Московский государственный университет имени М.В. Ломоносова

Научно-исследовательский вычислительный центр МГУ

На правах рукописи

Стефанов Константин Сергеевич

Комплекс инструментальных средств разработки программ для вычислительных систем с параллельной архитектурой

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата физико-математических наук

Москва

2007

Работа выполнена в *Научно-исследовательском вычислительном центре МГУ имени М.В. Ломоносова.*

Научный руководитель: д. ф.-м. н., чл.-корр. РАН
Воеводин Владимир Валентинович

Официальные оппоненты: д. ф.-м. н., чл.-корр. РАН
Абрамов Сергей Михайлович

д. ф.-м. н., профессор
Машечкин Игорь Валерьевич

Ведущая организация: Южно-Уральский государственный университет

Защита состоится «____»_____ 2007 г. в ____ часов на заседании диссертационного совета *К 501.001.11 Московского государственного университета имени М.В. Ломоносова* по адресу: *119991, г. Москва, Ленинские горы 1, стр. 4, МГУ, НИВЦ, конференц-зал.*

С диссертацией можно ознакомиться в библиотеке *Научно-исследовательского вычислительного центра МГУ имени М.В. Ломоносова.*

Автореферат разослан «____»_____ 2007 г.

Ученый секретарь
диссертационного совета,
к. ф.-м. н., *Суворов В.В.*

Общая характеристика работы

Актуальность темы

В последнее время все более широкое распространение получают компьютерные методы решения задач. Из-за высокой вычислительной сложности многих методов значительная часть задач не может быть решена без привлечения параллельных вычислительных систем и суперкомпьютеров. Ставшие неотъемлемой частью архитектуры суперкомпьютерных систем элементы параллелизма активно используются в структуре процессоров вплоть до персональных ЭВМ. Вместе с этим, уже более чем тридцатилетняя практика работы с параллелизмом показывает, насколько трудным является этот процесс. Создание программного обеспечения, помогающего эффективно осваивать вычислительные системы с параллельной архитектурой, а также стоящая в этом же ряду задача адаптации большого числа существующих программ под новые компьютеры является важной и актуальной проблемой.

При адаптации последовательных программ для параллельных компьютеров необходимо определять целое множество свойств, которые указывают на имеющийся потенциальный параллелизм исследуемых программ и возможные способы их преобразования. Несмотря на то, что между необходимыми свойствами имеется много общего, набор их весьма велик и специфичен для разных архитектур. Выявление этих свойств обычно возлагается на пользователя, поскольку методы, использующиеся в штатных компиляторах параллельных систем, как правило оказываются не в состоянии обеспечить достижение нужного уровня производительности.

Для помощи пользователям создаются специальные технологии параллельного программирования и разрабатываются автономные системы анализа структуры программ. Данное направление исследований активно развивается и у нас в стране, и за рубежом, в создание подобных систем внесли свой вклад многие ученые, такие как С.М. Абрамов, В.В. Воеводин, Вл.В. Воеводин, А.П. Ершов, И.Б. Задыхайло, В.А. Крюков, А.Л. Ластовецкий, P. Feautrier, K. Kennedy, D. Kuck, M. Lam, W. Pugh, M. Wolfe и ряд других.

Обычно методы анализа программ основываются на проверке некоторого набора достаточных условий информационной независимости отдельных фрагментов программ. Найденные свойства используются для адаптации программ, причем методы адаптации часто ориентированы на конкретный класс параллельных архитектур. Такой подход прост в реализации, применяемые алгоритмы работают быстро, но есть и целый ряд серьезных недостатков, побудивших выбрать иное направление исследований. Во-первых, использование достаточных условий не дает гарантии того, что программа действительно не обладает нужным свойством, если проверяемое условие не выпол-

няется. Во-вторых, учитывая появляющиеся все время новые архитектуры компьютеров, их слияние и модификации, создание отдельных систем анализа под каждый тип компьютера приводит к тому, что их выпуск не успевает за потребностями в таких системах.

Более общий подход к выбору методов анализа программ реализован в системе V-Ray, разработанной в лаборатории Параллельных информационных технологий НИВЦ МГУ и использующей в качестве основы для анализа программ граф алгоритма (параметрическое описание информационной истории реализации программ). В работах В.В. Воеводина и Вл.В. Воеводина сформулированы и доказаны условия существования информационной зависимости в программах, являющиеся не только достаточными, но и необходимыми. Эти критерии применимы к широкому классу программ, называемому линейным классом. Система предназначена для анализа программ на языке Фортран-77.

