Российская Академия Наук
Сибирское Отделение
Институт систем информатики им. А. П. Ершова

На правах рукописи

УДК 004.432.42

Идрисов Ренат Искандерович

МЕЖПРОЦЕДУРНЫЙ АНАЛИЗ И РАСПАРАЛЛЕЛИВАНИЕ ПОТОКОВЫХ ПРОГРАММ НА БАЗЕ ГРАФА ИСПОЛНЕНИЙ ВЫЗОВОВ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Новосибирск 2010

Работа выполнена в Учреждении Российской академии наук Институте систем информатики им. А.П. Ершова СО РАН

Научные руководители: Евстигнеев Владимир Анатольевич
доктор физико-математических наук, профессор

Касьянов Виктор Николаевич
доктор физико-математических наук, профессор

Официальные оппоненты: Легалов Александр Иванович
доктор технических наук, профессор

Скопин Игорь Николаевич
кандидат физико-математических наук, доцент

Ведущая организация: Южный Федеральный Университет
(г. Ростов-на-Дону)

Защита состоится 16 июня 2010 года в 15 часов 30 минут на заседании диссертационного совета ДМ003.032.01 при Институте систем информатики им. А.П. Ершова Сибирского отделения РАН по адресу: 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6.

С диссертацией можно ознакомиться в читальном зале ИСИ СО РАН (г.Новосибирск,пр.Акад.Лаврентьева,6).

Автореферат разослан _____ 2010 г.

Ученый секретарь
Диссертационного совета,

к.ф.–м.н Мурзин Ф.А.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность проблемы

В данной работе пойдёт речь о статическом анализе программ, а именно о той части, которая отвечает за взаимодействие процедур и функций программы.

Межпроцедурный анализ относится в первую очередь к анализу потока данных, который поступает при вызове в процедуру и из неё. В рамках данной работы под межпроцедурным анализом будем понимать анализ влияния вызова процедуры на контекст вызова и определение влияния контекста вызова на исполнение вызываемой процедуры.

Такой вид анализа приобретает особую ценность в распараллеливающих компиляторах. Если не анализировать код вызываемых процедур, придётся предположить, что все параметры и глобальные переменные могут измениться в результате вызова. Это существенно снизит эффективность результирующего кода, потому что, например, циклы, содержащие вызовы, не будут распараллелены в большинстве случаев. А распараллеливание на сегодняшний день также является неотъемлемой частью оптимизирующего транслятора, поскольку в последнее время увеличение вычислительных мощностей связано уже не с ускорением отдельного, а с добавлением дополнительных вычислителей и созданием различного рода суперкомпьютеров и кластеров.

Оптимизации и работа компилятора вообще заключается в преобразовании внутреннего представления. Внутренние представления варьируются от приближенных к языкам высокого уровня до приближенных к уровню машинного кода. Параллелизм наиболее универсально описывается при помощи представлениях, основанных на потоке данных. В таких представлениях не требуется тратить дополнительные усилия для анализа зависимостей. Примером могут служить многочисленные проекты, развивающие SSA форму. Одним из таких представлений является IR2, рассматриваемое в данной работе. К такому представлению может быть приведён даже императивный язык. Известны проекты, где Fortran транслируется в IF1, представление с аналогичной структурой, но другим набором вершин.

Потоковая структура IR2 является наиболее универсальным описанием параллелизма, что важно для переносимости кода. При решении вычислительных задач, удобнее всего не вдаваться в конкретные детали распараллеливания своей задачи и создавать код, который мог быть исполнен на системах с различной конфигурацией.

Все перечисленные причины делают анализ и оптимизацию потоковых структур актуальными.

Цель работы – адаптация известных оптимизаций и распараллеливания для потоковой модели вычислений. Разработка новых алгоритмов анализа и оптимизации.

Методы исследования. В диссертационной работе использовались понятия и методы теории построения компиляторов, теории смешанных вычислений, теории алгоритмов, теории графов, теории схем программ и элементы теории множеств.

Научная новизна.

1. Предложен новый метод анализа значений элементов массивов, который позволяет оптимизировать всюду завершаемые частичные вычисления.
2. Предложен новый подход к межпроцедурному анализу, основанный на графе исполнений вызовов, в рамках которого предложены эффективные алгоритмы осуществления анализа.
3. Впервые исследованы параллельные свойства и преобразования потоковых программ в рамках иерархического графового представления IR2.

Практическая ценность.

