УДК 004

СРЕДСТВА ИНТЕРАКТИВНОГО ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ В СИСТЕМЕ SAPFOR

Н. А. Катаев[0000-0002-7603-4026]

ИПМ им. М.В. Келдыша РАН, г. Москва;

kataev_nik@mail.ru

Аннотация

Автоматизация параллельного программирования затрагивает различные этапы в разработке параллельной программы, начиная от профилирования исходной программы, ее преобразования и приведения к виду, допускающему эффективное распараллеливание, и заканчивая построением параллельной версии программы и ее последующей оптимизацией. Немалое значение имеет выбор целевой модели параллельного программирования, с одной стороны, позволяющей задействовать разнообразие существующих на данный момент аппаратных ресурсов, а с другой, упрощающей разработку автоматизированных средств и позволяющей пользователю изучить решения, принимаемые системой автоматизированного распараллеливания. Система SAPFOR (System FOR Automated Parallelization) объединяет различные подходы, направленные на автоматизацию программирования, и позволяет пользователю принимать активное участие в процесс распараллеливания программ. Кроме того, распараллеливание выполняется в модели DVMH, позволяющей разрабатывать эффективные параллельные программы для гетерогенных вычислительных кластеров.

В настоящей статье рассмотрен подход к автоматизированному распараллеливанию программ, реализованный в системе SAPFOR. Отдельное внимание уделено архитектуре системы и реализации подсистемы интерактивного взаимодействия с пользователем. Рассмотрено применение интерактивной оболочки в процессе распараллеливания и приведены результаты распараллеливания некоторых программ из набора NAS Parallel Benchmarks 3.3.1 с ручным распараллеливанием, выполненным с помощью OpenCL.

***Ключевые слова**: анализ программ, преобразование программ, автоматизация распараллеливания, графический интерфейс пользователя, SAPFOR, DVM, LLVM*

ВВЕДЕНИЕ

Многочисленные исследования, проводимые в направлении разработки средств автоматизации параллельного программирования, а также разнообразие существующих и разрабатываемых инструментов показывают, что для снижения сложности параллельного программирования и минимизации ошибок, связанных с распараллеливанием программ, необходимы комплексный подход и разработка целого набора программных средства. 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]. Поэтому при разработке системы автоматизированного распараллеливания SAPFOR (System FOR Automated Parallelization) [8] мы ведем исследования по трем ключевым направлениям:

- Исследование характеристик и свойств распараллеливаемой программы. Помимо выявления участков кода, которые могут быть выполнены параллельно, и обнаружения препятствующих этому зависимостей по данным необходимо определять потенциал от возможного распараллеливания участков программы, в том числе выполняя профилирование программы на различных наборах входных данных. Кроме того, необходимо понимать взаимосвязь тех или иных участков кода, уметь выявлять фрагменты программы, которые связаны с выполнением разных задач и, более того, не могут выполняться одновременно на одних и тех же данных. Такого рода проблемы многовариантности [9] программных комплексов, приводят к тому, что одновременное распараллеливание разных участков кода

может требовать принятия противоречивых решений, например, при построении распределения данных, и найти компромиссное решение, учитывающие все варианты, может быть невозможно. Но в связи с невозможностью одновременного выполнения тех или иных путей в программе это, в конечном счете, и не требуется.

- Автоматическое распараллеливание «хорошо» написанных потенциально параллельных программ. Возможности существующих автоматически распараллеливающих компиляторов по-прежнему сильно ограничены. Распараллеливание в первую очередь выполняется на системы с общей памятью (CPU, GPU). При этом достаточно успешные и активно развивающиеся подходы, такие как модель многогранников (полиэдральная модель), накладывают существенные ограничения на структуру распараллеливаемых участков программы [1–3]. Зависимость выполнения программы от входных данных, сложный граф потока управления, использование косвенной адресации и адресной арифметики существенно затрудняют применение данных подходов и могут сделать невозможным эффективное распараллеливание в контексте всей программы. Проблема стоит особенно остро, если цель распараллеливания – гетерогенный вычислительный кластер, и требуется выполнить не только распределение вычислений, но и распределение данных, используемых в вычислениях. Одним из способов преодоления возникающих сложностей является следование идеям неявного параллелизма [10] при разработке последовательных программ и создании соответствующих автоматически распараллеливающих компиляторов. В этом случае, если программа написана на некотором подмножестве стандартного языка параллельного программирования и, возможно, содержит в исходном коде специального вида указания, описывающие ее свойства, не поддающиеся автоматическому анализу, сложность автоматического распараллеливания снижается [4, 5].
- Автоматизированное приведение программ к потенциально параллельному виду. Часто, даже имея возможность аккуратно проанали-

зировать программу и точно выявить и описать все присущие ей свойства, влияющие на распараллеливание, распараллелить программу в исходном виде оказывается невозможным либо такое распараллеливание оказывается неэффективным. Могут потребоваться как преобразования, связанные с ограничениями, накладываемыми теми или иными средствами параллельного программирования (например, требование канонической формы циклов, накладываемое OpenMP или требование необходимости тесной вложенности циклов при объединении их пространств итераций и распараллеливании гнезда циклов целиком), так и преобразования, связанные с устранением зависимостей по данным в программе. При этом стоит отметить, что для распараллеливания программ часто требуется выполнять похожие преобразования [11], в то же время выбор порядка их выполнения оказывает значительное влияния на возможность дальнейшего распараллеливания программы [12]. Однако компилятор не всегда способен автоматически определить требуемые преобразования и выстроить их в правильную последовательность. В этом случае решением может быть предоставление пользователю системы набора автоматически выполняемых преобразований и возможности управлять процессом их применения к распараллеливаемой программе. Это позволяет значительно снизить трудоемкость приведения программы к виду, допускающему автоматическое распараллеливание.

При разработке программных комплексов, направленных на автоматизацию распараллеливания, не менее важным является выбор целевого языка параллельного программирования, в терминах которого создается параллельная программа. Система SAPFOR в качестве целевой модели параллельного программирования опирается на модель DVMH [13, 14].

Данная модель является высокоуровневым расширением стандартных последовательных языков программирования C и Fortran. Так как для задания спецификаций параллельного программирования используются невидимые для стандартных компиляторов директивы (специального вида комментарии и

прагмы), то параллельная программа остается в то же время последовательной, что существенно облегает ее дальнейшее сопровождение, а также делает возможным проанализировать решения, принятые системой. Исследовать исходный код автоматически сгенерированных параллельных программ часто оказывается затруднительно, даже если выходным является язык высокого уровня (сгенерированный код может существенно отличаться от исходного) [3], не говоря уже о ситуации, когда компилятор генерирует код в терминах низкоуровневого представления [1, 2].

