ИПМ им.М.В.Келдыша РАН

Абрау-2020 • Труды конференции

Н.А. Катаев, С.А. Черных

Автоматизация распараллеливания программ в системе SAPFOR

Рекомендуемая форма библиографической ссылки

Катаев Н.А., Черных С.А. Автоматизация распараллеливания программ в системе SAPFOR // Научный сервис в сети Интернет: труды XXII Всероссийской научной конференции (21-25 сентября 2020 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2020. — С. 362-376.
https://doi.org/10.20948/abrau-2020-24
https://keldysh.ru/abrau/2020/theses/24.pdf

Видеозапись выступления

Автоматизация распараллеливания программ в системе SAPFOR

Н.А. Катаев[1], С.А. Черных[2]

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

Аннотация. Система SAPFOR (System FOR Automated Parallelization) объединяет различные подходы, направленные на автоматизацию параллельного программирования. С одной стороны SAPFOR включает средства статического и динамического анализа и опирающийся на них автоматически распараллеливающий компилятор. С другой стороны для получения эффективной параллельной программы автоматически часто требуется выполнять предварительное преобразование исходной последовательной программы, а также указывать дополнительные свойства, которые не удается определить средствами анализа. Для этой цели в системе SAPFOR реализованы различные преобразования, которые могут быть выполнены автоматически по запросу пользователя. Для управления процессом распараллеливания используется интерактивная оболочка, которая позволяет объяснить пользователю решения, принимаемые системой, а также задать дополнительные свойства распараллеливаемой программы. В данной статей рассматривается подход к автоматизированному распараллеливанию C программ для систем с общей памятью, реализованный в SAPFOR. Рассматривается применение интерактивной оболочки в процессе распараллеливания и особенности связанные с отображением программы на ускорители в модели DVMH. В статье приводится сравнение автоматического распараллеливания некоторых программ из набора NAS Parallel Benchmarks 3.3.1 с ручным распараллеливанием, выполненным с помощью OpenCL.

Ключевые слова: автоматизированное распараллеливание, анализ программ, преобразование программ, графический интерфейс пользователя, SAPFOR, DVM, LLVM

Automation of program parallelization in SAPFOR

N.A. Kataev[1], S.A. Chernikh[2]

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

Abstract. Systems FOR Automated Parallelization (SAPFOR) combines various approaches to automation of parallel programming. On the one hand, SAPFOR implements static and dynamic analysis techniques and an automatic parallelizing compiler which relies on them. On the other hand, in order to increase performance of a generated parallel program, it is often necessary to carry out preliminary transformations of the original sequential program. Moreover, specifications of additional properties, which cannot be determined by analysis tools, are required in many cases. For this purpose, SAPFOR implements source-to-source transformations which can be performed automatically at the request of the user. Graphic user interface is used to maintain the parallelization process. It allows the user to investigate the decisions which the system makes, as well as to set additional properties of the original program. This paper presents an approach, which is implemented in SAPFOR, to automate parallelization of C programs for the compute systems with shared memory. We also highlight opportunities of the graphic user interface and consider application of DVMH directive based programming model to automate the mapping of programs to accelerators. This paper compares performance of the automatic parallelization of some programs from NAS Parallel Benchmarks 3.3.1 with manual parallelization based on OpenCL.

Keywords: automated parallelization, program analysis, program transformation, graphical user interface, SAPFOR, DVM, LLVM

1. Введение

Создание инструментов автоматизации разработки параллельных программ стало особенно актуальным в связи с развитием вычислительных устройств разной архитектур, для использования которых требуется применение различных технологий параллельного программирования. Исследования ведутся в разных направлениях: разрабатываются как полностью автоматически распараллеливающие компиляторы [1-3], так и инструменты, накладывающие ограничения на применение стандартных конструкций последовательных языков программирования и опирающиеся на дополнительные указания пользователя о свойствах последовательной программы [4, 5]. Большое внимание уделяется инструментам, которые позволяют исследовать распараллеливаемую программу, но принятие решений о распараллеливании оставляют за пользователем [6, 7].

