);          /* S1 */
    Y[i][j] = Y[i][j] * sqrt(X[i][j - 1]);           /* S2 */
  }
```

Между инструкциями данной вложенности присутствует перекрестная зависимость.

Используя различный порядок следования искомых коэффициентов (шаг 5), были получены следующие функции отображения (табл. 1).

Таблица 1. Функции отображения

Функции отображения	**Выполняемые преобразования**
$\Theta_{S_1}(\vec{x}_{S_1}) = 2j-1, \Theta_{S_2}(\vec{x}_{S_2}) = 2j-2$	Перестановка циклов, разрезание цикла по i
$\Theta_{S_1}(\vec{x}_{S_1}) = i+j-2, \Theta_{S_2}(\vec{x}_{S_2}) = i+j-2$	Волновое распараллеливание
$\Theta_{S_1}(\vec{x}_{S_1}) = 2i, \Theta_{S_2}(\vec{x}_{S_2}) = 2i+1$	Разрезание цикла по j

На основе данных функций были получены три параллельные версии исходной вложенности соответственно.

Вариант для CPU (перестановка циклов, разрезание цикла по *i*):

```
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    y[p2][1] = y[p2][1] * sqrt (x[p2][1-1]);
#pragma omp parallel for shared (x, y) private (p1, p2)
  for (p1=1;p1<=2*N-2;p1++)
    for (p2=1;p2<=N;p2++)
    {
      if ((p1+1)%2 == 0)
        x[p2][(p1+1)/2] = x[p2][(p1+1)/2] * sqrt (y[p2-1][(p1+1)/2]);
      if (p1%2 == 0)
        y[p2][(p1+2)/2] = y[p2][(p1+2)/2] * sqrt (x[p2][((p1+2)/2)-1]);
    }
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    x[p2][N] = x[p2][N] * sqrt (y[p2-1][N]);
```

Вариант для CPU (волновое распараллеливание):

```
for (p1=0;p1<=2*N-2;p1++)
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=MAX(1,p1-N+2);p2<=MIN(N,p1+1);p2++)
  {
    x[p2][p1-p2+2] = x[p2][p1-p2+2] * sqrt (y[p2-1][p1-p2+2]);
    y[p2][p1-p2+2] = y[p2][p1-p2+2] * sqrt (x[p2][p1-p2+2-1]);
  }
```

Вариант для CPU (разрезание цикла по *j*):

```
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    x[1][p2] = x[1][p2] * sqrt(y[1-1][p2]);
#pragma omp parallel for shared (x, y) private (p1, p2)
  for (p1=3;p1<=2*N;p1++)
    for (p2=1;p2<=N;p2++)
    {
      if ((p1+1)%2 == 0)
        y[(p1-1)/2][p2] = y[(p1-1)/2][p2] * sqrt (x[(p1-1)/2][p2-1]);
      if (p1%2 == 0)
        x[p1/2][p2] = x[p1/2][p2] * sqrt(y[(p1/2)-1][p2]);
    }
#pragma omp parallel for shared (x, y) private (p2)
  for (p2=1;p2<=N;p2++)
    y[N][p2] = y[N][p2] * sqrt(x[N][p2-1]);
```

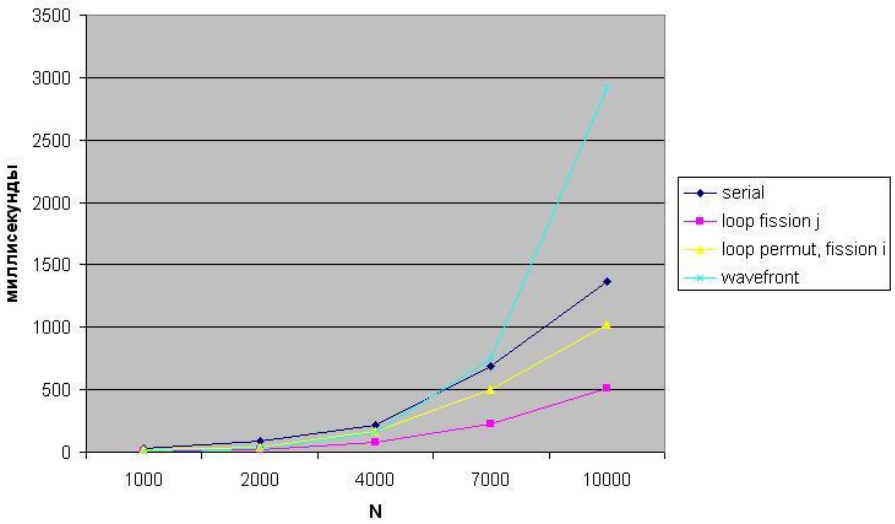

Рис. 1. Результаты временных замеров

Для оценки быстродействия программ выполнялись замеры времени их выполнения (рис. 1) для входных массивов различной длины (различных значений *N*). Для вычислений в данной работе использовалось следующее программное обеспечение и оборудование. Операционная система Debian Linux 6.0.4, CPU: Intel Pentium Dual-Core 2,6 ГГц E5300, RAM – 2048 Мб. Сборка программ выполнялась с помощью компилятора icc, с опциями –fast, –openmp, –xSSSE3.

Литература

1. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. Ростов-на-Дону: Издательство Ростовского университета, 2004. 192 с.
2. Pouchet L., Bastoul C. Iterative optimization in the polyhedral model: part 1, one-dimensional time // Proceedings of the International Symposium on Code Generation and Optimization, CGO'07. Washington: IEEE Computer Society, 2007. P. 144-156.
3. Ахо А., Лам М., Сети Р. Компиляторы. Принципы, технологии и инструментарий, 2-е изд. М.: Вильямс, 2008. 1184 с.
4. Lim A., Lam M. Maximizing parallelism and minimizing synchronization with affine transform // Proceedings of the 24th ACM SIGPLAN-SIGACT. N.Y.: ACM, 1997. P. 215–227.
5. Касьянов В.Н., Мирзуитова И.Л. Реструктурирующие преобразования: алгоритмы распараллеливания циклов // Программные средства и математические основы информатики. Новосибирск: Институт систем информатики им. А.П. Ершова СО РАН, 2004. С. 142–188.
6. Feautrier P. Some efficient solutions to the affine scheduling problem. One-dimensional time // International Journal of Parallel Programming. Norwell: Kluwer Academic Publishers, 1992. V. 21. P. 313-348.