На правах рукописи

Метелица Елена Анатольевна

АВТОМАТИЗАЦИЯ РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ СО СЛОЖНЫМИ ИНФОРМАЦИОННЫМИ ЗАВИСИМОСТЯМИ

Специальность 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание учёной степени
кандидата физико-математических наук

Ростов-на-Дону – 2024

Работа выполнена в Федеральном государственном автономном образовательном учреждении высшего образования «Южный федеральный университет».

Научный руководитель:	Штейнберг Борис Яковлевич, д.т.н., с.н.с., заведующий кафедрой алгебры и дискретной математики Института математики, механики и компьютерных наук им. И.И. Воровича Федерального государственного автономного образовательного учреждения высшего образования «Южный федеральный университет».
Официальные оппоненты:	Легалов Александр Иванович, д.т.н., профессор департамента программной инженерии, факультета компьютерных наук Федерального государственного автономного образовательного учреждения высшего образования «НИУ «Высшая школа экономики». Терехов Андрей Николаевич д.ф-м.н., профессор, заведующий кафедрой Системного программирования Федерального государственного бюджетного образовательного учреждения высшего образования «Санкт-Петербургский государственный университет».
Ведущая организация:	Федеральное государственное бюджетное учреждение науки Институт систем информатики имени А. П. Ершова Сибирского отделения РАН.

Защита состоится «04» февраля 2025 г. в 11 часов на заседании диссертационного совета № 24.1.237.02, созданного на базе ИПМ им. М.В. Келдыша РАН, 125047, Москва, Миусская пл., д.4.

С диссертацией можно ознакомиться в библиотеке и на сайте диссертационного совета № 24.1.237.02 https://www.keldysh.ru/council/1/.

Автореферат разослан «____» _____ 2024 г.

Учёный секретарь
диссертационного совета 24.1.237.02 М.Г. Широбоков
к.ф.-м.н., доцент

Общая характеристика работы

Актуальность работы

Существуют задачи, решение которых занимает много времени. К ним относятся численные методы решения задач математической физики, обработки изображений, машинного обучения, исследования космоса, создания новых лекарств, прогнозирования погоды и др. Иногда такие задачи не решаются из-за отсутствия подходящих высокопроизводительных вычислительных систем и соответствующего программного обеспечения.

Зачастую программным обеспечением используются не все возможности вычислительной системы[1]. Поэтому актуальной является проблема эффективного использования ресурсов компьютера. Есть библиотеки, ориентированные на решение часто используемых задач, например, BLAS для задач линейной алгебры. Такие библиотеки эффективны за счёт того, что они учитывают особенности конкретной вычислительной системы или процессора. Однако существует много задач и процессоров, для которых не созданы библиотеки.

В то же время развивается разработка многоядерных процессоров: К1879ВМ8Я (НТЦ «Модуль»), процессоры «Эльбрус», Colossus™ MK1 IPU и Colossus™ MK2 IPU, Untether AI Boqueria, Moffett AI S4 Antoum, SambaNova SN30 RDU, Amazon Web Services Trainium1, Tesla Dojo D1.

Для создания высокопроизводительных программ, решающих задачи, требующие больших объёмов вычислений, могут использоваться оптимизирующие компиляторы. Есть исследования[1], которые показывают, что современные оптимизирующие компиляторы плохо оптимизируют программы. В основе оптимизирующих компиляторов лежит теория преобразования программ. Однако не все возможные оптимизирующие преобразования реализованы в современных компиляторах, тем более что усложнение вычислительных архитектур влечет усложнение оптимизирующих преобразований программ. В частности, в ускорении нуждаются численные алгоритмы решения дифференциальных уравнений в частных производных в задачах математического моделирования и другие алгоритмы, в основе которых лежат гнёзда циклов итерационного типа.

Создание эффективных распараллеливающих компиляторов сократит сроки разработки высокопроизводительных программ и понизит их себестоимость за счет снижения требований к квалификации разработчиков.

Степень разработанности темы

В середине XX века были написаны первые работы по автоматическому распараллеливанию. В эту область внесли вклад U. Bondhugula, M. Лэм. М. Вольф, P. Feautrier, В.Ю. Волконский, А.П. Ершов, В.А. Вальковский, Б.Я. Штейнберг, В.Н. Касьянов, В.А. Евстигнеев, А.И. Легалов. В.А. Крюков, В.В. Воеводин, В.Э. Малышкин, R. Arora и др. Автоматической оптимизации и распараллеливанию обычно предшествует «ручная» оптимизация программ, существенных результатов в

[1] Gong Z., Chen Z., Szaday Z., Wong D., Sura Z., Watkinson N., Maleki S., Padua D., Veidenbaum A., Nicolau A. An empirical study of the effect of source-level loop transformations on compiler stability // Proceedings of the ACM on Programming Languages. 2018. № 2. P. 1–29.

которой достигли В.Д. Левченко, J. Dongarra, L. Lamport, Н. Лиходед, K. Goto, В.В. Корнеев и др. Основы теории преобразований программ закладывались А.П. Ершовым.

К компилирующим инструментам автоматического распараллеливания относятся распараллеливающие системы DVM, LuNA, SUIF, OPC и др. Существуют разработки в области интерактивного распараллеливания, например, Interactive Parallelization Tool (IPT). Современные компиляторы (ICC, GCC, LLVM, PGI-Compiler, ROSE-Compiler и др.) хорошо оптимизируют базовые блоки и самые глубоко вложенные циклы гнезда. Эти компиляторы являются многопроходными, но не всегда находят оптимальные цепочки преобразований. Преобразования «тайлинг», «скашивание» и «метод гиперплоскостей» используются в системах PLUTO, PolyMage, POLLY-llvm, MLIR и др. Эти системы используют внутреннее представление полиэдра (polyhedral model), которое ориентировано на анализ и преобразования гнёзд циклов. Для ускорения гнёзд циклов итерационного типа наиболее эффективной системой является PLUTO, что отражается в публикациях последних лет. Система PLUTO имеет открытый для доступа код, что позволило провести экспериментальное сравнение с программной реализацией предлагаемых в представленной работе методов на одном и том же компьютере. Можно отметить, что полиэдральное внутреннее представление ориентировано на преобразование гнёзд циклов, однако оно не позволяет использовать многие другие оптимизирующие преобразования. Универсальное внутреннее представление, основанное на AST, имеет дополнительные возможности для оптимизации программ.

В данной работе предлагаются новые методы ускорения программ, в частности гнезд циклов итерационного типа, которые часто встречаются в задачах математического моделирования. Эти методы основаны на анализе сложных информационных зависимостей в программе. Используется и обобщается понятие вектора расстояния зависимости, которое определяется на основе решетчатых графов. В тех случаях, когда компилятору недостаточно информации для определения зависимости, предлагается применять метод диалогового уточнения зависимостей, использующий символьный анализ текстов программ. Представленные алгоритмы автоматизации ускорения программ иллюстрируются на численных методах решения задач математической физики.

Цель диссертации состоит в разработке методов создания инструментов автоматизированной оптимизации программ, в частности, включающих гнёзда циклов итерационного типа.

Для достижения поставленной цели в рамках данной работы решаются следующие **задачи**:

1. Разработать на основе внутреннего представления оптимизирующей распараллеливающей системы (OPC) алгоритмы автоматизации выполнения преобразований тесных гнезд циклов: «скашивание» (Loop Skewing), «тайлинг» (Loop Tiling) и «метод гиперплоскостей» (Wavefront).
2. Разработать метод ускорения гнёзд циклов итерационного типа, включающий преобразования «скашивание», «тайлинг», «метод гиперплоскостей» и дополнительные оптимизации выражений и циклов.

3. Разработать метод диалогового анализа и преобразований текстов программ на основе символьного анализа.

Методы исследований

Поставленные задачи решались посредством методов теории преобразования программ, теории графов, теории языков программирования, линейной алгебры. Представленные в диссертации алгоритмы программно реализованы на основе методов объектно-ориентированного программирования на языке C++.