Система автоматизированного распараллеливания SAPFOR (System FOR Automated Parallelization) [8] объединяет различные подходы к созданию инструментов параллельного программирования. С одной стороны, SAPFOR использует автоматически распараллеливающий компилятор для построения программ в модели DVMH [9, 10], тем самым беря на себя ответственности за описание заложенного в программе параллелизма средствами DVMH-языков. С другой стороны, автоматически распараллеливаемая программа должна быть предварительно подготовлена пользователем к последующему распараллеливанию. Система позволяет пользователю описать свойства

программы, необходимые для распараллеливания, но недоступные для анализа, а также предоставляет инструменты для преобразования программы на уровне исходного кода.

Основные результаты работы, рассматриваемые в статье следующие:

1. Интерактивная оболочка системы SAPFOR, позволяющая пользователю управлять процессом распараллеливания программ (исследованием их свойств, преобразованием и получением параллельных версий), и подход к ее применению для организации автоматизированного распараллеливания.
2. Автоматическое распараллеливание предварительно преобразованных программ для последующего их выполнения на графических ускорителях.
3. Экспериментальная проверка предложенных подходов на некоторых Си-программах из пакета NAS Parallel Benchmarks 3.3.1 [18, 19], сравнение полученных с помощью системы SAPFOR параллельных программ в модели DVMH с существующими OpenCL версиями.

Данная статья организована следующим образом. В разделе 2 рассматриваются возможности интерактивной оболочки системы SAPFOR применительно к организации процесса распараллеливания и к исследованию свойств распараллеливаемой программы. Раздел 3 посвящен автоматическому отображению предварительно преобразованных программ на вычислительные системы, использующие графические ускорители. Описывается подход к оптимизации расстановки директив DVMH для сокращения объема коммуникаций между центральным процессором и ускорителем. В разделе 4 коротко рассматриваются некоторые аспекты реализации компонент системы SAPFOR. Раздел 5 посвящен практическому применению системы SAPFOR при распараллеливании трех программ из пакета NAS Parallel Benchmarks 3.3.1. Выводы и планы дальнейшего развития системы, приведены в заключении.

2. Поддержание процесса распараллеливания программ

Система SAPFOR предоставляет пользователю возможность в интерактивном режиме управлять процессом распараллеливания. Интерактивная оболочка является отдельным компонентом системы и опирается на организацию клиент-серверного взаимодействия с ядром системы, включающим подсистемы анализа, преобразования и принятия решений по распараллеливанию программ. При этом основная функциональность системы также доступна в виде консольного приложения, что позволяет упростить использование SAPFOR, если сборка анализируемого проекта осуществляется с помощью автоматизированных средств, таких как Make. Например, существующий Makefile может быть изменен для инсрументации и оптимизации программы перед запуском динамического анализа [11].

Интерактивная оболочка позволяет пользователю исследовать доступный в программе параллелизм и наглядно показывает проблемы, препятствующие ее автоматическому распараллеливанию. Так как SAPFOR ориентирована на извлечение параллелизма уровня циклов, интерактивная оболочка обеспечивает навигацию по циклам в программе. Для каждого цикла в программе доступна следующая информация: является ли поток управления внутри тела цикла безопасным для распараллеливания, находится ли цикл в канонической форме [12], присутствуют ли в цикле обращения к памяти, порождающие зависимости по данным.

Безопасность потока управления означает отсутствие побочных эффектов у вызовов функций из тела цикла, в том числе отсутствие операций ввода/вывода, а также отсутствие нескольких выходов из тела цикла. По запросу пользователя может быть построен подграф графа вызовов и отображена последовательность вызовов, приводящая к небезопасному потоку управления.

Зависимости по данным между различными итерациями цикла будут классифицированы в соответствии с возможностями их устранения (редукционные и индуктивные переменные, переменные, зависимость по которым может быть устранена за счет приватизации данных или за счет организации конвейерного выполнения цикла). Описание обнаруженных зависимостей доступно в отдельной вкладке, отображающей используемую в программе память в виде дерева псевдонимов [13]. При необходимости уточнить свойства отдельных переменных, интерактивная оболочка помогает пользователю сформировать описание данных свойств в JSON-формате, которое затем может быть использовано в процессе анализа. Кроме того, системой могут быть учтены свойства программы, обнаруженные в результате динамического анализа.

На Рис. 1 приведен пример отображения свойств одного из циклов модельной программы, решающий систему уравнений в частных производных методов переменных направлений (ADI). Красным цветом выделены переменные (в данном случае массив A), которые вызывают зависимость в данном цикле. При этом справа отображается информация о найденной зависимости: ее тип (anti и flow), степень уверенности анализатора в наличии зависимости (must), причины ее возникновения (запись(store)/чтение(load)) и расстояние зависимости. В данном случае зависимость является регулярной и может быть устранена за счет организации конвейера, таким образом, цикл может быть распараллелен автоматически. Соответствующая информация будет отображена в сводном окне по всем циклам программы.

