АВТОМАТИЧЕСКОЕ РАСПАРАЛЛЕЛИВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ ПРОГРАММ ДЛЯ МНОГОЯДЕРНЫХ КЛАСТЕРОВ

В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина

Предлагается использовать для разработки параллельной программы стандартные языки последовательного программирования, добавляя в программу спецификации тех ее свойств, которые необходимо знать для ее эффективного распараллеливания, но не удается извлечь при анализе программы. Описываются принципы функционирования автоматически распараллеливающего компилятора ПАРФОР. Приводятся результаты экспериментальной проверки компилятора.

Введение

Разработка программ для высокопроизводительных кластеров и других параллельных систем с распределенной памятью продолжает оставаться исключительно сложным делом, доступным узкому кругу специалистов и крайне трудоемким даже для них. Основная причина – это низкий уровень современной технологии автоматизации разработки параллельных программ. В настоящее время практически все параллельные программы для многоядерных кластеров (SMP-кластеров) разрабатываются с использованием низкоуровневых средств передачи сообщений (MPI). MPI-программы трудно разрабатывать, сопровождать и повторно использовать при создании новых программ. Использование гибридной модели параллельного программирования (MPI/OpenMP) является еще более сложной задачей. Поэтому вполне естественно, что прикладной программист хотел бы получить либо инструмент, автоматически преобразующий его последовательную программу в параллельную программу, либо высокоуровневый язык параллельного программирования, обеспечивающий эффективное использование современных параллельных систем.

Проведенные в 90-х годах активные исследования убедительно показали, что полностью автоматическое распараллеливание для систем с распределенной памятью реальных производственных программ возможно только в очень редких случаях.

Поэтому исследователи сосредоточились на двух направлениях:

- разработка высокоуровневых языков параллельного программирования (HPF, OpenMP-языки, DVM-языки, CoArray Fortran, UPC, Titanium, Chapel, X10, Fortress);
- создание систем автоматизированного распараллеливания (CAPTools/Parawise, FORGE Magic/DM, BERT77), в которых программист активно вовлечен в процесс распараллеливания.

Изначально высокие требования к эффективности выполнения параллельных программ и последовавшие затем изменения в архитектуре параллельных ЭВМ привели к тому, что в настоящее время нет ни одного общепризнанного высокоуровневого языка параллельного программирования для современных кластеров.

В системах автоматизированного распараллеливания на программиста стали возлагаться ответственные решения не только по уточнению свойств его последовательной программы (результатов анализа), но и по ее отображению на параллельную ЭВМ. Кроме того, отсутствие широко распространенного высокоуровневого языка параллельного программирования вынуждало использовать в качестве выходного языка систем автоматизированного распараллеливания низкоуровневые языки. Полученные на них программы сложны для восприятия и внесения в них небольших модификаций. Это приводило к необходимости повторять процесс распараллеливания при модификации исходной последовательной программы. Все это вызывало огромные трудности при внедрении таких систем.

Все больше специалистов стали предлагать использовать для разработки параллельных программ языки с неявным параллелизмом, при программировании на которых не требуется знать архитектуру параллельной ЭВМ, и автоматически отображать такие программы на параллельные машины. В качестве таких языков особенно привлекательно использовать Fortran и C/C++, поскольку их в основном и используют программисты при решении задач, наиболее остро требующих распараллеливания.

Такая постановка задачи автоматического распараллеливания вызывает изменение подхода к автоматизации распараллеливания программ: использовать диалог с программистом только для уточнения свойств последовательной программы, а выбор наилучших решений по ее распараллеливанию осуществлять полностью автоматически.

Преобразование последовательной программы в параллельную программу можно представить состоящим из двух основных этапов.

На первом этапе проводится автоматизированное исследование и преобразование программистом последовательной программы с целью получить *потенциально параллельную программу* (последовательную программу, которую можно автоматически преобразовать в эффективную параллельную программу) и задать необходимые ее свойства (через диалог или в виде специальных аннотаций в тексте программы). Например, в случае косвенной индексации элементов массива статический анализатор может сообщить о возможной зависимости между витками цикла, а программист может указать, что зависимости нет.

