Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа радиотехники и компьютерных технологий
Кафедра системного программирования ИСП РАН
Лаборатория (laboratory name)

Выпускная квалификационная работа бакалавра

Исследование и разработка методов машинного обучения

Автор:
Студент 082 группы
Иванов Иван Иванович

Научный руководитель:
научная степень
Денисов Денис Денисович

Научный консультант:
научная степень
Сергеев Сергей Сергеевич

Москва 2023

Аннотация

Исследование и разработка методов машинного обучения
Иванов Иван Иванович

В статье рассматривается работа цикловых оптимизации производительности приложений для процессорной архитектуры RISC-V на примере открытого компиляторного фреймворка LLVM. Обсуждается работа оптимизационного пасса LoopStrengthReduction, возможности его улучшения. Приведены примеры сравнения работы производительности LLVM и GCC. Результаты работы сравниваем на SPEC2016.

Abstract

Research and development of machine learning methods

Оглавление

Глава 1

Введение

В этой части надо описать предметную область, задачу из которой вы будете решать, объяснить её актуальность (почему надо что-то делать сейчас?). Здесь же стоит ввести определения понятий, которые вам понадобятся в постановке задачи.

Статическая компиляция по-прежнему является основным автоматическим способом повышения производительности программ на языках общего назначения (C/C++). Интересно, что актуальность наличия оптимизирующего статического компилятора не только не падает, но и возрастает.

Во-первых, появление новых процессорных архитектур влечет за собой необходимость не только переноса на них существующих промышленных компиляторов (GCC, LLVM), но и разработки специфических и настройки имеющихся оптимизаций (распределения регистров, программной конвейеризации, использования особых возможностей набора команд архитектуры).

Во-вторых, возникает необходимость использовать оптимизации с учетом профиля программы без затрат на инструментирование, с помощью снятия профиля программы через взятие проб (sampling) во время выполнения на нагрузках промышленного типа.

В-третьих, необходима поддержка новых стандартов программирования для эффективного использования современных многоядерных и гетерогенных архитектур (OpenMP, OpenCL, OpenACC, Cilk+). Наконец, все вышеперечисленное имеет смысл делать лишь в современных открытых компиляторных инфраструктурах типа GCC и LLVM, которые развиваются десятки лет и уже содержат все необходимые базовые машинно-зависимые и машинно-независимые оптимизации, а также средства для их создания.

Так как указанные компиляторы являются многоплатформенными, возникают естественные сложности при их использовании на различных архитектурах, таких как X86, ARM и набирающий популярность у нас в стране RISC-V. Особенности архитектур могут приводить к тому, что машинно-независимые оптимизации, в основном разработанные и протестированные на устройствах Intel или IBM Power, будут выполняться неоптимально для других типов устройств. Однако именно развитие таких оптимизаций важно, потому что помогает развивать компиляторы для всех архитектур, что ценится Open Source сообществом.

1.1 Индуктивная переменная.

Введём понятие **индуктивная переменной** - переменная в циклах, последовательные значения которой образуют арифметическую прогрессию. Наиболее распространенный пример — счётчик цикла. Выделяют **базовые** (изменяемая на одну и ту же

величину при итерации цикла) и **зависимые** (имеют более сложную математическую зависимость) индуктивные переменные.

```
for (int i = 0; i < size; ++i) {
    int to_store = 200 + 9 * i;
    mas[i] = to_store;
}
```

Мы должны выявить индукционные переменные, упростить вычисления, связанные с ними (так как переменная изменяется на каждой итерации цикла) и, по возможности, избавиться от большего числа.

Можно заметить, что во многих архитектурах при обращении к адрессу может быть добавлен индекс 2, 4, 8 ... что не увеличивает длительности исполнения и может быть использано в машинно-зависимых оптимизациях. При работе с индукционными переменными мы не должны исключать возможность их работы.

Глава 2

Постановка задачи

Здесь надо максимально формально описать суть задачи, которую потребуется решить, так, чтобы можно было потом понять, в какой степени полученное в результате работы решение ей соответствует. Текст главы должен быть написан в стиле технического задания, т.е. содержать как описание задачи, так и некоторый набор требований к решению

Глава 3

Обзор существующих решений

Здесь надо рассмотреть все существующие решения поставленной задачи, но не просто пересказать, в чем там дело, а оценить степень их соответствия тем ограничениям, которые были сформулированы в постановке задачи.

Глава 4

Исследование и построение решения задачи

Здесь надо декомпозировать большую задачу из постановки на подзадачи и продолжать этот процесс, пока подзадачи не станут достаточно простыми, чтобы их можно было бы решить напрямую (например, поставив какой-то эксперимент или доказав теорему) или найти готовое решение.

Глава 5

Описание практической части

Если в рамках работы писался какой-то код, здесь должно быть его описание: выбранный язык и библиотеки и мотивы выбора, архитектура, схема функционирования, теоретическая сложность алгоритма, характеристики функционирования (скорость/память).

Глава 6

Заключение

Здесь надо перечислить все результаты, полученные в ходе работы. Из текста должно быть понятно, в какой мере решена поставленная задача.

Литература

[1] *Mott-Smith, H.* The theory of collectors in gaseous discharges / H. Mott-Smith, I. Langmuir // *Phys. Rev.* — 1926. — Vol. 28.

[2] *Морз, Р.* Бесстолкновительный PIC-метод / Р. Морз // Вычислительные методы в физике плазмы / Ed. by Б. Олдера, С. Фернбаха, М. Ротенберга. — М.: Мир, 1974.

[3] *Киселёв, А. А.* Численное моделирование захвата ионов бесстолкновительной плазмы электрическим полем поглощающей сферы / А. А. Киселёв, Долгоносов М. С., Красовский В. Л. // Девятая ежегодная конференция «Физика плазмы в Солнечной системе». — 2014.