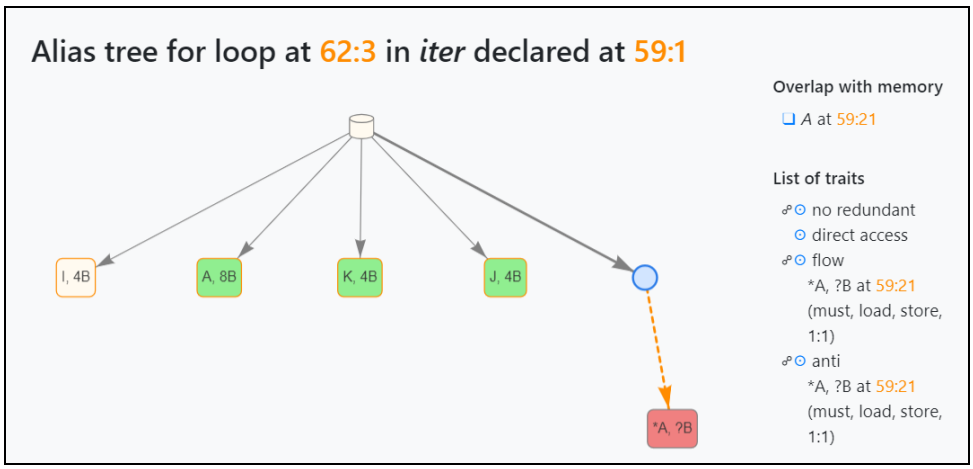

Рис 1. Отображение свойств цикла программы в графическом интерфейсе SAPFOR.

Обобщенная информация о циклах программы может быть визуализирована в виде отдельной вкладки. Дополнительно, непосредственно в исходном коде программы заголовок цикла сопровождается кратким описанием проблем, препятствующих распараллеливанию данного цикла, и по возможности рекомендациями по их устранению. Например, в случае наличия в цикле зависимостей по данным, вызванных вызовами функций, пользователю будет предложено пометить соответствующие вызовы функций для последующей подстановки их тел в исходном коде программы. В качестве альтернативы пользователю может быть предложено уточнить свойства указанных переменных вручную и не полагаться на возможности статического анализа SAPFOR.

Некоторые проблемы анализа могут быть устранены за счет задания свойств, влияющих на анализ всей программы в целом и описывающих общие факты, известные пользователю о программе. Примером одного из таких свойств является отсутствие побочного эффекта у математических функций стандартной библиотеки языка Си, которые в обычной ситуации сохраняют признак возникновения ошибки в специальной глобальной переменной. Если такое поведение не требуется при выполнении программы, то соответствующий побочный эффект может быть проигнорирован, в противном случае это препятствует распараллеливанию циклов, использующих математические операции.

Интерактивная оболочка также позволяет пользователю выбирать преобразования, которые должны быть выполнены над предварительно размеченным с помощью директив исходным кодом программы.

С использованием интерактивной оболочки процесс распараллеливания программы организован следующим образом:

1. Пользователь подготавливает программу к анализу и преобразованию средствами системы SAPFOR, задает опции

необходимые для синтаксического разбора программы (например, пути к заголовочным файлам, значения макросов). Текущая реализация диалогового режима ограничена возможностью анализа только одного файла с исходным кодом (допускается использование нескольких заголовочных файлов, подключаемых с помощью директив *include*). Данное ограничение относится в первую очередь к выполнению преобразований программ на уровне исходного кода, так как в этом случае требуется представление всей программы в виде единого дерева разбора. Также не желательно использование макроконструкций в области преобразования, так как дерево разбора Clang, используемое для преобразования, не содержит данных конструкций в явном виде, и оценить влияние макросов на допустимость преобразования часто оказывается невозможным. Например, макрос с одним и тем же именем может иметь разное определение в разных точках программы, и межпроцедурные преобразования, такие как подстановка функций, приведут к генерации некорректной программы. В этом случае SAPFOR принимает консервативное решение отказаться от выполнения преобразования.