Использование разработанного в рамках данной работы программного инструментария позволяет решать задачи анализа в комплексе, рассматривая целое множество целевых архитектур, преобразований, входных языков. Предлагаемая архитектура технологического инструментария позволяет получать конкретные системы для анализа нужных свойств программ из готовых блоков, дописывая недостающие модули по мере необходимости. Механизм модулей-экспертов позволяет создавать системы преобразования программ для конкретных целевых платформ, используя готовые блоки анализа и преобразования с фиксированным интерфейсом. Модули-интерфейсы позволяют использовать одни и те же блоки экспертов не только в интерактивных системах с графическим интерфейсом, но и в автоматизированном режиме совместной работы с внешними системами.

Целью диссертационной работы является разработка подходов к созданию и реализация комплекса инструментальных средств для решения множества задач анализа и преобразования структуры программ, возникающих при адаптации программного обеспечения к работе на вычислительных системах с параллельной архитектурой.

Основные положения, выносимые на защиту:

1. Разработаны подходы к решению множества задач анализа и преобразования структуры программ, актуальных для эффективного использования современных процессоров и суперкомпьютерных систем, с помощью единого комплекса инструментальных средств.

2. Разработаны принципы организации и архитектура технологического инструментария, ориентированного на создание как многофункциональных, так и специализированных автономных систем исследования и оп-

тимизации программ, а также на взаимодействие с внешними системами через интерфейсы различного уровня.

3. Реализован комплекс инструментальных программных средств, позволяющий эффективно решать широкий спектр задач анализа и преобразования структуры программ для их отображения на архитектуру современных компьютерных систем.

4. Проведена апробация комплекса инструментальных средств на различных прикладных программах, с ориентацией на различные компьютеры и целевые функции анализа, показавшие эффективность предложенных подходов и созданных средств для решения актуальных задач вычислительной практики.

Научная новизна

Разработаны подходы к решению различных задач анализа и преобразования структуры программ, актуальных для эффективного использования современных процессоров и суперкомпьютерных систем, с помощью единого комплекса инструментальных средств.

Разработаны принципы организации и архитектура технологического инструментария, ориентированного на создание как многофункциональных, так и специализированных автономных систем исследования и оптимизации программ, а также на взаимодействие с внешними системами через интерфейсы различного уровня.

Практическая значимость

Реализован комплекс инструментальных программных средств, позволяющий эффективно решать широкий спектр задач анализа и преобразования структуры программ для поддержки процесса их отображения на архитектуру современных микропроцессоров и суперкомпьютерных систем с параллельной архитектурой.

Предложенная архитектура комплекса позволяет легко конструировать конкретные системы анализа и преобразования программ с заданной целевой функциональностью. При этом получаемые системы могут быть как интерактивными, рассчитанными на работу в диалоге с пользователем, так и конверторами, работающими в полностью автоматическом режиме.

Зависимость от языка анализируемой программы локализована на уровнях внутреннего представления и зависимых от языка модулей, что упрощает добавление других языков для анализа. То же относится и к зависимости от платформы, на которой выполняется полученная система анализа: эта зависимость проявляется в относительно небольшом числе блоков уровня интерфейсов, что облегчает перенос системы.

Апробация работы

Проведена апробация комплекса инструментальных средств на различных прикладных программах, с ориентацией на различные компьютеры и целевые функции анализа, показавшие эффективность предложенных подходов и созданных средств для решения актуальных задач вычислительной практики.

Основные положения работы обсуждались на научно-исследовательских семинарах в НИВЦ МГУ и в ИВМ РАН. Результаты работы представлялись на первой Всероссийской научной конференции «Методы и средства обработки информации» (Москва, 2003 г.), на Ломоносовских чтениях в МГУ (в 2004, 2006 и 2007 годах), на всероссийской научной конференции «Научный сервис в сети Интернет: технологии распределенных вычислений» (Новороссийск, 2005 г.) и «Научный сервис в сети Интернет: технологии параллельного программирования» (Новороссийск, 2006 г.).

Механизмы взаимодействия с внешними системами через интерфейсы различного уровня и модули-эксперты апробированы в ходе успешного выполнения государственного контракта № 02.434.11.1002 от 25.04.2005, заключенного между НИВЦ МГУ и Роснаукой в рамках ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2002-2006 годы.

