МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский Государственный Университет

Республиканский конкурс научных работ студентов
высших учебных заведений Республики Беларусь

Информатика и информационные технологии. Программное обеспечение вычислительной техники и автоматизированных систем. Методы искусственного интеллекта

# К ВОПРОСУ ОБ АВТОМАТИЗАЦИИ РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ

Львович Виктория Дмитриевна, 5 курс
Перемотова Анна Николаевна, 5 курс

Кондратьева Ольга Михайловна,
старший преподаватель кафедры
многопроцессорных систем и сетей

Минск, 2016

# РЕФЕРАТ

Работа, 20 с., 4 рис., 3 табл., 10 источников.

ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ, АВТОМАТИЧЕСКОЕ РАСПАРАЛЛЕЛИВАНИЕ, ТРАНСЛЯЦИЯ ПРОГРАММ, ИНСТРУМЕНТ ANTLR, ТЕХНОЛОГИЯ OPENMP, БИБЛИОТЕКИ ПОДДЕРЖКИ МНОГОПОТОЧНОСТИ

Объектом исследования является процесс автоматизации разработки параллельных приложений.

Цель работы – разработка программных средств, которые могут быть использованы для автоматизации процесса разработки параллельных программ.

Методы исследования – системный и структурный анализ, методы проектирования программ.

Приведены описания двух авторских инструментов, которые позволяют автоматически получить параллельную программу одним из двух способов: распараллеливание циклов (из последовательной программы посредством расстановки директив OpenMP) и спецификация задачи поиска (задать параметры задачи поиска и получить параллельные реализации в нескольких вариантах).

Область применения – проектирование параллельных программ, обучение параллельному программированию.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

На сегодняшний день практически все высокопроизводительные системы используют преимущества такой технологии, как параллелизм. И это неслучайно, ведь данная технология позволяет наиболее эффективно использовать мощности современных вычислительных систем. Существование большого количества отлаженного и активно используемого последовательного кода «заставляет мечтать» о его автоматической трансформации в параллельный код. Возможность автоматического распараллеливания сулит очевидные выгоды:

- сокращение времени разработки параллельных программ;
- удешевление процесса разработки параллельных программ;
- снижение требований к квалификации программистов;
- повышение надежности разрабатываемых программ;
- упрощение переноса многих разработанных программ на параллельные компьютеры;
- эффективное проектирование программно-аппаратных комплексов с учетом новых методов распараллеливания.

Цель работы – помочь пользователю в разработке параллельного приложения. В результате исследования были разработаны программные инструменты, позволяющие упростить разработку многопоточных программ. Инструменты позволяют автоматически получить параллельную программу одним из двух способов:

- из последовательной программы посредством расстановки директив OpenMP;
- задать параметры задачи поиска и получить параллельные реализации в нескольких вариантах.

# 1 СРЕДСТВА ПОДДЕРЖКИ РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

## 1.1 Средства создания и проектирования параллельных программ

На практике существуют несколько технологических подходов к программированию для параллельных вычислительных систем [1]:

- Программирование на стандартных и широко распространненых языках программирования с использованием высокоуровневых коммуникационных библиотек и интерфейсов (API) для организации межпроцессного взаимодействия;
- Введение специальных "распараллеливающих" конструкций в язык программирования. При этом могут создаваться оригинальные параллельные языки или параллельные расширения существующих (с сохранением преемственности);
- Использование средств автоматического распараллеливания последовательных программ;
- Программирование на стандартных языках. Использование в качестве конструктивных элементов заранее распараллеленных процедур из специализированных библиотек;
- Использование инструментальных систем, облегчающих создание и проектирование параллельных программ;
- Использование специализированных прикладных пакетов.

Наш главный интерес лежит в области автоматического распараллеливания последовательных программ.

Вопрос упрощения разработки параллельных программ интересует многих. Помимо очевидных преимуществ, его решение позволило бы улучшить качество программного кода и структурировать распараллеливание. Безусловно, на сегодняшний день уже существует множество средств, каждое из которых обладает своими отличительными характеристиками. Важно понимать, что создание универсального средства работы с распараллеливанием на сегодняшний день не представляется возможным, поэтому даже самые лучшие средства обладают определенными ограничениями в использовании и применяются в конкретных случаях.

К наиболее известным интегрированным средам прототипирования, разработки и отладки параллельных программ относятся следующие [2].

*CODE* – графическая система создания параллельных программ. Параллельная программа представляется в виде графа, вершинами которого являются последовательные участки, а дуги соответствуют пересылкам данных. Последовательные участки могут быть написаны на любом языке, для пересылок используется PVM или MPI.

*HeNCE* (Heterogeneous Network Computing Environment) – графическая среда, разработанная для помощи в разработке параллельных программ, работающих на сети рабочих станций в модели PVM. Предназначена для вычислителей, желающих эффективно использовать имеющуюся сеть рабочих станций, не вдаваясь в подробности параллельного программирования. HeNCE включает графические средства построения, компиляции, запуска и анализа специальных HeNCE-программ (программист сам не пишет в терминах PVM), а также средства графического конфигурирования гетерогенного PVM-кластера для запуска программы. Обеспечивает высокий уровень абстракции для спецификации параллелизма. Программа описывается в виде направленного ациклического графа, в котором вершины соответствуют процедурам, а дуги – зависимостям, расставляемым пользователем.