Более того, DVMH-языки описывают параллелизм на нескольких уровнях. Параллелизм уровня целого кластера задает распределение данных и вычислений между узлами кластера. Параллелизм внутри узла распределяет данные и вычисления как между ядрами центральных процессорных устройств, так и позволяет задействовать ускорители, имеющиеся в узле.

Кроме того, DVM система, реализующая DVMH модель, включает целый ряд инструментов, облегчающих пользователю работу с параллельной программой, в том числе инструменты, направленные на исследование эффективности и корректности параллельной программы.

Система SAPFOR ориентирована на активное участие пользователя в процессе распараллеливания и наряду со средствами анализа и преобразования программы должна включать средства организации интерактивного взаимодействия с пользователем.

Данные средства должны позволять как описывать свойства программы, которые не удается выявить автоматически, так и управлять процессом распараллеливания в целом, задавая глобальные опции анализа программы и выбирая необходимые преобразования. При этом пользователь, в первую очередь, должен иметь возможность управлять преобразованиями программы на уровне исходного кода, так как явное конфигурирование последовательности оптимизационных проходов, например, так, как предлагается в [15], может оказаться для прикладного программиста слишком трудоемким. Инструмент, описываемый в статье [15], нацелен на оптимизацию программ для выполнения на встраиваемых системах, которые выпускаются огромными партиями и для которых

тонкая настройка последовательности оптимизационных проходов, применяемой к низкоуровневому представлению программы, является оправданной.

В данной статье рассматривается архитектура системы SAPFOR с точки зрения ее ориентированности на решение задач, стоящих перед системой. Отдельное внимание уделено организации интерактивного взаимодействия с пользователем и возможности создания различных систем визуализации на основе ядра системы SAPFOR, предоставляющего единый интерфейс для управления процессом распараллеливания.

Статья организована следующим образом. В разделе 2 рассмотрены архитектура системы SAPFOR и организация процесса распараллеливания с использованием подсистемы интерактивного взаимодействия с пользователем. Раздел 3 посвящен обзору возможностей интерактивной оболочки системы SAPFOR. В разделе 4 рассмотрены особенности реализации компонент системы SAPFOR и особенности взаимодействия между компонентами системы. Раздел 5 посвящен практическому применению системы SAPFOR при распараллеливании трех программ из пакета NAS Parallel Benchmarks 3.3.1 [16], также выполнено сравнение с ручным распараллеливанием данных программ с помощью OpenCL. Выводы приведены в заключении.

АРХИТЕКТУРА СИСТЕМЫ SAPFOR

Систему SAPFOR можно условно разделить на три части: ядро системы, отвечающее за анализ, преобразование и распараллеливание программы; средства динамического анализа программы; средства интерактивного взаимодействия с пользователем.

Ядро системы включает:

- фронтенды с поддерживаемых языков высокого уровня, генерирующие внутреннее представление программы в системе,
- средства анализа и преобразования внутреннего представления программы,
- средства для выполнения преобразований на уровне исходного кода программы,

- средства отображения результатов анализа, выполненного над внутренним представлением программы на конструкции высокоуровневого языка программирования.

Отображение результатов анализа между разными уровнями абстракции в системе SAPFOR особенно важно, так как для повышения качества и полноты анализа используется низкоуровневое представление программы в виде LLVM IR [17]. В то же время система должна отображать результаты своей работы в терминах исходного высокоуровневого языка программирования. Более того, использование высокоуровневого представления наряду с низкоуровневым позволяет учитывать некоторые особенности исходного языка (например, ограничения на допустимость пересечений по памяти между изменяемыми параметрами процедур в языке Fortran), которые могут теряться при переходе к LLVM IR.

В статье [18] подробно описан механизм сопоставления разных уровней представления программы, а также объяснен выбор LLVM в качестве базовой инфраструктуры для разработки SAPFOR.

Средства динамического анализа программ в первую очередь включают средства для анализа зависимостей по данным, которые не могут быть доказаны или опровергнуты во время статического анализа. В этом случае динамический анализ опирается на инструментацию программы на уровне LLVM IR, что позволяет, во-первых, выполнять выборочную инструментацию, а, во-вторых, выполнять ряд предварительных оптимизаций программы, не теряя при этом полноту проводимого анализа [19]. Свойства программы, выявленные в результате запуска ее инструментированной версии, передаются в ядро системы и учитываются в процессе всех анализов и преобразований исходной программы.

Для исследования свойств программы также могут быть полезны средства профилирования, входящие в состав DVM системы, но на данный момент они интегрированы в состав SAPFOR не полностью, и пользователю приходится вручную определять наиболее времяемкие участки кода, требующие распараллеливания в первую очередь.

Подсистема интерактивного взаимодействия с пользователем предназначена не только для того, чтобы отобразить результаты анализа программы и исследовать ее информационную структуру. В ее задачу также входит обеспечить

пользователя механизмом удобного задания дополнительных свойств программы и управления преобразованиями программы, выполняемыми на уровне исходного кода. В крайнем случае, пользователь может получить возможность вносить изменения в программу вручную.

С точки зрения архитектуры системы SAPFOR и организации взаимодействия между компонентами системы подсистема интерактивного взаимодействия является независимой и взаимодействует с ядром системы на основе фиксированного интерфейса. Фактически управление процессом распараллеливания опирается на организацию клиент-серверного взаимодействия, в котором ядро системы выступает в роли сервера и определяет интерфейс, доступный клиенту. Обмен информацией осуществляется через передачу сообщений в JSON формате. Таким образом, становится возможна реализации различных интерактивных сред в зависимости от требуемой функциональности и окружения, в котором должна функционировать система. Это могут быть как отдельное приложение с графическим интерфейсом, так и расширение для некоторой среды разработки.

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 интерактивной оболочки 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 нежелательно использование макроконструкций в области преобразования, так как дерево разбора Clang, используемое для преобразования, не содержит данные конструкций в явном виде, и оценить влияние макросов на допустимость преоб-

разования 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 использованы 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.

