ИПМ им.М.В.Келдыша РАН

Абрау-2019 • Труды конференции

И.А. Казаков, А.С. Колганов,
М. Ю. Кузнецов

Автоматизация преобразований последовательных программ для их последующего распараллеливания в системе SAPFOR

Рекомендуемая форма библиографической ссылки

Казаков И.А., Колганов А.С., Кузнецов М. Ю. Автоматизация преобразований последовательных программ для их последующего распараллеливания в системе SAPFOR // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2019. — С. 339-346. — URL: http://keldysh.ru/abrau/2019/theses/45.pdf doi:10.20948/abrau-2019-45

Размещена также презентация к докладу

Автоматизация преобразований последовательных программ для их последующего распараллеливания в системе SAPFOR

И.А. Казаков[1,2], А.С. Колганов[1], М. Ю. Кузнецов[1]

[1] ИПМ им. М.В. Келдыша РАН
[2] МГУ им. М.В. Ломоносова

Аннотация. Распараллеливание уже написанных последовательных программ часто требует значительных преобразований, так как в большинстве случаев данные программы хорошо оптимизированы для последовательного выполнения. Так как основной ресурс параллелизма содержится в циклах, то необходимо обеспечить в первую очередь их преобразование для устранения зависимостей и конфликтов, мешающих их анализу и распараллеливанию. К таким преобразованиям можно отнести: внос и вынос инварианта, объединение циклов в тесно-вложенные, расширение и сужение приватизируемых переменных, разбиение и слияние циклов и др. В данной статье будут рассмотрены два основных преобразования – расширение приватизируемых переменных, которое позволит устранить зависимость тесно-вложенного гнезда, и разделение циклов, которое позволит увеличить количество распараллеливаемых тесно-вложенных циклов. Полученные преобразования были проверены на некоторых программах NAS Parallel Benchmarks.

Ключевые слова: SAPFOR, распараллеливание программ, преобразование программ

Automating transformations of sequential programs for their parallelization in the SAPFOR system

I.A. Kazakov[1,2], A.S. Kolganov[1], M. U. Kyznetsov

[1] Keldysh Institute of Applied Mathematics RAS
[2] Lomonosov Moscow State University

Abstract. Parallelization of written sequential programs often requires significant transformations, since in most cases these programs are well optimized for sequential execution. It is necessary to provide their transformation in order to eliminate dependencies and conflicts that interfere with their analysis and parallelization, since the main resource of parallelism is contained in loops. The following transformations exist: introducing and carrying out of loop invariant, combining loops, expanding and

narrowing privatized variables in loops, splitting and merging loops, etc. This article will consider two major transformations – the expansion of privatized variables, which will eliminate the dependence of the closely nested loops, and the fission of loops, which will increase the number of parallelized closely nested loops. The result of transformations was tested on some programs of NAS Parallel Benchmarks suite.

Keywords: SAPFOR, program parallelization, program transformation

Введение

С прекращением роста частоты отдельно взятого процессора вырос интерес к параллельным вычислениям на многоядерных и многопроцессорных системах. При этом написание параллельных программ требует от программиста дополнительных знаний об особенностях аппаратуры, на которой будет выполняться программа. Задача усложняется из-за появления новых архитектур, таких как графические ускорители NVIDIA или сопроцессоры Intel. А для достижения параллелизма на всех уровнях требуется знание различных технологий параллельного программирования (CUDA, OpenMP, MPI, OpenACC и др.).

Ещё сложнее представляется задача по распараллеливанию последовательного кода. Это может потребовать значительных изменений или полной замены участков кода, которые предстоит выполнять параллельно. Значительным подспорьем здесь будут различные инструменты, осуществляющие помощь в распараллеливании, особенно, если они способны работать в автоматическом режиме.

Автоматизированное распараллеливание полагается на возможность выявления участков программного кода, которые могут быть выполнены параллельно. Это требует точного анализа исходного кода и выявления зависимостей по управлению и по данным. Однако, мощных средств анализа, порою, оказывается недостаточно для успешного распараллеливания. Зачастую необходимо использовать другие методы, например, спекулятивное выполнение участков программы. Другой подход, который успешно используется в современных оптимизирующих компиляторах – преобразование исходного кода программы.