*GRADE* (A Graphical Parallel Programming Environment) – интегрированный набор средств программирования для разработки приложений в модели передачи сообщений на гетерогенных кластерах, включающий в себя 6 средств: GRAPNEL – графический язык параллельного программирования, GPED – графический редактор для построения GRAPNEL-программ, GRP2C – препроцессор для генерации программ на PVM/C, Tape/PVM – средство генерации трасс, DDBG – распределенный отладчик, PROVE – средство визуализации трасс.

Данный проект является исследовательским проект, однако существует P-GRADE – профессиональная коммерческая версия системы.

*TRAPPER* – графическая среда программирования, поддерживающая все стадии жизненного цикла программных систем. Содержит компоненты построения параллельного ПО, конфигурирования аппаратных средств, распределения процессов по процессорам, графической отладки и мониторинга производительности. Состоит из 4 основных модулей: The Design Tool (построение параллельного приложения из отдельных процессов, обменивающихся сообщениями; модульная структура, иерархическая организация процессов, автоматическая генерация "шаблонов" программ), The Configuration Tool (графическое определение конфигурации виртуальной машины; генерация Make-файлов, отображение процессов на процессоры), The Visualization Tool (графическое изображение поведения процессов и взаимодействий, графическая отладка, позволяющая отлавливать различные ошибки распараллеливания), The Performance Tool (обнаружение критических для производительности секции кода, сбор статистики и ее отображение в виде диаграмм). Данный продукт является коммерческим и разработан немецкой компанией GENIAS.

*EDPEPPS* (An Environment for the Design and Performance Evaluation of Portable Parallel Software) – интегрированная среда для обеспечения быстрого, эффективного и гибкого проектирования переносимых параллельных программ. В EDPEPPS входят 3 основных средства: графическое средство конфигурирования для быстрого прототипирования, средство моделирования, основанное на виртуальной машине, исполняющей "скелетные" параллельные программы, средство визуализации для изображения интересующих пользователя характеристик, исходя из данных, полученных при моделировании. В данном исследовательском проекте все ориентировано на PVM.

*Reactor* – среда разработки распределенных и клиент-серверных приложений. Включает поддержку управления ресурсами, обработку исключительных ситуаций, поддержку нитей, сборку мусора и другие компоненты. Reactor является коммерческим продуктом, разработанным компанией Critical Mass.

*DEEP* (Development Environment for Parallel Programming) – интегрированная среда для параллельного программирования. Обеспечивает графический интерфейс, который связывает средства анализа производительности и отладки с исходным кодом программы. Помогает понять структуру и поведение параллельной программы. DEEP поддерживает языки Fortran (77, 90, 95) и C, а также смешанные программы на обоих языках. Модели программирования являются общая память (автоматическое распараллеливание циклов или OpenMP), параллелизм по данным (HPF, Data Parallel C) и передача сообщений (MPI). DEEP является коммерческим продуктом, разработанным компанией Pacific-Sierra Research.

*Converse* – среда для поддержки многоязыкового параллельного программирования. Поддерживаются парадигмы обмена сообщениями, объектно-ориентированного программирования, многопоточности. Среда Converse разработана в Parallel Programming Laboratory (PPL), University of Illinois at Urbana-Champaign.

## 1.2 Автоматизация распараллеливания программ

На сегодняшний день большинство высокопроизводительных систем использует преимущества параллелизма. И это неслучайно, ведь данная технология позволяет наиболее эффективно использовать вычислительные мощности систем. Однако процесс создания параллельной программы нельзя назвать простым. Чаще всего сначала создается последовательный алгоритм, который впоследствии преобразуется в параллельный. Автоматизация распараллеливания значительно упростила бы весь процесс благодаря ряду неоспоримых преимуществ, к которым можно отнести:

- ускорение времени разработки параллельных программ;
- удешевление разработки параллельных программ;
- снижение требований к квалификации программистов;

- повышение надежности разрабатываемых программ, поскольку объем исходного текста последовательной программы на порядок меньше, чем параллельной, а ее логическая структура проще;
- упрощение переноса многих разработанных программ с последовательных компьютеров на параллельные;
- эффективное проектирование программно-аппаратных комплексов с учетом новых методов распараллеливания.

Интенсивное развитие технологий высокопроизводительных параллельных вычислений привело к тому, что даже рядовые пользователи, не имеющие дорогостоящих высокопроизводительных вычислителей, получили возможность решать ресурсоемкие задачи.

Поскольку программы пишутся не для эффективного использования вычислительных мощностей, а для решения некоторых конкретных задач, распараллеливание и распределение вычислительных мощностей является моментом, затрудняющим их написание. При этом зачастую программисты формируют алгоритмы в привычной для них последовательной форме, на которую ориентируется большая часть программных средств, поддерживающих процесс разработки. Однако стоит отметить, что построение исходных программ изначально в параллельной форме чаще всего привязано к конкретной архитектуре. Что, безусловно, тормозит применение параллельных вычислителей. Поэтому нередко на многопроцессорных системах запускаются программы без распараллеливания. Соответственно имеет место неэффективное использование вычислительных мощностей системы. Таким образом, вместо распараллеленного используется обычный последовательный код, а значит, возможности параллельного программирования просто игнорируются.

