На правах рукописи

ГАРЕЕВ Роман Альбертович

МЕТОДЫ ОПТИМИЗАЦИИ ВЫПОЛНЕНИЯ ТЕНЗОРНЫХ ОПЕРАЦИЙ НА МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

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

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Челябинск — 2021

Работа выполнена в Федеральном государственном автономном образовательном учреждении высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина».

Научный руководитель: АКИМОВА Елена Николаевна, доктор физ.-мат. наук, доцент, ведущий научный сотрудник федерального государственного бюджетного учреждения науки «Институт математики и механики имени Н.Н. Красовского Уральского отделения Российской академии наук» (г. Екатеринбург)

Официальные оппоненты: СОКОЛОВ Андрей Владимирович, доктор физ.-мат. наук, профессор, ведущий научный сотрудник федерального государственного бюджетного учреждения науки Федеральный исследовательский центр «Карельский научный центр Российской академии наук» (г. Петрозаводск)

ЧЕРНЫХ Игорь Геннадьевич, кандидат физ.-мат. наук, заведующий лабораторией суперкомпьютерного моделирования федерального государственного бюджетного учреждения науки «Институт вычислительной математики и математической геофизики Сибирского отделения Российской академии наук» (г. Новосибирск)

Ведущая организация: Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (г. Нижний Новгород)

Защита состоится «31» марта 2021 г. в 12:00 час. на заседании диссертационного совета Д 212.298.18 при Южно-Уральском государственном университете по адресу: 454080, г. Челябинск, пр. Ленина, 76, ауд. 1007.

С диссертацией можно ознакомиться в библиотеке Южно-Уральского государственного университета и на сайте: http://www.susu.ru/ru/dissertation/d-21229818/gareev-roman-albertovich.

Автореферат разослан «____» _____ 2021 г.

Ученый секретарь диссертационного совета, доктор физ.-мат. наук, доцент М.Л. Цымблер

Общая характеристика работы

Актуальность темы. Тензорные операции имеют широкое применение в различных научных дисциплинах. В частности, операция свертывания или свертки тензоров применяется в машинном обучении, спектральных методах и квантовой химии. Тензорные операции используются при решении дифференциальных и интегральных уравнений, описывающих многие прикладные задачи. После дискретизации дифференциальные и интегральные уравнения сводятся к системам линейных и нелинейных уравнений. Для решения таких систем используются численные методы, применяющие матричные и матрично-векторные произведения, которые могут рассматриваться как частные случаи свертки тензоров.

Большинство подходов, используемых для сокращения времени выполнения операций над тензорами, основаны на *ручной настройке* (*manual-tuning*) и *автонастройке* (*auto-tuning*), требующих доступа к целевой аппаратной платформе и потенциально больших временных затрат. Вследствие этого ручная настройка и автонастройка неприменимы во время оптимизаций, выполняемых промышленными компиляторами по умолчанию во время кросс-компиляции. Ручная настройка выполняется специалистом при реализации программы. В случае автонастройки для каждой архитектуры процессора наилучшие параметры реализации находятся перебором и измерением времени выполнения программы.

Автонастройка и ручная настройка необходимы, если интерфейс BLAS (Basic Linear Algebra Subprograms) неэффективен или неприменим. Интерфейс BLAS не описывает ряд операций: свертки тензоров; обобщения матричного и матрично-векторных произведений на замкнутые полукольца, использующиеся для сокращения времени выполнения решений задач о путях; матрично-векторные операции для целочисленных типов данных. Эффективные реализации таких операций недоступны для многих архитектур. В описанных случаях могут применяться алгоритмы для сокращения времени выполнения, использующие модель целевой архитектуры процессора (ЦАП). Такие методы могут использоваться в процессе компиляции, ограниченной по времени, создавая высокопроизводительные реализации программ для решения задач большой размерности без ручной настройки, автонастройки и без доступа к целевой аппаратной платформе.

Оптимизации, использующие модель ЦАП, могут применяться в процессе компиляции, ограниченной по времени, с целью автоматического получения высокопроизводительных многопоточных реализаций тен-

зорных операций для многоядерных процессоров общего назначения с архитектурами x86-64, x86, ppc64le, aarch64 и др. без доступа к целевой аппаратной платформе.

Степень разработанности темы. Для уменьшения времени решения задач на больших сетках используются распараллеливание алгоритмов и многопроцессорные системы. Проблемам исследования и распараллеливания алгоритмов при решении прикладных задач посвящены работы В.В. Воеводина, Дж. Ортеги, В.Н. Фаддеевой и Д.К. Фаддеева, В.П. Ильина, Б.Н. Четверушкина, Е.Н. Акимовой, В.П. Гергеля, М.В. Якобовского, В.Н. Алеевой, Ю.Я. Болдырева, Л.Б. Соколинского, С.М. Абрамова, Б.Я. Штейнберга, Б.М. Глинского, М.Л. Цымблера, В.Э. Малышкина и др.

Существенный вклад в развитие тензорного исчисления внесли Дж. Риччи (Gregorio Ricci), Т. Леви-Чивита (Tullio Levi-Civita), А. Эйнштейн (Albert Einstein), М. Гроссман (Marcel Grossmann), В. Фойгт (Woldemar Voigt), Я. Схоутен (Jan Schouten), И.С. Сокольников, П.К. Рашевский, В.Ф. Каган, И.Н. Векуа, Б.Е. Победря, В.В. Лохин, Ф.И. Федоров, Н.Г. Ченцов, С.Г. Лехницкий, М.П. Шаскольская, Е.Е. Тыртышников, И.В. Оселедец.

Цель и задачи исследования. *Целью* данной работы является совершенствование методов обработки информации, получаемой со сложных систем, путем сокращения времени выполнения многопоточных реализаций тензорных операций на многоядерных процессорах общего назначения без ручной настройки и автонастройки.

Для достижения этой цели требовалось выполнить следующие *задачи*.

1. Разработать модель целевой архитектуры процессора с целью сокращения времени выполнения матрично-векторных операций и их обобщений на замкнутые полукольца с элементами из множества вещественных чисел.
2. Разработать новые алгоритмы выполнения тензорных операций константной сложности относительно размерности тензоров, уменьшающие время выполнения таких операций.
3. Разработать программную систему для автоматической оптимизации времени выполнения тензорных операций и их автоматического распараллеливания при компиляции программ для многоядерных процессоров общего назначения.

4. Провести вычислительные эксперименты, подтверждающие эффективность разработанной программной системы по сравнению с аналогами, использующими ручную настройку и автонастройку.

Соответствие паспорту специальности. Основными задачами, решавшимися в диссертации, являются разработка моделей, методов и алгоритмов для сокращения времени выполнения тензорных операций на многоядерных процессорах и их автоматическое распараллеливание, что соответствует п. 8 паспорта специальности 05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей.

