На правах рукописи

ВАСИЛЬЕВ ВЛАДИМИР СЕРГЕЕВИЧ

ОПТИМИЗАЦИЯ И ТРАНСФОРМАЦИЯ ФУНКЦИОНАЛЬНО-ПОТОКОВЫХ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

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

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

кандидата технических наук

Красноярск 2022

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

Научный руководитель:	доктор технических наук, профессор, **Легалов Александр Иванович**
Официальные оппоненты:	**Штейнберг Борис Яковлевич**, д-р техн. наук, старший научный сотрудник, Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет», кафедра алгебры и дискретной математики, заведующий кафедрой
	Васильчиков Владимир Васильевич, канд. техн. наук, Федеральное государственное бюджетное образовательное учреждение высшего образования «Ярославский государственный университет им. П.Г. Демидова», кафедра вычислительных и программных систем, заведующий кафедрой
Ведущая организация:	Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук

Защита состоится «03» марта 2023 г. в 14:00 часов на заседании диссертационного совета 24.2.404.05, созданного на базе Сибирского федерального университета, по адресу: 660074, г. Красноярск, ул. Киренского, д. 26, ауд. УЛК 112.

С диссертацией можно ознакомиться в библиотеке и на сайте Сибирского федерального университета по адресу http://www.sfu-kras.ru

Автореферат разослан «____» _____ 2023 г.

Учёный секретарь
диссертационного совета — Кайзер Юрий Филиппович

Общая характеристика работы

Актуальность работы.

Растущие требования к быстродействию вычислительных машин обуславливают активное внедрение параллельных вычислительных систем (ПВС). Существует множество архитектур таких систем, каждая из которых имеет свои специфические особенности, которые учитываются широко используемыми методами и инструментальными средствами параллельного программирования.

Решения, созданные для одной архитектуры ПВС, оказываются неэффективными для другой, поэтому развивается ряд проектов, направленных на автоматическое распараллеливание программ. Такие инструменты сталкиваются с проблемами выявления параллелизма и разделяемых в программе ресурсов, а также обеспечения эффективного их распределения между вычислителями с учётом особенностей архитектуры.

Альтернативным подходом к программированию ПВС является архитектурно-независимое параллельное программирование с использованием концепции неограниченного параллелизма. Он позволяет отойти от необходимости распараллеливания и привязки к конкретным архитектурам. В рамках этого подхода развивается направление, ориентированное на функционально-потоковое параллельное (ФПП) программирование.

На сегодняшний день для ФПП парадигмы разработан ряд инструментальных средств, позволяющих выполнять интерпретацию программ, их отладку и верификацию, но задачи, связанные с различными трансформациями ФПП программ, обеспечивающими перенос на реальные архитектуры до конца не решены. В частности отсутствуют инструменты анализа, оптимизации и трансформации этих программ в другие языки программирования. Существующие системы анализа и оптимизации кода не могут напрямую использоваться в связи с различием подходов к управлению вычислениями, а алгоритмы и используемые в них методы требуют переосмысления с учётом особенностей функционально-потоковой парадигмы параллельного программирования.

Повышение эффективности как трансформации, так и выполнения ФПП программ может быть обеспечено за счёт использования методов оптимизации и преобразования, многие из которых аналогичны методам, используемых в других парадигмах. Особенности ФПП программирования накладывает на реализацию этих методов свою специфику. Кроме того, возможна оптимизация кода за счёт применения приемов, которые непосредственно вытекают из особенностей данной парадигмы. Вопросы преобразования операторов функционально-потоковой модели параллельных вычислений в эквивалентные операторы императивных языков программирования, типичных для современных последовательных и параллельных архитектур не проработаны.

Степень разработанности темы. Инструментальные средства оптимизации кода разработаны для всех широко используемых языков программирования. Исследованиям методов распараллеливания программ для различных архитектур ПВС и их оптимизации посвящены работы В.Н. Касьянова, И.И. Левина, А.И. Дордопуло, Б.Я. Штейнберга, В.В. Воеводина. Из зарубежных специалистов в первую очередь стоит отметить работы Р. Гупты, М. Девера, Х. Сайто, Д. Ферранте, Г.В. Хамильтона, В. Чена и др. Однако в этих работах не затрагиваются методы повышения эффективности для архитектурно-независимых параллельных программ.

При оптимизации функционально-потоковых параллельных программ возникает ряд проблем:

- большинство известных методов оптимизации связано с синтаксическими конструкциями, характерными для императивных языков, отсутствующих в ФПП языках программирования;
- отсутствуют инструментальные средства, обеспечивающие эффективную трансформацию таких программ с учетом особенностей целевой платформы.

Известные попытки решения этих проблем в основном направлены на:

- оптимизацию фрагментов кода, исполняемых последовательно, при этом применяются как архитектурно-независимые, так и архитектурно-зависимые методы оптимизации, которые широко используются на практике, но не позволяют максимально эффективно использовать параллелизм вычислительной системы;

- распараллеливание последовательных программ с учетом особенностей архитектуры ПВС, например «Открытая распараллеливающая система» и «AutoPar» (некоторые из таких систем не выполняют автоматическое распараллеливание, но выдают рекомендации программисту);

- использование языков, придерживающихся концепции ресурсно-независимого параллельного программирования, например Sisal, Пифагор, Set@l. Такие языки позволяют описывать алгоритмы без учета ресурсных ограничений и отдельно задавать такие ограничения для различных архитектур ПВС. К этому подходу можно отнести языки функционально-потоковой парадигмы, например Пифагор и Smile. Однако исследований по оптимизации программ, написанных на функционально-потоковых языках программирования, обеспечивающих переносимость параллельных программ, не проводилось.

Следовательно, разработка методов и инструментальных средств, обеспечивающих оптимизацию функционально-потоковых параллельных программ, является актуальной задачей.