Указанные недостатки можно легко ликвидировать, создав специальные средства автоматического распараллеливания последовательных программ. При этом организация вычислительного процесса в компьютерной системе и, в частности, распределение фрагментов программы по процессорам будет является обязанностью соответствующих операционной системы и исполнительной среды, а распараллеливание программ – осуществляться специальной утилитой на этапе создания самой программы. Данный вариант реализации средства автоматического распараллеливания позволит не отказываться от существующих технологий разработки последовательных программ и вместе с тем даст возможность повысить эффективность использования многопроцессорных вычислительных систем.

На сегодняшний день существуют некоторые системы автоматического распараллеливания: BERT-77, KAP, FORGE, PIPS, VAST, V-Ray и др [3]. Однако подавляющее большинство представленных средств имеют ряд существенных недостатков, среди которых:

- большая стоимость коммерческих систем;
- трудность освоения;
- использование языка программирования, не входящего в арсенал современных средств.

Таким образом, создание системы автоматического распараллеливания, преобразующей исходную последовательную программу в параллельную, до сих пор является перспективным направлением.

Чаще всего в системах автоматического распараллеливания применяется граф зависимостей. Так в системе POLARIS в ходе всего процесса распараллеливания используется граф зависимостей по данным для всей программы, который модифицируется после каждого преобразования. В OPC и SUIF граф зависимостей строится только для преобразуемого фрагмента программы, что упрощает разработку преобразований и сокращает время компиляции. Кроме того, в OPC, в отличие от других распараллеливающих систем, используется решетчатый граф программы, позволяющий разрабатывать новые приемы распараллеливания с автоматически вставленными синхронизациями для циклов с

зависимыми итерациями, а модификация двух информационных графов отнимает вдвое больше времени на компиляцию и разработку. Система POLARIS работает только с программами на языке Fortran для SMP-архитектур. Система PARAWISE имеет возможность диалога при распараллеливании, но располагает ограниченным набором преобразований. Система SUIF (Stanford University Intermediate Format) акцентируется на внутреннем высокоуровневом представлении – тем самым подчеркивается его важность. Система работает с языком C, а для преобразования программ на Fortran используется конвертор f2c.

К многоязыковым системам следует отнести семейство оптимизирующих компиляторов GNU с многоуровневым внутренним представлением. В оптимизирующих компиляторах известных производителей имеются возможности автоматического распараллеливания циклов с независимыми итерациями. Но, чем выше уровень вложенности цикла в гнезде циклов, тем статистически реже его итерации независимы. А значит, осуществить распараллеливание сложнее или вообще невозможно. Чем выше уровень вложенности современных многоядерных процессоров, тем эффективнее распараллеливание.

## 1.3 Использование библиотек параллельных алгоритмов и классов

В настоящее время существует достаточно большое количество библиотек параллельных алгоритмов. Многие из них являются бесплатно распространяемыми и могут быть применены для эффективного решения некоторой задачи с использованием параллельных вычислений. Например, существуют такие библиотеки как:

- ATLAS (Automatically Tuned Linear Algebra Software) - библиотека, позволяющая автоматически генерировать и оптимизировать численное программное обеспечение для процессоров с многоуровневой организацией памяти и конвейерными функциональными устройствами. ATLAS требует некоторого времени для изучения основных параметров архитектуры целевого компьютера, а затем на основе этих параметров получает "оптимальный" код;
- Thrust – библиотека, которая позволяет реализовывать операции сортировки, сканирования, трансформации и сокращения данных;
- BlockSolve95 - параллельная библиотека для решения разреженных систем линейных уравнений. Реализована с помощью MPI;
- Aztec - параллельная библиотека итерационных методов для решения систем линейных уравнений. Позволяет описывать части распределенной матрицы с использованием глобальной адресации;
- NAMD - параллельная объектно-ориентированная библиотека для расчетов в области молекулярной динамики. Реализована на C++ с использованием шаблонов. Распространяется бесплатно.

Однако, в виду в виду сложности создания параллельной программы, создание универсальной библиотеки параллельных алгоритмов является чрезвычайно сложной задачей. Именно по этой причине все представленные библиотеки решают конкретный набор задач, связанный с определённой областью – задачи обработки данных, решения уравнений, проведения физических расчётов, реализация генетического алгоритма и т.д.

Другим, возможно более универсальным, способом создания параллельных программ может стать использование специализированной библиотеки классов некоторого языка программирования. Всего в мире по статистике придумано более восьми тысяч языков программирования. Каждый год их число увеличивается. Некоторые языки использует лишь узкая группа специалистов, другие получают широкое распространение. В основном наиболее популярными языками являются те, которые позволяют решать множество различных типов задач. Большинство из этих языков включает в себя и специальную библиотеку или даже несколько библиотек для написания параллельных программ.

Например, для языка программирования C++ существует довольно много кроссплатформенных многопоточных библиотек. В [4] представлен список из 20 библиотек:

- Rogue Wave SourcePro Threads Module;
- Boost Thread;
- C++ Standard Library Thread;
- commonc++;
- Dlib;
- Dthreads;
- JThread;
- just::thread;
- OpenMP;
- OpenThreads;
- Parallel Patterns Library;
- POCO C++ Libraries Threading;
- POSIX Threads;
- PTypes (C++ Portable Types Library);
- Qt QThread;
- Stapl;
- Threading Building Blocks (TBB);
- TinyThread++;
- ZThreads;
- jlibcpp.

Эти программные средства предназначены для написания параллельных программ на языке программирования C++. Они относятся к классу программных средств «библиотека» или «фрэймворк».