Методология и методы исследования. В диссертационной работе использован математический аппарат тензорного исчисления и теории лингвистических основ информатики. Для разработки программной системы автоматической оптимизации времени выполнения тензорных операций применялась методология объектно-ориентированного программирования и технология параллельного программирования OpenMP.

Научная новизна. Научная новизна работы состоит в разработке совокупности научно обоснованных технических решений, направленных на автоматическое получение высокопроизводительных многопоточных реализаций тензорных операций в процессе компиляции, ограниченной по времени, без доступа к целевой архитектуре многоядерного процессора общего назначения.

В процессе исследований получены следующие научные результаты.

1. Разработано расширение модели целевой архитектуры процессора Лоу, которое позволяет сократить время выполнения матрично-векторных операций и их обобщений на замкнутые полукольца с элементами из множества вещественных чисел.
2. Разработаны новые алгоритмы выполнения тензорных операций константной сложности относительно размерности тензоров, уменьшающие время выполнения таких операций. С помощью моделирования выполнения представленных алгоритмов на целевой архитектуре процессора выведены новые формулы, позволяющие получить значения параметров алгоритмов выполнения тензорных операций в зависимости от характеристик многоядерных процессоров общего назначения для архитектур x86-64, x86, ppc64le, aarch64. Впервые доказаны утверждения о существовании значений параметров, при которых отсутствует простой конвейера векторных инструкций модели целевой архи-

тектуры процессора для представленных алгоритмов при возможности мгновенной загрузки данных из памяти на векторные регистры.

3. Разработана новая программная система для автоматической оптимизации времени выполнения тензорных операций и их автоматического распараллеливания при компиляции программ для многоядерных процессоров общего назначения. Получена новая оценка производительности многопоточной программы.

Теоретическая и практическая значимость. Разработанные модель ЦАП и алгоритмы выполнения тензорных операций способствуют созданию новых методов сокращения времени выполнения операций линейной алгебры. Результаты, изложенные в диссертации, могут быть эффективно использованы при реализации численных решений задач математической физики, математической геофизики, механики, квантовой химии на многоядерных процессорах общего назначения.

Положения, выносимые на защиту. На защиту выносятся следующие новые научные результаты.

1. Разработано расширение модели целевой архитектуры процессора Лоу, которое обеспечивает сокращение времени выполнения матрично-векторных операций и их обобщений на замкнутые полукольца с элементами из множества вещественных чисел.

2. Разработаны новые алгоритмы выполнения тензорных операций константной сложности относительно размерности тензоров, уменьшающие время выполнения таких операций. Выведены новые формулы, позволяющие получить значения параметров алгоритмов выполнения тензорных операций в зависимости от характеристик многоядерных процессоров общего назначения для архитектур x86-64, x86, ppc64le, aarch64.

3. Разработана новая программная система автоматической оптимизации тензорных операций (AOTO) для автоматической оптимизации времени выполнения тензорных операций и их автоматического распараллеливания при компиляции программ для многоядерных процессоров общего назначения. Получена новая оценка производительности многопоточной программы, представленной группой полностью вложенных циклов. Автоматическая оптимизация времени выполнения обобщения матричного произведения внедрена в основной код Polly проекта LLVM.

4. С помощью экспериментов при решении обратной задачи гравиметрии, общей задачи о путях, оптимизации матрично-векторных операций и тензорных сверток подтверждена применимость программной

системы AOTO для оптимизации времени выполнения тензорных операций. Показано, что программная система AOTO по производительности скомпилированного кода существенно превосходит компиляторы общего назначения Clang и GCC и сопоставима со специализированным компилятором ICC, а также с библиотеками Intel MKL, OpenBLAS, BLIS, реализующими матричные и матрично-векторные произведения и с фреймворками TCCG и TBLIS, реализующими свертки тензоров.

Степень достоверности результатов. Результаты исследования подтверждаются данными экспериментов, выполненных в соответствии с общепринятыми стандартами.

Апробация работы. Основные положения диссертационной работы, разработанные модели, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных и всероссийских научных конференциях и семинарах.

1. 46-ая международная молодежная школа-конференция «Современные проблемы математики и ее приложений» (СоПроМат-2015). Екатеринбург, 25—31 января 2015 г. URL: https://sopromat.imm.uran.ru/.
2. 2nd International Workshop on Radio Electronics & Information Technologies (REIT'2017). Yekaterinburg, November 15, 2017. URL: https://reit-rtf.ru/2017b.html.
3. 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). Yekaterinburg, October 25—27, 2019. URL: https://sibircon.ieeesiberia.org.
4. Национальный Суперкомпьютерный Форум (НСКФ-2019). Переславль-Залесский, 26—29 ноября 2019 г. URL: http://2019.nscf.ru.
5. XIV международная конференция «Параллельные вычислительные технологии»(ПаВТ'2020). Пермь, 31 марта—2 апреля 2020 г. URL: http://agora.guru.ru/pavt2020.

Публикации. По теме диссертации опубликовано 7 работ. Работы [1, 2] опубликованы в журналах, включенных ВАК в перечень изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора и кандидата наук. Работы [3–6] опубликованы в изданиях, индексируемых в Scopus и Web of Science. Работа [7] опубликована в издании, индексируемом в РИНЦ.

Личный вклад автора. Все результаты, представленные в данной работе, получены автором лично. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. В работах [1, 4–6] научному руководителю Е.Н. Акимовой принадлежит постановка задачи, Р.А. Гарееву принадлежат все полученные результаты. В работе [6] Р.А. Гарееву принадлежат разделы 1, 2, 4, 5 (введение, описание метода автоматической оптимизации матрично-векторного произведения, описание результатов экспериментов, заключение, стр. 1—5), В.Е. Мисилову принадлежит раздел 3 (описание алгоритма решения обратной задачи гравиметрии, стр. 3—4). В работе [3] Р.А. Гарееву принадлежат разделы 1, 3, 4.1—4.3, 4.5, 4.6, 5—7 (введение, определение TC-подобного ядра и алгоритм его автоматической оптимизации, определение расширения модели ЦАП Лоу, описание результатов экспериментов, обзор работ, заключение, стр. 34:1—34:27), Т. Гроссару (Tobias Grosser) принадлежит раздел 2 (описание полиэдрального представления программы, стр. 34:3—34:4), М. Крузу (Michael Kruse) принадлежит раздел 4.4 (инфраструктура для модификации функций доступа к памяти полиэдрального представления программы, стр. 34:11).

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения и библиографии. В приложении 1 приведены основные обозначения, используемые в диссертации. Общий объем диссертации составляет 179 страниц, включая 37 рисунков, 34 таблицы. Библиография содержит 177 наименований.

Содержание работы

Во введении обосновывается актуальность темы проведенных исследований и дан обзор публикаций, близких к теме диссертации. Во введении сформулирована цель работы, научная новизна и практическая значимость результатов, кратко изложено содержание работы.