На втором этапе *автоматически распараллеливающий компилятор* преобразует потенциально параллельную программу в *параллельную программу* для заданной ЭВМ.

Достижение приемлемой эффективности выполнения параллельной программы может потребовать многократного повторения этих двух этапов, но *система автоматизации распараллеливания* должна быть ориентирована на сокращение таких итераций. Прежде всего, она должна позволить на инструментальной машине и еще до появления текста параллельной программы оценить эффективность ее работы на разных ЭВМ.

Эта способность быстрой оценки влияния разных решений по распараллеливанию на эффективность программы, которая сопровождается детальной разъясняющей информацией, очень важна для ускорения разработки параллельных программ, а также при обучении параллельному программированию.

Далее рассматриваются принципы построения автоматически распараллеливающего компилятора ПАРФОР и приводятся результаты его экспериментальной проверки.

1. Автоматически распараллеливающий компилятор ПАРФОР

Компилятор на вход получает текст потенциально параллельной программы вместе со специальными аннотациями, которую для простоты изложения будем называть просто “последовательной программой”. Автоматически распараллеливающий компилятор состоит из блоков анализа последовательной программы (*анализаторов*) и блоков отображения на параллельные ЭВМ (*экспертов*). Их взаимодействие осуществляется через специальную *базу данных*.

Допускается использование нескольких анализаторов для дополнения базы данных более точными результатами анализа. Анализаторы могут различаться методами анализа (статические или динамические) и алгоритмами (более быстрыми или более точными).

Эксперты могут отличаться алгоритмами отображения на параллельные ЭВМ и используемыми языками параллельного программирования. Функциями эксперта являются:

- построение схем распараллеливания;
- оценка схем для заданной ЭВМ;
- получение текста параллельной программы для лучшей схемы.

Под *схемой распараллеливания* понимается набор правил преобразования последовательной программы в параллельную.

На вход эксперту, кроме структуры и свойств последовательной программы, поступает описание ЭВМ (в нем задается архитектура – мультипроцессор, кластер или многоядерный кластер, число узлов и ядер в узлах, производительности ядер и параметры коммуникационной среды) и параметры задачи (значения переменных, которые определяют размеры массивов и количество витков циклов).

Выходом компилятора ПАРФОР является текст параллельной программы на языке Fortran DVM/OpenMP [1, 3]. Блок отображения (DVM/OpenMP эксперт) на многоядерный кластер можно рассматривать как дальнейшее развитие блока отображения на кластер (DVM эксперта [2]) для обеспечения распараллеливания программы внутри каждого узла средствами OpenMP.

Компилятор ПАРФОР ориентирован на автоматическое распараллеливание программ из класса задач, при решении которых используются разностные методы на статических структурных сетках. К тому же на данный момент последовательные программы должны удовлетворять следующим требованиям:

- быть написаны на языке Fortran 77;
- не содержать процедуры или допускать их инлайн-подстановку.

Язык Fortran DVM/OpenMP представляет собой расширение стандартного языка Fortran 95 директивами параллелизма. В результате компиляции программа на языке Fortran DVM/OpenMP преобразуется в программу на стандартном языке Fortran OpenMP, которая выполняется на узлах кластера и использует библиотеку MPI для организации взаимодействия между узлами.

Выбор этого языка параллельного программирования обусловлен тем, что он является высокоуровневым языком, что значительно упрощает преобразование последовательной программы в параллельную. Кроме того, для него есть развитые средства функциональной отладки и отладки эффективности, что существенно облегчает отладку алгоритмов распараллеливания.

2. Результаты экспериментальной проверки компилятора ПАРФОР

Компилятор ПАРФОР был испытан на реальных приложениях: тестах NAS [4] BT, LU, SP, и разработанных в ИПМ им. М.В. Келдыша РАН программах MHPDV [5, 3] и ZEBRA [6]. Тексты последовательных версий этих программ были получены из их DVM-версий путем удаления всех DVM-директив и инлайн-подстановки процедур в тело головной подпрограммы.