Однако, будучи достаточно универсальными, эти языки могут оказаться и более сложными в изучении. Поэтому, как вариант, в некоторых ситуациях при разработке программ для решения конкретных узких задач может возникнуть необходимость в разработке собственного специализированного языка программирования.

# 2 ОТ С-ПРОГРАММЫ К OPENMP-ПРОГРАММЕ

Технология OpenMP является отраслевым стандартом и очень успешным языком параллельного программирования для компьютеров с общей памятью. Значительная часть функциональности OpenMP реализуется при помощи директив компилятору. Они должны быть явно вставлены пользователем, что позволит выполнять программу в параллельном режиме.

OpenMP поддерживает такой стиль программирования, при котором параллелизм добавляется постепенно, пока имеющаяся последовательная программа не превратится в параллельную. Главная задача OpenMP – облегчить написание программ, ориентированных на циклы.

Ниже представлен разработанный программный инструмент, который преобразовывает последовательную программу, написанную на языке программирования C/C++, в параллельный код с помощью директив OpenMP. Поскольку наиболее ресурсоемким является выполнение циклов, основное внимание программа уделяется распараллеливанию циклов. На вид параллельных циклов накладываются достаточно жёсткие ограничения. В частности, предполагается, что программа не должна зависеть от того, какой именно поток и какую итерацию параллельного цикла она выполнит. Если тело цикла не удовлетворяет условиям распараллеливания, то пользователю предоставляется описание причины, не позволившей распараллелить цикл.

## 2.1 Инструмент автоматического распараллеливания

Параллельное программирование включает в себя все черты более традиционного, последовательного программирования, но в параллельном программировании имеются три дополнительных, четко определенных этапа:

- определение параллелизма: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно;
- выявление параллелизма: изменение структуры задачи таким образом, чтобы можно было эффективно выполнять подзадачи. Для этого часто требуется найти зависимости между подзадачами и организовать исходный код так, чтобы ими можно было эффективно управлять;
- выражение параллелизма: реализация параллельного алгоритма в исходном коде с помощью системы обозначений параллельного программирования.

Системой обозначений, используемой на третьем этапе, могут служить язык параллельного программирования, прикладной программный интерфейс, реализованный с помощью библиотечного интерфейса, или расширение, добавленное к существующему языку последовательного программирования.

Обычно программист создает последовательный алгоритм, а только потом его распараллеливает. Для этого сначала проводится анализ задачи с целью определения параллельных подзадач, затем – изменение самой структуры задачи для наиболее эффективного выполнения подзадач и наконец – реализация параллельного алгоритма в исходном последовательном коде с помощью некоторой системы обозначений параллельного программирования, зависящей непосредственно от языка программирования.

Автоматизация преобразования последовательного кода в параллельный является очень сложной задачей, так как предполагается, что компьютер самостоятельно выполнит все вышеописанные этапы. А значит, данную задачу можно отнести к задачам искусственного интеллекта. Стоит отметить, что на сегодняшний день такая задача не может быть полностью решена, так как недостаточно просто преобразовать исходный последовательный код в параллельный. Помимо этого, нужно максимально эффективно задействовать все вычислительные мощности системы.

Таким образом, для решения задачи автоматизации распараллеливания необходимо использовать методы, внешне аналогичные методам естественного интеллекта. Причем решение задачи должно быть эффективным. Во-первых, при этом необходимо использовать наиболее подходящие для той или иной задачи средства. Во-вторых, полученный параллельный код должен выполняться за наименьшее относительно других вариантов кода время.

Поскольку исходным языком программирования был выбран язык C/C++, наилучшим вариантом для перехода от последовательного кода к параллельному стало использование директив OpenMP [5, 6].

OpenMP – интерфейс программирования приложений который является отраслевым стандартом для создания параллельных приложений для компьютеров с совместным использованием памяти. Главная задача OpenMP – облегчить написание программ, ориентированных на циклы. Такие программы часто создаются для высокопроизводительных вычислений. Кроме того, в OpenMP были включены компоненты для поддержки таких параллельных алгоритмов как «одна программа, множество данных», «главный и рабочий процесс», конвейерный и т.д.

Технология OpenMP является одним из наиболее популярных средств программирования для компьютеров с общей памятью, базирующихся на традиционных языках программирования и использовании специальных комментариев. За основу берётся последовательная программа, а для создания её параллельной версии пользователю предоставляется набор директив, функций и переменных окружения. Предполагается, что создаваемая параллельная программа будет переносимой между различными компьютерами с разделяемой памятью, поддерживающими OpenMP.

Основная часть функциональности OpenMP реализуется при помощи директив компилятору, которые должны быть явно вставлены пользователем, что позволит выполнять программу в параллельном режиме. OpenMP поддерживает такой стиль программирования, при котором параллелизм добавляется постепенно, пока имеющаяся последовательная программа не превратится в параллельную. Стоит отметить, это преимущество является и самым слабым местом OpenMP. Если параллелизм добавляется постепенно, можно не выполнить широкомасштабную перестройку программы, которая часто необходима для достижения максимальной производительности.

OpenMP основывается на модели программирования «разветвление-объединение». Работа программы OpenMP начинается с одного потока. Когда необходимо добавить в программу параллелизм, выполняется разветвление на несколько потоков, чтобы создать группу потоков. Эти потоки выполняются параллельно в рамках фрагмента кода, который называется параллельным участком. В конце параллельного участка все потоки заканчивают свою работу и снова объединяются вместе. После этого исходный (главный) поток продолжает выполняться до тех пор, пока не начнется следующий параллельный участок или пока не наступит конец программы.