Первая глава, «Обзор известных подходов по сокращению времени выполнения тензорных операций», посвящена обзору известных подходов сокращения времени выполнения TC (Tensor Contraction). Описываются методы сокращения времени выполнения TC, использующие ручную настройку, автонастройку и моделирование вычислений на ЦАП. Рассматриваются способы сведения сокращения времени выполнения TC к сокращению времени выполнения MMM (matrix-matrix multiplication) и MVM (matrix-vector multiplication).

Во второй главе, «Модели, методы и алгоритмы автоматического сокращения времени выполнения тензорных операций», представлено определение расширения модели ЦАП Лоу с инструкциями предвыборки и операциями из замкнутых полуколец с элементами из множества вещественных чисел. Представлены новые алгоритмы для вычисления обобщений MMM и MVM на замкнутые полукольца. С помощью моделирования выполнения описанных алгоритмов на ЦАП выведены формулы, позволяющие получить значения параметров описанных алгоритмов в зависимости от характеристик многоядерных процессоров общего назначения. Разработан оригинальный алгоритм автоматического сокращения времени выполнения тензорных операций.

Рассмотрим определения замкнутого полукольца и операции MMA, используемые для описания новой модели программной системы.

Определение 1. *Замкнутым полукольцом* (*closed semiring*) называется алгебра $\{S, \oplus, \otimes, 0, 1\}$, где S — множество элементов; $\oplus$ и $\otimes$ — бинарные операции над S, обладающие следующими свойствами.

1. $\langle S, \oplus \rangle$ — идемпотентный коммутативный моноид.
2. $\langle S, \otimes \rangle$ — моноид.
3. 0 служит аннулятором для $\langle S, \otimes \rangle$: $a \otimes 0 = 0 \otimes a = 0$.
4. Умножение дистрибутивно относительно сложения.
5. Если $a_1, a_2, ..., a_i, ...$ — счетная последовательность элементов из S, то сумма $a_1 \oplus a_2 \oplus ... \oplus a_i \oplus ...$ существует и единственна. Ассоциативность, коммутативность и идемпотентность выполняется для бесконечных сумм. Операция $\otimes$ дистрибутивна относительно счетных сумм.

Определение 2. MMA[$\otimes$, $\oplus$] (Multiply-and-Add) — операция вида $C \leftarrow C \bar{\oplus} A \bar{\otimes} B$, где $\bar{\oplus}$ и $\bar{\otimes}$ — операции из замкнутого полукольца матриц $\{S^{N \times N}, \bar{\oplus}, \bar{\otimes}, \bar{O}, \bar{I}\}$, определенного над замкнутым полукольцом $\{S, \oplus, \otimes, 0, 1\}$.

Расширение модели ЦАП Лоу описывается следующим образом.

1. **Архитектура загрузки/сохранения и векторные регистры:** данные должны быть загружены в регистры процессора перед тем, как с ними могут быть выполнены вычисления. N_{REG} — количество векторных регистров. N_{VEC} — количество значений размерности S_{DATA}, которые может содержать векторный регистр. Если N_{REG} равно 0, N_{VEC}

присваивается значение 1. L_{VLOAD} — количество тактов процессора, которое должно быть совершено перед началом выполнения новой зависимой по данным векторной инструкции загрузки данных на векторный регистр из L_1. Предполагается, что команды для работы с памятью могут выполняться одновременно с командами арифметики с плавающей точкой.

2. **Векторные инструкции:** N_{VMMA} — количество операций VMMA, вычисляемых процессором за один такт. Отдельная VMMA выполняет N_{VEC} невекторных умножений и сложений, составляющих MMA. N_{VFMA} определяет пропускную способность процессора. L_{VMMA} — минимальное количество тактов, которое должно быть совершено перед началом выполнения новой зависимой по данным VMMA. L_{VMMA} определяет задержку VMMA-инструкции.
3. **Кэш-память:** весь кэш данных — множественно-ассоциативный кэш. Каждый уровень кэша L_i характеризуется следующими параметрами: S_{L_i} — размер L_i; W_{L_i} — степень ассоциативности; N_{L_i} — количество множеств; C_{L_i} — размер линии кэша, где $S_{\mathrm{L}_i} = N_{\mathrm{L}_i} C_{\mathrm{L}_i} W_{\mathrm{L}_i}$.
4. **Инструкции предвыборки:** N_{prefetch} — количество инструкций предвыборки, которое может быть выполнено за один такт. L_{prefetch} — количество тактов, составляющих задержку каждой инструкции. Каждая инструкция может загрузить данные, размер которых равен $C_{\mathrm{L_i}}$.

Расширение модели ЦАП Лоу является абстрактным описанием ЦАП и его принципов работы. Она используется для отображения реализуемого алгоритма на произвольный реальный процессор общего назначения и определения значений параметров алгоритмов. В рамках данной работы рассматривается расширение модели ЦАП Лоу, описывающее ЦАП с политикой вытеснения LRU (Least Recently Used), которая может не соответствовать реальному процессору.

Предложен алгоритм 1 вычисления MMA. На вход алгоритма подаются матрицы A, B и C. Матрица C содержит результат выполнения MMA. Во время выполнения алгоритма матрицы A и B разбиваются на блоки, как показано на рис. 1. Это позволяет многократно использовать элементы блоков, хранящихся в кэш-памяти. Размеры блоков N_c, K_c, M_c, N_r и M_r являются параметрами алгоритма. С целью обеспечения последовательного доступа к операндам MMA создаются выравненные в памяти массивы A_c и B_c, хранящие элементы перемножаемых матриц.

Алгоритм 1 вычисления MMA имеет следующий вид:

Алгоритм 1: Вычисление ММА.

```
1  begin
2      for j = 0...N − 1 step N_c do
3          for p = 0...K − 1 step K_c do
4              Копирование B(p:p + K_c − 1, j:j + N_c − 1) в B_c
5              for i = 0...M − 1 step M_c do
6                  Копирование A(i:i + M_c − 1, p:p + K_c − 1) в A_c
7                  for j_c = 0...N_c − 1 step N_r do
8                      for i_c = 0...M_c − 1 step M_r do
9                          for p_c = 0...K_c − 1 step 1 do
10                             𝕀 = i_c : i_c + M_r − 1, 𝕁 = j_c : j_c + N_r − 1
11                             C_c(𝕀, 𝕁) ⊕= A_c(𝕀, p_c) ⊗ B_c(p_c, 𝕁)
12                         end
13                     end
14                 end
15             end
16         end
17     end
18 end
```

