Автоматизация преобразований последовательных Фортран-программ, необходимых для их эффективного распараллеливания с помощью системы САПФОР

Н.А. Катаев[1], А.А. Буланов[2]

Институт Прикладной Математики им. М.В. Келдыша РАН[1], Московский Государственный Университет им. М.В. Ломоносова[2]

Автоматическое отображение последовательных программ на вычислительные системы с распределенной памятью может потребовать предварительного преобразования программ, ориентированного на данный класс систем. Использование системы САПФОР для распараллеливания прикладных программ позволило выделить преобразования, выполнение которых может быть автоматизировано. В статье представлены преобразования, повышающие возможность эффективного распараллеливания программ за счет устранения причин, препятствующих распараллеливанию циклов. Выполнение данных преобразований позволило автоматизировать получение последовательной реализации, эффективно отображаемой на современные кластеры автоматически распараллеливающим компилятором системы, для задачи гидродинамики.

1. Введение

Основной целью распараллеливания прикладной программы является ее эффективное выполнение на современных вычислительных системах. Отличительной чертой программирования для таких систем является совместное использование различных технологий параллельного программирования. Сложность применения данных технологий подчеркивает актуальность исследований в области автоматизации распараллеливания программ.

К автоматически распараллеливающим компиляторам можно отнести PLUTO [1], Par4all [2], Parallware [3], задать опции для автоматического распараллеливания можно при использовании компиляторов Intel. Основным языком программирования является C. Язык Fortran поддерживается в компиляторах Par4all (только Fortran 77) и Intel. Следует отметить, что компиляторы Intel и Parallware являются коммерческими. Распараллеливание ориентировано на системы с общей памятью, компилятор Par4all также обеспечивают распараллеливание для графических ускорителей. Считается, что автоматическое распараллеливание для систем с распределенной памятью пока не достижимо [4].

К автоматизированным системам распараллеливания относятся САПФОР [5], Intel Parallel Studio [6], Parawise [7], ДВОР [8]. Особенностью системы САПФОР является использование автоматически распараллеливающего компилятора – взаимодействие с пользователем осуществляется на этапе подготовки последовательной программы к автоматическому распараллеливанию. Пользователь, во-первых, предоставляет системе информацию о свойствах программы, которую не удалось получить автоматическими средствами анализа (статическими и динамическими). Во-вторых, он участвует в преобразовании последовательной программы в последовательную программу с целью устранения проблем, выявленных системой и мешающих автоматическому распараллеливанию.

Входным языком системы САПФОР является Fortran, результат распараллеливания представляет собой программу на языке Fortran OpenMP, Fortran DVM/OpenMP или Fortran DVMH. Модель DVMH [9] является расширением модели DVM [10] и обеспечивает распараллеливание программы на кластер с ускорителями. Основными компонентами САПФОР являются анализаторы последовательных программ (статические и динамические), блоки преобразования последовательных программ в параллельные программы (эксперты), диалоговая оболочка для взаимодействия с пользователем, генератор кода, создающий на основе принятых экспертом решений параллельную версию программы.

Целью данной работы является автоматизация преобразований последовательной программы, выполняемых на этапе подготовки ее к автоматическому распараллеливанию экспертом системы САПФОР.

2. Подходы к преобразованию программ

Большинство компиляторов использует заранее определенные последовательности фаз для оптимизации и распараллеливания программ. Данные последовательности могут сильно различаться в зависимости от цели проводимых оптимизаций: по памяти, по времени выполнения на одном процессоре или на нескольких. При распараллеливании необходимо учитывать архитектуры вычислительных систем, на которых предполагается выполнение программы. При этом оптимальность таких последовательностей сильно зависит от прикладной программы, а поиск наилучшей последовательности для конкретной программы сильно затруднен из-за огромных размеров пространства возможных оптимизационных последовательностей [11].

Распараллеливание в системе САПФОР представляет собой итерационный процесс, это обеспечивает возможность выбора трансформаций для решения проблем, мешающих эффективному распараллеливанию последовательной программы экспертом системы САПФОР, по мере их возникновения. Эксперт системы выявляет особенности программы, влияющие на качество ее распараллеливания (см. пример на **Рис. 1**).