Научная новизна

Разработана цепочка оптимизирующих преобразований гнёзд циклов итерационного типа, основанная на универсальном высокоуровневом древовидном внутреннем представлении. Такой подход отличается от известного метода, реализованного на полиэдральном внутреннем представлении, наличием возможности использовать дополнительные оптимизирующие преобразования для ускорения программ.

Разработан и теоретически обоснован новый метод обхода точек тайла, который повышает временную локальность данных и даёт ускорение.

Предложен метод определения оптимальных размеров тайлов для получения наилучшего ускорения.

Предложен метод диалогового уточнения информационных зависимостей, позволяющий распараллеливать более широкий класс циклов, чем позволяли прежние методы.

Теоретическая и практическая значимость работы

Теоретическая значимость состоит в развитии теории преобразования программ для компилирующих систем с высокоуровневым универсальным (древовидным) внутренним представлением:

- приводится обобщение вектора расстояний, которое необходимо для обоснования целевых преобразований гнёзд циклов итерационного типа;
- теоретически обосновывается целесообразность применения перестановки циклов внутри тайла;
- обосновывается эквивалентность преобразования "метод гиперплоскостей" с предлагаемым в данной работе вектором нормали;
- доказано, что результирующая и исходная программы имеют одинаковые условия сходимости (для итерационных алгоритмов), одинаковые погрешности машинных округлений и одинаковые условия переполнения регистров (превышения максимальных размеров типов данных);
- предложены методы расчета оптимальных размеров тайлов;
- предложен новый метод диалоговой оптимизации и распараллеливания циклов основанный на символьном анализе.

Практическая значимость работы заключается в том, что реализация преобразований программ для ускорения гнезд циклов на основе ОРС позволяет расширить множество оптимизируемых программ и повысить их ускорение.

Разработана «Программа, реализующая параллельный алгоритм Гаусса – Зейделя для задачи Дирихле», которая может служить прототипом решателя промышленного пакета прикладных программ для решения задач математического моделирования. Разработан диалоговый анализатор, который может рассматриваться как прототип блока полуавтоматического анализа программ в перспективных инструментах оптимизации. Реализованные преобразования программ «метод гиперплоскостей» и «скошенный тайлинг» в составе ОРС могут войти в состав распараллеливающих компиляторов для перспективных вычислительных систем.

Результаты работы использовались при выполнении грантов:

- Российского научного фонда № 22–21–00671;
- Правительства РФ № 075-15-2019-1928.

Основные положения, выносимые на защиту:

1. Предложена цепочка преобразований гнезд циклов итерационного типа, которая дает более высокое ускорение, чем известные методы; доказана эквивалентность данной цепочки.
2. Предложен метод диалогового анализа текстов программ, основанный на символьном анализе, который расширяет применимость оптимизирующих и распараллеливающих преобразований программ.
3. Предложен подход к реализации преобразований гнёзд циклов «скашивание циклов», «метод гиперплоскостей», «скошенный тайлинг» на основе внутреннего представления ОРС, что позволяет расширять цепочки из этих преобразований программ дополнительными преобразованиями, реализованными на основе универсального древовидного внутреннего представления.
4. Предложены методы обхода тайлов и определения их размеров, которые ускоряют выполнение программ.

Достоверность полученных результатов подтверждается корректным использованием математических формулировок и результатов теории преобразования программ, линейной алгебры, теории графов; тестированием корректности и производительности разработанного ПО; успешным применением разработанного ПО в научной деятельности; полученными и опубликованными результатами численных экспериментов по ускорению целевых алгоритмов.

Апробация работы

Результаты научной работы были апробированы на:

1) Всероссийской XXVII научной конференции «Современные информационные технологии: тенденции и перспективы развития» в 2020 году, доклад «Использование тайлинга для оптимизации итерационных алгоритмов»; (доклад без соавторов)
2) Национальном суперкомпьютерном форуме (НСКФ 2020) с докладом «Использование блочных алгоритмов для оптимизации гнёзд циклов»; (доклад без соавторов)
3) конференции 16th International Conference on Parallel Computing Technologies (PaCT-21) с докладом «Precompiler for the ACELAN-COMPOS package solvers»; (вклад

соискателя «методы оптимизации гнезд циклов итерационного типа и результаты численных экспериментов»)
4) Национальном суперкомпьютерном форуме (НСКФ 2021) с докладами «О сочетании распараллеливания и скошенного тайлинга» (вклад соискателя «методы оптимизации гнезд циклов итерационного типа, новый метод обхода точек тайла и результаты численных экспериментов для алгоритма Гаусса-Зейделя решения двумерной задачи Дирихле»), «О высокоуровневом автоматическом оптимизаторе программ» (вклад соискателя «методы оптимизации гнезд циклов итерационного типа, новый метод обхода точек тайла и результаты численных экспериментов для алгоритма Гаусса-Зейделя решения двумерной задачи Дирихле и обобщённой задачи Дирихле»
5) Национальном суперкомпьютерном форуме (НСКФ 2022) с докладом «Автоматизация распараллеливания программ с оптимизацией размещения и пересылок данных»; (вклад соискателя «анализ возможности распараллеливания алгоритма Гаусса-Зейделя для задачи Дирихле на вычислительной системе с распределенной памятью»)
6) конференции International scientific workshop OTHA Spring 2023 с докладом «Multidimensional convolution»; (вклад соискателя «методы ускорения алгоритма Гаусса-Зейделя для двумерной задачи Дирихле»)
7) конференции International Conference on Parallel Computing Technologies (PaCT-23) с докладом «Automatic Parallelization of Iterative Loops Nests on Distributed Memory Computing Systems»; (вклад соискателя «методы ускорения алгоритмов итерационного типа»)
8) конференции 12th International Young Scientists Conference in Computational Science с докладом «Combination of parallelization and skewed tiling» (вклад соискателя «методы ускорения алгоритмов итерационного типа»).

По теме диссертации опубликовано 11 работ, из которых 6 – статьи в изданиях, индексированных в БД Scopus или WoS, 2 статьи в журналах перечня ВАК, 1 свидетельство о государственной регистрации программы для ЭВМ, а остальные индексированы в РИНЦ.

Соответствие специальности

По своему научному содержанию диссертационная работа соответствует паспорту специальности 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей», пунктам: №1 «Модели, методы и алгоритмы проектирования, анализа, трансформации, верификации и тестирования программ и программных систем»; №2 «Языки программирования и системы программирования, семантика программ»; №8 «Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования».

Личный вклад автора

Автором на основе теории оптимизирующих преобразований программ разработан и реализован алгоритм автоматической оптимизации (и, в частности, распараллеливания) гнёзд циклов итерационного типа. Проведены численные

эксперименты преобразованных программ. В статьях с соавторами результаты, относящиеся к оптимизации алгоритмов итерационного типа, принадлежат автору. Сформулированы и доказаны условия эквивалентности цепочек преобразований.

Задача поставлена научным руководителем. В диссертации использована оптимизирующая распараллеливающая система, которая разрабатывалась многими студентами и аспирантами мехмата ЮФУ.

Структура и объём работы

Научно-квалификационная работа состоит из введения, четырёх глав, заключения, списка литературы из 148 наименований. Текст диссертации изложен на 152 (182 с приложениями) страницах и содержит 91 рисунок, 20 таблиц, 41 пример, 18 приложений. Проверка научно-квалификационной работы системой «Антиплагиат» показала 84,41% оригинальности текста.

Основное содержание работы

Во введении приводится обзор работ по теме исследования. Описываются известные системы автоматизированной оптимизации и распараллеливания программ. Для гнёзд циклов итерационного типа наиболее эффективной является система PLUTO основанная на полиэдральном внутреннем представлении. Далее будет представлено сравнение программ, преобразованных при помощи PLUTO и метода, предлагаемого в данной работе.

Обосновывается актуальность темы, формулируются цели и задачи исследования, описываются методы исследования, излагаются основные положения научной новизны и значение работы для теории и практики. Приведён список положений, выносимых на защиту. Изложена структура диссертации и краткое содержание работы по главам.