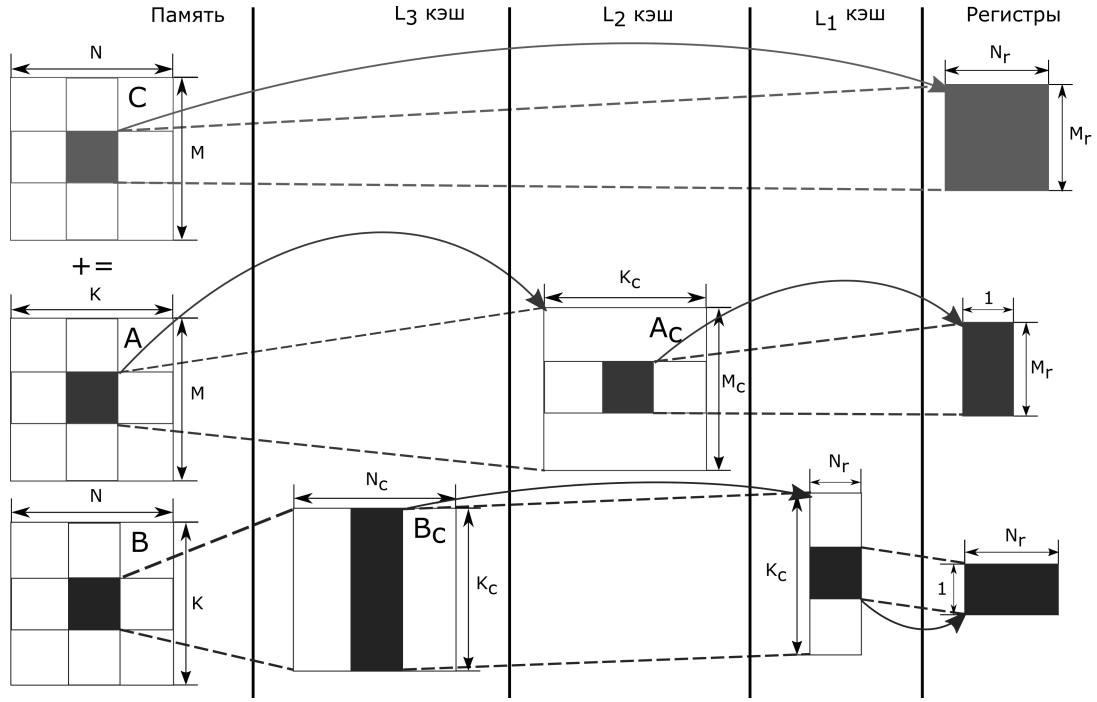

Рис. 1. Разбиение матриц A, B и C на блоки.

Выведены формулы, связывающие размеры блоков матриц с техническими характеристиками реального процессора:

$$N_r = \lceil \sqrt{\gamma}/N_{\text{VEC}} \rceil \, N_{\text{VEC}}, \, M_r = \lceil \gamma/N_r \rceil \, , \, K_c = \left\lfloor \frac{W_{\text{L1}}-1}{1+N_r/M_r} \right\rfloor N_{\text{L}_1} C_{\text{L}_1}/M_r S_{\text{DATA}},$$

где $\gamma = N_{\text{VEC}} L_{\text{VMMA}} N_{\text{VMMA}}$, $M_c = (W_{\text{L}_2} - 2) S_{\text{L}_2}/K_c S_{\text{DATA}} W_{\text{L}_2}$,

$N_c = \lfloor C_{B_c}/K_c S_{\text{DATA}} N_r \rfloor \, N_r$, C_{B_c} — количество байтов памяти, доступных для массива B_c.

Формулы для N_r и M_r обеспечивают отсутствие простоя конвейера векторных инструкций расширения модели ЦАП Лоу во время выполнения тела внутреннего цикла алгоритма 1. Формулы для K_c, M_c и N_c основаны на том, что элементы матриц, используемые чаще остальных, должны оставаться в кэш-памяти наименьшего из доступных уровней как можно дольше. Для их вывода рассматриваются элементы матриц, используемые циклами алгоритма, и применяется предположение о том, что кэш-память расширения модели ЦАП Лоу имеет политику вытеснения LRU.

Утверждение 1. Если данные могут быть мгновенно загружены из памяти на векторные регистры, то существуют значения параметров N_r и M_r алгоритма 1, при которых отсутствует простаивание конвейера векторных инструкций расширения модели ЦАП Лоу в процессе выполнения алгоритма 1.

Предложены алгоритмы 2 и 3 вычисления обобщения MVM на замкнутые полукольца с элементами из множества вещественных чисел для случаев транспонированной и нетранспонированной матрицы, соответственно.

На вход алгоритма 2 подаются матрица A, векторы x и y, имеющие размерность $M \times N$, M и N, соответственно. Вектор y содержит результат выполнения MVM, обобщенного на замкнутые полукольца с элементами из множества вещественных чисел. Обобщенное MVM имеет вид $y = A^{\mathsf{T}} \, \bar{\otimes} \, x \, \bar{\oplus} \, y$. При выполнении алгоритма операнды MVM разбиваются на блоки, влияя на предвыборку данных в L_1 и позволяя многократно использовать элементы блоков, хранящихся в кэш-памяти. Тело внутреннего цикла алгоритма 2 вычисляет N_b элементов вектора y, используя последовательность из N_b обобщенных сложений и умножений. Размеры блоков N_c, M_c, N_b, N_r и шаг предвыборки D являются параметрами алгоритма.

Алгоритм 2 вычисления обобщенного MVM для случая транспонированной матрицы имеет следующий вид:

Алгоритм 2: Вычисление обобщенного MVM. Случай транспонированной матрицы.

```
1  begin
2      for j_c = 0...N − 1 step N_c do
3          for i_c = 0...M − 1 step M_c do
4              for j_b = 0...N_c − 1 step N_b do
5                  Выполняем предвыборку M_c × N_b элементов
                    матрицы A с шагом D
                   for i_b = 0...M_c − 1 do
6                      I = i_c + i_b , acc(0:N_r − 1) = x(I)
7                      for j_r = 0...N_b − 1 step N_r do
8                          J = j_c + j_b + j_r : j_c + j_b + j_r + N_r − 1
9                          y(J) ⊕= A(I, J) ⊗ acc(0 : N_r − 1)
10                     end
11                 end
12             end
13         end
14     end
15 end
```

Выведены формулы, связывающие размеры блоков матрицы и векторов, а также шаги предвыборки с техническими характеристиками реального процессора:

$$N_b = \min\left\{ \max(N_{\text{VFMA}} L_{\text{VFMA}} N_{\text{VEC}}, (N_{\text{REG}} - 2) N_{\text{VEC}}), \left\lceil \frac{N_{\text{L}_1} C_{\text{L}_1}}{N_{\text{VEC}} S_{\text{DATA}}} \right\rceil N_{\text{VEC}} \right\},$$

$$N_r = N_{\text{VEC}},\ M_c = \min(W_{\text{L}_2} - 1, W_{\text{L}_1}),\ N_c = N_{\text{L}_2} C_{\text{L}_2} / S_{\text{DATA}},$$

$$D = \lceil L_{\text{prefetch}} / (\lceil M_c \lceil N_b S_{\text{DATA}} / C_{\text{L}_1} \rceil / N_{\text{prefetch}} \rceil + + M_c (\lceil N_b / N_{\text{VEC}} N_{\text{VFMA}} \rceil + L_{\text{VLOAD}})) \rceil.$$

Формулы для N_b и N_r выведены с целью обеспечения отсутствия простоя конвейера векторных инструкций расширения модели ЦАП Лоу в процессе выполнения внутреннего цикла алгоритма 2 и отсутствия вытеснения элементов матрицы A, предвыбираемой в L_1. Формулы для M_c,

N_c и D для обоих алгоритмов основаны на том, что элементы матриц, используемые чаще остальных, должны оставаться в кэш-памяти наименьшего из доступных уровней как можно дольше.