Публикации

По теме диссертации опубликовано 5 научных работ. На созданную в ходе работы программную систему получено свидетельство Роспатента об официальной регистрации программы для ЭВМ № 2006613632 от 19.10.2006.

Структура и объем диссертации

Диссертация состоит из введения, четырех глав, заключения и списка литературы (47 наименований).

Содержание работы

Во введении приводится общая характеристика работы и дается краткий обзор содержания диссертации.

Первая глава работы посвящена описанию развития программного обеспечения, используемого для распараллеливания программ. Описываются недостатки методов, применяемых в компиляторах для параллельных систем, и анализируются причины возникновения автономных систем исследования структуры программ.

В процессе развития вычислительных систем стали появляться параллельные компьютеры. Одними из первых были векторно-конвейерные. При создании компиляторов под эту платформу стали возникать трудности, которые

потом в полной мере проявились и на других параллельных системах. Основное, за счет чего мог проявиться выигрыш при использовании векторных компьютеров — это наличие встроенных векторных команд, манипулирующих сразу большими массивами данных. Наиболее подходящая кандидатура для преобразования в эффективные векторные команды — самые внутренние по вложенности циклы. Однако далеко не всякий цикл может быть преобразован в векторную команду. Появилось множество методов определения того, можно ли данный цикл представить как векторную команду или нет, причем каждый метод имел ограниченную область применимости. Универсального способа определения векторизуемости, пригодного для любых циклов, нет до сих пор.

Еще больше ситуация усложнилась с появлением других типов параллельных архитектур — архитектур с общей и распределенной памятью и многомашинных комплексов. Для архитектур с общей памятью необходимо определять, возможно ли разбиение программы на несколько потоков вычислений, выполняющихся параллельно, и находить точки синхронизации между отдельными потоками. Для архитектур с распределенной памятью, кроме того, необходимо организовать аккуратную работу с данными, в частности, нужно установить, какая часть данных требуется каждому потоку, и минимизировать обмены данными между разными процессорами.

Попытка создать компиляторы, автоматически извлекающие необходимые сведения из обычной программы и использующие их для получения кода параллельной вычислительной системы, оказалась не очень успешной. Практика показала, что извлечение подобных сведений крайне затруднено, и не существует достаточно универсального способа, пригодного для использования на широком множестве программ и широком классе вычислительных систем. В то же время в компиляторы, особенно для векторных систем и SMP-компьютеров, продолжали (и продолжают) включаться некие простые достаточные условия, на основе которых компилятор пытается обеспечить эффективное исполнение фрагментов входной программы. Никаких гарантий эффективности применяемых методов производители компиляторов не дают.

В результате забота о распараллеливании программ была перенесена с компилятора на пользователя. Стали активно развиваться системы и языки параллельного программирования (PVM, HPF, DVM, MPI, OpenMP и др.). Все они требуют от пользователя явного указания данных о параллельных свойствах программы. При этом опять возникает проблема, о которой уже упоминалось в связи с компиляторами для параллельных систем — достаточность предоставляемой информации для получения нужной производительности, а также эффективность и полнота ее использования, ведь для разных архитектур необходимы разные сведения.

В связи с этим стали создаваться автономные системы анализа, позволяющие по тексту программы определять некоторые ее свойства, которые впоследствии могли бы быть использованы при адаптации программы под параллельные компьютеры. Раньше (при использовании последовательных ЭВМ) процесс использования вычислительной техники для расчетов выглядел так: выбор алгоритма – запись его на языке программирования — компиляция — исполнение — анализ полученных результатов. Теперь же все большее значение приобретает новый этап — адаптация к архитектуре используемой параллельной системы.

В этой же главе сделан обзор препроцессоров и систем анализа программ (SUIF, серия препроцессоров KAP, пакет FORGE, Polaris, CAPO, открытая распараллеливающая система, Intel Threading Analysis Tools, ParaWise, система V-Ray и другие), что позволило выделить наиболее важные и проблемные вопросы предметной области, узкие места используемых технологий и одновременно уточнить постановку задачи диссертационной работы.

Целью работы является разработка подходов к созданию комплекса инструментальных средств для решения различных задач, возникающих при адаптации программного обеспечения к работе на параллельных вычислительных системах. Эти задачи включают, например, исследование программ на разных языках программирования, интерактивное исследование структуры больших программных комплексов, исследование тонкой информационной структуры отдельных фрагментов программ, описание потенциального параллелизма программы в автоматическом режиме, преобразование последовательной программы под конкретный язык программирования, предоставление интерфейсов для написания новых блоков анализа, обеспечение взаимодействия с внешними системами и другие.