Практическая часть данной работы описывает реализацию межпроцедурного анализа и распараллеливания в рамках системы функционального программирования (SFP), разрабатываемой в Институте Систем Информатики им. А.П. Ершова (ИСИ СО РАН). Основные оптимизирующие преобразования в SFP производятся на уровне внутреннего потоко-

вого представления IR2, в которое может быть транслирован даже императивный язык. Это внутреннее представление может описывать массивы и циклические конструкции, что не является обычным для внутреннего представления функционального языка программирования.

Разработанные методы реализованы в рамках компилятора Sisal 3.2. Компилятор Sisal 3.2 является частью системы SFP, разрабатываемой в рамках проекта ПРОГРЕСС. Компилятор будет использоваться другими разработчиками системы, которая ориентирована на решение математических задач на суперкомпьютерах и обучение функциональному программированию.

Апробация работы. Основные положения диссертации обсуждались на следующих конференциях и семинарах:

1. Семинар «Конструирование и оптимизация программ», Новосибирск, ИСИ СО РАН, 2004-2008 г.;
2. конференция-конкурс «Технологии Microsoft в информатике и программировании», Новосибирск, 2008 год.
3. XLIV Международная научная студенческая конференция «Студент и научно-технический прогресс», Новосибирск, 2006 г.
4. Четвертая азиатская международная школа-семинар «Проблемы оптимизации сложных систем» / ИВМиМГ 2008
5. Решетнёвские чтения / Красноярск 2009

Публикации. Результаты работы опубликованы в 9 печатных работах, из которых 4 полных статьи и 5 тезисов конференций.

Структура и объем работы

Диссертационная работа состоит из введения, трёх глав, заключения, списка литературы и двух приложений. Объем диссертации – 127 стр. Список литературы содержит 80 наименований. Работа включает 26 рисунков и графиков, полученных в результате расчетов на ЭВМ.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность проводимых исследований, сформулирована цель диссертационной работы, показана новизна и практическая значимость результатов.

В первой главе приведён обзор существующих методов межпроцедурного анализа императивных программ по четырём направлениям: анализ совмещений (alias analysis), анализ значений, анализ использования переменных и анализ контекста использования. Также приведено краткое описание системы функционального программирования (SFP) и внутренних представлений, в рамках которых были внедрены алгоритмы анализа и оптимизаций.

Анализ совмещений выявляет переменные, которые могут в некоторой определённой точке исполнения программы ссылаться на одну и ту же область памяти. Полученная информация помогает, в частности, исключить неэквивалентные преобразования программы.

Анализ значений – распространение информации о возможных значениях переменной внутрь процедуры, осуществляемое на стадии компиляции.

Анализ использования переменных направлен на получение для каждой процедуры множеств модифицируемых и используемых переменных.

Информация об этих множествах востребована оптимизирующими преобразованиями, в частности, на основе этой информации часть внутреннего или внешнего кода процедуры может быть исключена.

Для более точного анализа следует рассматривать компоненты массивов отдельно. В работе рассмотрены точные и неточные алгоритмы описания областей массивов: Регулярные секции (Regular Sections), Дескрипторы доступа к данным (Data Access Descriptors), Регионы (Regions), Образы (Atom Images), Линеаризация (Linearization), Омега-тест.

Анализ контекста использования направлен на уточнение оптимизаций путём анализа различных вариантов использования процедуры. Информация о совмещениях, диапазоны изменения переменных и, как следствие, набор возможных оптимизаций зависят от контекста вызова процедуры.

Язык Sisal. Аббревиатура SISAL образована от выражения: Streams and Iteration in a Single Assignment Language. Что буквально означает: «Язык однократного присваивания с потоками и итерациями». Для функциональных языков наличие циклических конструкций не характерно, более характерным является использование рекурсии. Наличие в Sisal циклических конструкций связано с ориентацией на научные вычисления.

Ориентация на научные вычисления и наличие неявного параллелизма делают Sisal потенциально привлекательным для решения вычислительных задач. Sisal является базовым входным языком системы поддержки параллельного программирования SFP. Основные оптимизации и распараллеливание производится на уровне back-end транслятора в рамках представления IR2.