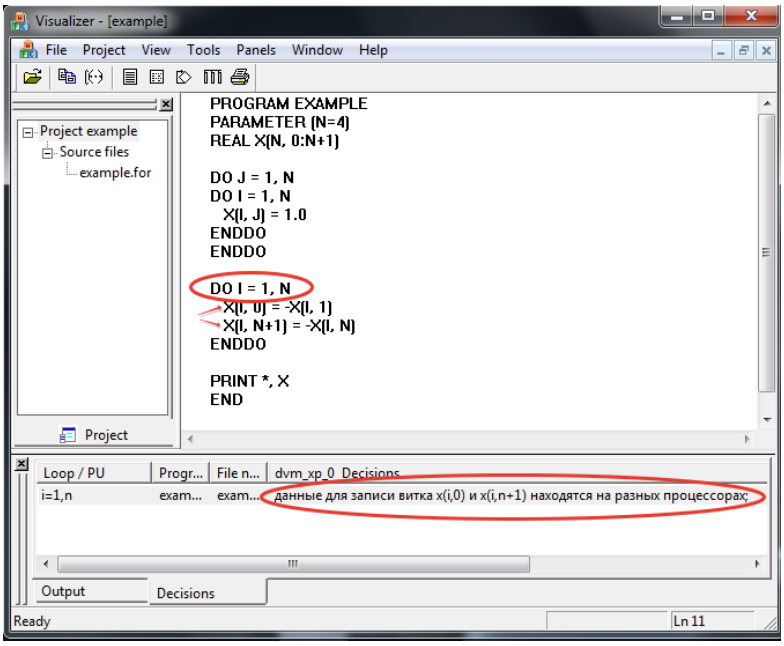

Рис. 1. Пример проблемы, мешающей эффективному отображению программы на параллельные компьютеры с распределенной памятью

Предлагается три подхода к преобразованию последовательной программы.

Первый подход состоит в автоматическом выборе способа устранения проблем. Решение о возможности устранения совокупности проблем принимается на основе удовлетворения каждой проблемы некоторому шаблону, описывающему ее решение. Процесс устранения проблем сводится к применению выбранных шаблонов. Каждая проблема рассматривается независимо от других, и в том случае если решения двух проблем конфликтуют между собой, необходимо участие пользователя для разрешения конфликта.

Второй подход состоит в активном вовлечении пользователя в процесс выбора решений выявленных проблем. Система предоставляет пользователю набор элементарных преобразований, для которых возможно автоматическое выполнение (например, разворачивание цикла, разделение цикла на несколько циклов, подстановку переменных и др.). Пользователь выбирает преобразование и указывает требующиеся характеристики, описывающие данное преобразование. Перед осуществлением преобразования система проверяет его допустимость. Допускается принудительное выполнения преобразования, если пользователь уверен в его допустимости, а требование консервативности средств анализа ограничивает его выполнение.

Если ни один из подходов не может быть применен, пользователь может вручную преобразовать исходный код программы, руководствуясь описаниями проблем, подготовленными экспертом системы САПФОР.

3. Автоматический выбор и применение шаблона

Использование системы САПФОР для распараллеливания прикладных программ [12-14]. позволило выделить проблемы, препятствующие эффективному распараллеливанию, для которых возможен автоматический поиск решения.

Одним из типов проблем является наличие в последовательной программе гнезд циклов с зависимостями по данным между различными итерациями циклов. В качестве шаблона решения проблем данного типа применяется разбиение гнезда циклов. Используемый алгоритм основан на алгоритме, предложенном в [15]. Чтобы цикл удовлетворял требованиям применимости алгоритма, системой САПФОР может выполняться автоматическое переупорядочивание операторов в цикле. В операторах, распределяемых по разным циклам, могут использоваться общие данные, вычисляемые внутри разделяемого цикла. В этом случае требующиеся вычисления либо будут продублированы в каждом из циклов, либо в программу будут добавлены вспомогательные массивы, в которых перед выполнением распределенных по циклам операторов будут размещены требующиеся данные.