2. Пользователь выполняет профилирование программы с целью определить наиболее времяемкие участки кода, которые требуется распараллелить в первую очередь. Для этих целей могут быть использован механизм интервалов, доступный в системе DVM, либо сторонние инструменты, такие как gprof. Дальнейшие работы по развитию интерактивной оболочки и системы SAPFOR в целом предполагают, что информация, полученная в результате профилирования программы, будет визуализирована и соотнесена с результатами последующего анализа программы на предмет возможности распараллеливания найденных времяемких фрагментов. На этом этапе также может быть выполнен динамический анализ программы, если пользователь считает, что возможностей статического анализа недостаточно.
3. Пользователь использует интерактивную оболочку, чтобы выполнить статический анализ программы (возможно дополнив его ранее полученными результатами динамического анализа, указав глобальные опции анализа или вручную задав свойства отдельных переменных и циклов программы) и изучить потенциал параллелизма доступного в программе и обнаруженного SAPFOR.
4. На основе результатов анализа, профилирования и рекомендаций системы пользователь выбирает фрагменты программного кода, которые должны быть распараллелены в первую очередь. Если распараллеливание данных фрагментов требует дополнительного

преобразования программы, пользователь может разметить исходный код, чтобы задать преобразования, которые требуется выполнить. Система SAPFOR автоматически преобразует программу по запросу пользователя, после чего процесс распараллеливания переходит снова к шагам (2) или (3). Если возможностей системы недостаточно для преобразования программы, то может потребоваться выполнение некоторых преобразований вручную.

5. Если программа приведена к виду, допускающему автоматическое распараллеливание, то пользователь может разметить области программы, которые требуется распараллелить с помощью директивы SAPFOR *region,* и система SAPFOR автоматически сгенерирует параллельный вариант программы либо с использованием OpenMP, либо в модели DVMH. В случае отсутствия директив *region* распараллеливание выполняется для всей программы.

3. Автоматическое отображение программ на ускорители

Одной из ключевых особенностей системы SAPFOR является возможность автоматического распараллеливания программ для последующего их выполнениях на ускорителях. Распараллеливание выполняется в модели DVMH с использованием языка C DVMH, являющегося директивным расширением языка программирования C. Распараллеливание заключается, во-первых, в необходимости определить циклы программы, которые могут быть выполнены параллельно, а, во-вторых, определить данные которые должны быть переданы на ускоритель и возвращены на процессор после вычислений. Непосредственно за передачу данных отвечает система поддержки выполнения программ, которая руководствуется указанными спецификациями, а также информацией, доступной во время выполнения программы, для оптимизации необходимых передач данных.

Допускается параллельное выполнение циклов *for*, представленных в канонической форме, для которых может быть гарантирована безопасность потока управление и отсутствие неустранимых зависимостей по данным, возможно использование редукционных переменных и переменных, которые могут быть объявлены приватными.

Циклы верхнего уровня (с учетом вызовов процедур), допускающие параллельное выполнение, помещаются внутрь вычислительного региона (директива DVMH *region*). Непосредственно перед каждым параллельным циклом помещается спецификация *parallel*. В случае тесной вложенности циклов, каждый из которых допускает параллельное выполнение, пространство итераций всех циклов объединяется для увеличения уровня

параллелизма (например, для распараллеливания гнезда, образованного двумя тесно вложенными циклами, на системы с общей памятью будет использована спецификация *parallel(2)*). Для описания устранимых зависимостей используются спецификации *reduction* и *private*.

Расстановка директив спецификации обменов данных между центральным процессором и ускорителем выполняется в несколько этапов.

1. На основе анализа графа потока данных для каждого параллельного цикла определяются входные (вычисленные до начала выполнения цикла), выходные (используемые после выхода из цикла) и локальные данные.
2. Анализируется граф вызовов функций в направлении снизу-вверх (сначала рассматриваются вызываемые функции, затем функции их вызывающие).
    a. Для параллельных циклов одинакового уровня вложенности, которые расположены внутри одной функции, исследуется граф потока данных между выполнениями каждого цикла. Если выходные данные первого цикла и входные данные второго цикла не используются ни на одном пути в графе управления между данными циклами, то директивы актуализации данных могут быть объединены и размещены до выполнения первого цикла (*actual*) и после завершения второго цикла (*get_actual*).
    b. Граф потока данных циклов, которые не являются параллельными и тело которых содержит параллельные циклы, анализируется с целью размещения директив актуализации данных непосредственно до и после выполнения всех итераций последовательных циклов. Примером таких циклов являются итерационные циклы, Проводимый анализ позволяет загрузить данные на GPU до начала итерационного процесса и вернуть обратно на центральный процессор после его завершения.
    c. Граф потока данных всей функции анализируется с целью расположить директивы актуализации данных до и после точки каждого вызова обрабатываемой функции.