Внутреннее представление IR1, используемое в компиляторе языка Sisal, основано на языке IF1 и состоит из множества ориентированных безконтурных графов, соответствующих функциям исходной программы. **Граф IR1** $G = (N, P, E, P^{in}, P^{out})$, где N – множество вершин, P – множество портов, $E \subseteq P \times P$ – множество дуг, $P^{in} \subseteq P$ – множество входных портов графа G, $P^{out} \subseteq P$ – множество выходных портов графа G. Каждой вершине $N_i \in N$ соответствуют два подмножества портов из P: входные порты $P_i^{in} \subseteq P$ и выходные порты $P_i^{out} \subseteq P$. Дуга $E_i=(P_1, P_2) \in E$, если для некоторых i и j $P_1 \in P_i^{out}$ и $P_2 \in P_j^{in} \cup P^{out}$ или $P_1 \in P_{in}$ и $P_2 \in P_j^{in} \cup P^{out}$.

Внутреннее представление IR2 также является потоковым представлением Sisal-программы. Граф IR2 изоморфен графу IR1, но в IR2 дугам сопоставляются переменные, а между вершинами возникает нестрогое отношение порядка. Основные оптимизационные и распараллеливающие преобразования производятся на уровне этого представления. Более фор-

мально: **граф IR2** для функции языка Sisal – это объект $(G, VAR, \sigma_\upsilon, \leq_\beta)$, где G – граф IR1, VAR – множество переменных, σ_υ – отображение $E \rightarrow VAR$, задающее привязку переменных к дугам графа G, $\leq_\beta$ – порядок на $N \times N$, задающий последовательность исполнения.

Внутреннее представление IR3 является классическим императивным представлением программы в виде косвенных троек[1].

Во **второй главе** приведены теоретические результаты данной диссертационной работы. Внутренние представления языка Sisal рассмотрены более подробно. Введены понятия непосредственной вложенности, достижимости вершин в IR2, исполнения вызова и графа исполнений вызовов. Сформулированы алгоритмы преобразования и анализа программ, записанных в виде внутреннего представления IR2.

Определение 3. Составными вершинами графа IR1 будем называть вершины, которые содержат вложенный граф IR1, например Function, Select, Loop, Select Case, Select Test, Loop Init, Loop body…

Определение 4. Будем называть вершины q_1 и q_2 непосредственно соединёнными, если $\exists\ e=(p_1, p_2) \in E$, где $p_1 \in P^{out}(q_1)$, а $p_2 \in P^{in}(q_2)$ или $p_1 \in P^{out}(q_2)$, а $p_2 \in P^{in}(q_1)$.

Определение 5. Для любой составной вершины n непосредственно вложенными будем называть такие вершины q, для которых $\exists\, e=(p_1, p_2) \in$

[1] **В. А. Серебряков** *Лекции по конструированию компиляторов* // ВЦ РАН — Москва, 1994 — 174 с.

E, $p_1 \in P^{out}(n)$, $p_2 \in P^{out}(q)$ или $\exists\ q_1, q_2 \ldots q_k$ такие, что $\exists e_1=(p_1, p_2) \in E$, $p_1 \in P^{out}(n)$, $p_2 \in P^{out}(q_1)$, а пары вершин $(q_1, q_2), (q_2, q_3), \ldots (q_{k-1}, q_k), (q_k, q)$ – непосредственно соединены.

Для анализа значений описаны алгоритмы протягивания значений, протягивания мультизначений, протягивания диапазонов и протягивания мультидиапазонов. Для каждого из алгоритмов доказаны временные характеристики (Теоремы 9, 10, 11 и 12), свойства детерминированности (Теорема 7), эквивалентности исходной программы относительно подстановки найденных значений (Теорема 8).

В главе также сформулирован общий критерий подстановки: подстановку константы можно совершить на стадии компиляции тогда и только тогда, когда может быть однозначно вычислен статический срез по переменной подстановки. Здесь срез – это подмножество операторов и выражений исходной программы, которые прямо или косвенно влияют на значения, вычисляемые в точке критерия среза, но не обязательно составляют исполняемую программу.

Для анализа массивов в системе SFP было внедрено новшество, связанное с тем, что описываются не только области чтения или записи, но, так же, области возможных принимаемых значений для элементов массива. Таким образом, анализ массивов становится естественным обобщением анализа скалярных величин. Здесь и далее результаты, относящиеся к Sisal, могут быть применены к любому транслируемому в IR2 языку.

Результатом вычисления в языке Sisal может являться «ошибка», это не приводит к завершению вычисления без получения остальных результатов. Таким образом, язык Sisal реализует модель всюду завершаемых частичных вычислений. Алгоритмы анализа значений переменных позволяют существенно оптимизировать работу с такими значениями.