Разбиение гнезда циклов также применяется для устранения проблемы, вызванной записью значений в разные элементы одного массива на одной итерации цикла (см. пример на **Рис.1**). Данная проблема характерна для отображения программ на системы с распределенной памятью. Перед выполнением вычислений данные должны быть распределены по процессорам таким образом, чтобы максимально снизить количество обменов. При параллельном выполнении итераций цикла процессорами системы необходимо, чтобы каждый процессор обладал необходимой ему частью распределенных данных. Для программы из примера на **Рис. 1** невозможно эффективно распределить данные и распараллелить выделенный цикл, несмотря на отсутствие в нем зависимостей по данным. На это указывает описание проблемы, полученное от эксперта. Результат успешного распараллеливания преобразованной программы приведен на **Рис. 2.**

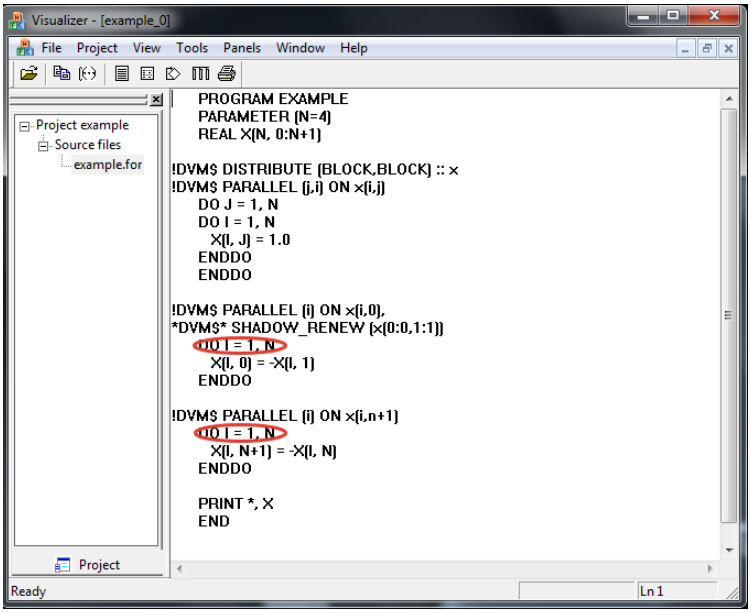

Рис. 2. Распараллеливание программы из примера на Рис. 1 после разбиения цикла

Разные массивы, к элементам которых выполняется доступ на одной итерации цикла, должны быть соответствующим образом выравнены. Это накладывает ограничения на возможные варианты распределения массивов по процессорам вычислительной системы. В случае если эксперту системы не удается выполнить автоматическое распределение данных, чтобы увеличить количество возможных вариантов распределения, выполняется максимально возможное разбиение циклов программы. В один цикл попадают операторы, в которых выполняется запись в одни и те же элементы одного и того же массива. После распараллеливания программы, выполняется объединение циклов на основе информации о выравнивании массивов и распределении вычислений, полученной от эксперта.

4. Распараллеливание прикладной задачи

Предложенный подход к автоматизации получения последовательной программы, эффективно отображаемой на кластер, был протестирован на двумерной задаче Каверна о течении несжимаемой жидкости или слабосжимаемого газа около прямоугольной выемки [16]. Ручное преобразование данной задачи описано в [12].

Автоматическое преобразование потребовало проведения 6 итераций.

Запуск эксперта системы САПФОР на исходной не преобразованной программе выявил следующую проблему: *«надо изменить оператор ввода-вывода на строке 520»*. В результате преобразования оператор вывода распределенного массива был заменен на оператор вывода скалярных переменных, содержащих предварительно скопированные элементы массива.

Следующий запуск эксперта выявил:

- четыре проблемы вида *«данные для записи витка ro(i,0) и ro(i, ny1) находятся на разных процессорах»*. В данном случае в указанных циклах не было зависимостей по данным, но на одной итерации выполнялась запись в разные элементы одного массива, указанные экспертом. Данная проблема аналогична проблеме, приведенной на **Рис. 1**. При проведении преобразований было обнаружено, что в каждом из циклов аналогичным образом организована запись в пять разных массивов. Так как на момент преобразований не была доступна информация о выравнивании данных массивов, каждый цикл был разбит максимальным образом на десять циклов.
- пять проблем вида *«данные для записи витка ro1(:, j) находятся на разных процессорах»*. Данная проблема указывает, что на итерации цикла записывается целиком измерение массива и свидетельствует о том, что не был распараллелен один из циклов гнезда. Причиной этого является не тесная вложенность циклов. В результате преобразования во внутренний цикл был внесен инвариант, расположенный между заголовками циклов.

Следующий запуск эксперта выявил четыре проблемы вида *«данные для записи витка ro1(i, j) и ro(i,j+1) находятся на разных процессорах»*. В указанных циклах были зависимости по данным, которые удалось устранить введением четырех временных массивов (по одному для каждого цикла) и разбиением каждого из циклов на два.