Первая глава носит вспомогательный характер. В ней описываются алгоритмы итерационного типа, приводятся элементы теории преобразования программ, граф информационных зависимостей, решетчатый граф, анализ информационных зависимостей с помощью векторов расстояний (distance vector), описание и анализ блочных и вспомогательных преобразований гнёзд циклов с использованием унимодулярных матриц преобразований.

Приведём некоторые определения для лучшего понимания результатов работы.

Гнездо циклов итерационного типа – это такое гнездо циклов, которое обладает следующими свойствами. Внешний цикл отвечает за обход итераций, а внутренние – за расчёт шаблона вычислений, который повторяется на итерациях внешнего гнезда. В данной работе рассматриваются такие гнёзда циклов итерационного типа, которые являются тесными, количество итераций не меняется на момент выполнения и в тело которых входят только операторы присваивания, безындексные переменные, константы, массивы в стиле языка C; массивы имеют линейные индексные выражения вида (i + <целочисленная константа>), где i – счетчик одного из циклов. Более точно: в теле итерационного гнезда циклов (LoopBody (i_1,i_2,...,i_n)) могут присутствовать генераторы вида u[i_1-c_1][i_2-c_2][…][i_n-c_n], (где c_1, c_2, …, c_n – целочисленные константы), в которых последовательность индексных выражений массива (i_1-c_1,i_2-c_2,…,i_n-c_n) соответствует последовательности

счетчиков гнезда циклов (i_1,i_2,…,i_n). Тогда использования этой же переменной u в правой части оператора присваивания должны иметь вид: u[i_1-d_1][i_2-d_2][…][i_n-d_n], где d_1, d_2, …, d_n – целочисленные константы. Общий вид такого гнезда циклов представлен ниже:

```
for (int i0 = 0; i0 < itera; i0++)
     for (int i1 = a1; i1 < b1; i1++)
          for (int i2 = a2; i2 < b2; i2++)
               …
                    for (int in = an; in < bn; in++) {
                         LoopBody(i1,i2,...,in);
                    }
```

Из-за специфики информационных зависимостей ни один цикл данного гнезда не может выполняться параллельно, поэтому перед распараллеливанием надо выполнить его преобразование.

Рассмотрим гнездо из n+1 вложенных друг в друга циклов. Пронумеруем операторы циклов в этом гнезде в соответствии с порядком вложенности, начиная с самого внешнего цикла. Обозначим счетчик j-го цикла – I_j, где j от 0 до n. Множество значений, которые может принимать вектор счетчиков циклов I = (I_0, I_1, …, I_n) в указанном гнезде, называется **пространством итераций** данного гнезда.

Если сделать раскрутку всех циклов гнезда, то получим код без циклов, состоящий из множества блоков, каждый из которых является копией тела исходного гнезда циклов. Каждая такая копия тела гнезда циклов соответствует набору значений счетчиков циклов гнезда. Будем обозначать копию вхождения v в раскрутке исходного гнезда циклов при значениях счетчиков I′ = (I'_0, I'_1, . . . , I'_n) следующим образом: v(I'_0, I'_1, . . . , I'_n) или v(I′) и называть представителем данного вхождения.

Сформулируем понятие **вектора расстояния (distance vector)**[2] для информационных зависимостей в гнезде циклов итерационного типа. Пусть размерность гнезда циклов итерационного типа равна n+1, u и v – вхождения n-мерного массива a, образующие дугу информационной зависимости (u, v) = (a[i_1-c_1][i_2-c_2][…][i_n-c_n], a[i_1-d_1][i_2-d_2][…][i_n-d_n]). Счетчик внешнего цикла (имеющего порядковый номер 0), вычисляющего итерации алгоритма, не входит в индексные выражения массивов (согласно определению гнезда циклов итерационного типа). Тогда существует такая пара представителей, для которых есть дуга в решетчатом графе (u(i'_0,i'_1,i'_2,…,i'_n), v(i''_0,i''_1,i''_2,…,i''_n)). Пример решетчатого графа изображен на рисунке 2.

Рассмотрим вектор длины n+1 разностей векторов счетчиков циклов (i''_0-i'_0, i''_1-i'_1, i''_2-i'_2,…,i''_n-i'_n). Все координаты, начиная с первой, не зависят от выбора пары представителей u, v. А нулевая координата t всегда неотрицательна (поскольку дуга информационной зависимости направлена от u к v). Тогда определим для гнезда циклов итерационного типа вектор расстояний дуги информационной зависимости (u, v): dist(a[i_1-c_1][i_2-c_2][…][i_n-c_n], a[i_1-d_1][i_2-d_2][…][i_n-d_n]) = (t,d_1-c_1,d_2-c_2,…,d_n-c_n), t >= 0.

[2] Maydan D.E., Hennessy J.L., Lam M.S. Efficient and exact data dependence // ACM SIGPLAN Notices. 1991. Vol. 26, № 6. P. 1–14.

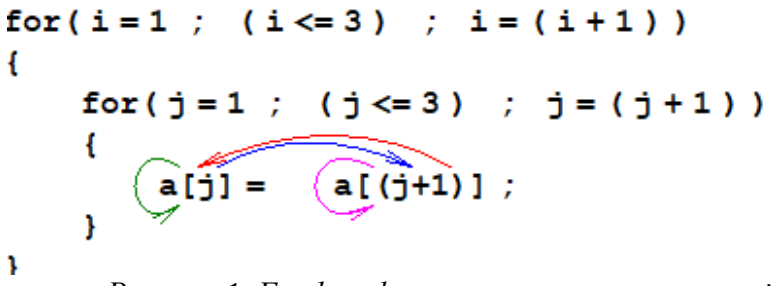

Рисунок 1. Граф информационных зависимостей программы, построенный ОРС. Красная дуга – дуга антизависимости

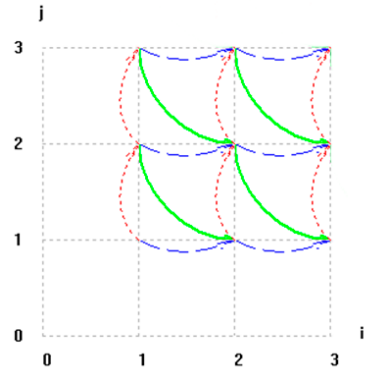

Рисунок 2. Решетчатый граф программы, построенный ОРС

Будем рассматривать далее вектор расстояния зависимости для гнезд циклов итерационного типа, полагая, что его нулевая координата равна 1, поскольку нам для преобразования «скошенный тайлинг» достаточна её положительность.

Пример 1. Проанализируем гнездо циклов, используя граф информационных зависимостей, решётчатый граф, и вычислим вектор расстояний.

На графе информационных зависимостей присутствуют дуги выходной, истинной и антизависимости (рисунок 1). Для дуги выходной зависимости вектор расстояния равен (1,0), истинной (1,-1), для антизависимости (0,1) (рисунок 2).

В данной работе решетчатый граф используется концептуально для иллюстрации информационных зависимостей.

Скашивание (Loop Skewing)[3] – преобразование, которое изменяет пространство итераций гнезда циклов: вектор счетчиков нового гнезда циклов I' равен вектору счетчиков исходного гнезда I, умноженного на нижнетреугольную матрицу с единицами на главной диагонали skew: I'=skew x I. Число f, которое находится в строке m и в столбце k (k<m) будем называть параметром скашивания цикла номер k относительно цикла под номером m. Матрица skew преобразования «скашивание» для двумерного гнезда циклов имеет вид: skew = $\begin{pmatrix} 1 & 0 \\ f & 1 \end{pmatrix}$.

Преобразование «скашивание циклов» является эквивалентным.

