ПРОФИЛИРОВАНИЕ И ЕГО ПРИМЕНЕНИЕ В ДИАЛОГОВОМ ВЫСОКОУРОВНЕВОМ ОПТИМИЗИРУЮЩЕМ РАСПАРАЛЛЕЛИВАТЕЛЕ

С.В. Полуян

Данная работа посвящена применению результатов профилирования для оптимального размещения данных в параллельной памяти с точки зрения оптимизации пересылок данных. Результаты этой работы используются в «Диалоговом высокоуровневом оптимизирующем распараллеливателе» (ДВОР) [1].

Особенностью данной работы является использование результатов профилирования программ не для преобразования циклов, а для размещения данных. Задача оптимального размещения данных в параллельной памяти имеет большое значение в ДВОР, так как из-за наличия информационных зависимостей [2] в программе не всегда возможно преобразовать распараллеливаемый цикл к виду, оптимальному с точки зрения пересылок данных, а выполнить оптимальное размещение на этапе компиляции можно всегда.

Для размещения данных в параллельной памяти используются *блочно-афинные размещения* [3], то есть такие размещения, при которых элемент массива X(i1,i2,...,im) находится в модуле памяти с номером u = (s1*i1+s2*i2+...+sm*im+s0) mod p, где s0,s1,s2,...sm - константы, зависящие только от массива X. В ДВОР реализован некоторый подкласс блочно-афинных размещений, содержащий такие размещения, как размещение массива по столбцам, по строкам и по диагонали.

В качестве лучшего размещения данных в ДВОР, выбирается то размещение, при котором количество пересылок между процессорами будет минимальным. Для расчета количества пересылок заводится *счетчик пересылок* и строится *дерево вложенности циклов* [4], в узлах которого содержится информация о циклах, а в листьях находятся вхождения массивов. С помощью этого дерева происходит вычисление того, в каком процессоре и к какому элементу массива происходит обращение. При этом если требуемый элемент массива не содержится в рассматриваемом процессоре, то счетчик пересылок увеличивается на единицу.

Из определения блочно-афинных размещений видно, что класс таких размещений достаточно велик, и перебрать все варианты размещения при большом количестве многомерных массивов во входной программе с целью отыскания оптимального размещения является достаточно сложной задачей. Для оптимизации этого процесса массивы размещаются в ДВОР в строго определенной последовательности, зависящей от величины массива и результатов профилирования входной программы.

Профилирование — это процесс сбора характеристик работы программы, таких как время выполнения, количество вызовов, степень покрытия программы [5] и других.

В данной работе *точками программы* будем называть выражения, время выполнения которых нужно вычислить, а *горячими точками* – участки программы, на выполнение которых тратится наибольшее количество времени.

Для сбора информации о горячих точках во входной программе в ДВОР реализовано профилирование, которое имеет следующие особенности:

1. позволяет получить время выполнения не для подпрограмм, а для наиболее мелких программных единиц - выражений.
2. является универсальным для разных языков программирования, так как реализовано на мультиязыковом внутреннем представлении, называемом в ДВОР Reprise [6], в узлах которого содержатся высокоуровневые конструкции языков программирования, а его структура не зависит от входного языка.
3. реализовано частичное обновление результатов профилирования при применении преобразований программ.

Алгоритмы профилирования, реализованные в ДВОР, позволяют выполнить глобальный анализ времени выполнения программы, в связи с чем их можно разделить на два типа:

- решающие внутрипроцедурную часть задачи профилирования;
- решающие межпроцедурную часть задачи профилирования.

Внутрипроцедурная часть профилирования основана на обходе дерева выражений в ДВОР. Начиная с первого выражения входной программы итерационный алгоритм вычисляет его время выполнения путем суммирования времени затраченного на каждую операцию, входящую в выражение. Для операторов ветвления вычисляется время работы каждой ветки, при этом временем работы самого оператора будет время работы самой долго работающей ветки. Основную сложность на внутрипроцедурном уровне представляют циклы: для того чтобы вычислить время работы цикла, вычисляется суммарное время всех выражений входящих в цикл, а затем умножается на число итераций цикла. В итоге, итерационный алгоритм завершит свою работу после того, как все выражения в подпрограмме окажутся вычисленными.

Межпроцедурная часть профилирования основана на обходе графа вызовов [7], начиная с точки входа в программу (для языка Си – это функции main). Общим временем работы подпрограммы будет время всех входящих в нее выражений. Также на межпроцедурном уровне вычисляется удельное время работы подпрограммы, под которым понимается время, затраченное на выполнение непосредственно самой

подпрограммы без учета дочерних вызовов. Удельное время выполнения нужно для того, чтобы отыскать действительно горячую точку программы, так как если удельное время не учитывать, то самой горячей точкой окажется функция main.

После того как профилирование входной программы в ДВОР выполнено, можно эффективно выбрать оптимальный способ размещения данных в параллельной памяти. Перед описанием алгоритма введем понятие вхождения переменной — это всякое появление этой переменной в тексте программы.

Описание алгоритма
Пока существуют неразмещенные массивы
{
1. Выбирается самая горячая точка программы, содержащая хотя бы одно вхождение еще неразмещенного массива.
2. Все неразмещенные массивы из горячей точки размещаются в порядке уменьшения их размера.

}
Конец алгоритма.

На данный момент мощность процессоров в вычислительных системах такова, что такие операции как сложение или умножение выполняются на порядок быстрее, чем пересылка данных между вычислительными устройствами, поэтому описанный в данной работе алгоритм может помочь улучшить характеристики параллельной программы.

ЛИТЕРАТУРА:

1. Диалоговый высокоуровневый оптимизирующий распараллеливатель программ (ДВОР). URL: http://www.ops.rsu.ru.
2. Allen R., Kennedy K. Optimizing compilers for Mordern Architetures // Morgan Kaufmann Publisher, Academic Press, USA, 2002, 790 p.
3. Штейнберг Б.Я. Оптимизация размещения данных в параллельной памяти, Изд-во Южного федерального ун-та, 2010. – 256 с.
4. Полуян С.В. Выбор оптимального размещения многомерных массивов в памяти компьютера // Известия высших учебных заведений. Северо-кавказский регион. № 3. 2010.
5. Касперски К. Техника оптимизации программ. Эффективное использование памяти. – Спб.: БХВ-Петербург, 2003. – 464 с.: ил.
6. Петренко В.В. Внутреннее представление REPRISE распаралелливающей системы // PACO’2008, Труды четвертой международной конференции «Параллельные вычисления и задачи управления», Москва.: 27-29 октября 2008 г. с.
7. Ахо, Альфред В., Лам, Моника С., Сети, Равви, Ульман, Джеффри Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2008. – 1184 с.