На вход алгоритма 3 подаются матрица A, векторы x, y, имеющие размерность $M \times N$, N и M, соответственно. Вектор y содержит результат выполнения MVM, обобщенного на замкнутые полукольца с элементами из множества вещественных чисел. Обобщенное MVM имеет вид $y = A \bar{\otimes} x \bar{\oplus} y$. При выполнении алгоритма операнды MVM разбиваются на блоки, влияя на предвыборку данных в L_1 и позволяя многократно использовать элементы блоков, хранящихся в кэш-памяти. Размеры блоков N_c, M_c, N_r и шаг предвыборки D являются параметрами алгоритма.

Алгоритм 3 вычисления обобщенного MVM для случая нетранспонированной матрицы имеет следующий вид:

Алгоритм 3: Вычисление обобщенного MVM. Случай нетранспонированной матрицы.

```
1  begin
2      for j_c = 0...N − 1 step N_c do
3          for i_c = 0...M − 1 step M_c do
4              acc[M_c][N_r]
5              for j_r = 0...N_c − 1 step N_r do
6                  if j_r mod 4C_L1/S_DATA == 0 then
7                      Предвыборка M_c × 4C_L1/S_DATA элементов A и
                         4C_L1/S_DATA элементов x с шагом D в L_1
8                  end
9              end
10             J = j_c + j_r : j_c + j_r + N_r − 1
11             acc(0, 0 : N_r − 1) ⊕= A(i_c, J) ⊗ x(J)
12             ...
13             acc(M_c − 1, 0 : N_r − 1) ⊕= A(i_c + M_c − 1, J) ⊗ x(J)
14         end
15         y(i_c) ⊕= ⊕_{i=0}^{N_r−1} acc(0, i)
16         ...
17         y(i_c + M_c − 1) ⊕= ⊕_{i=0}^{N_r−1} acc(M_c − 1, i)
18     end
19 end
```

Выведены формулы, связывающие размеры блоков матрицы и векторов, а также шаги предвыборки с техническими характеристиками реального процессора:

$$N_r = N_{\text{VEC}} \left\lceil \gamma / \min(W_{\text{L}_1}, W_{\text{L}_2} - 1) \right\rceil, M_c = \left\lceil \gamma / N_r \right\rceil, N_c = N_{\text{L}_2} C_{\text{L}_2} / S_{\text{DATA}},$$

где $\gamma = N_{\text{VMMA}} L_{\text{VMMA}} N_{\text{VEC}}$.

$$D = \lceil L_{\text{prefetch}} / (\lceil 4(M_c + 1) / N_{\text{prefetch}} \rceil + \\ + 4\mathbb{C}_{\text{L}_1} (\lceil (M_c N_r) / (N_{\text{VFMA}} N_{\text{VEC}}) \rceil + L_{\text{VLOAD}}) / N_r) \rceil.$$

Формулы для M_c и N_r выведены с целью обеспечения отсутствия простоя конвейера векторных инструкций расширения модели ЦАП Лоу в процессе выполнения внутреннего цикла алгоритма 3 и отсутствия вытеснения элементов матрицы A, предвыбираемой в L_1. Формулы для M_c, N_c и D для обоих алгоритмов основаны на том, что элементы матриц, используемые чаще остальных, должны оставаться в кэш-памяти наименьшего из доступных уровней как можно дольше.

Утверждение 2. Если данные могут быть мгновенно загружены из памяти на векторные регистры, то существуют значения параметров N_r, N_b и значения параметров M_c и N_r, позволяющие избежать простаивания конвейера векторных инструкций расширения модели ЦАП Лоу во время выполнения алгоритма 2 и цикла с индуктивной переменной i_c алгоритма 3, соответственно.

Автором вводится определение TC-подобного ядра, частным случаем которого является TC. TC-подобное ядро удовлетворяет требованиям полиэдральной модели, которая используется для моделирования и оптимизации доступа к памяти, производимого в гнездах циклов, не рассматривая отдельные вычисления.

Определение 3. TC-подобное ядро (англ. Tensor Contraction like kernel, сокр. *TC-like kernel*) — множество полностью вложенных циклов.

1. Ядро удовлетворяет требованиям полиэдральной модели.
2. Без ограничения общности TC-подобное ядро содержит три непустых множества циклов, имеющих одну индуктивную переменную и единичный шаг. Данные множества образуют три непустых набора $I = i_0 \dots i_{r-1}$, $J = j_0 \dots j_{s-1}$ и $P = p_0 \dots p_{t-1}$.
3. Тело цикла TC-подобного ядра, имеющего наибольшую глубину, может быть представлено в виде $\text{C}_{\pi_C(IJ)} = \text{E}(\text{A}_{\pi_A(IP)}, \text{B}_{\pi_B(PJ)})$, где $\text{A}_{\pi_A(IP)}$,

$\mathrm{B}_{\pi_B(PJ)}$, $\mathrm{C}_{\pi_C(IJ)}$ — обращения к тензорам A, B, C, соответственно; $\pi_C(IJ)$, $\pi_A(IP)$ и $\pi_B(PJ)$ — перестановки индексов; E — выражение, содержащее чтения из тензоров $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ и произвольное количество чтений из констант.

Построен алгоритм 4 сокращения времени выполнения TC с использованием алгоритмов вычисления обобщенных MMM и MVM, описанных в предыдущих разделах. В ходе выполнения алгоритма к полиэдральному представлению программы применяются оптимизации циклов.

Алгоритм 4 оптимизации времени выполнения TC-подобного ядра имеет следующий вид:

Алгоритм 4: Оптимизация TC-подобного ядра.

Входные данные:

Утверждение S полиэдрального представления программы. S является частью TC-подобного ядра и, вследствие этого, представимо в виде C[i][j] = E(A[i][p], B[p][j], C[i][j]), где i, j, p — индукционные переменные циклов; A[i][p], B[p][j], C[i][j] — обращения к матрицам A, B, C, соответственно; E — выражение, содержащее операции чтения из матриц A, B, C.

1. Отождествить каждый цикл с его индукционной переменной.
2. Переставить циклы так, чтобы i, j, p приобрели наибольшую глубину и следующий порядок: j, p и i, где i имеет наименьшую глубину.
3. Разбить i, j, p на блоки размера M_c, N_c, K_c, соответственно; получить циклы i_c, j_c и p_c; переставить i_c и p_c.
4. Разбить i_c, j_c, p_c на блоки размера M_r, N_r и 1, соответственно; получить циклы i_r, j_r, p_r; удалить p_r.
5. Выделить безусловные области итерирования i_r, j_r и выполнить их размотку.
6. Выполнить копирование элементов матриц A и B во временные массивы A_c , B_c и векторизовать код в p_c.