Где доступность не шла в ущерб качеству решения задач, использовались существующие средства и технологии, и вместе с этим проводилась разработка архитектуры нового технологического инструментария, позволяющего получать системы анализа для конкретных задач из набора готовых модулей и подключения новых.

Во второй главе описываются эволюционные компоненты разработанного комплекса инструментальных средств, опирающиеся на уже существующие подсистемы и пакеты. Приведено описание подходов, использованных для добавления анализа программ на языке Си к существующей системе анализа, рассчитанной на Фортран-77. Описываются причины создания модулей целевых преобразований (экспертов) и рассматриваются особенности их реализации.

Значительная часть возникающих в последнее время задач связана с анализом программ, написанных на языке Си. Языковые средства постоянно развиваются, нельзя ориентироваться только на Фортран и его модифика-

ции, поэтому многоязыковость — важное свойство создаваемого комплекса инструментальных средств.

Исходно в системе V-Ray использовались собственные средства разбора программы на языке Фортран-77. Эти средства не были оформлены в виде одной подсистемы и были распределены по различным частям самой системы. Примененный в V-Ray подход эффективен для Фортрана-77, так как особенности записи программы в фиксированной форме позволяют легко найти границы отдельных операторов (начала и концы строк с учетом строк продолжения и строк комментариев) и место расположения ключевых слов, определяющих конкретный оператор, делая необязательным полный синтаксический разбор программы. Но для языка Си применение такого подхода сталкивается со значительными трудностями.

В качестве библиотеки для разбора входной программы был выбран пакет Sage++. Основная причина такого выбора — заявленное авторами пакета наличие готовых модулей для разбора программ на языках Фортран-77 и Си, что давало возможность приступить к реализации требуемой функциональности.

Одним из важных этапов при анализе больших программных комплексов является исследование их общей структуры. Этот этап, необходимый почти во всех практически значимых задачах, предполагает исследование взаимосвязей между отдельными частями программы по управлению и по общим данным. В рамках работы разработан и реализован подход для исследования общей структуры программ на языке Си на базе существующей системы исследования программ V-Ray и системы разбора программ Sage++ с сохранением возможности исследования программ на языке Фортран.

При реализации методов исследования связи по управлению между программными единицами в программах на языке Си из системы V-Ray были взяты блоки визуализации графа вызовов и работы с его изображением (разметка, отсечение части вершин и т.п.). Использование существующего блока построения графа вызовов оказалось неэффективным. Он жестко ориентирован на язык Фортран, определенные в нем операторы вызова процедур и способы вызова функций.

В рамках новых технологий разработана архитектура и создана единая для входных языков подсистема построения графа вызовов. По дереву разбора программы, построенному на этапе загрузки проекта Sage++, определяются все программные единицы исследуемой программы и их характеристики, осуществляется проход по всем операторам программы. Для всех выражений всех операторов происходит поиск всех вызовов функций, и соответствующие дуги добавляются к графу вызовов.

Разработана методика и реализован блок определения максимальной глубины вложенных в программную единицу циклов для соответствующей раз-

метки. Этот блок учитывает циклические конструкции, явно описываемые языками программирования Си и Фортран. Определение GOTO-циклов (циклов, записанных при помощи операторов перехода) осуществляется при помощи блока из системы V-Ray, определяющего такие циклы по графу управления процедуры.

Исследование связи между программными единицами по общим данным осуществляется при помощи графа использования общей памяти. Это двудольный граф, в котором одно множество вершин соответствует процедурам, другое — глобальным переменным.

Для Си определение глобальных переменных производится поиском всех объявлений переменных вне программных единиц. Идентификация обращений к этим переменным производится на основе прохода по всем операторам, всем выражениям в этих операторах и поиска всех обращений к переменным.

Исследование управляющей структуры отдельных подпрограмм — один из ключевых этапов анализа программ. Основной объект исследования — это граф управления фрагмента программы, который необходим почти во всех возникающих на практике задачах, в частности, для нахождения GOTO-циклов, для построения графов зависимостей по данным, для выделения вычислительных ядер программ.