ПОДСИСТЕМА ИНТЕРАКТИВНОГО ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ

В данный момент интерактивная оболочка реализована в виде расширения для кроссплатформенного редактора Microsoft Visual Studio Code [20] и позволяет пользователю исследовать доступный в программе параллелизм и наглядно показывает проблемы, препятствующие ее автоматическому распараллеливанию.

Редактор Microsoft Visual Studio Code предоставляет удобный механизм для разработки расширений и имеет открытые исходные коды. Кроме того, он позволяет запускать расширения на удаленной рабочей станции, если к ней возможен доступ с использованием SSH. В этом случае, даже при работе напрямую на рабочей станции, становится возможным получить преимущества от интерактивного взаимодействия с системой SAPFOR.

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 [21] и может ли автоматически распараллеливающий компилятор системы SAPFOR распараллелить данный цикл, в том числе устранив имеющиеся в нем 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.

Пример, показывающий сводные результаты анализа для циклов модельной программы и решающий систему уравнений в частных производных методов переменных направлений (ADI), приведен на рис. 1. Зеленым подсвечены циклы программы, допускающие автоматическое распараллеливание. Видно, что итерационный цикл не может быть распараллелен в том числе из-за наличия в нем дополнительных операторов выхода из цикла и операторов ввода вывода.

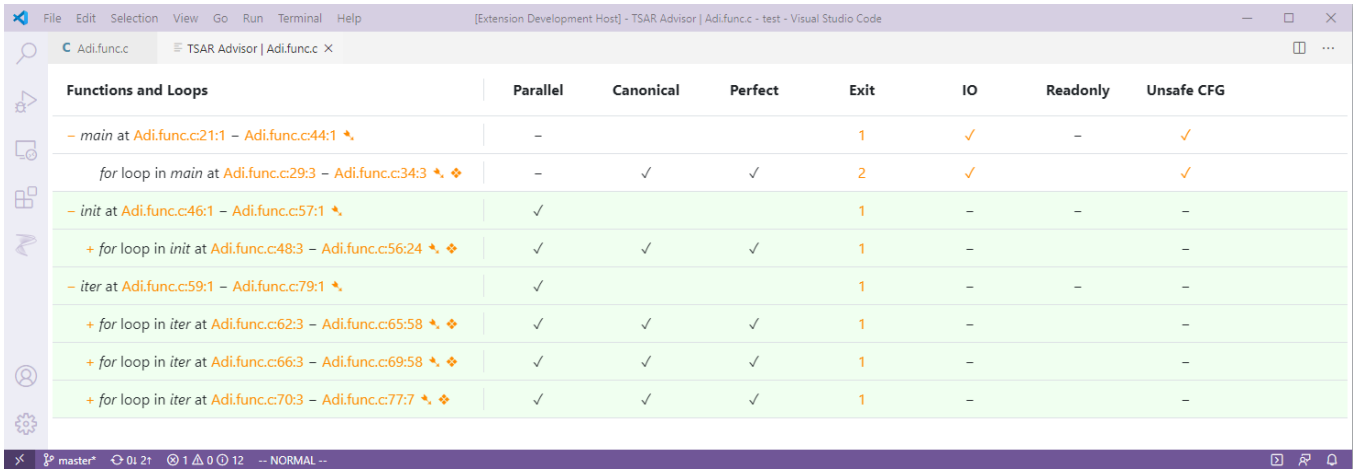

Рисунок 1. Сводная информация по циклам программы ADI

На рис. 2 приведен пример графа вызовов для всей программы ADI. Зеленым выделены функции, содержащие параллельные циклы. Серым цветом показаны библиотечные функции, о которых у SAPFOR может не быть достаточно информации, если их реализация не доступна в виде исходных кодов, и которые не могут быть преобразованы на уровне исходного кода, так как расположены за пределами программы пользователя.

С другой стороны, SAPFOR использует информацию о таких функциях, которую гарантирует реализация стандартной библиотеки. Это позволяет в определенных ситуациях гарантировать отсутствие побочного эффекта у вызываемой функции или быть уверенным, что возвращаемый функцией указатель уникален и не может указывать на память, ранее выделенную в программе.

При выборе дуги графа вызовов, справа отображается список всех вызовов функции, в которую ведет выбранная дуга.