Скашивание меняет векторы расстояний информационных зависимостей: матрица преобразования «скашивание циклов» для примера 1 имеет вид: $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, вектор расстояний для дуги истинной информационной зависимости: (a[j], a[j+1]) (dist(a[j], a[j+1]) = (1, -1)) после скашивания будет иметь вид: $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \text{x} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Метод гиперплоскостей (Loop Wavefront)[4] – это преобразование гнезда циклов, которое меняет обход точек пространства итераций следующим образом. Пространство итераций разбивается на параллельные гиперплоскости (с общим вектором нормали). Гиперплоскости обходятся в том порядке, который указывает вектор нормали. Точки, находящиеся на одной гиперплоскости, обходятся в лексикографическом порядке. В данной работе для преобразования гнёзд

[3] Wolfe M. Loop skewing: the wavefront method revisited. / Int J Parallel Program. 1986. Vol. 15, № 4. P. 279–293.
[4] Lamport L. The parallel execution of DO loops // Commun. ACM. 1974. Vol. 17, № 2. P. 83–93.

итерационного типа метод гиперплоскостей будет применяться поэтапно к парам соседних циклов. Метод гиперплоскостей выполняется таким образом, чтобы точки, находящиеся на одной гиперплоскости, были информационно независимы, что позволяет выполнять их параллельно.

Тайлинг (Loop Tiling)[5,6] – это преобразование программ, которое разбивает пространство итераций исходного гнезда цикла параллельными плоскостями (гиперплоскостями) на блоки (тайлы) меньшего размера и просматривает точки пространства итераций поблочно. Прямоугольный тайлинг – это тайлинг, у которого блоки являются прямоугольными параллелепипедами, грани которых перпендикулярны соответствующим координатным осям. Это преобразование является комбинацией преобразований «гнездование циклов» и «перестановка циклов». Чтобы прямоугольный тайлинг оставался эквивалентным преобразованием, в гнезде циклов должны отсутствовать информационные зависимости, для которых вектор расстояний имеет отрицательные координаты.

```
for (ii = 0; ii < N1 / d1; ii += 1)
 for (jj = 0; jj < N2 / d2; jj += 1)
  for (i = ii * d1; i < (ii + 1) * d1; i = i + 1)
   for (j = jj * d2; j < (jj + 1) * d2; j = j + 1)
    LoopBody(i, j);
```

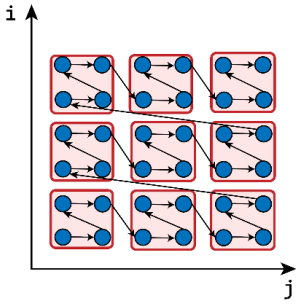

Рисунок 3. Пространство итераций двумерного гнезда циклов после тайлинга. Дуги показывают порядок выполнения точек пространства итераций. Точки каждого тайла обведены

Скошенный тайлинг (Skewed Loop Tiling)[6] – преобразование, которое является комбинацией скашивания и прямоугольного тайлинга.

```
for (int ii = 0; ii < (N - initN - 1) / d1 + 1; ii++)
 for (int jj = 0; jj < ((M - initM) + (N - initN) - 1 -
1) / d2 + 1; jj++)
  for (int i = ii * d1; i < min((ii + 1) * d1, N -
initN); i++)
   for (int jjj = max(i, jj * d2); jjj < min((jj + 1) *
d2, (M - initM) + i); jjj++)
LoopBody(i, jjj - i);
```

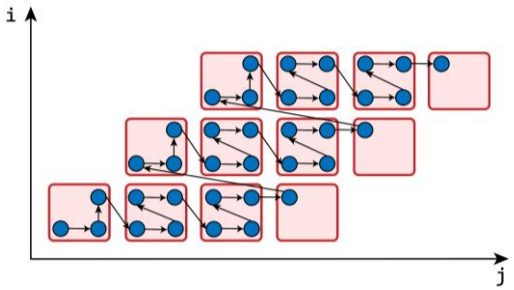

Рисунок 4. Пространство итераций двумерного гнезда циклов после скошенного тайлинга. Дуги показывают порядок выполнения точек пространства итераций. Точки, принадлежащие одному тайлу, обведены

[5] Irigoin F., Triolet R. Supernode Partitioning // POPL'88. 1988. P. 319–329.
[6] Wolf M.E., Lam M.S. A loop transformation theory and an algorithm to maximize parallelism // IEEE Transactions on Parallel and Distributed Systems. 1991. № 2(4). P. 452–471.

Гнёзда циклов, для которых прямоугольный тайлинг не применим, иногда можно преобразовывать, используя скошенный тайлинг, поскольку за счёт скашивания изменяются векторы расстояний информационных зависимостей.

В рамках научного исследования приведённые преобразования реализованы в оптимизирующей распараллеливающей системе (OPC), имеющей высокоуровневое внутреннее представление.

Во второй главе приводится разработанный алгоритм оптимизации и распараллеливания итерационных гнёзд циклов.

На вход алгоритму подается программа на языке C, содержащая тесное гнездо циклов итерационного типа.

Алгоритм оптимизации итерационных гнёзд циклов сводится к выполнению цепочки преобразований:

1. Вычисление параметров преобразования «скашивание» на основе анализа информационных зависимостей (параграф 2.1).
2. Применение скашивания (если необходимо) (параграф 2.2).
3. Применение тайлинга (параграф 2.2).
4. Перестановка циклов внутри тайла для повышения временно́й локальности данных (параграф 2.3).
5. Применение метода гиперплоскостей (параграф 2.4).
6. Применение дополнительных преобразований (параграф 3.8).
7. Применение прагм OpenMP для параллельного выполнения (параграф 2.4).

Результатом применения алгоритма является преобразованная программа на языке C. Полученная программа может компилироваться любым из компиляторов (GCC, ICC, LLVM, MS-Compiler и другие), включая компиляторы с закрытым кодом.

Для определения подходящего алгоритма тайлинга (прямоугольный или скошенный) производится анализ дуг графа информационных зависимостей. Для каждой дуги вычисляется вектор расстояний (distance vector). Если хотя бы в одном векторе расстояний присутствуют отрицательные координаты, то перед применением тайлинга выполняется скашивание.

Параметрами тайлинга является целочисленный массив размеров тайлов tile_sizes[i], где i от 0 до n и матрица скашивания (skew).

Далее описывается **метод изменения обхода точек тайла**, который достигается перестановкой циклов внутри тайла, что повышает временную локальность данных. Обосновывается целесообразность её применения.

Описывается применение метода гиперплоскостей и прагм OpenMP для распараллеливания преобразованной программы.

Теорема 1. После применения тайлинга тайлы, лежащие на одной гиперплоскости с вектором нормали (1,1,…,1), попарно не связаны информационными зависимостями, и, следовательно, могут выполняться параллельно.

Следствие 1. Применение метода гиперплоскостей с вектором нормали (1,1,…,1) и последующее распараллеливание являются эквивалентными.

Теорема 2. Алгоритм оптимизации итерационных гнёзд циклов и его расширение являются эквивалентными.

В конце второй главы рассматриваются перспективы развития и применения преобразований гнёзд циклов на основе ОРС. Сравниваются низкоуровневое и высокоуровневое внутренние представления для реализации преобразований программ. Анализируется возможность применения полученного алгоритма для распараллеливания на вычислительную систему с распределённой памятью.

В третьей главе приводятся метод выбора оптимальных размеров тайлов и численные эксперименты выполнения программ, преобразованных при помощи разработанного алгоритма оптимизации итерационных гнёзд циклов.

Численные эксперименты были проведены на компьютере с процессором Intel Core i7-9700 (Coffee Lake), тактовая частота 3,00 GHz, размер кеш-памяти: L1=256Kb; L2=2Mb; L3=12Mb; компиляторы GCC-10.1.0 и ICC с опцией компилятора -O3. Ускорение демонстрируется относительно исходной последовательной программы и вычисляется по формуле: *Ускорение = (Время выполнения исходной программы) / (Время выполнения преобразованной программы).*

Пиковая производительность процессора для чисел с плавающей точкой одинарной точности: P = 3 ГГц * 8 ядер * 32 ФЛОПС/такт = 768 ГФЛОПС; для чисел двойной точности: P = 3 ГГц * 8 ядер * 16 ФЛОПС/такт = 384 ГФЛОПС.