Выходные данные: Оптимизированный код.

Когда размер одного из наборов индексов I и J TC-подобного ядра равен 1, аналогичным способом могут быть получены оптимизированные формы обобщенного MVM, представленного в разделе 2.3.

Третья глава, «Программная система автоматической оптимизации тензорных операций», посвящена разработке программной системы ПС АОТО на основе моделей, алгоритмов и формул, предложенных во второй главе. ПС АОТО автоматически сокращает время выполнения тензорных операций без доступа к целевой аппаратной платформе и без выполнения автонастройки и ручной настройки. Приводится описание архитектуры ПС АОТО. Рассматриваются возможные реализации ПС АОТО. Предложен метод автоматического распараллеливания программ ПС АОТО на многоядерных процессорах общего назначения с общей памятью.

ПС АОТО является приложением командной строки. В качестве аргументов командной строки принимаются значения выделенных параметров расширения модели ЦАП Лоу. Исходные тексты созданных программ доступны по адресам **http://github.com/llvm/llvm-project.git** и **http://bitbucket.org/gareevroman/polly-groman-fork.git**.

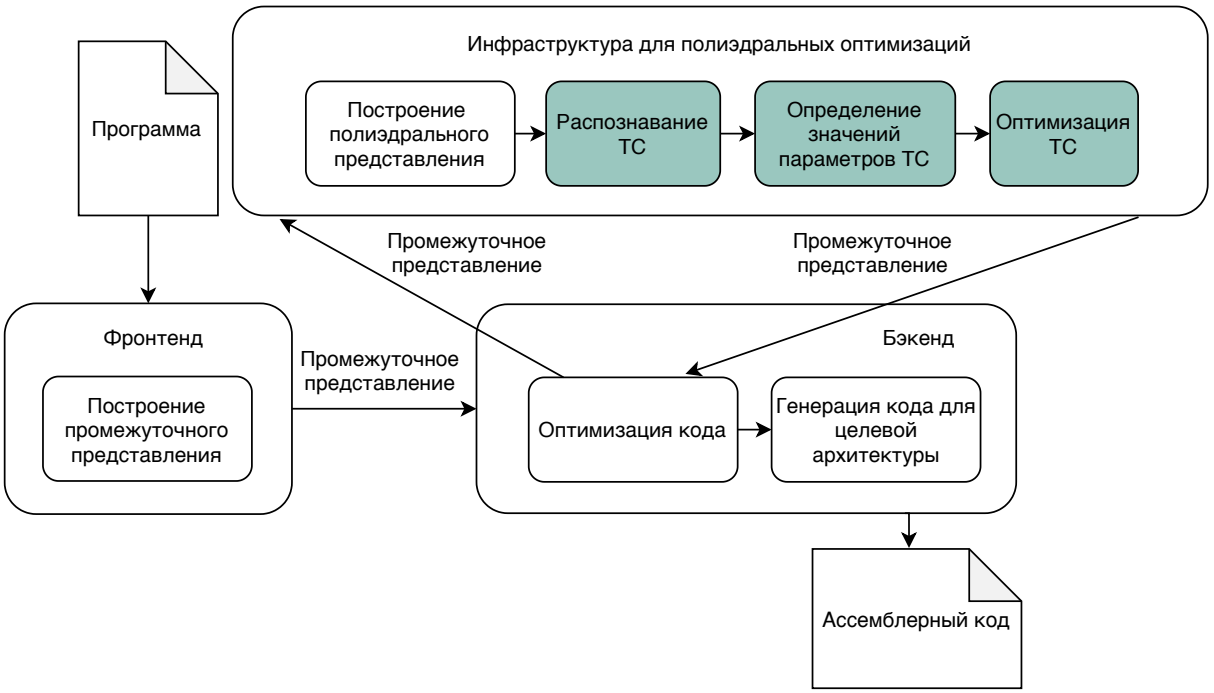

Рис. 2. Диаграмма программной системы; темным цветом выделены части, созданные в рамках данной работы.

ПС АОТО имеет три взаимодействующие между собой части, представленные на рис 2.

1. Фронтенд, выполняющий лексический, синтаксический и семантический анализ программы, а также построение промежуточного кода, являющегося программой в кодах виртуальной машины на языке эквивалентном исходному.

2. Инфраструктура для полиэдральных оптимизаций, выполняющая построение полиэдрального представления и его модификации, такие как оптимизация циклов и доступов к памяти. В результате работы инфраструктуры для полиэдральных оптимизаций создается промежуточный код, соответствующий оптимизированному полиэдральному представлению. В рамках данной работы была создана и реализована оптимизация полиэдрального представления программы, позволяющая распознать TC-подобное ядро, определить значения параметров реализации TC-подобного ядра и автоматически оптимизировать время его выполнения.
3. Бэкенд, выполняющий оптимизацию промежуточного кода программы с использованием низкоуровневых и высокоуровневых оптимизаций. Для выполнения оптимизаций, в частности, применяется инфраструктура для полиэдральных оптимизаций. В дальнейшем бэкенд генерирует ассемблерный код для одной из поддерживаемых целевых аппаратных платформ.

В рамках данной работы в качестве фронтенда, выполняющего компиляцию программы, использован фронтенд Clang. Фреймворк Polly применен в качестве инфраструктуры для построения полиэдрального представления и в качестве основы для реализации распознавания TC-подобного ядра и его оптимизации, описанных в предыдущей главе. **Для случая MMM, обобщенных на замкнутые полукольца с элементами из множества вещественных чисел, указанная оптимизация была внедрена в основной код Polly проекта LLVM.** Внешняя библиотека LLVM Core использована в качестве бэкенда, выполняющего оптимизацию промежуточного кода программы и генерацию ассемблерного кода для ЦАП (рис. 2).

Для компиляции программ с помощью представленной в рамках данной работы реализации ПС AOTO нужно запустить фронтенд Clang с нужными опциями и значениями параметров целевой аппаратной платформы. ПС AOTO принимает на вход следующие значения параметров расширения модели ЦАП Лоу для многоядерного процессора общего назначения: W_{L_i}, S_{L_i}, N_{VMMA} и L_{VMMA}. Остальные параметры расширения за исключением L_{prefetch} и N_{prefetch} выводятся ПС AOTO автоматически, используя, в том числе доступную в LLVM Core информацию о целевой аппаратной платформе. Поддержка оптимизации времени выполнения MVM, обобщенного на замкнутые полукольца с элементами из множества вещественных чисел, не является целью данной работы. Ее реализация планируется в будущих версиях ПС AOTO.