Требуется не только проанализировать широко известные методы оптимизации кода на предмет применимости к ФПП парадигме, но и создать новые методы, связанные с особенностями этой парадигмы.

Цель диссертационной работы заключается в повышении эффективности выполнения функционально-потоковых параллельных программ за счет разработки новых методов их оптимизации и трансформации.

Для достижения поставленной данной цели в работе решаются следующие **задачи**:

1. Анализ существующих методов оптимизации программ и возможности их применения в ФПП парадигме.

2. Разработка методов и алгоритмов оптимизации программ с учётом специфики ФПП парадигмы.

3. Исследование и разработка методов и алгоритмов преобразования ФПП программ в императивные.

4. Разработка инструментальных средств, обеспечивающих комплексную поддержку анализа, оптимизации и преобразования ФПП программ в программы, выполняемые на реальных вычислительных системах.

Методы исследований

Поставленные задачи решались посредством элементов теории графов, теории языков программирования, математической логики, теории конечных автоматов, технологии трансляции и теории алгоритмов.

При разработке основных положений диссертации применялись методы системного анализа вычислительных систем, функционально-потокового и объектно-ориентированного проектирования и программирования.

Прикладное программное обеспечение реализовано на языке C++.

Научная новизна:

1. Предложены методы и алгоритмы оптимизирующих трансформаций информационных графов функционально-потоковых параллельных программ, обеспечивающие сокращение времени выполнения и объема используемой памяти.

2. Разработаны методы преобразования управляющих графов функционально-потоковых языков программирования, позволяющие оптимизировать стратегию управления

вычислениями в таких языках за счет устранения избыточных управляющих зависимостей, что позволяет ускорить выполнение функционально-потоковых параллельных программ.

3. Предложен и реализован метод трансформации функционально-потоковых параллельных программ в эквивалентные императивные программы, позволяющий более эффективно исполнять их на существующих вычислительных системах за счет представления программы на уровне системы команд.

Теоретическая и практическая значимость работы.

Предложенные в работе методы и алгоритмы вносят вклад в развитие архитектурно-независимого параллельного программирования. На их основе разработаны инструментальные средства, интегрированные в существующую систему разработки программ на языке функционально-потокового параллельного программирования Пифагор.

Результаты работы использовались:

- при выполнении научно-исследовательских работ в рамках федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" по теме "Инструментальная поддержка архитектурно-независимой разработки параллельных программ на основе функционально-потоковой парадигмы параллельного программирования", № 14.A18.21.0396 в 2013-2014 гг.;
- при выполнении Грантов РФФИ: № 13-01-00360 "Методы и средства эволюционной разработки программного обеспечения с применением процедурно-параметрической парадигмы программирования", №17-07-00288 "Архитектурно-независимая разработка параллельных программ на основе функционально-потоковой парадигмы" в 2017-2020 гг.;

Выполнено экспериментальное внедрение результатов диссертационного исследования в ООО «Геология Восточной Сибири» при выполнении опытно-конструкторских работ.

Основные положения, выносимые на защиту:

1. Разработанные методы оптимизации информационных графов функционально-потоковых параллельных программ обеспечивают ускорение вычислений и сокращают необходимый для этого объем памяти.
2. Применение методов оптимизации управляющих графов ускоряет вычисления за счет устранения избыточных управляющих зависимостей.
3. Предложенный метод трансформации функционально-потоковых параллельных программ в императивную форму позволяет формировать эквивалентное представление программы на уровне системы команд и обеспечивает более эффективное ее выполнение на существующих вычислительных системах.

Достоверность полученных результатов подтверждается корректным использованием математического аппарата, работоспособностью разработанного программного обеспечения (ПО), успешным применением разработанного ПО в научной деятельности по развитию представленных методов оптимизации и трансформации программ.

Апробация работы.

Основные положения и результаты исследований докладывались и обсуждались на открытых межкафедральных семинарах ИКИТ СФУ, восьми отраслевых, всероссийских и международных конференциях и семинарах, в том числе: "МНСК-2015: Информационные технологии" (Новосибирск, 2015); "Научный сервис в сети Интернет" (Новороссийск, 2017); "Суперкомпьютерные дни в России" (Москва, 2018).

По теме диссертации опубликовано 24 работы, из которых: 7 статей в изданиях, рекомендуемых ВАК, 2 статьи в журналах базы данных Scopus и 7 свидетельств о государственной регистрации программ для ЭВМ.

Соответствие специальности. По своему научному содержанию диссертационная работа соответствует паспорту специальности 2.3.5 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»: пункту №1

(Модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования); пункту № 2 (Языки программирования и системы программирования, семантика программ); пункту № 8 (Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования).

Личный вклад автора. Основные результаты являются новыми и получены лично автором. Вклад автора в научные работы, опубликованные в соавторстве с научным руководителем, заключается в следующем: анализе особенностей ФПП парадигмы в контексте задачи оптимизации; разработке методов и алгоритмов оптимизации функционально-потоковых параллельных программ; разработке методов и алгоритмов преобразования ФПП программ в императивную форму; создании инструментальных средств трансформации ФПП программ и их интеграции с существующими инструментальными средствами программирования. Все выносимые на защиту результаты принадлежат лично автору.

Совместно с научным руководителем автор осуществлял постановку целей и задач, выбор методов для реализации, анализ полученных результатов.

В совместных публикациях автора с Матковским И.В., Ушаковой (Кропачевой) М.С., Савченко Г.В., Постниковым А.И., Непомнящим О.В. изложены теоретические идеи дополнения функционально-потоковой модели вычислений и ее промежуточных представлений, используемых элементами инструментальной среды программирования. Практическим результатом выполненных совместных исследований являются свидетельства о регистрации программ для ЭВМ, в них вклад автора пропорционален вкладу в соответствующие теоретические исследования.

Представление изложенных в диссертации и выносимых на защиту результатов, полученных в совместных исследованиях, согласованно с соавторами.