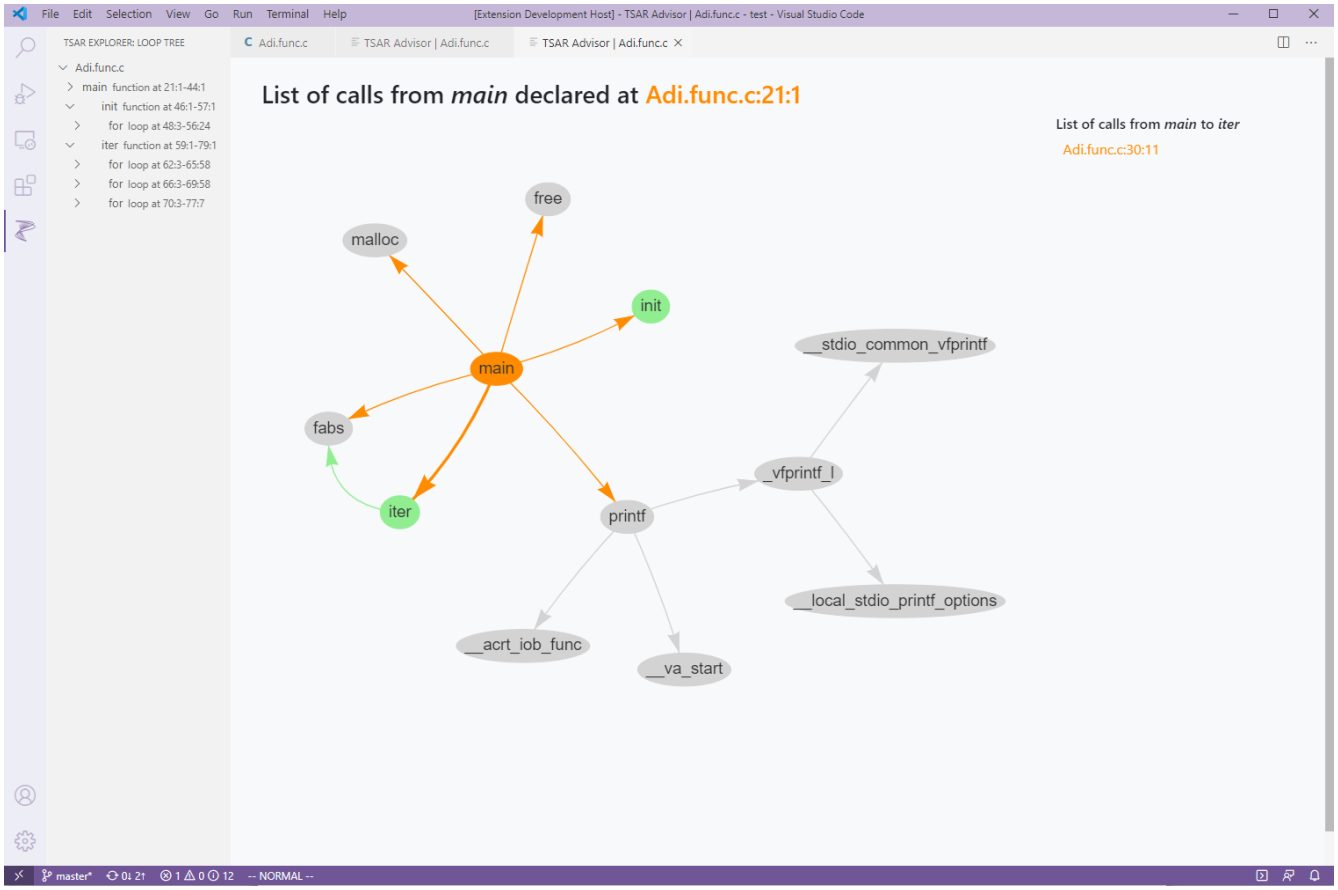

Рисунок 2. Граф вызовов для программы ADI

Для каждого цикла в отдельности можно исследовать, присутствуют ли в нем обращения к памяти, порождающие зависимости по данным. 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 [18].

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. 3 приведен пример отображения свойств одного из циклов программы 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 (свойство Parallel для цикла на рис. 1).

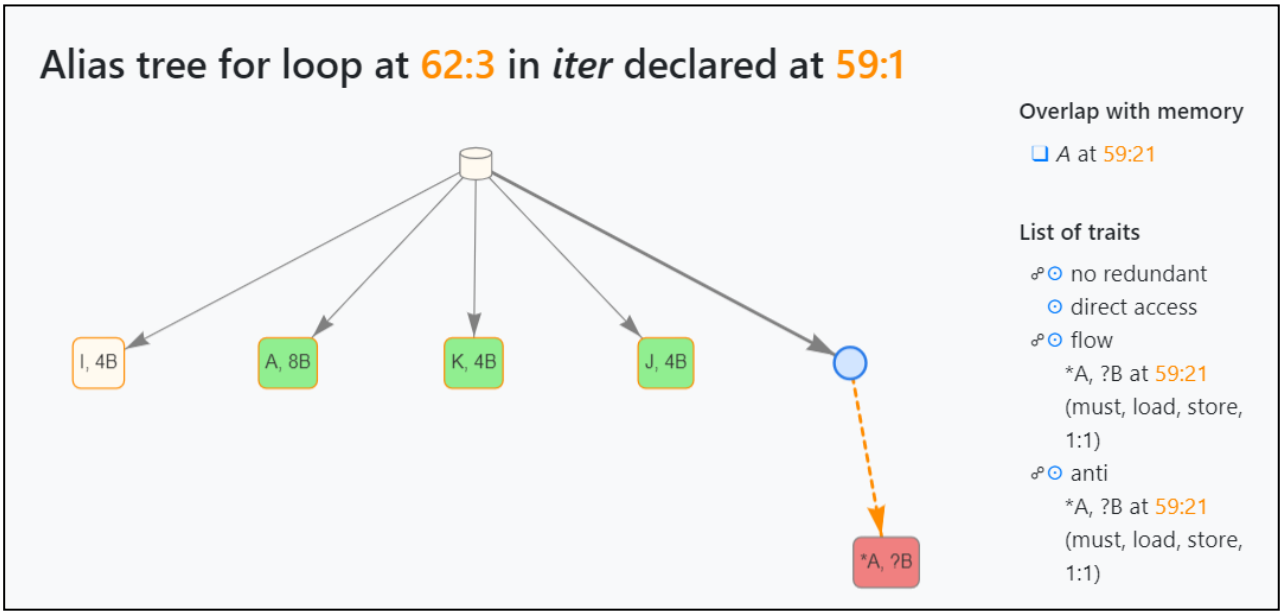

Рисунок 3. Свойства переменных одного из циклов программы ADI

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 C, 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.

Задать глобальные свойства анализа, а также дополнительные опции компиляции программы и указать файлы, содержащие результаты динамического анализа, можно в начале обработки программы системой SAPFOR (рис. 4).