DVM-система [1, 2], созданная в ИПМ им. М.В. Келдыша РАН при активном участии аспирантов и студентов факультета ВМК МГУ им. М.В. Ломоносова, предназначена для разработки параллельных программ научно-технических расчетов. В ней используются языки C-DVMH и Fortran-DVMH [3], являющиеся расширением стандартных языков C99 и Фортран95 спецификациями параллелизма, оформленными в виде директив компилятору. Важным дополнением к DVM-системе является система SAPFOR [4].

Один из способов расширить возможности системы SAPFOR – это улучшить способность определять циклы, которые возможно распараллелить. Как правило, циклы тесно связаны с массивами, с которыми они работают. Часто существуют зависимости между итерациями циклов, например, в случае

вычислений, основанных на ранее полученных значениях или в итеративных алгоритмах. Из-за подобных зависимостей распараллеливание цикла не представляется возможным.

Разрешить зависимости в циклах можно несколькими способами. Один из способов – выполнение преобразований данного цикла для устранения зависимостей. В данной статье будут рассмотрены два преобразования – расширение приватных переменных цикла и разделение циклов.

В разделе 2 приведено краткое описание системы SAPFOR. В разделе 3 дается обзор похожих на SAPFOR систем распараллеливания. Раздел 4 описывает рассматриваемые в данной статье автоматические преобразования циклов в системе SAPFOR.

Система SAPFOR

Система автоматизированного распараллеливания (SAPFOR) предназначена для распараллеливания программ на языке Фортран и Си. Эта система осуществляет анализ исходного кода последовательной программы и выполняет её преобразования в параллельную в модели DVMH [3] на основе этого анализа. Она ориентирована на ситуацию, когда пользователь имеет последовательную программу, а ему требуется эффективно решать задачи на параллельных ЭВМ разной архитектуры. В силу разных причин (например, из-за архитектурных отличий ЭВМ или отличий в их программном обеспечении), не всегда возможно создать единую параллельную программу, эффективную для всех этих ЭВМ. Даже если это принципиально возможно, пользователю может оказаться удобнее иметь несколько более простых вариантов параллельной программы вместо одного очень сложного.

Работа системы SAPFOR состоит из этапов анализа и преобразования, каждый из которых состоит из набора проходов. Проход – это функциональный модуль, решающий простую задачу. Проход делится на две фазы, сначала каждый файл исходной программы единообразно обрабатывается, а затем происходит объединение собранных данных. Причём наличие обеих фаз не является обязательным. Например, строящий граф вызовов функций проход сначала находит все функции, просмотрев все файлы, а потом, на основе полученной информации о вызовах, объединяет их в граф. А у прохода, который приводит все циклы к виду DO-ENDDO (к циклам без метки), нет части объединения собранных данных, так как такое преобразование циклов можно проводить независимо для каждого файла.

Как правило, проходы, осуществляющие какие-либо преобразования, опираются на проходы, проводящие предварительный анализ. Если говорить о преобразованиях для распараллеливания, то вопрос стоит не только об эффективности, но и о самой возможности распараллеливания. Если на этапе анализа были выявлены зависимости по данным в цикле, то его распараллеливание становится либо трудной, либо невозможной задачей. Поэтому, предъявляются высокие требования к проводимому анализу.

Обзор существующих решений

Существует множество различных преобразователей исходного кода. Рассмотрим некоторые из наиболее популярных.

ДВОР [5] – Диалоговый Высокоуровневый Оптимизирующий Распараллеливатель программ. Данная система предназначена для создания инструментов разработки параллельных программ на основе автоматического анализа и преобразования последовательных программ.

В данном проекте заложен новый подход к реализации переносимости параллельного программного обеспечения. Суть этого подхода состоит в том, что библиотеки эффективных прикладных программ разрабатываются не параллельными, а последовательными, но допускающими автоматический анализ информационных зависимостей и эффективное автоматическое или полуавтоматическое распараллеливание. При появлении новых вычислительных параллельных архитектур следует добавлять только новый генератор выходного кода ДВОР, ориентированный на новую архитектуру и заново преобразовывать программы библиотеки этим распараллеливателем.

ParaWise [6] – выполняет распараллеливание исходного кода на C или Фортран, генерируя на выходе переносимый и масштабируемый код на базе технологий MPI, OpenMP или их комбинации. Инструмент позволяет улучшать и настраивать генерируемый параллельный MPI и OpenMP код через интерактивное взаимодействие с пользователем. Отвечая на вопросы про свою программу, пользователь предоставляет контекст для ParaWise, помогая ему сгенерировать эффективный параллельный код. На данный момент проект больше не поддерживается.