Следующий запуск эксперта выявил четыре проблемы вида *«часть гнезда, данные для записи витка tmp1(i+1,j) и e1(i,j) находятся на разных процессорах»*. Данные проблемы указывали на циклы из предыдущего запуска. Причиной их возникновение стало использование введенных временных массивов. Для решения данной проблемы необходимо разбить указанные циклы. Вынесение временного массива в отдельный цикл не решает данную проблему, так как в этом случае должен быть заведен новый временный массив с такой же структурой и операциями записи данных. Вынесению второго массива препятствовали зависимости по данным, которые удалось устранить введением еще четырех временных массивов (по одному для каждого цикла). Затем каждый из четырех циклов был разбит на два.

Следующий запуск эксперта выявил 2 проблемы: *«часть гнезда, измерение 1 массива ro1 не распределено»*. Причиной возникновения данных проблем является вызов внутри цикла оператора вывода данных. Устранить данную проблему невозможно, но она не влияет на распараллеливание остальной части программы. В результате данного запуска программа была успешно распараллелена.

В результате разбиения циклов было добавлено 36+4+4=44 новых цикла. На следующем шаге количество циклов было уменьшено за счет объединения циклов, выполненного на основе информации о выравнивании массивов и распределении вычислений, полученной от эксперта. Количество добавленных циклов было сокращено до 8.

5. Программная реализация

Выбор подходящих шаблонов и выполнение определяемых ими преобразований реализован в виде отдельного компонента – преобразователя системы САПФОР. Преобразователь взаимодействует с другими компонентами и пользователем системы двумя способами.

Во-первых, через базу данных (внутреннее представление) системы САПФОР.

Во-вторых, через специальные комментарии с префиксом `!PRG`, добавляемые в исходный код программы. Данные комментарии позволяют задавать дополнительные свойства программы, которые не удалось выявить анализаторам системы САПФОР, или уточнить описание проблемы, возникшей при распараллеливании и выделенной экспертом системы САПФОР, для более эффективного подбора шаблона ее решения. Пример комментариев, задаваемых перед циклами программы, приведен в **Таблице 1**.

Таблица 1. Пример специальных комментариев применяемых в системы САПФОР

Специальный комментарий	Использование
`!PRG INDEPENDENT (<variable-name>)`	Позволяет указать на отсутствие межитерационных зависимостей по данным в цикле.

`!PRG PRIVATE (<variable-name>)`
`!PRG FIRST_PRIVATE (<variable-name>)`
`!PRG LAST_PRIVATE (<variable-name>)`
	Позволяют указать приватные (приватные по входу или по выходу) переменные в цикле.

`!PRG FLOW|ANTI|OUTPUT <from> <to> [<dist>]`	Позволяет указать наличие зависимости по данным между двумя операторами и расстояние указанной зависимости.
`!PRG REDUCTION (<variable-name> (<op>) [, <array-name>, <index-number>]` `<op> ::= SUM|PRODUCT|MAX|MIN|AND|OR|EQV|NEQV|MAXLOC|MINLOC`	Позволяет указать редукционные переменные для цикла. Опциональный параметр `<array-name>` используется в случае редукции `MAXLOC` или `MINLOC`.

Преобразователь написан на языке программирования Си++ с использованием библиотеки Sage++ [17] для выполнения преобразований исходного кода программы.

6. Заключение

Эффективное автоматическое распараллеливание последовательных программ во многих случаях невозможно без выполнения их преобразования. Применение подхода, когда в процессе распараллеливания выполняется заранее фиксированная последовательность фаз анализа и преобразований, не учитывает особенности каждой конкретной программы. Проведенное таким образом автоматическое распараллеливание часто оказывается неэффективным. Итерационный процесс распараллеливания программ, поддерживаемый системой автоматизированного распараллеливания САПФОР, позволяет применять только те преобразования, которые необходимы для устранения проблем, препятствующих распараллеливанию.

Выполняемые перед распараллеливанием преобразования не выводят программу из класса последовательных программ, оставляя ее понятной для пользователя, детально незнакомого с особенностями параллельного программирования. Выполнение преобразований может выполняться в автоматическом или полуавтоматическом режиме.

Реализация данных возможностей в системе САПФОР позволила автоматизировать получение последовательных реализаций программ, эффективно отображаемых на современные кластеры автоматически распараллеливающим компилятором системы САПФОР, и была проверена на прикладной задаче моделирующей течение несжимаемой жидкости или слабосжимаемого газа около прямоугольной выемки.