Автоматизация распараллеливания предполагает осуществление лексического, синтаксического и семантического анализа. Поэтому было решено, что все преобразования исходного кода в параллельный код будут осуществляться посредством лексического и синтаксического анализаторов, сгенерированных с помощью инструмента ANTLR [7].

Таким образом, последовательная программа проходит следующие этапы обработки:

- лексический и синтаксический анализ;
- семантический анализ (проверка ограничений на циклы);
- генерация кода (вставка директив OpenMP) или выдача сообщения об ошибке (вставка комментария о причине, по которой невозвожно распараллелить цикл).

## 2.2 Функциональные возможности инструмента

Созданная программа позволяет преобразовывать исходный последовательный программный код на языке C/C++ в параллельный программный код. При этом подразумевается,

что последовательный код соответствует всем правилам исходного языка программирования и написан корректно.

С левой стороны вводится исходный программный код, а справа по нажатию на кнопку «Преобразовать» появляется уже распараллеленный. Все добавленные строки выделены зеленым цветом. Если же по каким-то причинам невозможно распараллелить тело цикла, то красным цветом выводится соответствующий комментарий. Таким образом, полученный программный код можно сразу использовать, ничего не корректируя, и при этом, видя, почему тот или иной цикл нельзя распараллелить. Предполагается, что программист, использующий данную программу, может в случае необходимости отредактировать сгенерированный код и лишь потом сохранить его. В программе есть возможность задать некоторые настройки, влияющие на те или иные директивы OpenMP: задать число потоков, определить, как будут распределены между ними итерации. Графический интерфейс программы представлен на рисунке 1.

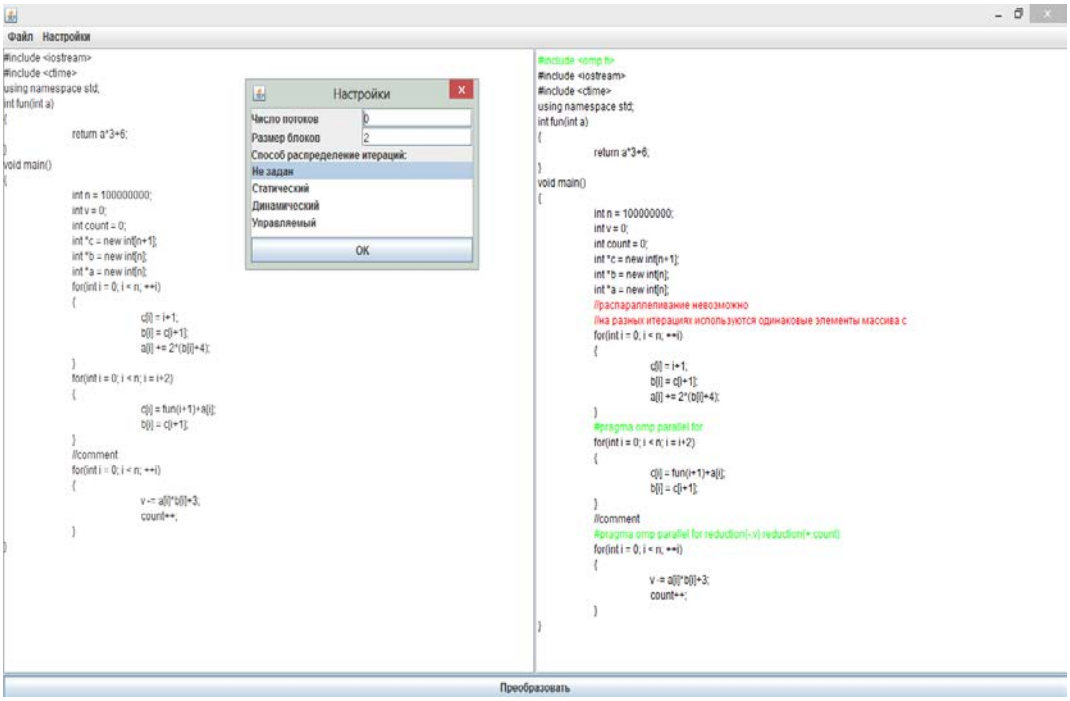


Рисунок 1 – Графический интерфейс пользователя

Сравнить полученную программу с системами автоматического распараллеливания очень сложно, так как большинство из них являются коммерческими. Это один из фактов, который подтверждает актуальность задачи автоматического распараллеливания и сложность ее полноценной реализации. Единственной некоммерческой системой, использующей возможности OpenMP, которую удалось найти является PIPS. Эта система представляет собой автоматический распараллеливатель для научных программ. PIPS основан на линейных методах алгебры, выявлении зависимостей, перестановке циклов. Однако PIPS не преобразовывает исходный код в код на языке C/C++ с использованием директив OpenMP. PIPS работает в первую очередь с Fortran 77.

## 2.3 Результаты использования инструмента

Здесь представлены результаты следующего эксперимента:

1) берем последовательную программу;
2) подаем ее на вход разработанного автоматического распараллеливателя;
3) получаем параллельную программу и проводим с ней вычислительные эксперименты с целью определения ее эффективности.