Таким образом, удается сократить объем передаваемых данных между выполнениями разных вычислительных регионов. Дополнительно выполняется объединение соседних вычислительных регионов, чтобы сократить накладные расходы на инициализацию региона.

4. Детали реализации

Система SAPFOR опирается на LLVM [14] версии 7.1.0 для выполнения анализа программ, фронтенд Clang применяется для реализации преобразований исходного кода. При этом допустимость выполнения данных преобразований проверяется с использованием как высокоуровневого дерева разбора Clang AST, так и низкоуровневого LLVM IR [12].

Проводимый статический анализ должен принимать консервативные решения и не допускать преобразований, которые потенциально могут привести к некорректному поведению программы. Система SAPFOR предполагает наличие зависимостей по данным в циклах или пересечений по памяти между указателями, если не может быть доказано обратное.

Чтобы сократить количество ситуаций, не поддающихся точному анализу, предварительно выполняются дополнительные преобразования LLVM IR [13]. Это также позволяет задействовать доступные в LLVM анализы, такие как определение редукционных переменных и переменных индукции. Дополнительные преобразования LLVM также необходимы для делинеаризации массивов, линеаризованных в LLVM IR [15]. Делинеаризация массивов необходима для реализации анализа зависимостей по данным.

Интерактивная оболочка реализована в виде плагина к кроссплатформенному редактору Microsoft Visual Studio Code [16]. Разработанный плагин выступает в роли клиентского приложения, посылающего запросы к серверной части системы SAPFOR, ответственной за выполнение анализа, преобразования и распараллеливания последовательной программы. Обмен данным осуществляется в виде сообщений, представленных в JSON-формате. Реализация интерактивной оболочки в виде плагина, а не отдельного приложения, позволяет воспользоваться широкими возможностями Visual Studio Code по поддержке процесса разработки программ на языках C/C++ и сосредоточится на разработке возможностей, необходимых для поддержания интерактивного процесса распараллеливания [17].

5. Распараллеливание программ NAS NPB 3.3

Мы воспользовались системой SAPFOR и разработанной интерактивной оболочкой, чтобы выполнить распараллеливание программ CG, BT, EP из пакета NAS Parallel Benchmarks 3.3.1 [18, 19]. Для оценки ускорения полученного после распараллеливания данных программ в модели DVMH был использован 6-ядерный процессор Intel Xeon CPU E5-1660 v2, 3.70 GHz с включенным Hyper Threading (2 потока на ядро, всего 12 потоков) и отключенным Turbo Boost. В качестве ускорителя была использована GPU GeForce GTX 1660 Ti. Компиляция программ выполнялась с помощью Intel Compiler 19.0.2.187 и NVIDIA Compiler 10.2,

использовалась опция оптимизации -O3. На Рис. 2 приведено ускорение программ относительно исходных последовательных программ. Автоматическое распараллеливание выполнялось для преобразованных последовательных версий программ. Также приведено ускорение программ, вручную написанных с помощью OpenCL [19]. Результаты были получены для трех классов A, B и C, определяющих размер (в порядке увеличения) используемых данных для каждой программы [18].