В главе также рассмотрены алгоритмы оптимизации циклических конструкций, удаления «мёртвого» кода и выноса инвариантных значений, которые основываются на данных, полученных в результате анализа. Доказаны временные характеристики алгоритмов (Теоремы 15 и 17).

Далее рассматривается межпроцедурный анализ.

Определение 18. Будем называть функцию *F'* «исполнением» функции *F*, если *F'* получается из *F* при конкретизации её входных параметров, в соответствии с данными, которые выявляются на стадии статического анализа. А так же если функция *F'* тождественна *F*.

Определение 19. Графом исполнений вызовов будем называть иерархический граф, в котором операторы вызова исполнений функции соединены дугами с исполнениями, которые они вызывают. Исполнения, относящиеся к одному фрагменту иерархии, относятся к одной функции исходной программы. Каждая функция исходной программы порождает один фрагмент в иерархическом графе исполнений вызовов.

Свойство 1. Граф вызовов является вырожденным случаем графа исполнений вызовов, в котором каждый фрагмент состоит из единственной вершины.

Свойство 2. Одной исходной программе могут соответствовать несколько графов исполнений вызовов.

Из определений можно отметить, что особенностью этого графа является его иерархичность. Вершины, относящиеся к функциям исходной программы, содержат вложенные вершины, которые обозначают копии тел функций, создаваемые в процессе анализа (рис. 1).

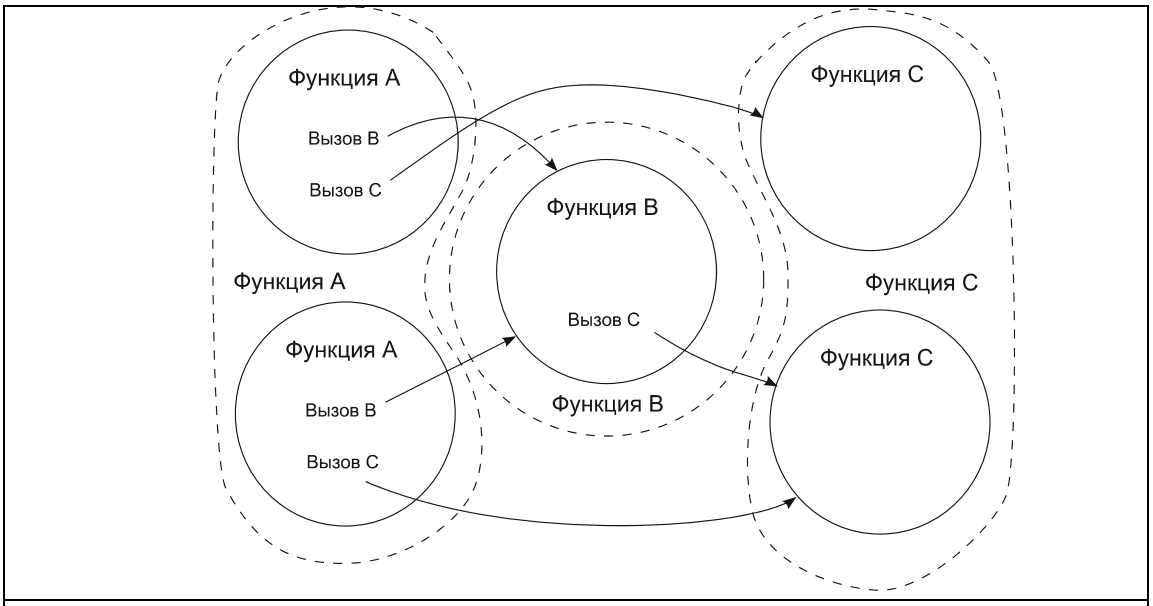

Рис. 1. Граф исполнений вызовов

Наличие копий в таком виде обусловлено тем, что оптимизирующие алгоритмы могут создавать временные копии функции, которые не будут являться отдельными функциями в результирующем коде. В этом случае нам не требуется дополнительно ассоциировать копии с копируемой функцией, поскольку связь указана явным образом, также явно ассоциируется вызываемая копия для каждой из дуг вызова.

На основе этого графа рассматривается межпроцедурный анализ без ограничений на копирование функций и межпроцедурный анализ с ограничениями на копирование функций, который реализован в рамках системы SFP. Доказываются завершимость (Теоремы 19 и 23) и свойства детерминизма (Теоремы 20, 21 и 22) алгоритмов межпроцедурного анализа.