Структура и объем работы. Диссертация состоит из введения, четырёх разделов, заключения, списка сокращений и трех приложений. Работа содержит 143 страницы основного текста, 69 рисунков и 6 таблиц. Список использованных источников содержит 120 наименований.

Основное содержание работы

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

В первой главе изложены результаты анализа методов оптимизации программ и промежуточных представлений, используемых оптимизирующими компиляторами, описаны основные особенности ФПП языков, существенные при оптимизации программ. Показано, что к ФПП программам можно как применять некоторые широко известные методы оптимизации, так и разрабатывать новые, специфичные для функционально-потоковой парадигмы программирования.

В контексте задачи оптимизации кода рассмотрены особенности функционально-потоковой парадигмы программирования. На основе проведенного анализа определены основные задачи диссертационного исследования и требования к разрабатываемым методам оптимизации.

Рассмотрены методы оптимизации программ применяемые в других языках программирования. Показано, что ряд методов применим к ФПП программам, но неудачное применение оптимизирующего преобразования может приводить к деградации характеристик программы. Функционально-потоковые языки программирования обладают особенностями, позволяющими выполнять новые, специфичные только для них, оптимизирующие преобразования:

- ФПП программа представляет собой набор реверсивных информационных (РИГ) и управляющих (УГ) графов, за счет этого возможны раздельные преобразования информационных и управляющих зависимостей;
- функционально-потоковые языки поддерживают специфическую модель вычислений, содержащую правила эквивалентных преобразований программ, поэтому часть этих преобразований можно выполнить до этапа исполнения программы и при этом добиться улучшения ее характеристик;
- по дугам управляющего графа передаются сигналы готовности, которые обрабатываются управляющими автоматами, в ряде случаев управляющие автоматы отдельных операторов имеют избыточную структуру и могут быть заменены более простыми версиями.

Полученные результаты позволили перейти к разработке методов и алгоритмов оптимизации функционально-потоковых параллельных программ.

Во второй главе описаны методы и алгоритмы оптимизирующих преобразований ФПП программ.

Несмотря на отсутствие явно выделенной конструкции цикла, ФПП языки программирования позволяют записывать повторяющиеся вычисления с помощью рекурсии или операций над параллельными списками. Для обоих вариантов задания повторяющихся вычислений предложены алгоритмы преобразования ФПП программ на основе широко используемого метода оптимизации, основанного на выносе инварианта за тело цикла.

Показано, что метод оптимизации неиспользуемого кода применим к ФПП программам без каких-либо специфических доработок.

Алгоритмы для таких широко известных методов оптимизации как «inline-подстановка» и «удаление общих подвыражений» учитывают иерархическую вложенность специфических для ФПП парадигмы задержанных списков. Для более эффективного поиска общих подвыражений предложено использовать ярусно-параллельную форму (ЯПФ) программы, так как узлы, выполняющие одинаковые вычисления, гарантированно находятся на одном ярусе ЯПФ. Для каждого узла РИГ на основе его информационных зависимостей вычисляется код (хэш), узлы каждого яруса ЯПФ группируются в зависимости от кода и номера задержанного списка как показано на рис. 1.

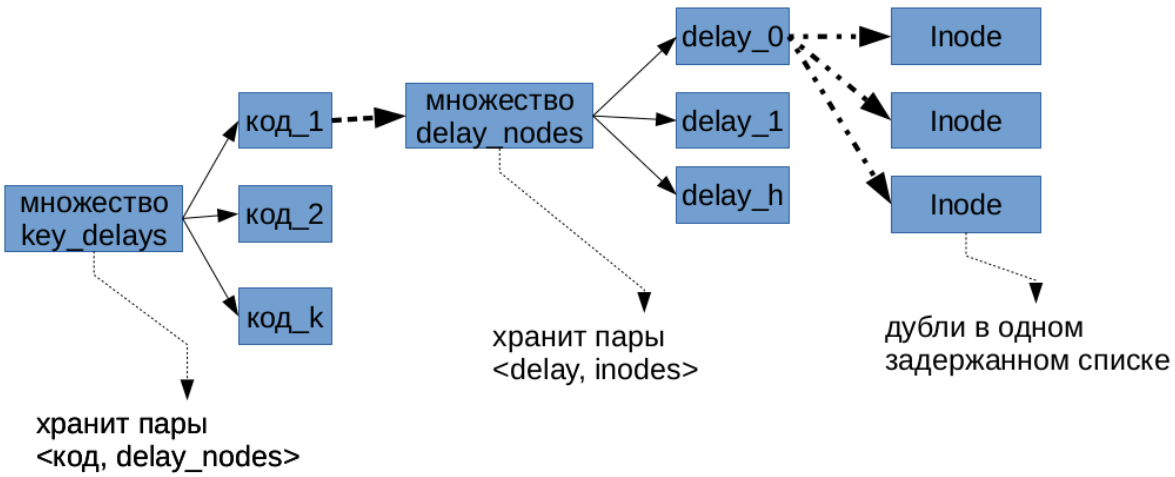

Рисунок 1 – Структура данных для хранения узлов одного яруса ЯПФ

Описан эффективный алгоритм построения ЯПФ для ФПП программ на основе РИГ. Такая форма представления оказалась эффективной и для реализации алгоритма преобразования ФПП программ в императивную форму, рассмотренного в четвертой главе работы.

Рассмотрены эквивалентные преобразования, описанные в модели функционально-потоковых параллельных вычислений. Предложен специфичный для ФПП парадигмы метод оптимизации, выполняемый на уровне РИГ, заключающийся в переносе части таких преобразований с этапа выполнения на этап оптимизации кода, а именно: удалению параллельных списков из одного элемента; интерпретации параллельных списков; раскрытию параллельных списков, вложенных в список данных.