В рамках работы разработаны и реализованы методы исследования управляющей структуры отдельных функций для программ на языке Си. Реализация осуществлялась на основе системы V-Ray с адаптацией ее блоков исследования управляющей структуры к программам на Си с сохранением в получившемся средстве возможности анализа программ на Фортране.

На первом этапе построения графа управления происходит нахождение всех линейных участков исследуемой программной единицы, что позволяет построить множество вершин графа управления. Далее, при последовательном проходе по всем линейным участкам находятся все возможные передачи управления между этими участками, и в граф добавляются соответствующие дуги.

Исследование информационной структуры подпрограмм — важнейший этап при анализе программ с целью их дальнейшего распараллеливания. Именно исследование информационной структуры дает возможность установить фрагменты программы, которые возможно исполнять в параллельном режиме, найти векторизуемые циклы, выявить потенциальный параллелизм исследуемой программы.

Основное понятие при исследовании информационной структуры программы — это информационная зависимость. В рамках работ по созданию комплекса инструментальных средств разработан подход и реализован способ быстрого построения графа зависимостей по данным для программ на языке Си.

Для работы данного блока из системы V-Ray для каждого оператора необходимо определять два множества: переменные, значения которых используются в данном операторе, и переменные, значения которых изменяются. Блок, определяющий используемые и изменяемые переменные в данном операторе, берет все выражения в операторе и осуществляет рекурсивный обход дерева разбора каждого выражения. При обходе для каждого поддерева определяется параметр — способ обращения (использование или изменение) к результату вычисления соответствующего поддерева. Изначально способ использования определяется типом оператора, в котором используется выражение, и конкретным местом вхождения выражения в оператор.

Для облегчения создания инструментальных средств, осуществляющих преобразование программы под конкретную архитектуру, в рамках комплекса введено понятие модулей целевых преобразований, которые в дальнейшем будут называться модулями-экспертами или просто экспертами. Таким образом предоставляется возможность написания собственных модулей с необходимой дополнительной функциональностью без знания деталей организации и внутренних структур системы V-Ray.

В рамках данной работы выполнена реализация интерфейса для подключения модулей-экспертов к системе V-Ray. Эксперт — это блок, имеющий некоторую предопределенную целевую функцию. Он компилируется вместе со всей системой, но связи его с остальной системой подчиняются определенному интерфейсу, через который эксперт пользуется функциональностью системы по анализу и преобразованию программ.

Третья глава содержит описание базовых принципов архитектуры технологического инструментария, позволяющие эффективно строить специализированные системы для решения конкретных задач, возникающих при исследовании последовательных программ. Описываются принципы разбиения инструментария на логические уровни: внутреннее представление, зависимые и не зависимые от языка модули, уровень модулей целевых функций (экспертов) и уровень интерфейсов. Определяются возможные взаимодействия между модулями инструментария, расположенными на разных уровнях, с целью локализации зависимости от языка анализируемой программы и ОС выполнения инструментального комплекса. Рассмотрено разделение модулей на модули анализа и преобразований. Рассматриваются отдельные уровни с примерами расположенных на них модулей.

В основу инструментария закладываются три принципа: локализация зависимостей от конкретного языка анализируемой программы, модульность инструментария и универсальность архитектуры.

Под универсальностью мы понимаем возможность последующего построения на основе инструментария специализированных систем и конверторов, предназначенных для решения максимально широкого спектра задач, кото-

рые возникают в процессе адаптации программ под параллельные вычислительные системы.