Программа BT (Block Tridiagonal Solver) – нахождение конечно-разностного решения 3-х мерной системы уравнений Навье-Стокса для сжимаемой жидкости или газа. Используется трех диагональная схема, метод переменных направлений. При выполнении программы задавался класс C (сетка 162x162x162) и класс A (сетка 64x64x64).

Программа LU (Lower-Upper Solver) отличается от BT тем, что применяется метод LU-разложения с использованием алгоритма SSOR (метод верхней релаксации), а программа SP (Scalar Pentadiagonal Solver) – тем, что используется скалярная пяти диагональная схема.

Программа MHPDV осуществляет трехмерное моделирование сферического взрыва во внешнем магнитном поле с помощью решения уравнений идеальной магнитной гидродинамики.

Программа ZEBRA производит расчет нейтронных полей атомного реактора в диффузионном приближении.

Полученные в результате автоматического распараллеливания программы на языке Fortran DVM/OpenMP могут рассматриваться как программы для кластера (на языке Fortran DVM) и как программы для мультипроцессора (на языке Fortran OpenMP). Поэтому ниже приводятся времена выполнения эталонных вариантов (распараллеленных вручную) и вариантов, полученных в результате автоматического распараллеливания, сначала для кластера, затем для мультипроцессора.

Таблица 1. Времена выполнения вариантов программ на кластере МВС-100К (сек) (“авт” – автоматическое распараллеливание компилятором ПАРФОР, “ручн” – ручное)

Варианты	Число процессоров 1	Число процессоров 8	Число процессоров 64	Число процессоров 256	Число процессоров 1024
BT-авт (класс C)	мало памяти	1255.97	182.70	54.64	21.36
BT-ручн	мало памяти	817.88	128.01	30.27	7.19
LU-авт (класс C)	3482.40	1009.49	148.78	40.33	25.55
LU-ручн	2103.14	858.26	122.61	34.99	19.97
SP-авт (класс C)	1982.00	–	–	–	–
SP-ручн	2601.85	–	–	–	–
MHPDV-авт	3703.23	500.78	89.32	34.75	12.78
MHPDV-ручн	3574.29	486.74	79.63	32.15	10.98
ZEBRA-авт	75.09	11.13	1.96	–	–
ZEBRA-ручн	75.62	10.18	1.85	–	–

Программу SP автоматически распараллелить для кластера не удалось (никаких DVM-директив в нее не было вставлено). Необходима ее серьезная модификация, чтобы избавиться от имеющихся там зависимостей между витками циклов, препятствующих их параллельному выполнению (первые три витка цикла модифицируют один и тот же элемент массива, который не является редукционной или приватной переменной). Такие циклы можно распараллелить, если обеспечить, чтобы зависимые витки выполнялись на одном процессоре, что и было сделано при ручном распараллеливании.

Увеличение времени выполнения автоматически распараллеленных вариантов программ BT и LU на малом количестве процессоров объясняется тем, что на этих программах подстановка процедур сильно повлияла на оптимизацию кода компиляторами. Увеличение времени выполнения на большом количестве процессоров происходит из-за того, что DVM-эксперт не сумел организовать доступ к удаленным данным так же эффективно, как это смогли сделать разработчики DVM-версий вручную.

На программах MHPDV и ZEBRA времена выполнения автоматически распараллеленных вариантов и вручную распараллеленных – близки. Увеличение количества процессоров для программы ZEBRA (свыше 64-х) уже не ускоряет ее выполнение.

Таблица 2. Времена выполнения вариантов программ на одном узле кластера СКИФ-МГУ (сек) (“авт” – автоматическое распараллеливание компилятором ПАРФОР, “авт-Интел” – автоматическое распараллеливание компилятором Интел, “ручн” – ручное)