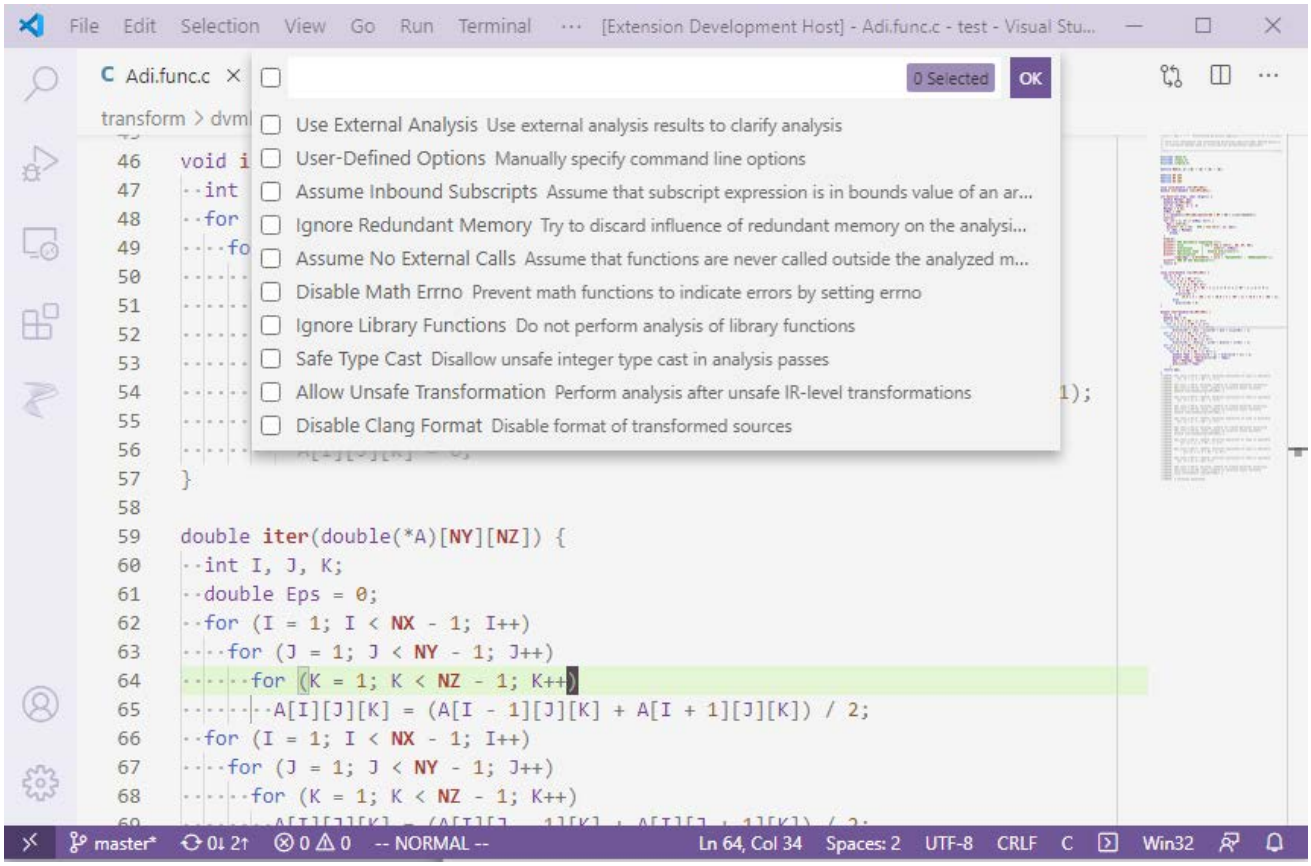

Рисунок 4. Окно задания глобальных свойств анализа программы

ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ МЕЖДУ КОМОПНЕНТАМИ SAPFOR

Ядро системы SAPFOR основано на инфраструктуре компиляторов LLVM, в данный момент используется LLVM версии 11. Абстрактное синтаксическое дерево, генерируемое компилятором Clang (Clang AST), используется для выполнения преобразований на уровне исходного кода, а также для получения свойств программы, зависящих от конкретного языка программирования (например, структура организации циклов программы и факт их тесной вложенности).

Интерактивная подсистема взаимодействия с пользователем разрабатывается на языке TypeScript в виде расширения для Visual Studio Code.

Схема процесса анализа и распараллеливания программы, а также взаимодействия ядра системы с подсистемой интерактивного взаимодействия изображена на рис. 5.

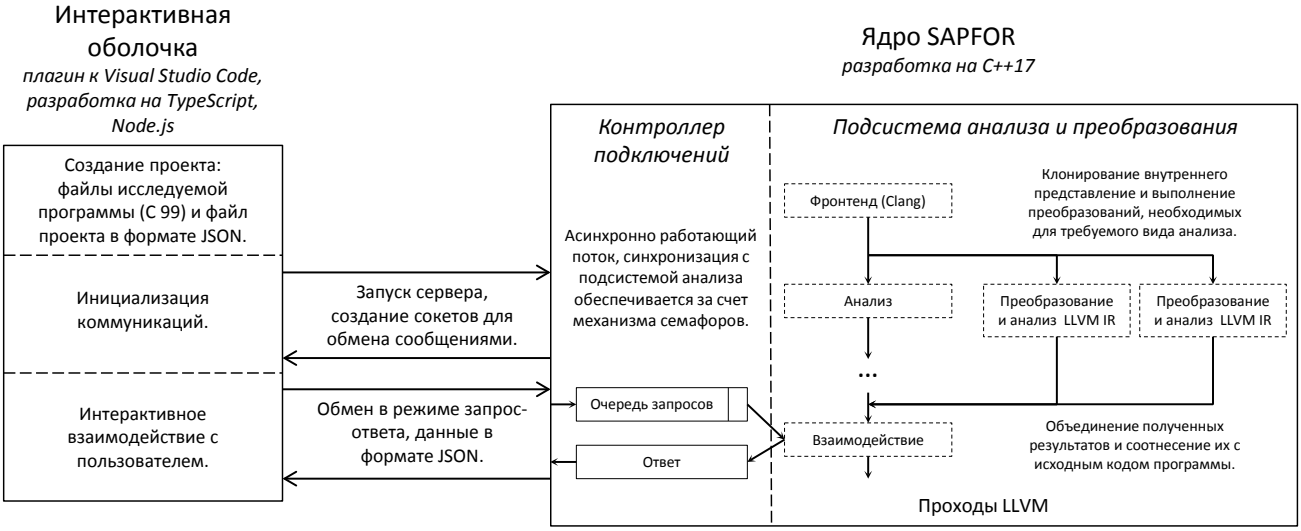

Рисунок 5. Схема взаимодействия компонент системы SAPFOR

Перед началом анализа и преобразования программы пользователь создает проект, содержащий файлы с исходным кодом, а также файл в формате JSON, описывающий правила компиляции каждой отдельной единицы трансляции. После того, как будет установлено соединение с ядром системы, обмен информацией между интерактивной оболочкой и ядром системы будет осуществляться в виде передачи сообщений в формате JSON. Для этого будут использованы механизм сокетов и соответствующие его реализации, доступные в Node.js и C++.

Получив файлы с исходными кодами программы, ядро системы SAPFOR строит Clang AST и LLVM IR, а также устанавливает соответствие между представлениями анализируемых объектов программы на данных уровнях абстракции.

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 [18]. 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 [22]. Делинеаризация массивов необходима для реализации анализа зависимостей по данным.

Система SAPFOR позволяет одновременно запустить несколько потоков анализа программы и тем самым выполнять разные последовательности преобразований низкоуровневого LLVM IR, наиболее подходящие для определения разных свойств программы. Для этого создаются копии LLVM IR исходной программы, а после завершения анализа каждой преобразованной копии обобщенная информация о полученных результатах отображается сначала на исходный LLVM IR, а затем на объекты исходной программы, представленные в виде Clang AST.