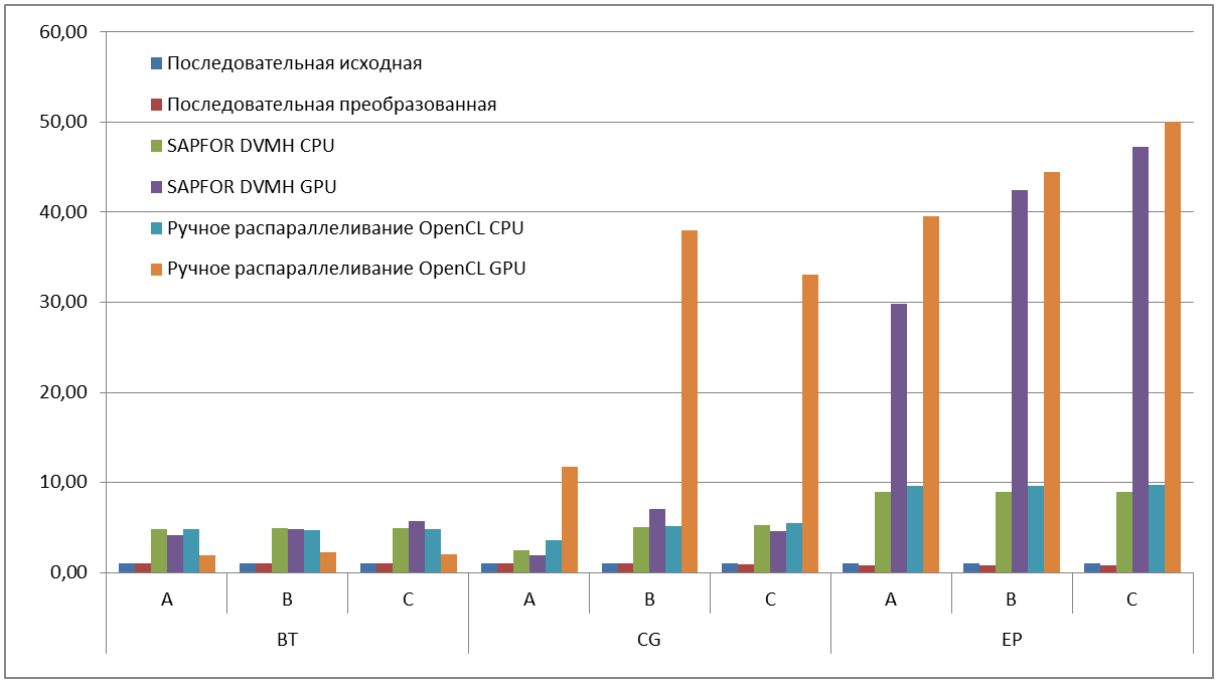

Рис. 2. Ускорение программ из пакета NAS Parallel Benchmarks (NPB)

Стоит отметить, что программы, разработанные вручную с помощью OpenCL, существенно отличаются от исходных программ, что в общем случае вызывает дополнительные трудности по сопровождению и модификации программ. В тоже время программы, распараллеленные в модели DVMH, остаются написанными в последовательном стиле, а параллелизм в них выражен за счет высокоуровневых спецификаций, заданных в виде директив. Существенное преимущество ручного распараллеливания программы CG над автоматическим вызвано в первую очередь дополнительной векторизацией некоторых внутренних циклов и использованием общей памяти ускорителя.

Основным преобразованием, которое потребовалось для распараллеливания программ является подстановка функций. Это связано с тем, что текущая реализация межпроцедурного анализа в SAPFOR достаточно грубо оценивает участки памяти, используемые и изменяемые внутри вызовов функций. Чтобы обнаружить приватные массивы в программах BT и EP был дополнительно применен динамический анализ.

Программа EP также потребовала некоторых ручных преобразований, связанных с устранением редукционного массива фиксированного размера (заменой его на несколько скалярных переменных) и устранением приватного массива большого размера (объединение цикла инициализации массива и использования массива в один цикл с перевычислением требующихся элементов на каждой итерации цикла). Редукционные массивы в данный момент не поддерживаются в языке C DVMH в случае использования ускорителей. Суммарный размер всех копий массива после его приватизации превышал объем памяти, доступной на GPU, что делало невозможным выполнение программы.

6. Заключение

В данной работе рассмотрена организация процесса распараллеливания программ в системе автоматизированного распараллеливания SAPFOR. Разработанная интерактивная оболочка упрощает участие пользователя в процессе распараллеливания, позволяя ему изучить принимаемые системой решения и при необходимости дать соответствующие рекомендации системе. Пользователь может влиять как на выполнение анализа программы, подсказывая системе наличие тех или иных свойств программы, так и на выполнение преобразований исходного кода, необходимых для распараллеливания программы. Полученная в итоге программа на последовательном языке программирования может быть автоматически распараллелена системой SAPFOR. В данной работе рассматривается результат успешного применения предложенного подхода к распараллеливанию нескольких программ на языке C из набора программ NAS Parallel Benchmarks 3.3.1. Приводятся результаты распараллеливания в модели DVMH для систем с общей памятью (многоядерные процессоры и ускорители) и их сравнение с ручным распараллеливанием с помощью OpenCL. Проведенные исследования показали необходимость расширить набор поддерживаемых преобразований исходного кода и уделить дополнительное внимание межпроцедурному анализу программ.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проекты 18-01-00851-а, 20-01-00631-а.