Результаты некоторых экспериментов представлены в таблицах 1-3. В таблице 1 представлены результаты эксперимента по вычислению значения числа π, в таблице 2 – скалярное произведение матрицы на вектор, в таблице 3 – перемножение матриц.

Таблица 1 – Вычисление числа $\pi$

| Размерность задачи | Время выполнения последовательной программы, ms | Параллельная программа на 2 процессорах | | |
|---|---|---|---|---|
| | | Время выполнения, ms | Ускорение | Эффективность |
| 100 000 000 | 1781 | 1116 | 1,60 | 0,80 |
| 1 000 000 000 | 18566 | 11033 | 1,68 | 0,84 |

Таблица 2 – Скалярное произведение матрицы на вектор

| Размерность задачи | Время выполнения последовательной программы, ms | Параллельная программа на 2 процессорах | | |
|---|---|---|---|---|
| | | Время выполнения, ms | Ускорение | Эффективность |
| 100*100*100 | 297 | 203 | 1,46 | 0,73 |
| 1000*100*100 | 2871 | 1856 | 1,54 | 0,77 |

Таблица 3 – Вычисление произведения матриц

| Размерность задачи | Время выполнения последовательной программы, s | Параллельная программа на 2 процессорах | | |
|---|---|---|---|---|
| | | Время выполнения, s | Ускорение | Эффективность |
| 100*100*100*100 | 30,12 | 21,53 | 1,40 | 0,70 |
| 1000*100*100*1000 | 295,12 | 201,30 | 1,47 | 0,73 |

Таким образом, разработанный инструмент позволяет автоматизировать распараллеливание, которое в свою очередь дает возможность эффективнее использовать вычислительные возможности компьютера.

# 3 ОТ ПОСТАНОВКИ ЗАДАЧИ К ПАРАЛЛЕЛЬНОЙ ПРОГРАММЕ

## 3.1 Инструмент генерации параллельных программ на основе шаблонов

В рамках данной работы также предложен язык, предназначенный для формализации задач поиска, и разработана его грамматика. Фрагмент кода на этом языке имеет следующий вид:

```
use boostthread library;
number of threads = 4;
search for “luck”;
in “D:\CP\DATA”;
print filenames and time;
```

Был создан программный инструмент, который осуществляет перевод программы, написанной на авторском языке, на язык программирования C++. Результирующая программа имеет следующие характеристики:

- является многопоточной;
- используется модель делегирования [10];
- может быть реализована в одном из трех вариантов: OpenMP, Boost.Thread и TinyThread++.

Boost – это набор бесплатных портативных библиотек. Все библиотеки хорошо взаимодействуют со стандартными библиотеками C++, могут использоваться в широком спектре приложений и помогать в решений множества различных задач. Библиотека Boost.Thread позволяет использовать несколько потоков для выполнения некоторого участка кода. Эта библиотека предоставляет классы и функции для того чтобы управлять самими потоками, а также позволяет синхронизировать данные между потоками. Библиотека TinyThread++ – это очень маленькая портативная реализация базовых классов потоков C++. Основная её задача – сделать разработку многопоточных C++-приложений как можно проще.

Реализация приложения поиска состоит из трёх основных частей:

- ANTLR;
- Java-приложение
- C++-приложение.

Сначала была разработана и написана грамматика для генератора ANTLR. Эта грамматика определяет параметры, которые будут использоваться при генерации кода для осуществления поиска. Этими параметрами являются – используемая библиотека, количество потоков, ключевое слово для поиска, путь к файлу или файлам, а также в каком виде нужно отобразить результат. При помощи этой грамматики были сгенерированы классы, написанные на Java, которые далее используются в приложении (лексер, парсер и слушатель событий исходного языка).

Java-приложение используется для реализации пользовательского интерфейса приложения. Окно Java-приложения является редактором для ввода кода программы на исходном языке программирование. У редактора есть такие удобные функции как подсказки при вводе текста, выделение значений параметров другим цветом, диалоговое окно для ввода директории. После написания программы пользователь может нажать на кнопку запуска. При нажатии на эту кнопку будет сгенерирован код на языке C++, используя параметры, которые были получены из исходной пользовательской программы при помощи лексер, парсер и слушателя событий в Java. Сгенерированная C++ - программа будет автоматически скомпилирована и запущена. Результат её работы будет отображён пользователю в отдельном окне.

Стоит отметить, что C++ - программа должна удовлетворять модели делегирования многопоточных приложений.

## 3.2 Пользовательский интерфейс и возможности инструмента

Для взаимодействия с пользователем используется Java-приложение. Оно имеет графический интерфейс, который представляет собой окно с полем для ввода кода программы на упрощенном языке программирования, разработанным при помощи ANTLR, а также панели с кнопками для запуска и изменения кода пользовательской программы.

Поле ввода в окне программы имеет функцию выведения подсказок для ключевых слов собственного языка программирования. Вид подсказок изображён на рисунке 2.