РАСПАРАЛЛЕЛИВАНИЕ ПРОГРАММ ИЗ НАБОРА NAS PARALLEL BENCHMARKS 3.3.1

Мы воспользовались системой SAPFOR и разработанной интерактивной оболочкой, чтобы выполнить распараллеливание программ CG, BT, EP из пакета NAS Parallel Benchmarks 3.3.1 [16, 23]. 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. 6 приведено ускорение программ относительно исходных последовательных программ. Автоматическое распараллеливание выполнялось для преобразованных последовательных версий программ. Также приведено ускорение программ, вручную написанных с помощью OpenCL [23]. Результаты были получены для трех классов A, B и C, определяющих размер (в порядке увеличения) используемых данных для каждой программы [16].

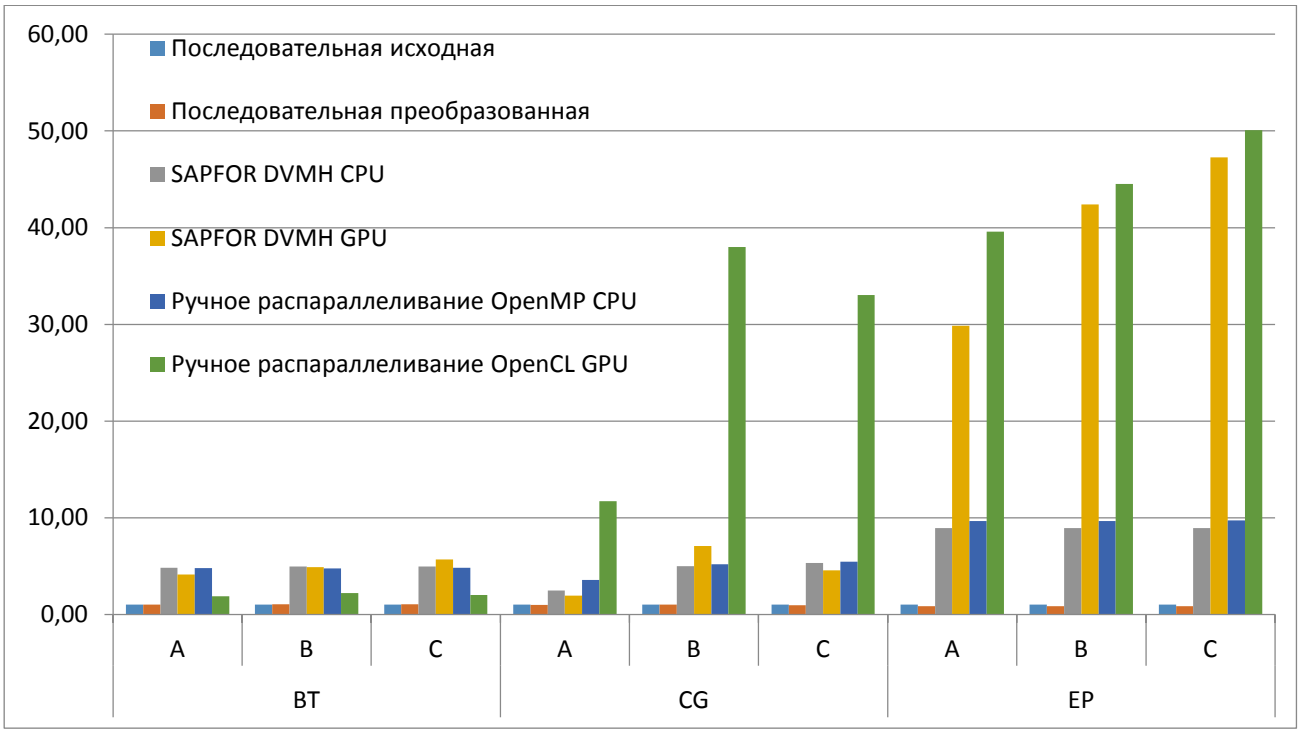

Рисунок 6. Ускорение выполнения программ из пакета NAS Parallel Benchmarks (NPB)

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 то же 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. В Таблицах 1 и 2 приведен фрагмент кода до и после преобразования соответственно.

Таблица 1. Фрагмент программы EP, требующий ручного преобразования.

```
double x[2*NK]; // NK is 65536, NP depends on an input
#pragma dvm parallel(1) private(x) ...
for (k = 1; k <= NP; k++) {  ...
 for (i = 0; i < NK; i++) {    ...
   x[i] = r46 * (*x4);
 }
 for (i = 0; i < NK; i++) {
   x1 = 2.0 * x[2 * i] - 1.0;
   x2 = 2.0 * x[2 * i + 1] - 1.0;
   t1 = x1 * x1 + x2 * x2;
   ...
 }
 ...
}
```

Таблица 2. Фрагмент программы EP после ручного преобразования.

```
#pragma dvm parallel(1) ...
for (k = 1; k <= NP; k++) { ...
  for (i = 0; i < NK; i++) {
    double x_2i, x_2i1;
    { ...
      x_2i = r46 * (*x4);
    }
    { ...
      x_2i1 = r46 * (*x4);
    }
    x1 = 2.0 * x_2i - 1.0;
    x2 = 2.0 * x_2i1 - 1.0;
    ...
  }
  ...
}
```

Директивы языка C DVMH, приведенные в Таблицах 1 и 2, были добавлены в исходный код программы EP системой SAPFOR автоматически.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрены архитектура системы SAPFOR, а также ключевые моменты, которые были приняты во внимание при проектировании системы. Система автоматизированного распараллеливания SAPFOR разрабатывается с целью добиться снижения трудоемкости ручного распараллеливания программ и уменьшить количество возникающих при этом ошибок, вносимых программистом. В этой связи перед системой ставятся три основные задачи: исследование свойств распараллеливаемой программы, автоматическое распаралле-

ливание некоторого класса программ, относящихся к так называемым «хорошо» написанным потенциально параллельным программам, и разработка средств автоматизации выполнения преобразований программ для приведения программы к потенциально параллельному виду.

Система SAPFOR ориентирована на тесное взаимодействие с пользователем в процессе анализа и преобразования программ. Поэтому наряду со средствами как статического, так и динамического анализа система включает интерактивную подсистему, реализующую взаимодействие с пользователем в процессе распараллеливания. Использование фиксированного интерфейса, предоставляемого ядром системы, упрощает разработку интерактивной оболочки и делает возможным ее реализацию в виде, определяемым, в том числе, средой функционирования системы. Таким образом, могут быть реализованы как расширение для существующих сред разработки программного обеспечения, так и отдельное приложение с графическим интерфейсом пользователя.