При описании распараллеливания в системе SFP обосновывается возможность оценки временной сложности программ, записанных в виде графа IR2, при помощи инструмента временных развёрток. Доказываются свойства используемых оптимизирующих алгоритмов, которые не увеличивают время исполнения параллельной программы.

Третья глава посвящена экспериментальным результатам работы Sisal – программ. На данный момент компилятор Sisal, описанный в этой работе существует в виде Source-to-Source транслятора из Sisal 3.2 в C#. В дальнейшем планируется трансляция в язык IL внутреннего представле-

ния платформы .NET. Вместе с компилятором предоставляются средства для поддержки системы типов языка Sisal в C# и средства для организации ввода-вывода. Данные в программу поступают из xml-файла с параметрами. Аналогичным образом сохраняются результаты работы программы.

В рамках данной работы был доработан Back-End транслятор системы SFP языка Sisal. Объём исходного кода Back-End транслятора составлял 575кб, в результате изменений, отражённых в данной работе размер кода увеличился до 882кб. Библиотека поддержки во время исполнения программы (Run-Time Library) составляла 14кб исходного кода C# и была увеличена до 50кб.

Трансляцию Sisal-программы можно разбить на следующие части:

1. **Трансляция из исходного текста в представление IR1**

   Эта часть называется Front-End транслятором, на этой стадии производится лексический и семантический анализ текста. Эта стадия была полностью оставлена без изменений и не относится к результатам данной диссертационной работы
2. **Трансляция из внутреннего представления IR1 в представление IR2**

   Трансляция в IR2 была доработана. Добавлена поддержка строковых переменных, переменных типа «запись» и переменных с плавающей запятой двойной точности. Редукции суммы, максимального значения.
3. **Трансляция из IR2 в граф исполнений вызовов**

   Эта часть была добавлена в рамках данной работы. При трансляции разбираются связи между функциями, которые присутствуют в IR2 в виде символических ссылок.
4. **Межпроцедурный анализ и оптимизации**

   Эта часть также была добавлена в компилятор и является основным результатом данной работы. Внутрь межпроцедурного анализа практически без изменений были внесены оптимизации лишнего копирования и выноса инвариантных значений из циклов.
5. **Трансляция из графа исполнений вызовов в IR2**

На этой стадии производится клонирование (создание конкретизированных копий) функций программы и подстановка тел функций на место вызова.

6. Распараллеливание

Распараллеливание также изначально отсутствовало в компиляторе Sisal и является полностью результатом работы автора. Сейчас распараллеливание ориентировано на платформу SMP и тяжеловесные нити. Компилятор выбирает наиболее крупные участки кода, которые можно исполнить параллельно на заданном количестве вычислителей.

7. Трансляция из IR2 в IR3

Эта стадия была доработана не только с целью добавления новых типов и языковых конструкций, которые были отмечены на шаге 2, но и специальных конструкций, необходимых для распараллеливания и оптимизаций значений ошибочности. Были добавлены такие конструкции как ThreadFork, ThreadJoin, введено само понятие функции-нити, которая исполняется параллельно с другими функциями-нитями. Конструкция UnWarp для получения значения, содержащего свойство ошибочности из переменной, которая изначально его не содержала.

8. Оптимизации IR3

Здесь были добавлены оптимизации меток в программе с целью повышения читаемости результирующего кода, так же были доработаны существующие оптимизации удаления лишних присваиваний в виду добавления новых языковых конструкций.

9. Трансляция в C#

Добавлена поддержка нитей и ранее не используемых языковых конструкций, которые были указаны в пункте 2.

Приведены тесты компилятора на некоторых вычислительных задачах, одна из которых называлась **smooth**. Эта программа являлась фрагментом большой аэродинамической задачи от NASA, реализованной на Sisal 1.3 и транслированной в Sisal 3.2 для тестирования производительности. Задача генерирует и обрабатывает 3-мерные массивы заданного размера. Для проверки корректности работы программ, сгенерированных компилятором, вы-

ходные данные сохранялись в файл и сравнивались при помощи утилиты fc (file compare) с эталонной версией. Входные данные были получены при помощи генератора псевдослучайных чисел платформы .NET.