Литература

1. PLUTO – An automatic parallelizer and locality optimizer for multicores. URL: http://pluto-compiler.sourceforge.net/ (дата обращения 28.11.2014).
2. Par4All – Par4All 1.4.5 documentation. URL: http://www.par4all.org/ (дата обращения 28.11.2014).
3. Source-to-Source Parallelizing Compiler – Appentra. URL: http://www.appentra.com/ (дата обращения 28.11.2014).

4. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В. Автоматическое распрраллеливание последовательных программ для многоядерных кластеров. // Труды Всероссийской научной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи”, Новороссийск, сентябрь 2010 г. – М.: Изд-во МГУ, 2010, С. 12-15.
5. Бахтин В.А., Бородич И.Г., Катаев Н.А., Клинов М.С. Ковалева Н.В., Крюков В.А., Поддерюгина Н.В. Диалог с программистом в системе автоматизации распараллеливания САПФОР // Вестник Нижегородского университета им. Н.И. Лобачевского. – Н. Новгород: Изд-во ННГУ, №5 (2) 2012, С. 242– 245.
6. Intel Parallel Studio. URL: http://software.intel.com/en-us/intel-parallel-studio-home (дата обращения 28.11.2014) .
7. Automatic Parallelization for Multi-processor/Multi-cores systems http://www.parallelsp.com/ (дата обращения 28.11.2014).
8. Юрушкин М.В., Петренко В.В., Штейнберг Б.Я., Алымова Е.В.,Абрамов А.А., Баглий А.П., Гуда С.А., Дубров Д.В., Кравченко Е.Н., Морылев Р.И., Нис З.Я., Полуян С.В., Скиба И.С., Шаповалов В.Н., Штейнберг О.Б., Штейнберг Р.Б. Диалоговый высокоуровневый оптимизирующий распараллеливатель (ДВОР) // Труды Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи”, Новоросийск, сентябрь 2010 г., – М.: Изд-во МГУ, с. 71-75.
9. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л.. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. – Вестник Южно-Уральского государственного университета, серия "Математическое моделирование и программирование", №18 (277), выпуск 12 – Челябинск: Издательский центр ЮУрГУ, 2012, с. 82-92.
10. Коновалов Н.А., Крюков В.А., Михайлов С.Н., Погребцов А.А.. «Fortran DVM - язык разработки мобильных параллельных программ», Ж. "Программирование", № 1, 1995, С. 49-54.
11. L. Almagor, Keith D. Cooper, Alexander Grosul, Timothy J. Harvey, Steven W. Reeves, Devika Subramanian, Linda Torczon, Todd Waterman. Finding effective compilation sequences // Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems, Washington, DC, June 11-13 2004, –
ACM New York, NY, USA 2004, P. 231-239
12. Бахтин В.А., Бородич И.Г., Катаев Н.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л. Распараллеливание с помощью DVM-системы некоторых приложений гидродинамики для кластеров с графическими процессорами //: Труды Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: поиск новых решений”, Новороссийск, сентябрь 2012 г. – М.: Изд-во МГУ, 2012. – С. 444–450.
13. Катаев Н.А., Клинов М.С., Поддерюгина Н.В. Преобразования последовательных программ при их распараллеливании с помощью системы САПФОР // Труды международной научной конференции “Параллельные вычислительные технологии (ПаВТ’2013)”,Челябинся, апрель 2013 г. – Изд-во ЮУрГУ, 2013, С. 387-393.
14. Бахтин В.А., Клинов М.С., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н. Автоматическое отображение программ на языке Фортран на кластеры с графическими процессорами // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика – Челябинск, Изв-во ЮУрГУ, 20414 №3(3) , С. 86-96.
15. Штейнберг Б.Я. Разбиение циклов для исполнения на суперкомпьютере с архитектурой перестраиваемого конвейера. // Искусственный интеллект. Научно-теоретический журнал. Институт проблем искусственного интеллекта НАНУ. Украина, Донецк, ДонДИШИ, “Наука и Освита”, 2002, №3 , С. 331-338.
16. Давыдов А.А., Четверушкин Б.Н., Е.В. Шильников “Моделирование течений несжимаемой жидкости и слабосжимаемого газа на многоядерных гибридных вычислительных системах” // Журнал “Вычислительная математика и математическая физика”, 2010, Т. 50, № 12, С. 2275-2284
17. pC++/Sage++ Home Page. URL: http://www.extreme.indiana.edu/sage/ (дата обращения: 01.12.2012).