В данной работе рассмотрен пример реализации системы интерактивного взаимодействия в виде расширения кроссплатформенного редактора Microsoft Visual Studio Code. Разработанная интерактивная оболочка упрощает участие пользователя в процессе распараллеливания, позволяя ему изучить решения, принимаемые системой, 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 статье представлены результаты успешного применения предложенного подхода к распараллеливанию нескольких программ на языке C из набора программ NAS Parallel Benchmarks 3.3.1. Приведены результаты распараллеливания в модели DVMH для систем с общей памятью (многоядерные процессоры и ускорители) и их сравнение с ручным распараллеливанием с помощью OpenCL и OpenMP.

Исходные коды системы SAPFOR доступны на GitHub [24].

СПИСОК ЛИТЕРАТУРЫ

1. *Grosser T., Groesslinger A., Lengauer C.* Polly – performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters. 2012. Vol. 22, No. 04. 1250010. https://doi.org/10.1142/S0129626412500107.

2. *Grosser T., Hoefler T.* Polly-ACC Transparent compilation to heterogeneous hardware // ICS '16: Proceedings of the 2016 International Conference on 373 Supercomputing June 2016. P. 1–13. https://doi.org/10.1145/2925426.2926286.

3. *Bondhugula U., Hartono A., Ramanujam J., Sadayappan P.* A practical automatic polyhedral parallelizer and locality optimizer // ACM SIGPLAN Notices. 2008. Vol. 43, No. 6. P. 101–113. https://doi.org/10.1145/1379022.1375595.

4. *Vandierendonck H., Rul S., De Bosschere K.* The Paralax Infrastructure: Automatic Parallelization with a Helping Hand // Proceedings of the 19th international conference on Parallel architectures and compilation techniques (PACT'10). 2010. P. 389–400. https://doi.org/10.1145/1854273.1854322.

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. IEEE Computer Society. Washington, DC, USA, 2015. P. 138–149. 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. P. 1–6.

7. Intel Parallel Studio. URL: https://software.intel.com/en-us/parallel-studioxe.

8. *Клинов М.С., Крюков В.А.* Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. № 2. С. 128–134.

9. *Бахтин В.А., Жукова О.Ф., Катаев Н.А., Колганов А.С., Крюков В.А., Кузнецов М.Ю., Поддерюгина Н.В., Притула М.Н., Савицкая О.А., Смирнов А.А.* Распараллеливание программных комплексов. Проблемы и перспективы // Труды XX Всероссийской научной конференции «Научный сервис в сети Интернет», Но-

вороссийск, Россия, 17–22 сентября 2018 г. М.: ИПМ им. М.В. Келдыша, 2018. С. 63–72.
URL: http://keldysh.ru/abrau/2018/theses/33.pdf. https:// doi.org/10.20948/abrau-2018-33

10. *Hwu W.-m., Ryoo S., Ueng S.-Z., Kelm J.H., Gelado I., Stone S.S., Kidd R.E., Baghsorkhi S.S., Mahesri A.A., Tsao S.C., Navarro N., Lumetta S.S., Frank M.I., Patel S.J.* Implicitly parallel programming models for thousand-core microprocessors // Proceedings of the 44th annual Design Automation Conference (DAC '07), ACM, New York, NY, USA. 2007. P. 754–759. https://doi.org/10.1145/1278480.1278669.

11. *Blume W., Eigenmann R*. Performance analysis pf parallelizing compilers on the Perfect Benchmarks programs // IEEE Transactions on Parallel and Distributed Systems. 1992. Vol. 3, Issue 6. P. 643–656. https://doi.org/10.1109/71.180621.

12. *Wolfe M.* Scalar vs. parallel optimizations // CSETech. 210. 1990. URL: https://classes.cs.uoregon.edu/16S/cis410parallel/Documents/scalar-paralleloptimizations-wolfe.pdf

13. *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. 1995. V. 21, No. 1. P. 35–38.

14. *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». 2012. №18 (277), выпуск 12. Челябинск: Издательский центр ЮУрГУ. С. 82–92.

15. *Kulkarni P., Zhao W., Moon H*., Cho K., Whalley D., Davidson J., Bailey M., Paek Y., Gallivan K. Finding effective optimization phase sequences // Proceedings of the 2003 ACM SIGPLAN Conference on Languages, Tools, and Compilers for Embedded Systems. 2003. P. 12–23. https://doi.org/10.1145/780731.780735.

16. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.html.

17. *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. https://doi.org/10.1109/CGO.2004.1281665.

18. *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. 2018. V. 965. Springer, Cham. P. 487–499. doi:10.1007/978-3-030-05807-4_41.

19. *Катаев Н.А., Смирнов А.А., Жуков А.Д.* Определение зависимостей по данным средствами динамического анализа системы SAPFOR // Электронные библиотеки. Тематический выпуск «Научный сервис в сети Интернет». Часть 1. 2020. Том 23. № 3. С. 473–493. https://doi.org/10.26907/1562-5419-2020-23-3-473-493.

20. Visual Studio Code. URL: https://code.visualstudio.com/, last accessed 2020/11/25.

21. OpenMP Application Programming Interface. https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-1.pdf, last accessed 2020/11/25.

22. *Катаев Н.А., Василькин В.Н.* Восстановление многомерной формы обращений к линеаризованным массивам в системе SAPFOR // Электронные библиотеки. Тематический выпуск «Научный сервис в сети Интернет». Часть 2. 2020. Том 23. № 4. С. 770–787. https://doi.org/10.26907/1562-5419-2020-23-4-770-787.

23. *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. https://doi.org/10.1109/IISWC.2011.6114174.

24. SAPFOR. https://github.com/dvm-system.

INTERACTION WITH THE USER IN THE SAPFOR SYSTEM

N. A. Kataev

Keldysh Institute of Applied Mathematic RAS

kataev_nik@mail.ru

Abstract

Automation of parallel programming is important at any stage of parallel program development. These stages include profiling of the original program, program transformation, which allows us to achieve higher performance after program parallelization, and, finally, construction and optimization of the parallel program. It is also important to choose a suitable parallel programming model to express parallelism available in a program. On the one hand, the parallel programming model should be capable to map the parallel program to a variety of existing hardware resources. On the other hand, it should simplify the development of the assistant tools and it should allow the user to explore the parallel program the assistant tools generate in a semi-automatic way. The SAPFOR (System FOR Automated Parallelization) system combines various approaches to automation of parallel programming. Moreover, it allows the user to guide the parallelization if necessary. SAPFOR produces parallel programs according to the high-level DVMH parallel programming model which simplify the development of efficient parallel programs for heterogeneous computing clusters. This paper focuses on the approach to semi-automatic parallel programming, which SAPFOR implements. We discuss the architecture of the system and present the interactive subsystem which is useful to guide the SAPFOR through program parallelization. We used the interactive subsystem to parallelize programs from the NAS Parallel Benchmarks in a semi-automatic way. Finally, we compare the performance of manually written parallel programs with programs the SAPFOR system builds.