Под раскрытием параллельных списков внутри списка данных имеется ввиду устранение операций группировки в параллельный список при условии, что затем этот список становится элементом списка данных. Например:

$$(A, [B, C, D], E) \rightarrow (A, B, C, D, E).$$

Оптимизация возможна если в информационном графе есть узел DataList, задающий операцию группировки в список данных и узел ParList, задающий операцию формирования параллельного списка, а также существует дуга, соответствующая потоку данных от ParList к DataList. Такое преобразование можно выполнить по следующему **алгоритму**:

1. Получить количество дуг (ParLength), входящих в ParList;
2. Для всех дуг, входящих в DataList, если их слот имеет большее значение, чем слот параллельного списка, прибавить к номеру слота значение ParLength;
3. Для всех дуг, входящих в параллельный список, изменить узел-приемник на список данных, при этом номер слота увеличить на значение номера слота дуги ParList-> DataList;
4. Удалить дугу между параллельным списком и списком данных и узел ParList.

Предложены алгоритмы оптимизации на основе каждого такого преобразования, на примерах показана их широкая применимость. В результате таких преобразований сокращается объем работы, необходимый для выполнения программы, а значит и время ее выполнения.

Показано, что в управляющем графе программы, построенном по РИГ часть управляющих дуг является избыточной. Удаление таких дуг не изменяет порядок вычислений, но сокращает количество сигналов готовности, передаваемых при выполнении программы, а значит является оптимизирующим преобразованием. Особый тип управляющих зависимостей возникает между узлами, вложенными в задержанный список и узлами, зависящими от операции раскрытия этого списка. Код, вложенный в задержанный список, не получит управление до момента раскрытия списка. Следовательно, если узел, выполняющий раскрытие, имеет некоторые зависимости, то эти зависимости имеют и все узлы, вложенные в список.

На рис. 2 группировка в задержанный список показана пунктирной рамкой, узел G, зависящий по управлению от узла C, выполняет интерпретацию (раскрытие) этого списка. В задержанный список вложены узлы X и Y, при этом Y имеет прямую управляющую зависимость от узла C. Узел Z, выделенный толстой линией, является константой (следовательно не имеет зависимостей), представляющей задержанный список. При интерпретации задержанной константы Z выполнится раскрытие списка, которое приведет к вычислениям узлов X и Y. Узел G не начнет выполняться пока не завершит выполнение C, следовательно до этого момента не получат управление узлы X и Y. Таким образом, узлы X и Y имеют косвенную управляющую зависимость узла C, а значит дуга C-Y является избыточной.

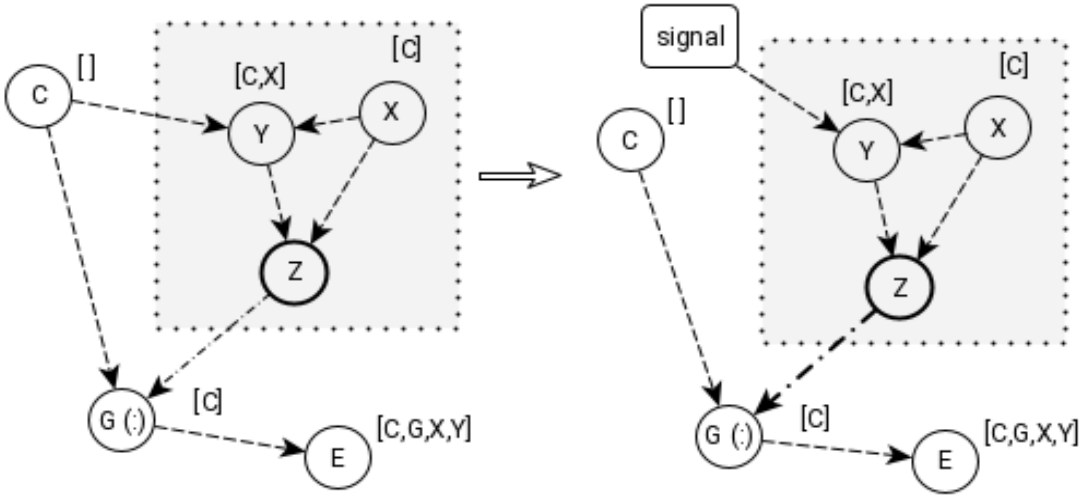

Рисунок 2 – Управляющие зависимости задержанных списков

Предложен метод и разработаны алгоритмы оптимизации, заключающиеся в удалении из управляющего графа избыточных дуг для обоих типов управляющих зависимостей.

Для оптимизации прямых управляющих зависимостей достаточно обойти управляющий граф "сверху вниз" и для каждого узла $Node_i$ рассчитать множество $Deps_i$, задающее зависимости, по следующему **алгоритму**:

1. $Deps_i = \varnothing$;

2. Для каждой, входящей в узел $Node_i$, дуги j выполнить:

2.1 Если удаление дуги j не изменит зависимости узла *i*, то есть:

$$Deps_i \subset Deps_j,$$

то дуга *j* — избыточная, выполняется ее удаление;

2.2 Иначе множество зависимостей узла j добавляется к множеству $Deps_i$, то есть: $Deps_i = Deps_i \cup Deps_j$.

Оптимизация задержанных управляющих зависимостей выполняется аналогично, однако список зависимостей узлов, вложенных в задержанный список, дополняется зависимостями, которые имеет узел раскрытия списка.

Управляющие автоматы в ФПП языках программирования контролируют процесс выполнения операторов. Они обрабатывают поступление сигналов готовности, передаваемых по дугам управляющего графа и инициируют процесс вычислений при наступлении предусмотренных автоматом условий. Показано, что наиболее общий и универсальный вид управляющих автоматов, допускающий поступление на любой вход оператора параллельных списков, зачастую оказывается избыточным.

Разработан алгоритм, позволяющий в ряде случаев определить на этапе трансляции, что узел РИГ не является параллельным списком. Предложено ввести в ФПП модель вычислений упрощенные версии управляющих автоматов и подставлять их вместо универсального автомата, если на вход оператора не может поступить параллельный список.