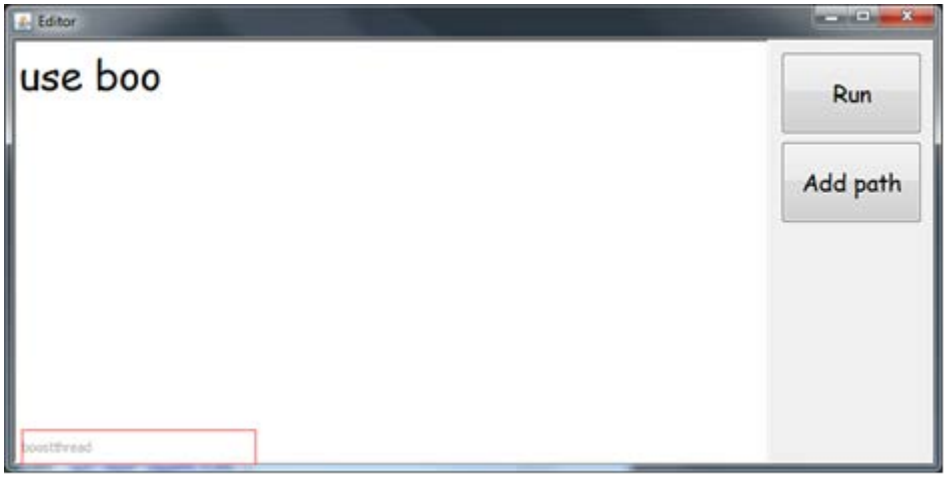


Рисунок 2 – Главное окно приложения

Также при вводе кода программы пользователем есть функция выделения цветом значений параметров. Пример изображён на рисунке 3.

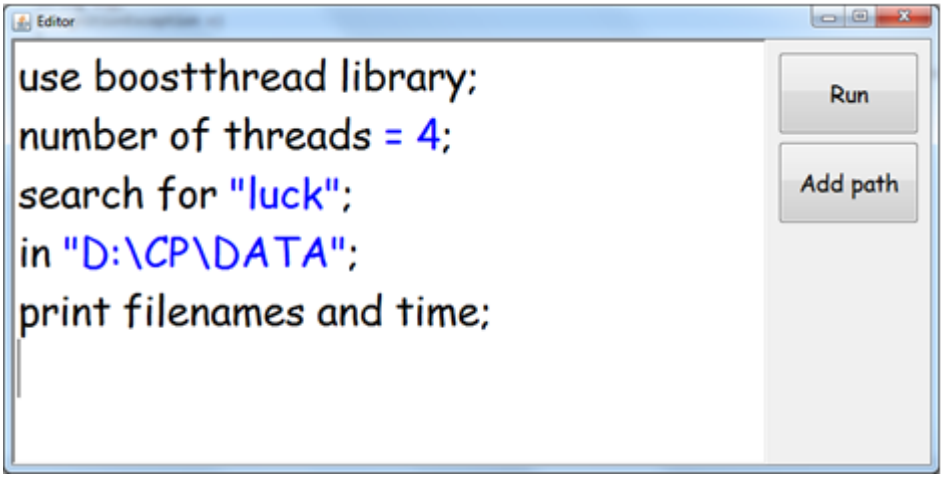


Рисунок 3 – Отображение значений параметров

Слева в окне приложения расположена панель с кнопками: «Run» и «Add path». Кнопка «Run» имеет несколько функций. При нажатии на неё пользователем сначала происходит разбор текста программы, введённой пользователем программы, выделяются значения необходимых параметров и записываются как поля класса. После этого происходит проверка, что выбранная пользователем конфигурация для программы на C++ является оптимальной. Для этих целей используется полученная ранее статистика работы приложения – на различных процессорах, операционных системах. Если программа пользователя удовлетворяет всем критериям оптимальности, то происходит генерация кода на языке C++, а после этого запуск данной программы. В том случае, если конфигурация не является оптимальной, то появится диалоговое окно, которое предложат пользователю внести изменения в программу, либо

продолжить далее без модификаций. Пример диалогового окна на рисунке 4. Если пользователь выберет опцию внесения изменений, то диалоговое окно закроется и она сможет вновь отредактировать код программы, выбрав улучшенные параметры. Если пользователь откажется, то сгенерируется C++-программа в соответствии с его значениями параметров и будет запущена.

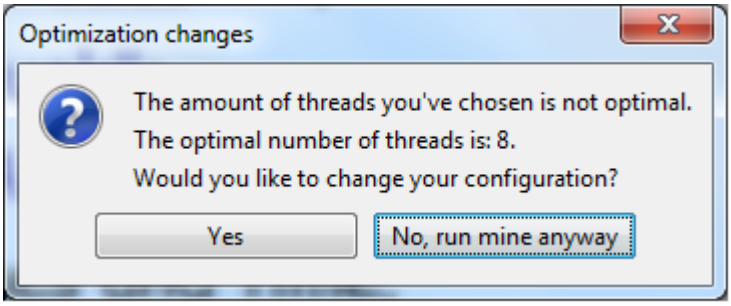


Рисунок 4 – Диалоговое окно для оптимизации значений

Для генерации программы на языке C++ используются заранее написанные шаблоны для разных библиотек потоков, которые хранятся в текстовых файлах. Эти шаблоны представляют собой программы, в которых не заданы лишь параметры, которые определяются пользователем. При создании cpp файлов значения этих параметров будут добавлены в код.

Для запуска сгенерированных программ на языке C++ под управлением операционной системы Windows используются компилятор MinGW и make-файлы. При этом в коде Java-программы необходимо сформировать команду для компилятора, а также выполнить запуск полученного в результате работы компилятора исполняемого файла.