Варианты	Число ядер 1	Число ядер 2	Число ядер 4	Число ядер 8
BT-авт (класс A)	119.04	61.39	39.56	34.66
BT-ручн	109.78	55.88	37.65	34.73
BT-авт-Интел	115.12	109.38	106.81	106.45
LU-авт (класс A)	110.42	60.56	39.01	33.35
LU-ручн	106.58	57.09	37.88	35.53
LU-авт-Интел	105.04	91.05	83.97	83.94
SP-авт (класс A)	81.08	76.64	75.42	76.62
SP-ручн	102.82	51.42	33.19	32.91
SP-авт-Интел	78.82	72.09	70.21	69.85
MHPDV-авт	300.19	148.87	82.89	51.60
MHPDV-ручн	333.80	169.97	91.25	57.55
MHPDV-авт-Интел	297.70	260.14	252.82	261.10
ZEBRA-авт	45.91	21.85	11.71	5.92
ZEBRA-ручн	53.68	26.17	14.72	8.11
ZEBRA-авт-Интел	55.38	55.09	55.55	55.21

Из-за отсутствия версий программ, распараллеленных вручную для мультипроцессора при помощи языка Fortran OpenMP, для сравнительной оценки качества распараллеливания компилятором ПАРФОР использовались вручную распараллеленные программы на языке Fortran DVM и программы, автоматически распараллеленные компилятором фирмы Интел.

Заключение

Разработка компилятора ПАРФОР для современных кластеров и исследование его применимости на реальных приложениях показали, что удалось решить принципиальную задачу – обеспечить эффективное автоматическое отображение последовательной программы на кластер. Для эффективного отображения на многоядерные кластеры требуется использование двух уровней параллелизма (модель передачи сообщений между узлами и модель общей памяти в узлах), которое было реализовано в компиляторе.

В настоящее время ведется развитие компилятора в двух направлениях – расширение входного языка Fortran 77 до языка Fortran 95 и снятие ограничений на использование процедур. Кроме того, в рамках создания системы автоматизации распараллеливания ведется разработка ее диалоговой подсистемы и средств отладки параллельных программ. Эти работы поддержаны программой Союзного государства СКИФ-ГРИД, Программами президиума РАН №14 и №15, грантами РФФИ № 08-01-00479 и № 10-07-00211. Их успешное выполнение позволит перейти к практическому внедрению системы автоматизированного распараллеливания программ на языке Fortran и ее дальнейшему развитию применительно к кластерам с неоднородными узлами (содержащими в качестве ускорителей графические процессоры).

ЛИТЕРАТУРА:

1. В.А. Бахтин, Н.А. Коновалов, В.А. Крюков, Н.В. Поддерюгина "Fortran OpenMP/DVM – язык параллельного программирования для кластеров" // Материалы второго Международного научно-практического семинара "Высокопроизводительные параллельные вычисления на кластерных системах", г. Нижний Новгород, 26-29 ноября 2002 г., с.28-30
2. М.С. Клинов, В.А. Крюков "Автоматическое распараллеливание Фортран-программ. Отображение на кластер" // Вестник Нижегородского университета им. Н.И. Лобачевского, 2009, № 2, с.128-134
3. В.А. Бахтин, Н.А. Коновалов, Н.В. Поддерюгина, С.Д. Устюгов "Гибридный способ программирования DVM/OpenMP на SMP-кластерах" Труды Всероссийской научной конференции "Научный сервис в сети Интернет: технологии параллельного программирования" (сентябрь 2006 г., г. Новороссийск), Изд-во Московского Университета, с.128-130
4. The NAS Parallel Benchmarks [Электронный ресурс] – Режим доступа: http://www.nas.nasa.gov (дата доступа 26.01.2009)
5. S.D. Ustyugov "Three Dimensional Numerical Simulation of MHD Solar Convection on Multiproccesors Supercomputer Systems" // Proc. of the NSO 23 Workshop "Solar MHD: Theory and Observations – a High Spatial Resolution Perspective", Sunspot, New Mexico, USA, 18-22 July, 2005
6. V.I. Arzhanov, A.V. Voronkov, A.S. Golubev, E. Zemskov, N.A. Konovalov, V.A. Krukov "Development of explicit cyclic schemes with Chebyshev's polinomials for space neutron kinetics on the multiprocessor computers" // Proc. of INFORUM-99 Annual Meeting on Nuclear Technology, Karlsruhe, Germany, 18-20 May 1999, p. 19-23