Литература

1. Grosser T., Groesslinger A., Lengauer C. Polly - performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters, 22(04):1250010 (2012)
2. Grosser T., Hoefler T. Polly-ACC Transparent compilation to heterogeneous hardware // ICS '16: Proceedings of the 2016 International Conference on

Supercomputing June 2016. — P. 1-13 (2016). — https://doi.org/10.1145/2925426.2926286
3. Bondhugula U., Hartono A., Ramanujam J., Sadayappan P. A practical automatic polyhedral parallelizer and locality optimizer // SIGPLAN Notices. — 43(6):101-113 (2008)
4. H. Vandierendonck et al. The Paralax Infrastructure: Automatic Parallelization with a Helping Hand // PACT, 2010.
5. Baghdadi R., Beaugnon U., Cohen A., Grosser T., Kruse M., Reddy C., Verdoolaege S., Betts A., Donaldson A.F., Ketema J., Absar J., Haastregt S., Kravets A., Lokhmotov A., David R., Hajiyev E. Pencil: A platform-neutral compute intermediate language for accelerator programming // Proceedings of the 2015 International Conference on Parallel Architecture and Compilation (PACT), PACT '15. — P. 138–149, Washington, DC, USA, 2015. IEEE Computer Society — https://doi.org/10.1109/PACT.2015.17
6. Kim M., Kim H., Luk C.-K. Prospector: A Dynamic Data-Dependence Profiler To Help Parallel Programming // 2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10), 2010.
7. Intel Parallel Studio. URL: https://software.intel.com/en-us/parallel-studio-xe
8. Клинов М.С., Крюков В.А. Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник Нижегородского университета им. Н.И. Лобачевского, № 2, 2009. — С. 128–134.
9. Konovalov, N.A., Krukov, V.A, Mikhajlov, S.N., Pogrebtsov, A.A.: Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. vol. 21, no. 1, 1995. — P. 35-38.
10. 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
11. Катаев Н.А., Смирнов А.А., Жуков А.Д. Динамический анализ зависимостей по данным 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. 400-412. — URL: http://keldysh.ru/abrau/2019/theses/62.pdf doi:10.20948/abrau-2019-62
12. Катаев Н.А., Козырев В.И. Преобразование программ на высокоуровневом языке программирования на основе результатов анализа низкоуровневого представления программ в системе SAPFOR // Параллельные вычислительные технологии – XIII международная

конференция, ПаВТ'2019, г. Калининград, 2–4 апреля 2019 г. Короткие статьи и описания плакатов. — Челябинск: Издательский центр ЮУрГУ, 2019 — С. 251-262.
13. Kataev N.A. Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, vol 965. — Springer, Cham, 2018 — P. 487-499. — https://doi.org/10.1007/978-3-030-05807-4_41
14. Lattner, C., Adve, V. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04). — Palo Alto, California, 2004
15. Катаев Н.А., Василькин В.Н. Восстановление обращений к многомерным массивам 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. 413-423. — URL: http://keldysh.ru/abrau/2019/theses/90.pdf doi:10.20948/abrau-2019-9
17. Катаев Н.А., Колганов А.С., Смирнов А.А. Поддержка интерактивности в системе САПФОР // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2017. — С. 243-249. — URL: http://keldysh.ru/abrau/2017/57.pdf doi:10.20948/abrau-2017-57
18. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.html.
19. Seo S., Jo G., Lee J. Performance Characterization of the NAS Parallel Benchmarks in OpenCL // 2011 IEEE International Symposium on. Workload Characterization (IISWC), 2011. — P. 137-148

References

1. Grosser T., Groesslinger A., Lengauer C. Polly - performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters, 22(04):1250010 (2012)
2. Grosser T., Hoefler T. Polly-ACC Transparent compilation to heterogeneous hardware // ICS '16: Proceedings of the 2016 International Conference on Supercomputing June 2016. — P. 1-13 (2016). — https://doi.org/10.1145/2925426.2926286
3. Bondhugula U., Hartono A., Ramanujam J., Sadayappan P. A practical automatic polyhedral parallelizer and locality optimizer // SIGPLAN Notices. — 43(6):101-113 (2008)
4. H. Vandierendonck et al. The Paralax Infrastructure: Automatic Parallelization with a Helping Hand // PACT, 2010.

