Динамический анализ качества работы оптимизации асинхронной программной подкачки данных в компиляторе LCC для архитектуры «Эльбрус»

Левченко Дмитрий Николаевич

Аспирант

Физтех-школа Радиотехники и Компьютерных Технологий МФТИ, Москва, Россия

E-mail: levchenko.dn@phystech.edu

Научный руководитель *— к.ф.-м.н. Нейман-заде Мурад Искендер-оглы*

Микропроцессоры на архитектуре «Эльбрус» построены по принципу широкого командного слова (VLIW - very long instruction word)[1]. VLIW-процессоры не обладают возможностями по переупорядочиванию инструкций во время исполнения. Вся работа по оптимизации и распараллеливанию кода производится в оптимизирующем компиляторе, а процессор последовательно исполняет готовые широкие команды, сформированные компилятором.

Одной из самых важных оптимизаций для процессоров на архитектуре «Эльбрус» является оптимизация APB[2]. APB (Array Prefetch Buffer) - это программно-аппаратный механизм, который позволяет асинхронно предподкачивать данные по адресам, линейно зависящих от номера итерации цикла. В отличие от полностью аппаратных префетчеров, которые работают без программной поддержки, данная оптимизация требует явных инструкций в коде о начале предподкачки с указанием необходимых параметров (адрес начала, шаг предподкачки и др.). Такой способ предподкачки позволяет сильно упростить логику работы аппаратуры.

Так как механизм APB обладает рядом накладных расходов на своё применение, то компилятор должен уметь предсказывать их и оценивать эффект от применения оптимизации. Однако, не обладая всей динамической информацией о коде программы в случае однопроходной компиляции, сделать это точно невозможно. Главным методом для решения проблемы является сборка пользовательской программы в режиме PGO (Profile Guided Optimization) с использованием профильной информации. Однако данный подход обладает рядом недостатков. В частности при изменении кодовой базы необходимо получить всю профильную информацию заново, что для боль-

ших кодовых баз с существенным числом вариантов путей исполнения, может занимать недопустимо большое количество времени.

В данной работе был предложен иной подход к решению обозначенных проблем без сбора профильной информации. В составе оптимизирующего компилятора LCC было реализовано отдельное инструментирование для оптимизации APB которое внедряет инструментирующий код на этапе компиляции и передаёт некоторые параметры эвристик оптимизации или параметры архитектуры, под которую компилируется пользовательский код. Для сбора статистики и вывода отчёта программисту реализована библиотека поддержки. После исполнения пользовательской программы на тестовых данных, библиотека поддержки формирует отчёт для циклов в случаях, когда оптимизация APB не применилась. Причиной неприменения может служить то, что компилятор неверно оценил среднее число итераций или оптимизация применилась, но накладные расходы оказались больше полезного эффекта. В данных случаях пользователю рекомендуется подсказка "#pragma prefetch"или "#pragma noprefetch"соответственно или "pragma loop count(N) где N - среднее число итераций цикла.

Для верификации работы была выбрана задача 523.xalancbmk из состава пакета SPEC CPU 2017[3]. Выданный для данной задачи отчёт инструментирования содержал подсказки вида "#pragma noprefetch"для 1 цикла и "#pragma prefetch"для 4 циклов. Расстановка данных подсказок позволила ускорить задачу на 5.5% в режиме компиляции -O3 -fwhole.

Литература

1. Ким А. К., Перекатов В. И., Ермаков С. Г. Микропроцессоры и вычислительные комплексы семейства "Эльбрус". — СПб.: Питер, 2013.
2. Нейман-заде М. И., Королёв С. Д. Руководство по эффективному программированию на платформе «Эльбрус», АО «МЦСТ», 2024
3. SPEC CPU® 2017 industry-standardized, CPU intensive suites for measuring and comparing compute intensive performance // SPEC Organization: [Электронный ресурс]. URL: `https://www.spec.org/cpu2017/`. (дата обращения 09.02.2025).