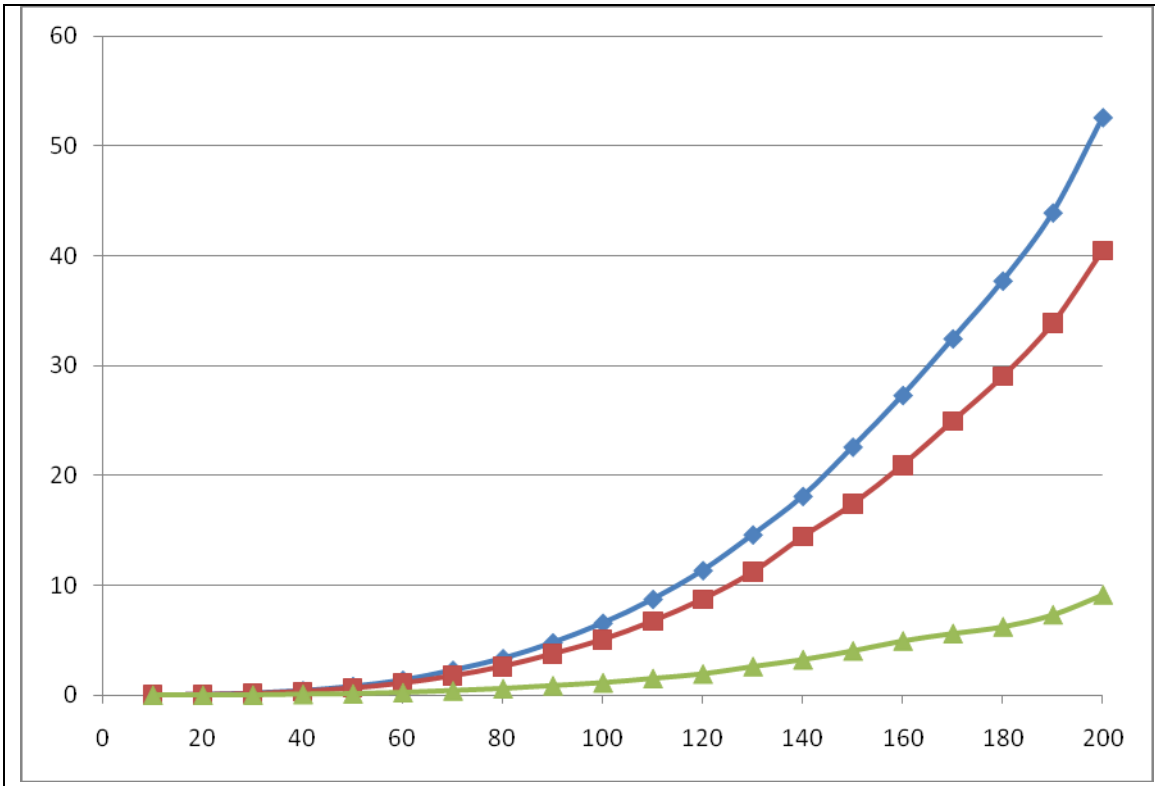

Рис. 2. Время выполнения smooth (по y, в секундах) в зависимости от размера обрабатываемых массивов. Здесь ромбом обозначается Sisal без оптимизаций, квадратом – Sisal без межпроцедурных оптимизаций, а треугольником – Sisal с полным набором оптимизаций.

Поскольку, в Sisal каждое значение может иметь тип «ошибка» и за счёт того, что обычный компилятор C#, работая с такой системой типов, не использует оптимизаций проверки ошибочности значений – оптимизированный Sisal код оказывается эффективней. Тесты производились на ПК с процессором Intel Core 2 Duo 3.16GHz на одном ядре, поскольку требовалось сравнить последовательные реализации.