Рассматривается автоматическое распараллеливание программ, выполняемое ПС АОТО. По умолчанию инфраструктура для полиэдральных оптимизаций определяет самый внешний цикл группы вложенных циклов, распараллеливание которого не нарушает зависимости по данным. Получена оценка многопоточной производительности программы, представленной группой полностью вложенных циклов.

Утверждение 3. Если цикл L содержится в группе полностью вложенных циклов, то значение многопоточной производительности программы (ГФлоп/сек) может быть вычислено как $F(L)C_{\mathrm{R}}$, где

$$F(L) = N_{\mathrm{O}}/(C_{\mathrm{PAR}} + C_{\mathrm{W}}C_{\mathrm{R}});$$

L — оцениваемый цикл; N_{O} — количество операций с плавающей запятой, вычисляемых циклом L; C_{PAR} — количество тактов процессора, требуемых для создания группы из N_{THREADS} потоков; C_{W} — количество секунд, требуемых для выполнения цикла L после распараллеливания; C_{R} — тактовая частота процессора.

На практике используются формулы для приближенного вычисления значений N_{O} и $C_{\mathrm{W}}C_{\mathrm{R}}$. Для вывода формул используются характеристики векторных инструкций, описываемых расширением модели ЦАП Лоу.

Автоматическое распараллеливание программ на основе представленной формулы планируется внедрить в ПС АОТО.

Четвертая глава, «Оценка эффективности разработанных алгоритмов», посвящена оценке эффективности ПС АОТО при решении обратной задачи гравиметрии, общей задачи о путях, оптимизации ТС, МММ и MVM. Выполнено сравнение с оптимизированными библиотеками, компиляторами и фреймворками.

Рассмотрено сокращение времени решения трехмерной структурной обратной задачи гравиметрии о восстановлении поверхности раздела между средами по известному скачку плотности $\Delta\sigma$ и гравитационному полю $\Delta g(x, y, 0)$, измеренному на некоторой площади земной поверхности. Предположим, что $z = h$ — асимптотическая плоскость для данной поверхности раздела ζ такая, что $\lim\limits_{x,y\to\pm\infty} \zeta(x, y) = h$. Функция $\zeta = \zeta(x, y)$, описывающая искомую поверхность раздела, удовлетворяет нелинейному двумерному интегральному уравнению Фредгольма первого рода

$$f\Delta\sigma \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty}\left\{\frac{1}{\sqrt{((x-x')^2+(y-y')^2+\zeta^2(x',y'))}}-\right.$$
$$\left.-\frac{1}{\sqrt{((x-x')^2+(y-y')^2+h^2)}}\right\}dx'dy' = \Delta g(x,y,0), \tag{1}$$

где f — гравитационная постоянная.

После дискретизации области $\Pi = \{(x,y) : a \leqslant x \leqslant b, c \leqslant y \leqslant d\}$ на сетке $N \times M : (x_i, y_j), \quad i = 1, ..., N, \quad j = 1, \ldots, M$ с шагами Δx и Δy и аппроксимации интегрального оператора A по квадратурным формулам имеем систему нелинейных уравнений в операторном виде $A(z) = F$.

Для решения системы будем использовать метод Левенберга—Марквардта

$$z^{k+1} = z^k - \gamma\left[\left(A'(z^k)\right)^{\intercal} A'(z^k) + \alpha I\right]^{-1} \times A'(z^k)^{\intercal}(A'(z^k) - F), \tag{2}$$

где z^k — приближенное решение на k-ой итерации, $A'(z^k)$ — производная Фреше оператора A в точке u^k, I — единичный оператор, γ — демпфирующий множитель, α — параметр регуляризации.

В качестве начального приближения используется $z^0 \equiv h$. Критерием останова метода (2) является условие $\left\|A(z^k) - F\right\| / \|F\| \leqslant \varepsilon_1$ при некотором $\varepsilon_1 < 0$.

На каждом шаге итерационного метода Левенберга—Марквардта имеем систему линейных алгебраических уравнений

$$Bz = b,$$

где $B = \left(A'(z^k)\right)^{\intercal} A'(z^k) + \alpha I,$

$b = \left[\left(A'(z^k)\right)^{\intercal} A'(z^k) + \alpha I\right] z^k - \gamma A'(z^k)^{\intercal}(A'(z^k) - F).$

Для решения СЛАУ (3) используется метод минимальных невязок

$$z^{l+1} = z^l - \frac{\langle B(Bz^l - b), Bz^l - b\rangle}{\|B(Bz^l - b)\|^2}(Bz^l - b), \tag{3}$$

где z^l — приближенное решение на l-ой итерации метода минимальных невязок.

Критерием останова (3) является условие $\left\|Bz^l - b\right\| / \|b\| \leqslant \varepsilon_2$ при некотором $\varepsilon_2 < 0$.

На каждом шаге итерационного процесса (2) с использованием метода минимальных невязок (3) вычисляется большое количество MMM и MVM. Для сокращения их времени выполнения может применяться ПС AOTO.

Проведено сравнение реализации решения задачи гравиметрии на основе предложенного автором алгоритма вычисления MVM с реализациями на основе MVM оптимизированных библиотек. Эксперименты показали, что в среднем результаты предложенного алгоритма отличаются не более чем на 1% от реализации на основе Intel MKL и превосходят библиотеки OpenBLAS и BLIS.

Выполнено сравнение однопоточной и многопоточной производительностей ПС AOTO с библиотеками, содержащими оптимизированную реализацию MMM для различных типов данных, и компиляторами. Выполнена оценка погрешности автоматической векторизации LLVM Core в реализации ПС AOTO для случая MMM.

В результате экспериментов по сравнению однопоточной производительности установлено, что ПС AOTO достигает 1.63-кратного ускорения по сравнению с компилятором ICC; 20-кратного ускорения по сравнению с фронтендом для компиляции Clang и компиляторами GCC, IBM XLC; 83.3% производительности рассмотренных библиотек Intel MKL, ARMPL, OpenBLAS и BLIS. Установлено, что ПС AOTO достигает 86.2% многопоточной производительности рассмотренных библиотек Intel MKL, OpenBLAS и BLIS.

Выполнено сравнение однопоточной и многопоточной производительностей ПС AOTO с компиляторами для оптимизации времени выполнения решений общей задачи о путях, применяющих обобщение матричного произведения на замкнутые полукольца с элементами из множества вещественных чисел.

Рассмотрим постановку общей задачи о путях. Предположим, что $G = (V, E, w)$ — взвешенный ориентированный граф, где $V = \{1, 2, ..., N\}$ — множество вершин, $E \subseteq V \times V$ — ребра графа и $w : E \to S$ — весовая функция, определенная над замкнутым полукольцом $\{S, \oplus, \otimes, *, 0, 1\}$.

Вес пути $p = \langle i, k_1, k_2, ..., k_m, j \rangle$ из вершины i в вершину j равен произведению всех входящих в него ребер. Вес путей длины 0, начинающихся и заканчивающихся в одной и той же вершине, определяется как $1 \in S$. $w(i, j) = 0$, если $(i, j) \notin E$. Задача состоит в нахождении суммы весов всех путей из вершины i в вершину j для всех i и j.