ROSE [7] – это целая инфраструктура для проведения преобразований исходного кода и проведения различных анализов. Данная инфраструктура поддерживает такие языки как C (C89 и C98), C++ (C++98 и C++11), UPC, Фортран (77, 95, 2003), OpenMP, Java, Python, PHP. ROSE нацелена в основном на использование статического анализа, оптимизацию программ, свободное преобразование программ, машинно-зависимые оптимизации, сложную оптимизацию циклов и анализ эффективности.

В большинстве своём существующие инструменты предлагают необратимую трансформацию исходного последовательного кода в параллельную его версию. Парадигма DVM-системы подразумевает, что исходный код программы остаётся последовательным, а техническая реализация параллельности сокрыта от пользователя внутри DVM-директив. Так как выходным языком для системы SAPFOR является DVMH-язык, то все преобразования, которые проводит система, должны быть source-to-source (выходная программы должна быть на том же языке, например, Фортран).

Преобразования циклов: расширение приватных переменных и разделение цикла

Тесно-вложенным циклом будем называть такой цикл, который состоит из нескольких последовательно вложенных друг в друга циклов так, что только у последнего тело может состоять более чем из одного оператора. Далее в тексте не будет делаться различия между циклами и тесно-вложенными циклами.

При попытке распараллеливания последовательного цикла по переменной IT может возникнуть состояние гонки за значениями общих переменных. Рассмотрим следующий пример:

```
DO IT = 1, N_ITER
   DO I = 0, N
      B (I) = foo (I, IT)
   ENDDO
   DO I = 1, N - 1
      A (I) = A (I) + B (I - 1) / B (I + 1)
   ENDDO
ENDDO
```

В данном примере массиву B присваиваются значения, а затем они используются на этой же итерации. Также существует зависимость по массиву A для цикла по переменной IT. Поэтому этот цикл нельзя выполнять параллельно. Для устранения выше описанных проблем, можно применить преобразование – расширение приватных для цикла по IT массивов A и B, то есть увеличить размерность массива на одно измерение, размер которого будет равен N_ITER. После преобразования цикл будет выглядеть следующим образом:

```
DO IT = 1, N_ITER
   DO I = 0, N
      Bit (I, IT) = foo (I, IT)
   ENDDO
   DO I = 1, N - 1
      Ait (I, IT) = Ait (I, IT) + Bit (I - 1, IT) / Bit (I + 1, IT)
   ENDDO
ENDDO
```

Теперь выполнение цикла по IT не вызовет ситуации гонки за значения нового массивов B_{it} и A_{it}. Такое же преобразование можно проводить и для обычных скалярных переменных. Расширение переменной – это превращение её в массив, размерность которого совпадает с количеством итераций цикла, для которого проводится преобразование (данное преобразование можно применять для многомерного тесно-гнездового цикла).

Техническая реализация расширения приватных переменных требует, чтобы новые полученные массивы были динамическими (если до этого они были статическими), так как их размеры по расширяемым измерениям определяются количеством итераций циклов. В случае, если стартовое, конечное значения итеративной переменной цикла или шаг вычисляются с использованием вызовов функций, то будут заведены дополнительные переменные для того, чтобы хранить возвращаемые значения. Эти переменные будут использованы для определения границ циклов и размеров измерений расширяемых массивов, чтобы сохранить корректность программы и оставить количество вызовов функций прежним.

Другое преобразование – разделение циклов. Если в цикле есть несколько групп независимых друг от друга операторов, то этот цикл можно разделить на такое же количество циклов. Для определения групп независимых операторов используется Омега-тест и достигающие определения. Омега-тест предназначен для определения зависимостей между обращениями к массивам на разных итерациях цикла и в явном виде позволяет получить зависимости между операторами. Достигающие определения позволяют получить связи между оператором, в котором переменная используется, и операторами, в которых эта переменная изменяется.