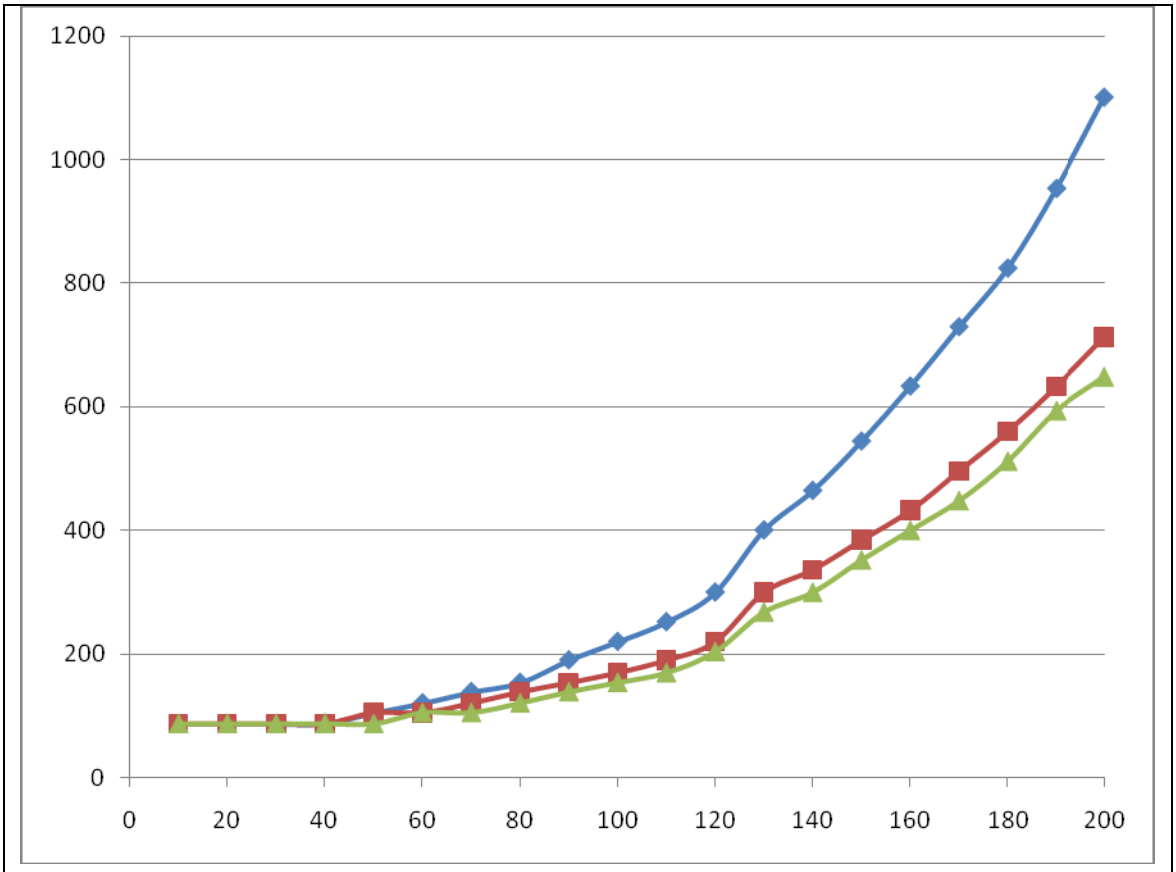

Рис. 3. Расход в Мб для задачи smooth, обозначения трендов аналогичны предыдущему графику

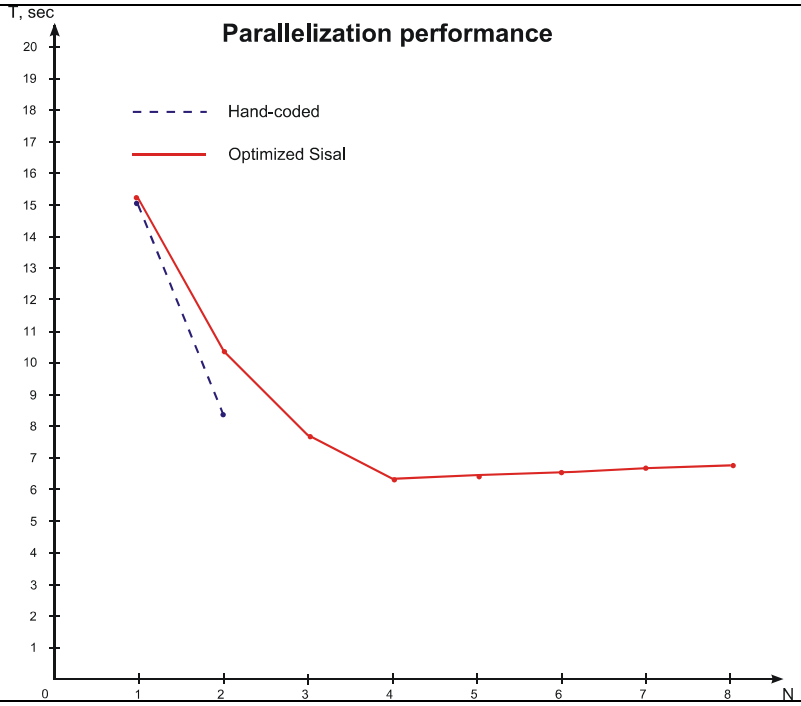

Рис. 4. Распараллеливание программы перемножения матриц

Распараллеливающая система реализована для архитектуры SMP при помощи библиотек Threading платформы ".NET". Следующие тесты производились на четырёхпроцессорной вычислительной машине. По оси X отложено количество вычислителей, заданное при компиляции Sisal-программы. После N=4 наблюдается плавный рост времени вычисления, поскольку возникают дополнительные расходы на поддержание большего числа параллельных нитей.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