Большинство функциональных языков программирования поддерживают оптимизацию кода за счет замены хвостовой рекурсии циклом. Это связано с тем, что такие языки не позволяют в явном виде описывать циклы, а рекурсивные вызовы приводят к необоснованным накладным расходам. Показано, что для применения к программам на ФПП языках такого вида оптимизирующего преобразования необходимо добавить в модель вычислений этих языков специального оператора, но оставить его недоступным для непосредственного использования программистом.

При замене рекурсивного вызова *f*, выполняющего преобразование аргумента *X*, на функцию *repeat* осуществляется следующая модификация оператора интерпретации.

X:f → X:repeat.

Семантика операции repeat прописана в разделе 2.2.7 диссертации. Для выполнения преобразования может использоваться следующий **алгоритм** :

1. Найти оператор рекурсивного вызова *Rec*;
2. Сформировать из узлов пути между Rec и возвратной вершиной множество RecPath;
3. Если каждый узел RecPath является либо оператором группировки в список, либо интерпретацией константы — заменить оператор рекурсивного вызова на *repeat*.

В третьей главе приведены структуры данных, обеспечивающие высокое быстродействие разработанных алгоритмов, а также результаты работы созданной системы оптимизации ФПП программ.

Проведен анализ наиболее требовательных к вычислительным ресурсам операций, выполняемых в ходе оптимизирующих преобразований программ. Показано, что при оптимизации целесообразно задавать программу в виде графа определений и использований. Рассмотрены варианты хранения данных графа, упорядоченными по различным параметрам. Наиболее подходящим для реализации из рассмотренных вариантов является хранение информационного графа в виде матрицы смежности, связи в которой задаются ссылками, а список вершин не упорядочен. В табл. 1 приведены оценки вычислительной сложности для наиболее затратных операций, выполняемых при оптимизации кода для графа из M дуг и N узлов, каждому из которых инцидентно по K дуг.

Таблица 1 – Асимптотические оценки вычислительной сложности типовых операций при задании графа списками смежности

Операция			Оценка трудоемкости	
			узел задается парой (type, id)	узел задается ссылкой
узла	поиск	по (type, id)	$O(log_2(N))$	$O(log_2(N))$
		по value/delay/...	*O(N)*	*O(N)*
	удаление		$O(K \cdot log_2(N))$	$O(K \cdot K)$
	добавление		*O(1)*	*O(1)*
	изменение		*O(1)*	*O(1)*
дуги	поиск (проверка существования)		$O(log_2(N) + K) \approx O(log_2(N))$	$O(log_2(N) + K) \approx O(log_2(N))$
	удаление		$O(log_2(N) + K) \approx O(log_2(N))$	$O(log_2(N) + K) \approx O(log_2(N))$
	добавление		$O(log_2(N))$	$O(log_2(N))$
	изменение		$O(log_2(N))$	$O(log_2(N))$
поиск (type, id) узла приемника/источника данных			*O(K)*	*O(K)*
полный обход графа			$O(N \cdot N)$	*O(N+M)*
обход графа на L уровней в глубину			$O((K*log_2(N))^L)$	$O(K^L)$

Для задания управляющего графа наиболее рационально использовать списки инцидентности. Схематично внутреннее представление управляющего графа приведено на рис. 3. Узлы графа хранятся внутри связного списка и между собой связаны посредством дуг, задающих use-def и def-use зависимости. Отдельный вид зависимостей возникает от сигналов и задержанных списков (динамических связей).