Вновь обратимся к примеру, полученному при расширении приватных массивов A и B. В результате получилось два внутренних цикла, один из которых только использует результаты вычислений другого. Такой цикл можно разделить на два, в результате чего получаем следующий код:

```
DO IT = 1, N_ITER
   DO I = 0, N
      B_it (I, IT) = foo (I, IT)
   ENDDO
ENDDO

DO IT = 1, N_ITER
   DO I = 1, N - 1
      A_it (I, IT) = A_it (I, IT) + B_it (I - 1, IT) / B_it (I + 1, IT)
   ENDDO
ENDDO
```

Такое преобразование позволяет увеличить ресурс параллелизма программы. В данном случае, вместо выполнения N_ITER итераций цикла параллельно можно выполнять N_ITER * (N + 1) и N_ITER * (N - 1) итераций параллельно. Это особенно важно при отображении программы на графические процессоры и многоядерные процессоры внутри узла кластера.

Рассмотрим другой пример кода, для которого требуется разделение циклов:

```
DO I = 0, N
   A (1, I) = foo (I)
   A (2, I) = bar (I)
ENDDO
```

Так как система SAPFOR создает выходную программу в модели DVMH, то требуется, чтобы выполнялось правило собственных вычислений для операторов присваивания, то есть операции над операндами должны проводиться на процессоре, где эти операнды находятся. При выполнении цикла параллельно это означает, что все операнды из тела цикла должны находиться на одном процессоре. В данном примере операторы независимы и могут выполняться параллельно, но если оба измерения массива A будут распределены между процессорами, то нельзя гарантировать, что A (1, I) и A (2, I) попадут на один и тот же процессор. Однако, разделение этого цикла на два позволит корректно выполнять данный фрагмент кода параллельно.

Заключение

Система SAPFOR была дополнена механизмами расширения приватных массивов и скалярных переменных и разделения циклов. Для этого были использованы уже существующие её компоненты определения границ циклов, зависимостей между обращениями к массивам и вычисления достигающих определений. Эти два механизма были реализованы в качестве новых проходов преобразования последовательной программы.

Для проверки корректности проводимых преобразований было проведено тестирование на тестах BT и SP пакета NAS Parallel [8]. Преобразованные версии программ при запуске показывают одинаковые выходные результаты (для чисел с плавающей точкой с точностью до 13-го знака). Время работы исходной и преобразованной программы отличается не более чем на несколько процентов, что является допустимым в рамках погрешности замеров. При этом, полученные программы имеют больший потенциал для распараллеливания.

Разработанный механизм расширения приватных массивов позволяет распараллелить большее количество циклов, что существенно расширяет класс программ, которые могут быть успешно распараллелены системой SAPFOR. А разработанный механизм разделения циклов позволяет увеличить ресурс параллелизма распараллеливаемых циклов, что является критически важным при использовании графических процессоров или нитей центрального процессора.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проекты 17-01-00820 и 18-01-00851.

Литература

1. Коновалов Н.А., Крюков В.А. DVM-подход к разработке параллельных программ для вычислительных кластеров и сетей // Открытые системы, 2002. — № 3.
2. Описание DVM-системы. URL: http://dvm-system.org/
3. Документация по языкам C-DVMH и Fortran-DVMH. URL: http://dvm-system.org/ru/#dvm-docs
4. Kolganov A.S, Kataev N.A. The experience of using DVM and SAPFOR systems in semi-automatic parallelization of an application for 3D modeling in geophysics // Springer The Journal of Supercomputing, 2018. — P. 286-294. — doi: 10.1007/s11227-018-2551-y.
5. ДВОР. URL: http://www.ops.rsu.ru/about.shtml
6. Автоматический распараллеливатель ParaWise. URL: http://www.parallelsp.com/
7. Инфраструктура ROSE. URL: http://rosecompiler.org/?page_id=11
8. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.htm

References

1. Konovalov N.A., Krukov V.A. DVM-podhod k razrabotke parallel'nyh programm dlya vychislitel'nyh klasterov i setej // Otkrytye sistemy, 2002. — № 3.
2. DVM-system. URL: http://dvm-system.org/
3. C-DVMH and Fortran-DVMH languages. URL: http://dvm-system.org/ru/#dvm-docs
4. Kolganov A.S, Kataev N.A. The experience of using DVM and SAPFOR systems in semi-automatic parallelization of an application for 3D modeling in geophysics // Springer The Journal of Supercomputing, 2018. — P. 286-294. — doi: 10.1007/s11227-018-2551-y.
5. OPS. URL: http://www.ops.rsu.ru/about.shtml
6. ParaWise Automatic Parallelisation. URL: http://www.parallelsp.com/
7. ROSE Documentation. URL: http://rosecompiler.org/?page_id=11
8. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.htm