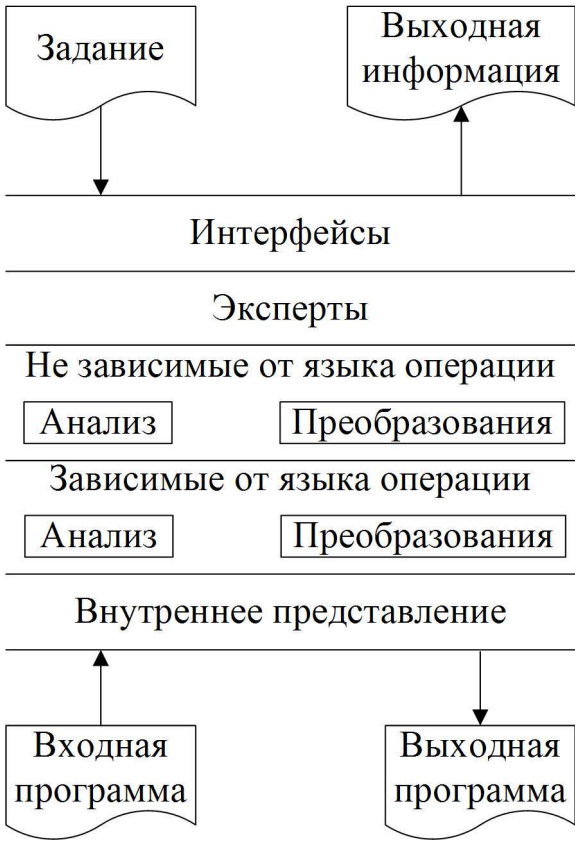

Рис. 1. Логические уровни блоков инструментального комплекса

Все блоки системы логически разделяются на уровни (рис. 1): *внутреннее представление*, *зависимые от языка* (входной программы) блоки, *не зависимые от языка* блоки, *эксперты* и *интерфейсы*. Уровни зависимых и не зависимых от языка операций дополнительно делятся на блоки определения свойств (анализ) и блоки преобразований. Более полная схема с примерами расположенных на каждом уровне блоков и возможных взаимодействий между ними приведена в тексте диссертации.

Эксперт — это блок, имеющий фиксированную целевую функцию. Интерфейс предоставляет доступ к функциям системы внешним пользователям, в роли которых, в частности, могут быть и другие программы. Все взаимодействие комплекса или созданной на его основе новой системы с внешним миром — с пользователем либо с другими программами — происходит через интерфейсы.

Уровень внутреннего представления составляют блоки, отвечающие непосредственного за работу с текстом программы (ее разбор или генерацию нового текста) и за действия со структурами данных, составляющих представ-

ление разобранной программы. Интерфейс блоков уровня внутреннего представления позволяет производить различные действия с объектами программы.

Уровни зависимых и не зависимых от языка блоков содержат основные подсистемы комплекса — анализ и преобразование программ, и предоставляют функциональность, необходимую для построения экспертов. Блоки анализа служат для определения свойств входной программы, которые затем могут быть использованы другими блоками для выполнения своих функций, и дают на выходе объекты-свойства, содержащие информацию о входной программе. Блоки преобразований служат для выполнения базовых преобразований программы, которые могут быть использованы в качестве составных частей другими блоками преобразований или модулями-экспертами.

Деление блоков на зависимые и не зависимые от языка определяется возможностью определить свойство или преобразование в терминах, общих для всех анализируемых языков. Зависимые от языка блоки — это блоки определения свойств или проведения преобразований, которым нужно знать специфику языка, на котором написана анализируемая или преобразуемая программа, в частности, блок определения наличия в данном операторе вызова процедуры или функции. Не зависимые от языка блоки анализа определяют те, обычно более „высокоуровневые“, свойства, которые можно определить общим для всех языков процедурного типа образом, опираясь на информацию, поставляемую зависимыми от языка блоками.

Не зависимые от языка блоки преобразования предоставляют возможности по преобразованию программы, необходимые блокам экспертов. По аналогии с независимыми от языка блоками анализа, обычно на данном уровне проводятся более „высокоуровневые“ преобразования чем те, которые проводятся на уровне зависимых от языка преобразований. Например, преобразование, вводящее новую переменную, (не зависимое от языка) должно опираться на добавление описания переменной, которое свое для каждого языка программирования.

В блоках экспертов всегда (в не зависимых от языка блоках — по возможности) используется информация, поставляемая блоками не зависимого от языка уровня. Такая архитектура облегчит решение проблем, возникающих при добавлении новых анализируемых языков.

Блоки, реализующие целевую функциональность интерактивной системы или конвертора, расположены на уровне экспертов. Эксперт — это модуль системы, реализующий нужную пользователю конкретную функцию в автоматическом или полуавтоматическом режиме, например, модуль, распараллеливающий и преобразующий программу под конкретную систему параллельного программирования.

В рамках рассматриваемого технологического инструментария вся целевая функциональность системы оформляется модулями уровня экспертов, в отличие от экспертов, которые добавлялись в систему V-Ray, для расширения функциональности, не предусмотренной изначально при проектировании V-Ray.