В параграфе 3.1 приводится метод выбора оптимальных размеров тайлов (d1, d2, d3) на примере алгоритма Гаусса – Зейделя решения двумерной задачи Дирихле уравнения Лапласа. Выбор величин d1 и d2 определяется по формуле:

$$d1 = d2 \leq \sqrt{|L1| + 4} - 2.$$

Для процессора, на котором проводились вычисления, объем L1-кеш равен 32К и теоретические значения получаются d1=d2=64 числа типа double. Численные эксперименты показали при наиболее близких значениях размеров тайла d1=64, d2=50, d3=50 ускорение **18,57**, которое незначительно отличается от лучшего ускорения **18,67** при d1=64, d2=40, d3=40 (таблица 1).

Демонстрируется влияние величины d3 на ускорение при фиксированных значениях d1, d2. При последовательном выполнении (таблица 2) преобразованной программы наблюдается постепенное увеличение ускорения в пределах 1.1 секунды и при увеличении значения прирост уменьшается. Это связанно с затратами на вычисление первого сечения тайла. Данные для первого сечения подгружаются из оперативной памяти, а для последующих – большая часть из кеш-памяти и регистров.

При распараллеливании наблюдается неравномерное повышение ускорения преобразованной программы при увеличении размера тайла d3 (таблица 2).

Предлагается теоретически оптимальное значение d3 равным d2, которое подтверждается численными экспериментами (таблица 1).

Для выбранной задачи были проведены дополнительные численные эксперименты при фиксированном значении d3=500. Взяты ближайшие размеры тайла, наиболее близкие к теоретическим d1=64, d2=50. Время выполнения для тайла с такими размерами 0.555 (ускорение 20 раз). Полученное ускорение отличается на ~5.6% от наибольшего достигнутого ускорения при d1=128, d2=50, время выполнения 0,526 (ускорение 21,19).

Таблица 1. Результаты оптимизации алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнения Лапласа. Размерность задачи – 256 x 4000 x 4000, тип данных – double. Компилятор – GCC, опция – -O3. Время выполнения исходного алгоритма – 11,202 сек.

Число потоков	Последовательно		8 потоков		16 потоков	
Размеры блоков (блоков)	Время выполнения	Ускорение	Время выполнения	Ускорение	Время выполнения	Ускорение
d1=32, d2=40, d3=40	3,075	3,64	1,060	10,57	0,649	17,26
d1=32, d2=80, d3=80	2,954	3,79	1,015	11,04	0,677	16,55
d1=32, d2=100, d3=100	3,088	3,63	1,031	10,87	0,687	16,31
d1=64, d2=20, d3=20	3,373	3,32	1,172	9,56	0,707	15,84
d1=64, d2=40, d3=40	**2,852**	**3,93**	0,996	11,25	**0,600**	**18,67**
d1=64, d2=50, d3=50	2,884	3,88	**0,969**	**11,56**	0,603	18,57
d1=64, d2=80, d3=80	2,914	3,85	0,972	11,53	0,652	17,19

Рассчитаем долю пиковой производительности для лучшего ускорения (0,6 секунд). Производительность: $\frac{16.384*10^9}{0.6} \approx 27.31$ ГФЛОПС. Доля пиковой производительности $= \frac{27.31}{384} * 100\% \approx 7.11\%$.

Таблица 2. Влияние величины d3 на ускорение алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнения Лапласа. Размерность задачи – 256 x 4000 x 4000, тип данных – double. Компилятор – GCC, опция – -O3. Время выполнения исходного алгоритма – 11,2024 сек.

	Последовательно				16 потоков			
	d1=32		d1=64		d1=32		d1=64	
d3	Время (сек.)	Ускорение	Время (сек.)	Ускорение	Время (сек.)	Ускорение	Время (сек.)	Ускорение
4	3,823	2,92	3,643	3,06	1,400	7,96	1,223	9,11
8	3,359	3,32	3,257	3,42	1,013	11,00	0,947	11,77
20	3,088	3,61	2,990	3,73	0,817	13,65	0,758	14,70
40	3,007	3,71	2,880	3,87	0,650	17,13	0,605	18,41
80	2,959	3,77	2,815	3,96	0,692	16,09	0,664	16,79
125	2,920	3,82	2,825	3,94	0,654	17,04	0,639	17,45
200	2,899	3,84	2,755	4,05	0,691	16,13	0,672	16,58
250	2,895	3,85	2,754	4,05	0,615	18,12	0,605	18,43
400	2,907	3,83	2,754	4,05	0,807	13,80	0,751	14,84
500	2,887	3,86	2,731	4,08	0,573	19,46	0,558	19,97
1000	2,877	3,87	2,723	4,09	0,970	11,48	0,891	12,51
2000	2,876	3,88	2,709	4,11	1,727	6,45	1,585	7,03

После применения предложенного алгоритма для компиляции могут использоваться различные компиляторы. Таблица 3 демонстрирует влияние используемого компилятора на ускорение. При использовании компилятора ICC (версии 2021.01) исходная программа выполняется медленнее, чем при использовании компилятора GCC (версии 6.3.0–1).

Таблица 3. Сравнение компиляторов GCC и ICC при оптимизации алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнения Лапласа. Итерации – 256, размер сетки – 4000 x 4000, тип данных – float. Выполнение на 16 потоках

Компилятор	GCC (-O3)		ICC (-O3)	
Время выполнения исходной программы (сек.)	16,835		10,915	
Размеры блоков	Время (сек.)	Ускорение	Время (сек.)	Ускорение
d1=32, d2=25, d3=25	0,616	17,72	0,677	24,88
d1=32, d2=40, d3=40	0,590	18,51	0,547	**30,80**
d1=32, d2=50, d3=50	**0,565**	**19,31**	0,600	28,04
d1=32, d2=80, d3=80	0,620	17,60	0,685	24,59
d1=64, d2=25, d3=25	0,580	18,82	0,625	26,94
d1=64, d2=40, d3=40	0,578	18,87	0,535	**31,48**
d1=64, d2=50, d3=50	**0,539**	**20,26**	0,571	29,48
d1=64, d2=80, d3=80	0,606	18,02	0,701	24,03

Рассчитаем долю пиковой производительности для лучшего ускорения (0,5388 секунд) при использовании компилятора GCC. Производительность: $\frac{16.384*10^9}{0.5388} \approx 30.41$ ГФЛОПС. Доля пиковой производительности $= \frac{30.41}{768} * 100\% \approx 3.96\%$.

Предложенный в данной работе алгоритм применялся к разным прикладным задачам. Рассмотрены программы: алгоритм Гаусса – Зейделя для решения задачи Дирихле (таблицы 1, 2, 3), алгоритм Гаусса – Зейделя для решения задачи Дирихле уравнения Пуассона (таблица 4), задача теплопроводности (таблица 5), обобщенный алгоритм Гаусса – Зейделя для решения задачи Дирихле (таблица 6). Таблицы демонстрируют зависимость ускорения от размера тайла (блока).

Таблица 4. Результаты оптимизации алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнения Пуассона. Размерность задачи – 256 x 4000 x 4000, тип данных – float. Время выполнения исходного алгоритма – 14,746 сек.