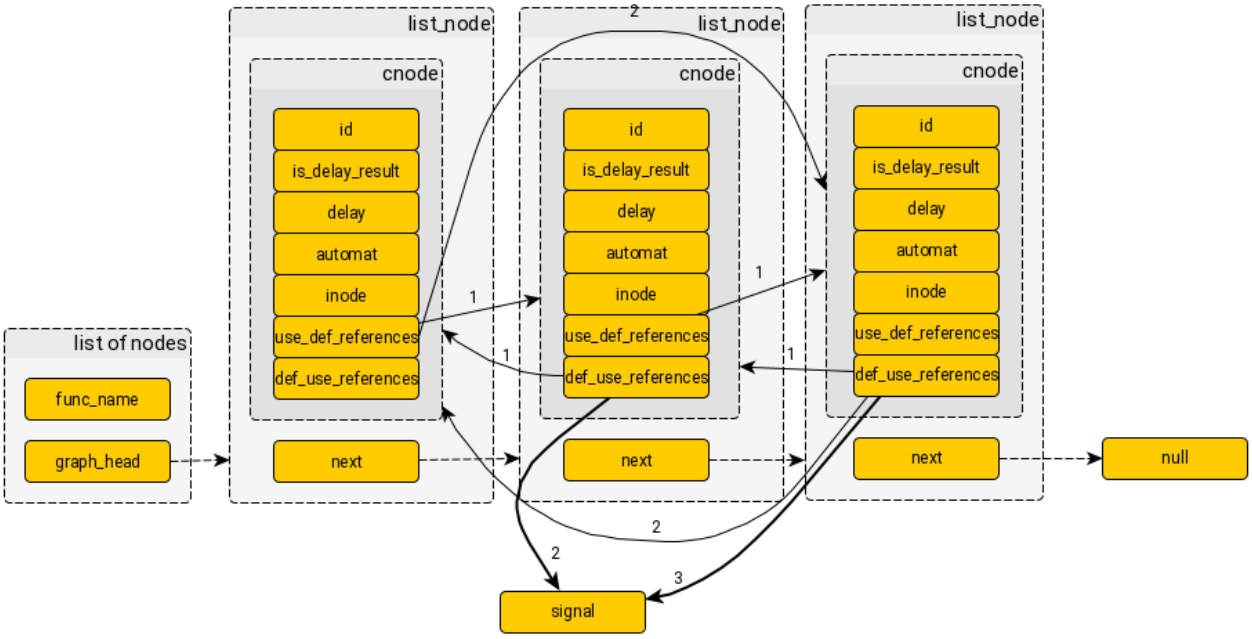

Рисунок 3 – Схема внутреннего представления управляющего графа программы

С использованием этих структур данных и ряда алгоритмов, описанных во второй главе работы, реализовано инструментальное средство, выполняющее оптимизацию кода программ функционально-потокового языка параллельного программирования Пифагор. В настоящий момент в среде оптимизации реализованы: inline-подстановка функции вместо вызова, удаление общих подвыражений, два алгоритма выноса инварианта, три алгоритма оптимизации на основе эквивалентных преобразований, удаление неиспользуемого кода, замена хвостовой рекурсии циклом, два метода удаления лишних управляющих зависимостей.

Система оптимизации применена к программе, выполняющей вращение трехмерной фигуры вокруг оси. Результаты оптимизации оценивались по изменению количества элементарных операций и вызовов функций, выполняемых при интерпретации программы. На рис. 4 приведена зависимость доли удаленных операций от объема обрабатываемых данных: с ростом объема вычислений, растет и эффективность оптимизации. На рассмотренном примере доля удаленных операций превышает 60%. В табл. 2 показано количество выполняемых операций после применения к этой программе методов, давших наибольший эффект. Очень результативным в этом случае оказался вынос инварианта из цикла, это связано с тем, что в исходном коде программы при вращении каждой точки формируется абсолютно одинаковая матрица вращения и перемещение этого фрагмента кода приводит к тому, что матрица формируется всего один раз.

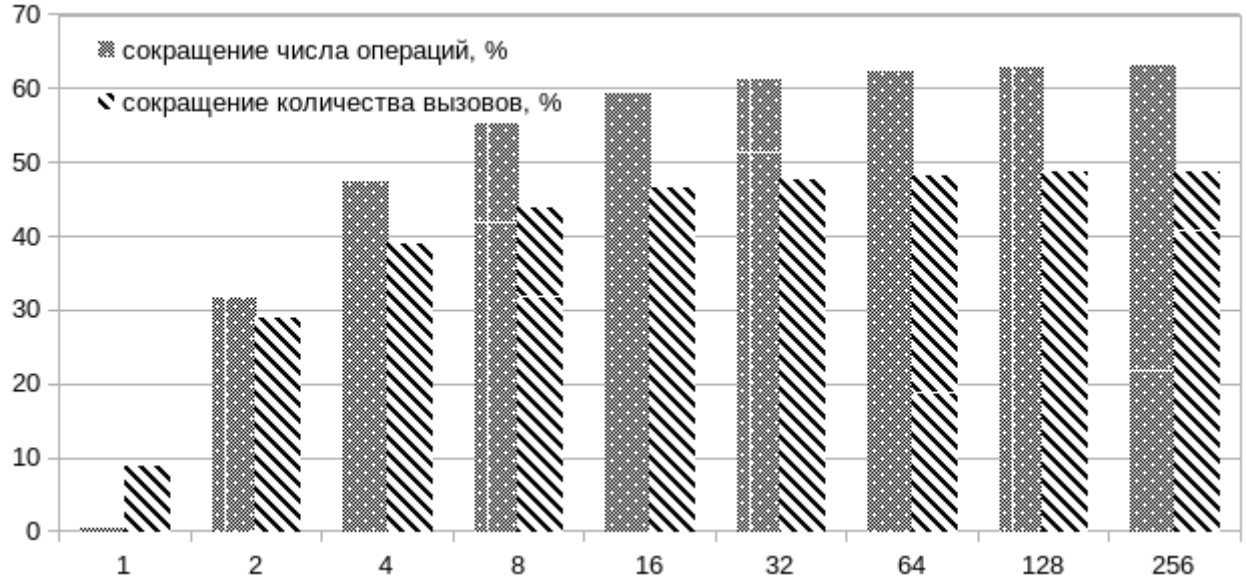

Рисунок 4 – Оценка результатов оптимизации в зависимости от объема обрабатываемых данных

Таблица 2 – Результаты применения отдельных методов оптимизации при вращении фигуры из 128 точек

Операции	Исходный вариант	inline-подстановка	Оптимизация инварианта
Все операции	86918	86406	32437
Вызовы функций	5760	5248	2966

Между методами оптимизации имеются зависимости, с учетом которых выстроен наиболее эффективный порядок применения оптимизирующих преобразований.

В четвертой главе предложен метод и алгоритмы трансформации функционально-потоковых параллельных программ в императивную форму. Наибольшая эффективность исполнения программ достигается при их компиляции, то есть генерации кода на уровне системы команд процессора. Для этого необходима информация о типах переменных, поэтому преобразование выполняется в императивные языки со статической типизацией.

Описаны проблемы выполнения такой трансформации, показано, что преобразование произвольной ФПП программы в императивную форму крайне сложно.

В связи с тем, что язык Пифагор использует динамическую типизацию, была предложена семантика типов данных для реверсивных информационных графов. Определено подмножество наиболее часто используемых синтаксических конструкций ФПП языков, для которых возможна трансформация в императивную форму. Описаны правила трансформации этих конструкций:

- преобразование задержанных списков, используемых для организации ветвления, в соответствующие императивные конструкции;
- формирование на императивных языках списка аргументов функций и возвращаемого значения, использование этих аргументов;
- трансформация параллельных списков, используемых для задания повторяющихся вычислений в конструкции циклов;
- преобразования специфических операторов функционально-потоковой модели вычислений в различные императивные конструкции в зависимости от контекста их применения в программе: оператора транспонирования, дублирования, вычисления длины списка, арифметических операторов и оператора генерации. Для арифметических

операторов, передаваемых в качестве аргументов функции, предложено описывать на ФПП языках «функции-обертки» с указанием типов обрабатываемых данных.