5. Baghdadi R., Beaugnon U., Cohen A., Grosser T., Kruse M., Reddy C., Verdoolaege S., Betts A., Donaldson A.F., Ketema J., Absar J., Haastregt S., Kravets A., Lokhmotov A., David R., Hajiyev E. Pencil: A platform-neutral compute intermediate language for accelerator programming // Proceedings of the 2015 International Conference on Parallel Architecture and Compilation (PACT), PACT '15. — P. 138–149, Washington, DC, USA, 2015. IEEE Computer Society — https://doi.org/10.1109/PACT.2015.17
6. Kim M., Kim H., Luk C.-K. Prospector: A Dynamic Data-Dependence Profiler To Help Parallel Programming // 2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10), 2010.
7. Intel Parallel Studio. URL: https://software.intel.com/en-us/parallel-studio-xe
8. Klinov M.S., Krukov V.A. Avtomaticheskoe rasparallelivanie Fortran-programm. Otobrazhenie na klaster // Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo, № 2, 2009. — P. 128–134.
9. Konovalov, N.A., Krukov, V.A, Mikhajlov, S.N., Pogrebtsov, A.A.: Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. vol. 21, no. 1, 1995. — P. 35-38.
10. Bakhtin V.A., Klinov M.S., Krukov V.A., Podderiugina N.V., Pritula M.N., Sazanov Iu.L. Rasshirenie DVM-modeli parallelnogo programmirovaniia dlia klasterov s geterogennymi uzlami // Vestnik Iuzhno-Uralskogo gosudarstvennogo universiteta, seriia "Matematicheskoe modelirovanie i programmirovanie", №18 (277), vypusk 12 — Cheliabinsk: Izdatelskii tsentr IuUrGU, 2012. — P. 82-92
11. Kataev N.A., Smirnov A.A., Zhukov A.D. Dinamicheskii analiz zavisimostei po dannym v sisteme SAPFOR // Nauchnyi servis v seti Internet: trudy XXI Vserossiiskoi nauchnoi konferentsii (23-28 sentiabria 2019 g., g. Novorossiisk). — M.: IPM im. M.V.Keldysha, 2019. — S. 400-412. — URL: http://keldysh.ru/abrau/2019/theses/62.pdf doi:10.20948/abrau-2019-62
12. Kataev N.A., Kozyrev V.I. Preobrazovanie programm na vysokourovnevom iazyke programmirovaniia na osnove rezultatov analiza nizkourovnevogo predstavleniia programm v sisteme SAPFOR // Parallelnye vychislitelnye tekhnologii – XIII mezhdunarodnaia konferentsiia, PaVT'2019, g. Kaliningrad, 2–4 aprelia 2019 g. Korotkie stati i opisaniia plakatov. — Cheliabinsk: Izdatelskii tsentr IuUrGU, 2019 — P. 251-262.
13. Kataev N.A. Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, vol 965. — Springer, Cham, 2018 — P. 487-499. — https://doi.org/10.1007/978-3-030-05807-4_41
14. Lattner, C., Adve, V. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International

Symposium on Code Generation and Optimization (CGO'04). — Palo Alto, California, 2004
15. Kataev N.A., Vasilkin V.N. Vosstanovlenie obrashchenii k mnogomernym massivam v sisteme SAPFOR // Nauchnyi servis v seti Internet: trudy XXI Vserossiiskoi nauchnoi konferentsii (23-28 sentiabria 2019 g., g. Novorossiisk). — M.: IPM im. M.V.Keldysha, 2019. — S. 413-423. — URL: http://keldysh.ru/abrau/2019/theses/90.pdf doi:10.20948/abrau-2019-9
16. Visual Studio Code. URL: https://code.visualstudio.com/
17. Kataev N.A., Kolganov A.S., Smirnov A.S. Podderzhka interaktivnosti v sisteme SAPFOR // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18-23 sentiabria 2017 g., g. Novorossiisk). — M.: IPM im. M.V.Keldysha, 2017. — P. 243-249. — URL: http://keldysh.ru/abrau/2017/57.pdf doi:10.20948/abrau-2017-57
18. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.html.
19. Seo S., Jo G., Lee J. Performance Characterization of the NAS Parallel Benchmarks in OpenCL // 2011 IEEE International Symposium on. Workload Characterization (IISWC), 2011. — P. 137-148