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

REFERENCES

1. *Grosser T., Groesslinger A., Lengauer C*. Polly - performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters. 2012. Vol. 22, No. 04. 1250010. https://doi.org/10.1142/S0129626412500107.

2. *Grosser T., Hoefler T.* Polly-ACC Transparent compilation to heterogeneous hardware // ICS'16: Proceedings of the 2016 International Conference on 373 Supercomputing June 2016. P. 1–13. https://doi.org/10.1145/2925426.2926286.

3. *Bondhugula U., Hartono A., Ramanujam J., Sadayappan P.* A practical automatic polyhedral parallelizer and locality optimizer // ACM SIGPLAN Notices. 2008. Vol. 43, No. 6. P. 101–113. https://doi.org/10.1145/1379022.1375595.

4. *Vandierendonck H., Rul S., De Bosschere K.* The Paralax Infrastructure: Automatic Parallelization with a Helping Hand // Proceedings of the 19th International Conference on Parallel architectures and compilation techniques (PACT'10). 2010. P. 389–400. https://doi.org/10.1145/1854273.1854322.

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. IEEE Computer Society. Washington, DC, USA, 2015. P. 138–149. 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. P. 1–6.

7. Intel Parallel Studio. URL: https://software.intel.com/en-us/parallel-studioxe

8. *Klinov M.S., Kriukov V.A.* Avtomaticheskoe rasparallelivanie Fortran-programm. Otobrazhenie na klaster // Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo. 2009. No. 2. S. 128–134.

9. *Bakhtin V.A., Zhukova O.F., Kataev N.A., Kolganov A.S., Kriukov V.A., Kuznetsov M.Iu., Podderiugina N.V., Pritula M.N., Savitskaia O.A., Smirnov A.A.* Rasparallelivanie programmnykh kompleksov. Problemy i perspektivy // Trudy XX Vserossiiskoi nauchnoi konferentsii «Nauchnyi servis v seti Internet», Novorossiisk, Rossiia, 17–22 sentiabria 2018 g. M.: IPM im. M.V. Keldysha, 2018. S. 63–72.

http://keldysh.ru/abrau/2018/theses/33.pdf. https:// doi.org/10.20948/abrau-2018-33.

10. *Hwu W.-m., Ryoo S., Ueng S.-Z., Kelm J.H., Gelado I., Stone S.S., Kidd R.E., Baghsorkhi S.S., Mahesri A.A., Tsao S.C., Navarro N., Lumetta S.S., Frank M.I., Patel S.J.* Implicitly parallel programming models for thousand-core microprocessors // Proceedings of the 44th annual Design Automation Conference (DAC'07), ACM, New York, NY, USA. 2007. P. 754–759. https://doi.org/10.1145/1278480.1278669.

11. *Blume W., Eigenmann R*. Performance analysis pf parallelizing compilers on the Perfect Benchmarks programs // IEEE Transactions on Parallel and Distributed Systems. 1992. Vol. 3, Issue 6. P. 643–656. https://doi.org/10.1109/71.180621.

12. *Wolfe M.* Scalar vs. parallel optimizations // CSETech. 210. 1990. URL: https://classes.cs.uoregon.edu/16S/cis410parallel/Documents/scalar-paralleloptimizations-wolfe.pdf.

13. *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. 1995. V. 21. No. 1. P. 35–38.

14. *Bakhtin V.A., Klinov M.S., Kriukov 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". 2012, No. 18 (277), vypusk 12. S. 82–92.

15. *Kulkarni P., Zhao W., Moon H*., Cho K., Whalley D., Davidson J., Bailey M., Paek Y., Gallivan K. Finding effective optimization phase sequences // Proceedings of the 2003 ACM SIGPLAN Conference on Languages, Tools, and Compilers for Embedded Systems. 2003. P. 12–23. https://doi.org/10.1145/780731.780735.

16. NAS Parallel Benchmarks. https://www.nas.nasa.gov/publications/npb.html.

17. *Lattner C., Adve V.* LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // In: Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO’04), Palo Alto, California. 2004. https://doi.org/10.1109/CGO.2004.1281665.

18. *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, 2018. V. 965. Springer, Cham. P. 487–499. doi:10.1007/978-3-030-05807-4_41.

19. *Kataev N.A., Smirnov A.A., Zhukov A.D.* Opredelenie zavisimostei po dannym sredstvami dinamicheskogo analiza sistemy SAPFOR // Elektronnye biblioteki. Tematicheskii vypusk «Nauchnyi servis v seti Internet». Chast 1. 2020. Tom 23. № 3. S. 473–493. https://doi.org/10.26907/1562-5419-2020-23-3-473-493.

20. Visual Studio Code. https://code.visualstudio.com/, last accessed 2020/11/25.

21. OpenMP Application Programming Interface. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-1.pdf, last accessed 2020/11/25.

22. *Kataev N.A., Vasilkin V.N.* Vosstanovlenie mnogomernoi formy obrashchenii k linearizovannym massivam v sisteme SAPFOR // Elektronnye biblioteki. Tematicheskii vypusk «Nauchnyi servis v seti Internet». Chast 2. 2020. Tom 23. № 4. S. 770–787. https://doi.org/10.26907/1562-5419-2020-23-4-770-787.

23. *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. https://doi.org/10.1109/IISWC.2011.6114174.

24. SAPFOR. URL: https://github.com/dvm-system.

СВЕДЕНИЯ ОБ АВТОРЕ

КАТАЕВ Никита Андреевич – научный сотрудник ИПМ им. М.В. Келдыша, специалист в области системного программирования. Сфера научных интересов – компиляторные технологии, автоматизация распараллеливания программ.

Nikita Andreevich KATAEV – Researcher of KIAM RAS, a specialist in system programming. Research interests include compiler technologies, semi-automatic program parallelization.

email: kataev_nik@mail.ru; ORCID: 0000-0002-7603-4026.

Материал поступил в редакцию 30 ноября 2020 года