На рис. 5 приведен пример преобразования повторяющихся вычислений, заданных на языке Пифагор с помощью конструкции интерпретации параллельных списков, в эквивалентный код на языке программирования C++.

```
// код на Пифагор:
//@{@(@int),@(@double)}->@(@double)
ex15 << funcdef X {
  return << (X:1, X:2):#:[]:+;
}
```

```
// эквивалентный код на C++:
vector<double > ex15 (vector<double > arg_0 ,vector<double > arg_1 ){
        size_t var1_size = arg_0.size();
        vector<double > var1(var1_size);
        for (size_t var1_iter = 0; var1_iter < var1_size; ++var1_iter){
                var1[var1_iter] = arg_0[var1_iter] + arg_1[var1_iter];
        }
        return var1;
}
```

Рисунок 5 – Пример преобразования цикла

Показано, что наиболее трудоемкой операцией при реализации системы трансформации программ является поиск в РИГ подграфов, соответствующих преобразуемым шаблонам. При этом целесообразно использовать ЯПФ программы, но необходимо учитывать вложенность узлов РИГ в задержанные списки. Предложены структуры данных и алгоритм трансформации ФПП программ в императивную форму, обеспечивающие высокое быстродействие разработанных инструментальных средств. Корректность работы системы проверена на ряде примеров, при этом в результате преобразования получен исходный код на языке C++, который успешно компилируется и решает поставленную перед ним задачу.

Основные результаты работы

Работа посвящена повышению эффективности исполнения функционально-потоковых параллельных программ посредством разработки методов их оптимизации и трансформации в императивную форму. Полученные результаты имеют значение для развития теоретических основ программирования, включающих языки программирования, технологии программирования, автоматизацию программирования.

1. Разработаны методы и алгоритмы оптимизации ФПП программ, применяемые на уровне информационных графов программы. Эти методы включают в себя: оптимизацию инварианта, удаление неиспользуемого кода, inline-подстановку, оптимизацию общих подвыражений, замену хвостовой рекурсии циклом и хвостового вызова длинным переходом, ряд преобразований на основе модели функционально-потоковых параллельных вычислений, удаление избыточных дуг в управляющих графах и оптимизацию управляющих автоматов. Эти методы позволяют сократить объем вычислений и объем памяти, потребляемый программой в процессе исполнения.
2. Созданы методы оптимизации программ, применяемые на уровне управляющих графов функционально-потоковых языков программирования, позволяющие обнаруживать и устранять избыточные управляющие дуги. За счет такого преобразования не

только сокращается количество сигналов готовности, передаваемых во время выполнения, но может быть уменьшен общий объем вычислений, если результаты учитываются системой интерпретации.

3. Предложен и реализован метод трансформации ФПП программ в императивные, основанный на обнаружении и преобразовании заранее определенного набора шаблонов. Метод повышает эффективность выполнения программ на традиционных архитектурах вычислительных систем, приводит к возможности применения к программам множества методов оптимизации, уже реализованных для императивных языков программирования. За счет такого преобразования реализуется новый способ исполнения функционально-потоковых параллельных программ без использования интерпретатора, это является перспективным направлением исследований.

4. Реализовано инструментальное средство оптимизации ФПП программ. Созданный оптимизатор интегрирован в существующую систему разработки функционально-потоковых параллельных программ и может служить основой для создания других систем трансформации кода.

Проведенные исследования могут развиваться по различным направлениям. Возможно дополнение системы оптимизации кода поддержкой новых методов и улучшением существующих. Результаты исследований по трансформации ФПП программ в императивное представление могут быть использованы при построении ФПП языка со статической типизацией. За счет дальнейшего развития системы интерпретации ФПП программ в части сбора статистической информации, может быть создана система профилирования кода. Результаты профилирования целесообразно использовать для более обоснованного применения оптимизирующих преобразований, а также при создании системы трансформации ФПП программ в параллельные императивные программы для различных вычислительных архитектур.

Публикации по теме диссертации

В изданиях, рекомендованных ВАК:

1. Легалов, А.И. Событийная модель вычислений, поддерживающая выполнение функционально-потоковых параллельных программ / А.И. Легалов, Г.В. Савченко, В.С. Васильев // Системы. Методы. Технологии. — 2012. — № 1 (13). — С. 113–119.

2. Васильев, В.С. Особенности преобразования хвостовой рекурсии в функционально-потоковом языке параллельного программирования / В.С. Васильев, А.И. Легалов, А.И. Постников // Системы. Методы. Технологии. 2013. №3. С. 106-111.

3. Васильев, В.С. Оптимизация параллельных списков функционально-потокового языка программирования "Пифагор" / В.С. Васильев, И.Н. Рыженко, И.В. Матковский // Системы. Методы. Технологии. - 2014. - № 3, С. 102-107.

4. Легалов, А.И. Инструментальная поддержка создания и трансформации функционально-потоковых параллельных программ / А.И. Легалов, В.С. Васильев, И.В. Матковский, М.С. Ушакова // Труды Института системного программирования РАН. — 2017. — Т. 29, № 5. — С. 165–184.

5. Васильев, В. С. Оптимизация инварианта цикла в языке Пифагор / В.С. Васильев, А.И. Легалов // Моделирование и анализ информационных систем. – 2018. – Т. 25. – № 4. – С. 347-357.

6. Васильев, В.С. Оптимизация графов потока управления в промежуточных представлениях языка функционально-потокового параллельного программирования / В.С. Васильев, А.И. Легалов // Научный вестник Новосибирского государственного технического университета. – 2020. – № 4. – С. 37-46.

7. Васильев, В. С. Трансформация функционально-потоковых параллельных программ в императивные / В.С. Васильев, А.И. Легалов, С.В. Зыков // Моделирование и анализ информационных систем. – 2021. – Т. 28. – №. 2. – С. 198-214.