Блоки интерфейсов предназначены для реализации различных способов взаимодействия построенной системы с внешним миром. У каждой системы, созданной при помощи инструментария, должен быть один главный блок интерфейса, который определяет основные правили взаимодействия системы с пользователем. В блоках уровня интерфейсов должна быть локализована зависимость от среды, в которой исполняется система. Именно блоки этого уровня знают о желательном расположении файлов или о вызовах, требуемых для открытия этих файлов.

Четвертая глава описывает особенности реализации базовых модулей технологического инструментария.

Приведено сравнение средств для разбора программ и обоснован выбор пакета для построения компиляторов Eli в качестве основы для реализации уровня внутреннего представления.

Реализация модулей внутреннего представления основана на системе Eli. В комплект примеров к ней входят описания грамматики языка Фортран-77, а также описание вычислений атрибутов, осуществляющее проверку семантической корректности входной программы (проверка синтаксической корректности происходит в процессе синтаксического разбора).

Часть блоков, входящих в состав создаваемого инструментария, может быть реализована при помощи вычислений атрибутов на абстрактном синтаксическом дереве разбора программы. В качестве примера рассмотрен блок построения графа вызовов, его зависимая и не зависимая от языка части.

Модули анализа могут быть реализованы либо на внутренних проблемно-ориентированных языках Eli, либо на Си и Си++, на которых реализуется остальная часть инструментария (вспомогательные функции, часть модулей анализа и преобразований, модули экспертов и интерфейсы).

Рассмотрены способы передачи информации от блоков, написанных на внутренних языках Eli к остальным блокам при помощи вызовов функций либо использования макроподстановок для задания других действий синтаксисом, подобным синтаксису вызова функции. В частности, показан способ использования конструкций языка Си++, прямое использование которых не предполагается, так как исходно Eli ориентирован на язык Си.

Показана технология использования внешних модулей (т.е. отдельных программ, выполняющих функции модулей системы) и описано использование внешнего модуля отображения графа. В качестве примера интеграции с внешними системами рассмотрено выделение блока определения независимо-

сти итераций цикла из V-Ray и сопряжение его в качестве модуля анализа с остальной системой.

В заключении формулируются основные результаты работы:

1. Разработаны подходы к решению множества задач анализа и преобразования структуры программ, актуальных для эффективного использования современных процессоров и суперкомпьютерных систем, с помощью единого комплекса инструментальных средств.

2. Разработаны принципы организации и архитектура технологического инструментария, ориентированного на создание как многофункциональных, так и специализированных автономных систем исследования и оптимизации программ, а также на взаимодействие с внешними системами через интерфейсы различного уровня.

3. Реализован комплекс инструментальных программных средств, позволяющий эффективно решать широкий спектр задач анализа и преобразования структуры программ для их отображения на архитектуру современных компьютерных систем.

4. Проведена апробация комплекса инструментальных средств на различных прикладных программах, с ориентацией на различные компьютеры и целевые функции анализа, показавшие эффективность предложенных подходов и созданных средств для решения актуальных задач вычислительной практики.

Список публикаций

[1] *Стефанов К. С.* Система анализа структуры программ // Труды Первой Всероссийской научной конференции «Методы и средства обработки информации» / Под ред. Л. Н. Королева. — М.: Издательский отдел факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова, 2003. — С. 498–502.

[2] *Стефанов К. С.* Система анализа информационной структуры программ // *Вычислительные методы и программирование.* — 2005. — Т. 6, № 1. — С. 124–135.

[3] *Стефанов К. С.* Использование пакета Sage++ для построения системы анализа структуры программ // Научный сервис в сети Интернет: технологии распределенных вычислений: Труды Всероссийской научной конференции (19-24 сентября 2005 г., г. Новороссийск). — М.: Изд-во МГУ, 2005. — С. 98–100.

[4] *Стефанов К. С.* Проект архитектуры системы анализа структуры программ // Научный сервис в сети Интернет: технологии параллельного программирования: Труды Всероссийской научной конференции (18-23 сентября 2006 г., г. Новороссийск). — М.: Изд-во МГУ, 2006. — С. 67–70.

[5] *Стефанов К. С.* Архитектура инструментального комплекса для анализа и преобразования структуры программ // *Системы управления и информационные технологии.* — 2007. — № 1.3(27). — С. 395–398.

[6] Св-во Роспатента о регистрации программы для ЭВМ № 2006613632. Система V-Ray исследования и преобразования структуры программ / В. В. Воеводин, Вл. В. Воеводин, К. С. Стефанов и др. — № 2006613343; Заяв. 5.10.2006.