Рассмотрим постановку общей задачи о путях в матричной форме. Определим матрицу весов $A = [a_{ij}] \in S^{N \times N}$, где $a_{ij} = w(i, j)$. Определим матрицу $D = [d_{ij}] \in S^{N \times N}$, где d_{ij} — сумма весов всех путей из вершины i в вершину j. D может быть вычислена как:

$$D = \bigoplus_{m \geq 0} A^m = \bar{I} \bar{\oplus} A \bar{\oplus} A^2 \bar{\oplus} A^3 \bar{\oplus} ...$$

Для решения общей задачи о путях может применяться блочная модификация алгоритма Флойда—Уоршелла, вычисляющая большое количество ММА. Если $\forall a \in S \; a \oplus 1 = 1$, то для решения общей задачи о путях может использоваться алгоритм последовательного возведения матрицы весов в квадрат с применением операций ММА.

Проведено сравнение реализаций решения частных случаев общей задачи о путях на основе алгоритма последовательного возведения матрицы весов в квадрат. Эксперименты показали, что ПС АОТО достигает 85% верхней теоретической границы производительности. Для всех рассматриваемых задач ПС АОТО достигла 1.4-кратного ускорения по сравнению с компилятором ICC и 151-кратного ускорения по сравнению с компиляторами Clang и GCC.

Выполнено сравнение однопоточной и многопоточной производительностей реализации предложенных автором алгоритмов с библиотеками, содержащими реализации MVM. В результате экспериментов установлено, что достигается производительность библиотек Intel MKL, OpenBLAS и BLIS. Выполнена оценка значений параметров, полученных с использованием формул, выведенных в разделе 2.

Выполнено сравнение однопоточной и многопоточной производительностей ПС АОТО с фреймворками, позволяющими получить оптимизированную реализацию ТС, и компиляторами. Экспериментально установлено, что ПС АОТО достигает 80-кратного ускорения по сравнению с рассмотренным фронтендом для компиляции Clang и компиляторами GCC, ICC; ПС АОТО сопоставима по производительности скомпилированного кода с фреймворками TCCG и TBLIS.

В заключении в краткой форме излагаются итоги выполненного диссертационного исследования, представляются отличия диссертационной работы от ранее выполненных родственных работ других авторов, даются рекомендации по использованию полученных результатов и рассматриваются перспективы дальнейшего развития темы.

Заключение

В ходе исследования были получены следующие основные результаты. Разработано расширение модели целевой архитектуры процессора Лоу, которое обеспечивает сокращение времени выполнения матрично-векторных операций и их обобщений. Разработаны новые алгоритмы выполнения тензорных операций константной сложности относительно размерности тензоров, уменьшающие время выполнения таких операций. Разработана новая программная система автоматической оптимизации тензорных операций (ПС АОТО) для автоматической оптимизации времени выполнения тензорных операций и их автоматического распараллеливания при компиляции программ для многоядерных процессоров общего назначения. Экспериментально подтверждена применимость ПС АОТО для оптимизации времени выполнения ТС.

Выделим преимущества ПС АОТО по сравнению с библиотеками, реализующими интерфейс BLAS. ПС АОТО использует формулы, позволяющие автоматически получить значения параметров алгоритмов выполнения тензорных операций в зависимости от характеристик многоядерных процессоров общего назначения. ПС АОТО автоматически сокращает время выполнения ТС, не описываемой интерфейсом BLAS. Интерфейс BLAS неприменим для реализации новых методов сокращения времени выполнения ТС, реализованных в ПС АОТО, а также в фреймворках TCCG и TBLIS, поддерживающих малое количество целевых архитектур процессоров (ЦАП). ПС АОТО автоматически сокращает время выполнения ММА, не описываемой интерфейсом BLAS, с целью сокращения времени выполнения решений общей задачи о путях для замкнутых полуколец с элементами из множества вещественных чисел. ПС АОТО применяется для автоматического получения высокопроизводительных реализаций ММТ и MVM для различных ЦАП и типов данных, включая не описываемые интерфейсом BLAS целочисленные типы данных.

Полученные методы могут использоваться для оптимизации программ в процессе компиляции; для упрощения создания новых реализаций BLAS; уменьшения времени выполнения автонастройки.

Основным направлением дальнейших исследований предполагается создание модели архитектуры графического процессора и получение значений параметров алгоритмов. Другим направлением дальнейших исследований предполагается разработка методов автоматического моделирования выполнения алгоритмов с целью автоматического получения значений параметров алгоритмов в зависимости от характеристик ЦАП.

Публикации автора по теме диссертации

Статьи в изданиях из перечня ВАК

1. Акимова Е.Н., Гареев Р.А. Аналитическое моделирование матрично-векторного произведения на многоядерных процессорах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 1. С. 69–82. DOI: 10.14529/cmse200105.
2. Гареев Р.А. Методы оптимизации обобщенных тензорных сверток // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 19–39. DOI: 10.14529/cmse200202.

Статьи в изданиях, индексируемых в SCOPUS, Web of Science

3. Gareev R., Grosser T., Kruse M. High-Performance Generalized Tensor Operations: A Compiler-Oriented Approach // ACM Transactions on Architecture and Code Optimization (TACO). 2018. Vol. 15, no. 3. P. 34:1–34:27. DOI: 10.1145/3235029.
4. Gareev R.A., Akimova E.N. Analytical Modeling of Matrix–Vector Multiplication on Multicore Processors // Mathematical Methods in the Applied Sciences. 2021. P. 1—31. DOI: 10.1002/mma.7045.
5. Akimova E.N., Gareev R.A. Algorithm of Automatic Parallelization of Generalized Matrix Multiplication // Proceedings of the 2nd International Workshop on Radio Electronics & Information Technologies (Ekaterinburg, November 15, 2017). CEUR Workshop Proceedings. 2017. Vol. 2005. P. 1–10.
6. Akimova E.N., Gareev R.A., Misilov V.E. Analytical Modeling of Matrix-Vector Multiplication on Multicore Processors: Solving Inverse Gravimetry Problem // SIBIRCON: 2019 International Multi-Conference on Engineering, Computer and Information Sciences (Novosibirsk, Russia, October 25-27, 2019). Boston, Massachusetts, USA, IEEE Xplore Digital Library, 2019. P. 0823–0827. DOI: 10.1109/SIBIRCON48586.2019.8958103.

Статья в издании, индексируемом в РИНЦ

7. Гареев Р.А. Сравнение средств генерации абстрактного синтаксического дерева из полиэдральной модели в библиотеках CLooG и ISL // Труды 46-й Международной молодежной школы-конференции “Современные проблемы математики и ее приложений - 2015”. Институт математики и механики УрО РАН им. Н.Н. Красовского, 2015. С. 200–202.