В индексируемых базах данных:

8. Legalov, A.I. A Toolkit For The Development Of Data-Driven Functional Parallel Programmes / A.I. Legalov, V.S. Vasilyev, I.V. Matkovskii, M.S. Ushakova // Communications in Computer and Information Science. — 2018. — Vol. 910. — P. 16–30.

9. Vasilev, V.S. Loop-invariant Optimization in the Pifagor Language / V.S. Vasilev, A.I. Legalov // Automatic Control and Computer Sciences. - 2018 . - Vol. 52, P. 843-849.

В других изданиях и материалах конференций:

10. Васильев, В.С. Оптимизирующие преобразования программ языка "Пифагор" / В.С. Васильев // Перспективы развития информационных технологий: сборник материалов VII Международной научно-практической конференции. - 2012. С.13-24.

11. Васильев, В.С. Оптимизация хвостовой рекурсии ФПЯП Пифагор / В.С. Васильев // Молодежь и наука: сборник материалов IX Всероссийской научно-технической конференции [Электронный ресурс]. — Красноярск: Сибирский федеральный ун-т, 2013. — Режим доступа: http://conf.sfu-kras.ru/sites/mn2013/section045.html, свободный.

12. Васильев, В. С. Оптимизация программ функционально-потокового языка Пифагор / В.С. Васильев // Перспективы развития информационных технологий: сборник материалов XX Международной научно-практической конференции. - Новосибирск. - 2014. С. 7-14.

13. Васильев, В. С. Оптимизация хвостового вызова в языке «Пифагор» / В.С. Васильев // Материалы 53-й Международной научной студенческой конференции МНСК-2015: Информационные технологии. - Новосибирск. - 2015, 297 С.

14. Легалов, А.И. Изменение стратегий управления вычислениями при архитектурно-независимом параллельном программировании / А.И. Легалов, В.С. Васильев, И.В. Матковский // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции. - М.: ИПМ им. М.В.Келдыша, 2017. с. 341-351.

15. Васильев, В. С. Оптимизация кода функционально-потокового языка программирования Пифагор / В.С. Васильев // Суперкомпьютерные дни в России. – Москва, 2018. – С. 997-998.

16. Васильев, В.С. Использование системы оптимизации функционально потоковых параллельных программ / В.С. Васильев // Приоритетные направления инновационной деятельности в промышленности: сборник научных статей международной научной конференции. - Казань, 2021. – С. 81-85.

17. Васильев, В.С. Внутренние представления в системе оптимизации функционально-потоковых параллельных программ / В.С. Васильев, О.В. Непомнящий // Современное программирование. - Нижневартовск, 2021. – С. 9-15.

Свидетельства о государственной регистрации программ для ЭВМ:

18. Легалов, А.И. Свидетельство о государственной регистрации программы для ЭВМ № 2013611617 Российская Федерация. Модуль формирования графического представления реверсивного информационного графа / А.И. Легалов, О.В. Непомнящий, В.С. Васильев, И.В. Матковский; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Сибирский федеральный университет" (СФУ). — № 2012660517; заявл. 03.12.2012; опубл. 29.01.2013г. — 1 с.

19. Васильев, В. С. Свидетельство о государственной регистрации программы для ЭВМ № 2013613531 Российская Федерация. Модуль формирования оптимизационного внутреннего представления программ ФПЯП «ПИФАГОР» / В.С. Васильев, А.И. Легалов, М.С. Кропачева; заявитель и правообладатель Федеральное государственное автономное

образовательное учреждение высшего профессионального образования "Сибирский федеральный университет" (СФУ). — № 2013611551; заявл. 21.02.2013; опубл. 10.04.2013г. — 1 с.

20. Васильев, В.С. Свидетельство о государственной регистрации программы для ЭВМ 2018666434 Российская Федерация. Программа для оптимизации функционально-потокового языка параллельного программирования / В. С. Васильев, А. И. Легалов, И. В. Матковский, О. В. Непомнящий; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Сибирский федеральный университет" (СФУ). — № 2018663789; заявл. 03.12.2018; опубл. 17.12.2018г. — 1 с.

21. Матковский, И.В. Свидетельство о государственной регистрации программы для ЭВМ 2018666239 Российская Федерация. Программа для интерпретации функционально-потокового языка параллельного программирования / И.В. Матковский, А.И. Легалов, В.С. Васильев, О.В. Непомнящий; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Сибирский федеральный университет" (СФУ). — № 2018663794; заявл. 03.12.2018; опубл. 13.12.2018г. — 1 с.

22. Матковский, И.В. Свидетельство о государственной регистрации программы для ЭВМ 2018666577 Российская Федерация. Программа для трансляции функциональнопотокового языка параллельного программирования / И.В. Матковский, А.И. Легалов, В.С. Васильев, А.И. Постников; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Сибирский федеральный университет" (СФУ). — № 2018663780; заявл. 03.12.2018; опубл. 18.12.2018г. — 1 с.

23. Матковский, И.В. Свидетельство о государственной регистрации программы для ЭВМ 2018663792 Российская Федерация. Программа для формирования управляющего графа для функционально-потокового языка параллельного программирования / И.В. Матковский, А.И. Легалов, В.С. Васильев, О.В. Непомнящий; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования "Сибирский федеральный университет" (СФУ). — № 2018663792; заявл. 03.12.2018; опубл. 13.12.2018г. — 1 с.

24. Васильев, В.С. Свидетельство о государственной регистрации программы для ЭВМ № 2021662788 Российская Федерация. Программа преобразования реверсивных информационных графов языка ПИФАГОР в программы на языке С++ / В. С. Васильев, О. В. Непомнящий; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Сибирский федеральный университет" (СФУ). — № 2021661762; заявл. 26.07.2021; опубл. 04.08.2021. — 1 с.