АЛГОРИТМЫ АВТОМАТИЧЕСКОГО ОТОБРАЖЕНИЯ ПОСЛЕДОВАТЕЛЬНЫХ ФОРТРАН-ПРОГРАММ НА КЛАСТЕР

М.С. Клинов

В рамках научно-технической программы Союзного государства "ТРИАДА" в 2005-2008 гг. была разработана экспериментальная версия автоматически распараллеливающего компилятора ПАРФОР. Компилятор осуществляет отображение Фортран-программы в параллельную программу для кластера на языке Fortran-DVM [1].

Компилятор ПАРФОР состоит из статического анализатора ВерБа [2], базы данных и DVM-эксперта, который осуществляет отображение на кластер с заданными характеристиками (количество и производительность процессоров, тип коммуникационной среды, ее латентность и пропускную способность).

Множество программ, на которые ориентирована текущая версия системы ПАРФОР, ограничено. Эти программы должны удовлетворять следующим требованиям:

- программы написаны на языке Fortran 77;
- не содержат процедуры или допускают их инлайн-подстановку;
- в обращениях к элементам массивов используются индексные выражения, которые линейно зависят от одной итерационной переменной;
- границы циклов и их шаги вычисляемы через подстановку именованных констант, объявленных в программе или заданные программистом.

Таким требованиям удовлетворяют, например, программы вычислений на статических регулярных сетках. Если индексные выражения или параметры циклов не удовлетворяют вышеуказанным требованиям, то это скажется на эффективности распараллеливания, но не приведет к отказу от распараллеливания

При отображении программы на кластер необходимо:

- распределить данные;
- распределить вычисления;
- организовать доступ к удаленным данным.

Но при автоматическом отображении возникает ряд трудностей:

- огромное количество возможных вариантов распределения данных (более чем 2**Nd, где Nd - сумма всех измерений по всем массивам, для тестов NAS Nd более 60)
- сложность построения вариантов распараллеливания (распределение вычислений и организация доступа к удаленным данным)
- сложность оценки их эффективности

Используемый в ПАРФОР подход к автоматическому распараллеливанию основывается на формировании множества наиболее перспективных вариантов распараллеливания программы и выборе наилучшего из них путем моделирования параллельного выполнения программы. Наиболее перспективные варианты - это те варианты, которые по грубым оценкам представляются компилятору наиболее эффективными. Более точные оценки получаются путем моделирования параллельного выполнения программы для каждого перспективного варианта распараллеливания программы.

Построение варианта распараллеливания опирается на то, что для каждого варианта распределения данных определяются параллельные циклы. Параллельными циклами могут быть многомерные циклы без зависимостей по данным, циклы с редукционными зависимостями, зависимостями по приватным переменным и регулярными зависимостями по массиву (конвейерное выполнение). Для параллельного цикла должно соблюдаться правило собственных вычислений (весь виток цикла будет выполняться на том процессоре, на котором находятся модифицируемые данные). Организация доступа к удаленным данным осуществляется при помощи специальных буферов и коммуникационных операций.

Моделирование параллельного выполнения построенных вариантов производится на уровне достаточно больших блоков программы [3], используется упрощенное моделирование параллельного выполнения (после моделирования нескольких первых витков циклов производится экстраполяция характеристик выполнения остальных витков), а также используются алгоритмы нахождения минимального прогнозируемого времени выполнения программы на заданном количестве процессоров.

Важным этапом работы алгоритма построения перспективных вариантов распараллеливания является выбор наиболее перспективных вариантов распределения данных. Этот выбор производится при помощи построения взвешенного графа.

Вершинами в графе являются измерения массивов, но не все массивы исходной программы могут быть распределенными в DVM, и не по всем измерениям массивов можно распределить вычисления. Поэтому соответствующие им вершины не будут добавляться в граф.

Дуги в графе отображают связь двух измерений каких-либо массивов между собой. Каждая дуга состоит из двух выражений (при каждой из вершин) и веса. Выражения задают индексы в обращениях операторов программы к элементам массивов. Дуги строятся для каждой пары обращений в одном цикле. Вес назначается дуге по некоторым правилам в зависимости от того, происходит ли обращение к элементу массива на запись или только на чтение. Вес дуги отображает временные задержки, которые возникнут, если распределение данных в программе не будет соответствовать соотношению выражений при дуге (т.е. будет им противоречить).

Временные задержки могут быть двух видов:

- Для непараллельных циклов возникают задержки из-за проверки правила собственных вычислений (на всех витках цикла каждый процессор проверяет, находится ли на нем данный элемент массива, и только своим элементам производит присваивания). Такие задержки замедляют выполнение цикла на каждом процессоре на время, сравнимое со временем его выполнения. Если данный цикл вызывается несколько раз (например, вложен в другой цикл), то задержки надо оценивать с учетом количества вызовов данного цикла. Поэтому вес = TLoop*Q, где TLoop - время выполнения цикла (время тела цикла, умноженное на количество итераций), а Q - количество вызовов данного цикла.
- Задержки из-за пересылок удаленных данных с одного процессора на другой. В худшем случае, всем процессорам пересылается весь массив, используемый на чтение. В DVM-эксперте используется модель коммуникаций, которая задается двумя величинами: латентностью и временными расходами на передачу каждого байта. Таким образом, вес = (Tstart+N*Tbyte)*Q, где Tstart и Tbyte - коммуникационные характеристики заданной ЭВМ, N - число байт в массиве, Q - количество вызовов цикла.

В случае обнаружения противоречивых требований к распределению массивов, веса дуг будут определять, какие из дуг надо удовлетворить, а какие следует проигнорировать.

При поиске минимального времени выполнения варианта распараллеливания программы возникает большое количество различных конфигураций процессоров, каждая из которых является многомерной решеткой процессоров. Для сокращения области перебора используется принцип рассмотрения всех конфигураций процессоров по классам, элементы которых различаются между собой только по одному измерению. Изменение времени выполнения программы среди элементов одного класса более закономерное, чем среди элементов разных классов. В каждом классе ищется один локальный минимум (ненужные конфигурации отбрасываются), лучший среди локальных минимумов всех классов считается искомым минимумом.

Работа компилятора ПАРФОР проверена на небольших тестах, а также на тесте NAS LU и программе mhpdv (трехмерного моделирования сферического взрыва во внешнем магнитном поле с помощью решения уравнений идеальной магнитогидродинамики), на которых было получено ускорение выполнения параллельной программы более чем в 100 раз по сравнению с исходной последовательной программой.

Работа выполнялась в рамках научно-технической программы Союзного государства "Развитие и внедрение в государствах-участниках Союзного государства наукоёмких компьютерных технологий на базе мультипроцессорных вычислительных систем" ("ТРИАДА").

Работа поддержана также грантом Президента РФ № НШ-383.2006.9 для ведущих научных школ и грантом РФФИ № 07-07-00221.

ЛИТЕРАТУРА:

1. DVM-система. http://www.keldysh.ru/dvm/
2. Баранова Т.П., Вершубский В.Ю., Ефимкин К.Н., Федосимов В.А. Развитие возможностей анализатора последовательных программ. // Труды Всероссийской научной конференции. "Научный сервис в сети Интернет: решение больших задач", Новороссийск, 22-27 сентября 2008. М.: Изд-во МГУ, 2008. С. 38-42.
3. Клинов М.С., Крюков В.А. Прогнозирование характеристик параллельного выполнения DVM-программ. // Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ: технологии параллельного программирования", Новороссийск, 18-23 сентября 2006. М.: Изд-во МГУ, 2006, С. 113-114.