### 3.3 Механизм работы приложения

Для получения многопоточных реализаций применяются реализованные на C++ программы-шаблоны поиска. Это позволяет сделать свой язык программирования очень простым: пользователю надо всего лишь указать параметры для поиска (используемую библиотеку, количество потоков, ключевое слово для поиска, файлы, в которых необходимо осуществлять поиск, а также вид возвращаемого результата). Весь остальной код генерируется с использованием шаблонов.

Несмотря на то, что при работе приложения код на языке C++ может быть сгенерирован с применением одной из трёх различных библиотек создания потоков, общий принцип работы эти программ будет схожим.

Сначала каждая из программ осуществит просмотр пути для поиска файлов, заданной в качестве константы. Будет выполнена проверка, что путь является корректным, что он существует. Также будет определено, является ли заданный путь – путём к некоторой директории или путём к конкретному файлу. Этот функционал реализован при помощи библиотеки Boost.Filesystem.

После этого будет определён размер файла для поиска или же суммарный размер фалов, если была указана директория. Затем в зависимости от количества потоков, также заданных параметром, будет определено количество байтов, которое должно быть просмотрено каждым из потоков. Естественно, что в данном случае наиболее оптимальным будет распределить всю работу равномерно между всеми потоками. Размер блока для просмотра, а также файлы, которые будет необходимо обработать каждому потоку, хранятся в специальном двумерном массиве. Может возникнуть ситуация, когда 1 файл будут считывать и обрабатывать несколько потоков. Это ситуация встречается, например, при работе с одним большим файлом.

Далее начинается процесс создания потоков. Этот этап различен для библиотек. В случае OpenMP используется специальная прагма для цикла «for», где определяется процесс по-

иска, который должен выполнить каждый поток. В случае библиотек Boost.Thread и TinyThread в программе заданы специальные функции и структуры. Функция определяет работу, которую должен выполнять каждый поток, а структура – параметры, которые будут переданы в функцию.

В процессе поиска потоком необходимого ключевого файла в выделенных ему файлах формируется специальный map. Ключом для этого этого словаря будет имя файла, где было найдено слово, а значением – количество раз, которое это слово встретилось в файле.

Результат поиска выводится в файл следующим образом:

```
Time: 5.2567
The number of occurences:
D:\CP\in\input1txt -> 10
D:\CP\in\input 2.txt -> 16
D:\CP\in\input3.txt -> 5
```

# ЗАКЛЮЧЕНИЕ

На сегодняшний день существует ряд систем автоматического распараллеливания. Подавляющее большинство этих систем являются дорогими коммерческими продуктами. Мы лишены даже возможности провести их сравнительный анализ.

Было представлено описание той работы, которую мы проделали, чтобы помочь пользователю в разработке параллельного приложения. Были разработаны программные инструменты, позволяющие упростить разработку многопоточных программ. Инструменты позволяют автоматически получить параллельную программу одним из двух способов:

- из последовательной программы посредством расстановки директив OpenMP;
- задать параметры задачи поиска и получить параллельные реализации в нескольких вариантах.

В данной работе мы сделали первый шаг по пути «автоматизации распараллеливания программ». Успешный или нет – судить читателю. Мы собираемся идти по нему дальше, к осуществлению «мечты об автоматическом преобразовании хорошего последовательного кода в эффективный параллельный код».

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Средства разработки параллельных программ [Электронный ресурс]. – Режим доступа: http://parallel.ru/tech/tech_dev/. – Дата доступа: 13.10.2016.
2. Средства создания и проектирования параллельных программ [Электронный ресурс]. – Режим доступа: http://parallel.ru/tech/tech_dev/build_par.html. – Дата доступа: 13.10.2016.
3. Средства автоматического распараллеливания [Электронный ресурс]. – Режим доступа: https://parallel.ru/tech/tech_dev/auto_par.html. – Дата доступа: 13.10.2016.
4. List of C++ multi-threading libraries [Electronic resource] – Mode of access: https://en.wikipedia.org/wiki/List_of_C%2B%2B_multithreading_libraries – Date of access: 13.10.2016.
5. Антонов, А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие. / А.С. Антонов. – М.: Изд-во МГУ, 2009. – 77 с.
6. Гергель, В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. / В.П. Гергель. – Нижний Новгород.: Изд-во НГУ, 2010. – 421 с.
7. ANTLR [Электронный ресурс]. – Режим доступа: http://www.antlr.org/. – Дата доступа: 13.10.2016.
8. Ахо, А. В., Лам, Моника С., Сети, Рави, Ульман, Джеффри Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд. / А. В. Ахо, М. С. Лам, Р. Сети, Дж. Д. Ульман. – М.: Вильямс, 2008. – 1184 с.
9. Intel Developer Zone // Введение в технологии параллельного программирования [Электронный ресурс]. – 2014. – Режим доступа: https://software.intel.com/ru-ru/articles/writing-parallel-programs-a-multi-language-tutorial-introduction/. – Дата доступа: 21.11.2015.
10. Хьюз, К., Хьюз, Тр. Параллельное и распределенное программирование с использованием C++. / К. Хьюз, Тр. Хьюз. – М.: ООО “И.Д. Вильямс”, 2004. – 672 с.

# СПИСОК ПУБЛИКАЦИЙ

Львович, В.Д., Перемотова А.Н.  К вопросу об автоматизации распараллеливания программ // Сборник работ 73 научн. конф. студ. и асп. БГУ. Минск: БГУ. Т.1. 2016. – С. 71-74.