1. Рассмотрены свойства и специфические особенности внутренних представлений потоковых языков. Введено понятие непосредственной вложенности вершин иерархического мультиграфа программы и доказана применимость преобразований и анализа графов потока данных к отдельным компонентам непосредственной вложенности.
2. Механизм исследования параллельных свойств императивных программ при помощи временных развёрток обобщён на граф IR2. Доказаны временные характеристики алгоритмов оптимизации и их влияние на параллельные характеристики программы.
3. Предложен и реализован в рамках проекта SFP новый метод межпроцедурного анализа, основанный на графе исполнений вызовов.
4. Впервые предложен и реализован анализ значений элементов n-мерных массивов, который показал свою эффективность в рамках оптимизации вычислений, результатом которых может быть значение «ошибка».
5. Компилятор системы SFP дополнен более развитой средой поддержки времени исполнения (RTL), конструкциями, обеспечивающими распараллеливание для платформы SMP, возможностью ввода и вывода данных посредством XML.
6. Компилятор проверен на реальных вычислительных задачах, которые показали эффективность внедрённых преобразований. Результаты вычислений сверены с неоптимизированной версией.

Все выносимые на защиту результаты принадлежат автору лично.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. **Стасенко А.П. Пыжов К. Идрисов Р.И.** Компилятор в системе функционального программирования SFP. // Вестник НГУ. Серия: информационные технологии. Т.1. Вып.2. Новосибирск: 2008. Стр. 73–90.
2. **Идрисов Р. И.** Временная развёртка внутреннего представления IR2 языка Sisal 3.1 // Методы и инструменты конструирования программ / РАН, Сиб. Отд-е, Ин-т систем информатики. – Новосибирск, 2007. – С. 31 – 37.
3. **Идрисов Р. И.** Методы межпроцедурного анализа // Методы и инструменты конструирования программ / РАН, Сиб. Отд-е, Ин-т систем информатики. – Новосибирск, 2007. – С. 38 – 55.
4. **Идрисов Р. И.** Межпроцедурный анализ в автоматических распараллеливающих системах // XLIV Междунар. науч. студенческая конф. «Студент и научно-технический прогресс»: Информационные технологии / НГУ – Новосибирск, 2006 – С. 10 – 11.
5. **Идрисов Р. И.** Russian supercomputer software // Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы / Материалы международной конференции в двух частях. Часть 2 / Петрозаводский Государственный Университет, 2006 – С. 33.
6. **Идрисов Р. И.** Протягивание констант в графе IR2 внутреннего представления языка SISAL // Конструирование и оптимизация параллельных программ / РАН Сиб. Отд-е, Ин-т систем информатики. – Новосибирск 2008. – С. 83-95.
7. **Идрисов Р. И., Пыжов К. А.** Распараллеливание и оптимизация на уровне внутреннего представления в компиляторе Sisal 3.1 // Технологии Microsoft в теории и практике программирования / Новосибирск 2008 – С. 128 – 129.
8. **Идрисов Р. И., Пыжов К.А.** Настоящее и будущее функциональных языков программирования на примере языков Sisal и F#. // Четвертая азиатская международная школа-семинар «Проблемы оптимизации сложных систем» / Труды ИВМиМГ СО РАН Выпуск 8, 2007, Серия: Информатика – С. 194-201.

9. **Идрисов Р. И.** Межпроцедурные оптимизации «ошибочности» значений для функционального языка, ориентированного на научные вычисления // Материалы XIII Международной научной конференции, посвящённой 50-летию Сибирского государственного университета имени академика М. Ф. Решетнёва/ Часть 2 – Красноярск 2009 – С. 425-426

Идрисов Р.И.

МЕЖПРОЦЕДУРНЫЙ АНАЛИЗ И РАСПАРАЛЛЕЛИВАНИЕ ПОТОКОВЫХ ПРОГРАММ НА БАЗЕ ГРАФА ИСПОЛНЕНИЙ ВЫЗОВОВ

Автореферат

Подписано в печать Объем 1,1 уч.-изд. л.
Формат бумаги 60 × 90 1/16 Тираж 100 экз.
Отпечатано в ЗАО РИЦ «Прайс-курьер»
630090, г. Новосибирск, пр. Ак. Лаврентьева, 6, тел. 334-22-02
Заказ №135