Число потоков	Последовательно		8 потоков		16 потоков	
Размеры тайлов (блоков)	Время выполнения	Ускорение	Время выполнения	Ускорение	Время выполнения	Ускорение
d1=128, d2=20, d3=20	4,701	3,14	1,576	9,36	0,915	16,11
d1=128, d2=25, d3=25	4,513	3,27	1,486	9,92	**0,864**	**17,08**
d1=128, d2=40, d3=40	4,420	3,34	1,441	10,23	0,874	16,88
d1=128, d2=80, d3=80	4,396	3,36	1,385	10,65	0,887	16,62
d1=128, d2=100, d3=100	4,340	3,40	**1,353**	**10,90**	0,876	16,83
d1=128, d2=125, d3=125	**4,323**	**3,41**	1,383	10,66	0,896	16,46
d1=128, d2=200, d3=200	4,566	3,23	1,540	9,58	1,101	13,40

Таблица 5. Результаты оптимизации алгоритма решения задачи теплопроводности. Размерность задачи – 128 x 4000 x 4000. Время выполнения исходного алгоритма – 5,592 сек. Тип данных – double

Число потоков	Последовательно		8 потоков		16 потоков	
Размеры тайлов (блоков)	Время выполнения	Ускорение	Время выполнения	Ускорение	Время выполнения	Ускорение
d1=64, d2=20, d3=20	7,000	0,799	2,476	2,259	1,847	3,028
d1=64, d2=40, d3=40	6,653	0,841	2,132	2,623	1,426	3,923
d1=64, d2=50, d3=50	6,508	0,859	2,084	2,683	**1,415**	**3,952**
d1=64, d2=80, d3=80	6,676	0,838	2,135	2,619	1,479	3,781

Листинг 1. Алгоритм Гаусса – Зейделя для решения обобщённой задачи Дирихле.

```
for (int k = 0; k < K; ++k )
     for (int i = 1; i < N - 1; ++i )
          for (int j = 1; j < M - 1; ++j )
               u[i][j] = A[i][j]*u[i-1][j] + B[i][j]*u[i][j-1] +
C[i][j]*u[i+1][j] + D[i][j]*u[i][j+1] + y0[i][j];
```

Проведено сравнение времени выполнения программ, полученных путём преобразования алгоритма Гаусса – Зейделя для решения обобщённой задачи Дирихле (листинг 1) с использованием алгоритма, описанного в главе 2 и реализованного в OPC, и системой PLUTO. В таблице 6 демонстрируется преимущество алгоритма, реализованного на универсальном внутреннем представлении (ВП) OPC, по сравнению с системой PLUTO, которая основывается на полиэдральном ВП.

Таблица 6. Сравнительные результаты (в OPC и PLUTO) оптимизации гнезда циклов из листинга 1. Время выполнения исходного гнезда циклов – 7,0842 сек. Тип данных – float. Размерность задачи – 256 x 2000 x 2000

	OPS				PLUTO			
	Последовательно		16 потоков		Последовательно		16 потоков	
Размер тайлов (блоков)	Время (сек.)	Ускорение	Время (сек.)	Ускорение	Время (сек.)	Ускорение	Время (сек.)	Ускорение
d1=8, d2=8, d3=8	3,844	1,84	1,384	5,11	4,176	1,70	0,624	11,34
d1=8, d2=16, d3=16	3,040	2,33	0,920	7,70	4,212	1,68	0,648	10,93
d1=32, d2=8, d3=8	3,257	2,17	0,718	9,87	4,186	1,69	0,659	10,75
d1=32, d2=16, d3=16	2,533	2,80	0,554	12,77	4,228	1,68	0,666	10,63
d1=32, d2=40, d3=40	2,126	3,33	0,457	15,48	5,909	1,20	0,983	7,20
d1=32, d2=50, d3=50	2,101	3,37	0,448	15,80	6,084	1,16	1,033	6,86
d1=32, d2=80, d3=80	2,268	3,12	0,525	13,47	6,520	1,09	1,192	5,94
d1=64, d2=8, d3=8	3,164	2,24	0,665	10,64	4,262	1,66	1,252	5,65
d1=64, d2=16, d3=16	2,463	2,88	0,502	14,11	4,256	1,66	1,256	5,64
d1=64, d2=40, d3=40	2,130	3,32	0,453	15,63	5,889	1,20	1,721	4,11
d1=64, d2=50, d3=50	2,096	3,38	0,434	16,33	6,074	1,17	1,833	3,86
d1=64, d2=80, d3=80	2,275	3,11	0,499	14,18	6,526	1,09	2,043	3,47

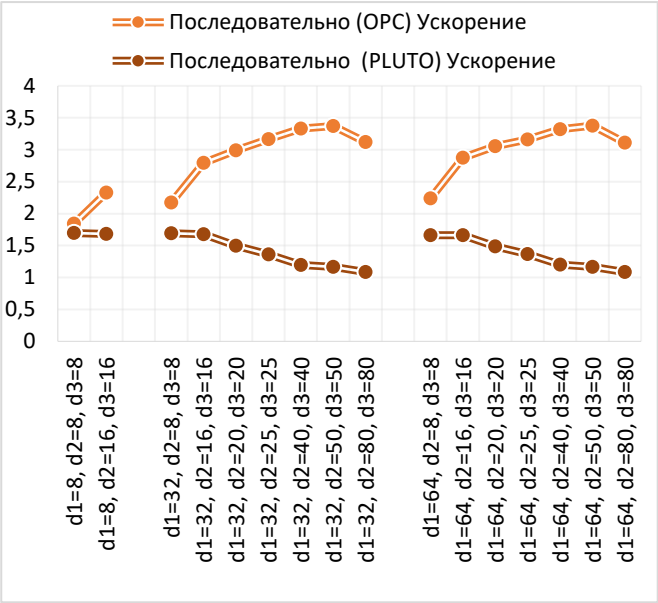

Рисунок 5. Сравнение ускорений при последовательном выполнении, полученных при помощи PLUTO и разработанного алгоритма оптимизации итерационных гнёзд циклов (в OPC) для гнезда циклов из листинга 1

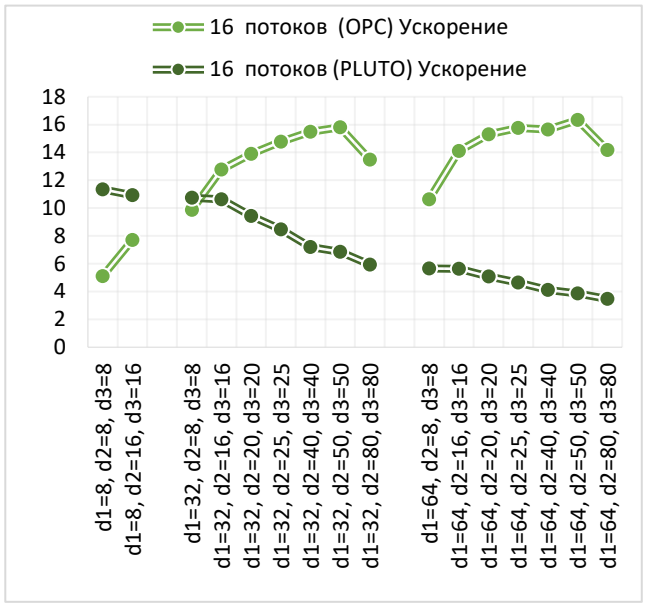

Рисунок 6. Сравнение ускорений при выполнении на 16 потоках, полученных при помощи PLUTO и разработанного алгоритма оптимизации итерационных гнёзд циклов (в OPC) для гнезда циклов из листинга 1

Наибольшее ускорение (в 16.32 раза) гнезда циклов достигается алгоритмом, реализованным в OPC, с размерами блоков 64x50x50. В то время как программа, преобразованная с использованием PLUTO, показывает своё лучшее ускорение (в 11.34 раза) на блоках меньшей размерности (8x8x8). Эта разница обусловлена тем, что разработанный в OPC алгоритм применяет перестановку циклов внутри тайлов.

В параграфе 3.8 рассматривается цепочка преобразований для оптимизации гнёзд циклов итерационного типа. Кроме преобразований, описанных в главе 2, используются дополнительные оптимизирующие преобразования (линеаризация, вынос общих инвариантных выражений). В работе [6] приведён код (более 50 строк) преобразованного необобщённого алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнения Лапласа. Также в работе [6] приведён пример применения линеаризации для упрощения выражений к рассматриваемому алгоритму.

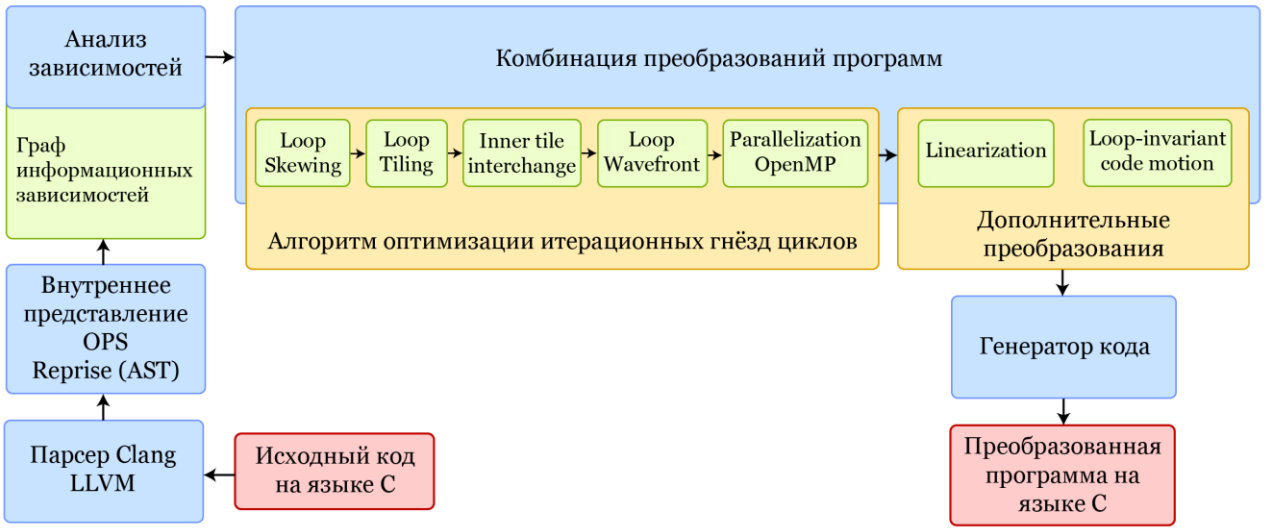

Рисунок 7. Схематическое представление цепочки преобразований для оптимизации гнёзд циклов итерационного типа

Приводится влияние дополнительных преобразований «линеаризация выражений» и известное в оптимизирующей компиляции преобразование «вынос инвариантных выражений» на ускорение программ, преобразованных алгоритмом, описанным в данной работе, на примере гнезда циклов из листинга 2.

Листинг 2.

```
for (int k = 0; k < K; ++k )
    for (int i = 1; i < N - 1; ++i )
        for (int j = 1; j < M - 1; ++j )
            u[i][j] = (u[i-1][j]+u[i+1][j]+[i][j-1]+u[i][j+1]+
            u[i-1][j-1]+u[i+1][j-1]+u[i+1][j-1]+u[i+1][j+1])/8.0;
```

Таблица 7. Сравнение влияния преобразований «линеаризация» и «вынос инвариантных выражений» на программу из листинга 2, предварительно преобразованную алгоритмом из главы 2. Выполнение на 16 потоках

Размеры блоков	Время без дополнительных преобразований	Время с дополнительными преобразованиями	Ускорение с дополнительными преобразованиями
d1=32, d2=8, d3=8	3,07	2,28	1,34
d1=32, d2=16, d3=16	3,03	1,99	1,52
d1=32, d2=20, d3=20	3,24	1,94	1,67
d1=32, d2=25, d3=25	3,39	1,92	1,77
d1=64, d2=8, d3=8	2,94	2,16	1,36
d1=64, d2=16, d3=16	2,93	1,89	1,55
d1=64, d2=20, d3=20	3,18	1,85	1,72
d1=64, d2=25, d3=25	3,33	1,85	1,80

Таким образом, за счёт применения дополнительных преобразований, после описанного алгоритма оптимизации из главы 2, для гнёзда циклов из листинга 2, достигается дополнительное ускорение в до 1.8 раза (таблица 7).

В четвёртой главе рассматривается метод диалогового анализа текстов программ. Суть данного подхода состоит в символьном анализе программы и формировании уточняющих вопросов пользователю. Высокоуровневое внутреннее представление ОРС позволяет анализировать код и формировать вопросы в доступном программисту виде. Актуальность этого метода демонстрируют следующие примеры.

Пример 1. Если хотим распараллелить цикл, вычисляющий сумму чисел с плавающей точкой (или сумму матриц), то будет изменён порядок выполнения операций сложения, что может привести к изменению погрешности вычислений. В диалоговом режиме компилятор может спросить у пользователя, можно ли пренебречь получаемой погрешностью и при положительном ответе распараллелить (рисунок 8).

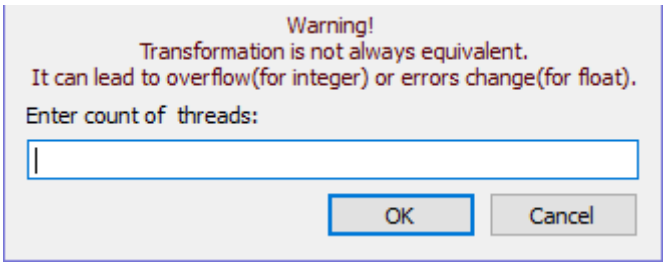

Рисунок 8. Окно диалогового анализатора, предупреждающее пользователя о возможном переполнении или изменении погрешности при распараллеливании

Пример 2. Пусть требуется распараллелить вычисление количества сочетаний C(n, k) = n!/(k!*(n-k)!) = 1/k! * $\prod_{i=0}^{k-1}(n-i)$ (листинг 3).

Листинг 3. Исходная программа для вычисления количества сочетаний.

```
int n, k, c;
for (int i = 1; i <= k; i++) { c *= n - i + 1; c /= i; }
```

К циклу (листинг 3) нельзя автоматически применить расщепление тела цикла (листинг 4) для параллельного вычисления числителя и знаменателя, поскольку может возникнуть переполнение.

Листинг 4. Преобразованная программа.

```
int c, n, k;
float c1(1), c2(1);
for (int i = 1; i <= k; i++) { c1 *= n - i + 1; }     // поток 1
for (int i = 1; i <= k; i++) { c2 /= i; }             // поток 2
c = round(c1*c2);
```

В диалоговом режиме пользователю можно предоставить предупреждение, подобное рисунку 8.

Для уточнения информационных зависимостей при оптимизирующих преобразованиях, таких так прямоугольный тайлинг, гнездование цикла и при распараллеливании, для архитектур SIMD, MIMD можно использовать диалоговый режим следующим образом. Выполняется поиск переменных, значение которых не

известно на этапе компиляции и которые влияют на применимость преобразований. На основе символьного анализа формируются предикаты для уточнения значения таких переменных. Пользователю задаётся вопрос об истинности вычисленного предиката (пример 3). Высокоуровневое внутреннее представление OPC позволяет анализировать код и формировать вопросы в доступном программисту виде.

Пример 3. Рассмотрим пример уточнения информационных зависимостей для векторизации цикла. Как это работает?

Листинг 5.

```
for(int i=0; i<10; i++) {
      x[i+2] = x[i+k] + A[i];
}      |         |
       V2        V1
```

Значение переменной k не известно на этапе компиляции, и оно влияет на информационные зависимости в гнезде циклов.

Для **распараллеливания на архитектуру SIMD**, в частности векторизации, требуется, чтобы отсутствовали дуги графа информационных зависимостей, направленные «снизу вверх» (от тех вхождений, к которым программа обращается позже, к тем вхождениям, к которым программа обращалась раньше), в данном случает от V2 к V1 не должно быть дуги информационной зависимости.

Диалоговый анализатор находит переменную k, значение которой не определено на этапе компиляции и влияет на наличие информационных зависимостей; берёт индексные выражения вхождений переменной x; в символьном виде формирует неравенство: $i + k \geq i + 2$; упрощает при помощи линеаризации выражений: $k \geq 2$.

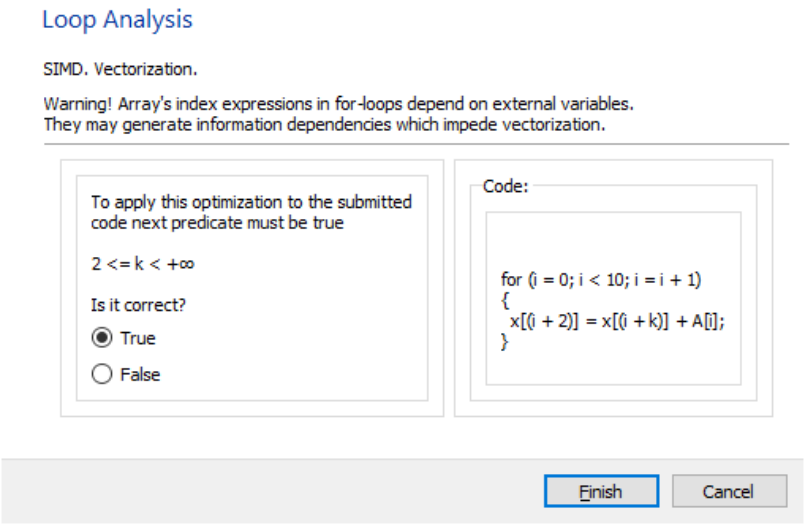

Рисунок 9. Окно диалогового анализатора в OPC. Сформулирован вопрос, уточняющий значение переменной k

Для **распараллеливания на архитектуру MIMD** нужно отсутствие циклически порождённых зависимостей (чтобы вхождения не обращались к одной

ячейке памяти на разных итерациях). Т.е. чтобы индексные выражения вхождений массива x V1, V2 были равны: i + 2 = i + k. После упрощения получаем: k = 2.

В заключении описаны **основные результаты работы:**

1. Доказана эквивалентность разработанной в оптимизирующей распараллеливающей системе (ОРС) цепочки преобразований: «скошенный тайлинг», «метод гиперплоскостей», «распараллеливание с использованием OpenMP», «перестановка циклов внутри тайла», «линеаризация», «вынос общих инвариантных выражений». Тем самым решается выносимое на защиту положение 1.
2. В ОРС реализованы преобразования «скашивание циклов», «метод гиперплоскостей», «скошенный тайлинг», тем самым решается положение 3, выносимое на защиту.
3. Теоретически и экспериментально обоснован новый метод обхода точек тайла, который повышает временную локальность данных и даёт ускорение. Тем самым частично решается положение 4, выносимое на защиту.
4. Предложен новый метод расчета *оптимальных* размеров тайла для получения лучшего ускорения. Тем самым завершается решение выносимого на защиту положения 4.
5. Проведены численные эксперименты, демонстрирующие ускорение программ, преобразованных с использованием разработанного алгоритма. Получены ускорения относительно исходных программ для: алгоритма решения задачи теплопроводности (в 3,95 раза); алгоритма Гаусса – Зейделя для решения задачи Дирихле уравнений Лапласа (в 20,26 раза) и Пуассона (в 17,08 раз). Для алгоритма Гаусса – Зейделя для решения обобщённой задачи Дирихле достигнуто ускорение в 16,33 раза, что на 40% эффективнее чем при использовании известной системы PLUTO. Тем самым подтверждается правильность полученных теоретических моделей и практическая значимость выполненной работы.
6. Предложен основанный на символьном анализе подход к диалоговому ускорению программ, который позволяет расширить класс задач, оптимизируемых полуавтоматически с помощью компилятора. Приводятся примеры программ, которые нельзя оптимизировать автоматически, но можно с помощью в предложенного метода. Тем самым решается положение 2, выносимое на защиту.

Приводится список прикладных задач, к которым могут быть применены представленные в диссертации методы.

Предложенные в данной работе методы могут быть реализованы при создании инструментов (компиляторов, конверторов) автоматического (и полуавтоматического) распараллеливания и оптимизации программ. Что позволит сократить время разработки высокопроизводительных программ, тем самым повышая производительность труда программистов.

Публикации по теме диссертации

Публикации из перечня ВАК:

1. Метелица Е.А. Обоснование методов ускорения гнёзд циклов итерационного типа // Программные системы: теория и приложения. 2024. Т. 15. № 1. С. 63–94. (К2)
2. Метелица Е.А., Штейнберг Б.Я. Об ускоряющих преобразованиях программ для решения обобщенной задачи Дирихле // Вычислительные методы и программирование. 2024.25, № 3. С. 292-301. (К1)

Свидетельство о государственной регистрации программ:

3. «Программа, реализующая алгоритм Гаусса – Зейделя для задачи Дирихле» №2021681898 от 10 января 2022 г. Василенко А.А., Метелица Е.А., Штейнберг Б.Я. (РИНЦ).

Публикации, индексируемые в международных базах данных Scopus, WoS:

4. Baglij A., Mikhailuts Y., Metelitsa E., Ibragimov R., Steinberg B., Steinberg O. On the Development of the Compiler from C to the Processor with FPGA Accelerator // Frontiers in Software Engineering Education. FISEE. Lecture Notes in Computer Science. 2019. Vol. 12271. Springer, Cham. [Электронный ресурс]: URL: https://doi.org/10.1007/978-3-030-57663-9_25 (дата обращения: 01.04.2024). (Scopus, **Q2**, РИНЦ).
5. Vasilenko A., Veselovskiy V., Metelitsa E., Zhivykh N., Steinberg B., Steinberg O., Precompiler for the ACELAN-COMPOS Package Solvers, Parallel Computing Technologies // PaCT. Lecture Notes in Computer Science. 2021. Vol. 12942. Springer, Cham. [Электронный ресурс]: URL: https://doi.org/10.1007/978-3-030-86359-3_8 (дата обращения: 01.04.2024). (Scopus**, Q2**, РИНЦ).
6. Bagliy A.P., Metelitsa E.A., Steinberg B.Y. Automatic Parallelization of Iterative Loops Nests on Distributed Memory Computing Systems // Parallel Computing Technologies. PaCT. Lecture Notes in Computer Science. 2023. Vol. 14098. Springer, Cham. [Электронный ресурс]: URL: https://doi.org/10.1007/978-3-031-41673-6_2 (дата обращения: 01.04.2024). (Scopus).
7. Gervich L., Metelitsa E., Steinberg B. Combination of parallelization and skewed tiling // Procedia Computer Science. 2023. Vol. 229. P. 228–235.
8. Steinberg B., Baglij A., Petrenko V., Burkhovetskiy V., Steinberg O., Metelica E. An Analyzer for Program Parallelization and Optimization // Proceedings of the 3rd International Conference on Applications in Information Technology (ICAIT'2018). New York: Association for Computing Machinery, 2018. P. 90–95. (Scopus, WoS, РИНЦ).
9. Gervich L.R., Guda S.A., Dubrov D.V., Ibragimov R.A., Metelitsa E.A., Mikhailuts Y.M., Paterikin A.E., Petrenko V.V., Skapenko I.R., Steinberg B.Ya., Steinberg O.B., Yakovlev V.A., Yurushkin M.V. How OPS (optimizing parallelizing system) may be useful for clang // Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR 17). New York: Association for Computing Machinery, 2017. P. 1–8. (Scopus, WoS, РИНЦ).

Другие публикации:

10. Метелица Е.А. Использование тайлинга для оптимизации итерационных алгоритмов // Современные информационные технологии: тенденции и перспективы развития: материалы XXVII научной конференции, г. Ростов-на-Дону – Таганрог, 24–26 сентября 2020 г. Ростов н/Д. – Таганрог: Изд-во ЮФУ, 2020. с. 185–188. (РИНЦ).
11. Метелица Е.А., Морылев Р.И., Петренко В.В., Штейнберг Б.Я. Основанная на OPC система обучения преобразованиям программ «Тренажер параллельного программиста», PLC-2017 / Всероссийская научная конференция памяти А.Л. Фуксмана. Ростов н/Д., 2017. с. 198. (РИНЦ).

Метелица Елена Анатольевна

Автоматизация распараллеливания программ со сложными информационными зависимостями

Автореферат диссертации на соискание ученой степени кандидата физ.-мат. наук

Подписано в печать 20.11.2024 г.
Бумага офсетная. Формат 60×84/16. Тираж 100 экз.
Усл. печ. лист 1,0. Уч.-изд. 1,0. Заказ № 9721.

Отпечатано в отделе полиграфической, корпоративной и сувенирной продукции
Издательско-полиграфического комплекса КИБИ МЕДИА ЦЕНТРА ЮФУ.
344090, г. Ростов-на-Дону, пр. Стачки, 200/1, тел. (863) 243-41-66.