РАЗРАБОТКА ПРОФИЛЬНЫХ ОПТИМИЗАЦИЙ ЦИКЛОВ В РАМКАХ ИНФРАСТРУКТУРЫ LLVM

Лобанов А.А., студент кафедры системного программирования СПбГУ, resaglow@gmail.com

Аннотация

При разработке инфраструктур компиляторов в последнее время большое внимание уделяется системам оптимизации. В частности, важное значение имеют оптимизации, использующие профильную информацию. В данной работе рассматривается использование профильной информации в оптимизациях циклов, реализованное в компиляторной инфраструктуре LLVM.

Введение

Одной из важных составляющих, определяющих качество программного продукта, является эффективность компилятора, применяемого при разработке. В частности, компилятор должен обладать возможностью максимально оптимизировать код в соответствии с выбранными эвристиками. Оптимизации практически всегда производятся над каким-то внутренним представлением компилятора. Методов оптимизации существует огромное количество: упорядочивание базовых блоков, специальные оптимизации времени связывания и многие другие. В частности, значительный интерес вызывают оптимизации циклов, так как циклы являются одними из самых нагруженных участков программ. В каком-то виде оптимизации циклов реализованы во всех распространённых компиляторах, однако далеко не все компиляторы используют для повышения качества таких оптимизаций профильную информацию – информацию о

нагрузке различных частей программы, полученную с предыдущих запусков. Обладая такой информацией, возможно получить представление о типичных запусках программы и сгенерировать код оптимальный специально для подобного характера работы программы. Одним из проектов, предоставляющих интерфейс как к оптимизации циклов, так и к генерацию профильной информации, является проект LLVM, в рамках которого выполняется данная работа.

Проект LLVM

Проект LLVM представляет собой комплекс библиотек, предоставляющий интерфейс инфраструктуры компиляторов. Он реализован в рамках распространённой в настоящее время retargetable-архитектуры.

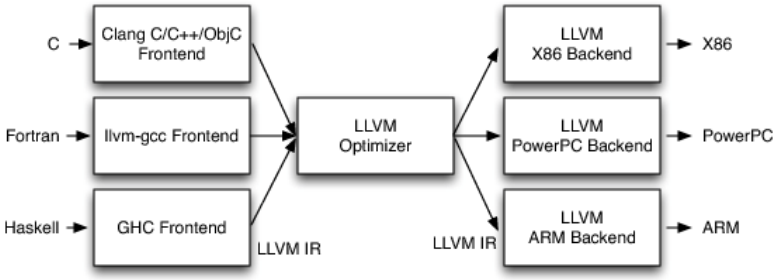

Рисунок 1. Общая архитектура LLVM.

Для реализации в рамках таких систем компияторов нового компилятора для нового языка требуется лишь написать "фронтенд" для этого языка, сразу получив возможность генерировать код для различных архитектур. Аналогично для поддержки всеми языками новой архитектуры необходимо реализовать только генератор кода для этой архитектуры.

Цели работы и обзор

Цель данной работы – исследовать способы повышения качества оптимизаций циклов с помощью профильной

информации [1, 2]. Существует несколько типов оптимизации циклов, которые оказываются пригодными для применения профильной информации. В частности, особое внимание было уделено специализации циклов [3] – вынесении частных случаев цикла с более оптимальной реализацией; векторизации циклов [4, 5] – распараллеливанию итераций на SIMD архитектурах, в том числе векторизации обработки чередующихся данных [6]; и раскрутке циклов. Отдельно был рассмотрен вопрос о балансе между потенциальным повышением производительности программы и размером кода, особо актуальный для embedded систем (SoC, IoT etc.). Эксперименты с оптимизациями проводились на основе специального SSA представления LLVM IR [7].

Результаты исследования

В рамках проводимого исследования были проанализированы несколько типов оптимизаций, способствующих повышению производительности или снижения размера кода циклов в LLVM IR. В частности, как непосредственные оптимизации циклов, были рассмотрены векторизация и раскрутка циклов, как вспомогательная оптимизация был рассмотрен частичный инлайнинг (partial inlining).

Для определения необходимости векторизации/раскрутки циклов в различных участках кода был разработан механизм, позволяющий оценить нагрузку на различные участки кода при типичном использовании программы.

Также для повышения качества был разработан механизм предсказания профильной информации на основании нескольких наборов последовательных профильных данных. Основой реализации механизма предсказания послужила нейросеть типа LSTM (long short-term memory), часто используемая в задачах предсказания временных рядов.

Наконец, информация, сохраняемая во время частичного инлайнинга во внутренние структуры данных компилятора, была использована в векторизации и раскрутки циклов для задания значений параметров:

- коэффициента векторизации;
- коэффициента интерливинга;
- наличия отсутствия скаляризованной версии цикла.

Сравнительный анализ различных комбинаций оптимизаций показал, что во-первых, при сезонных нагрузках на различные части программы предсказывающий алгоритм даёт гораздо лучшие результаты, что статичный; во-вторых параметры векторизации и раскрутки действительно более тщательно оптимизировать производительность для типичных запусков программы; наконец, на основе параметров оптимизаций была разработана система балансировки кода и производительности на основе данных о нагруженности циклов.

Заключение

В работе были предложены различные методы использования профильной информации в оптимизациях циклов в рамках инфраструктуры LLVM. Описаны особенности реализации данных методов, рассмотрены вспомогательные приёмы, позволяющие повысить эффективность методов. Помимо этого приведены результаты сравнительного анализа предложенных оптимизаций, выявлены наиболее существенно влияющие на качество кода оптимизации.

Литература

1. Документация информации о частоте базовых блоков в LLVM. — URL: http://llvm.org/docs/BlockFrequencyTerminology.html.

2. Документация метаинформации о весах веток CFG в LLVM. — URL: http://llvm.org/docs/BranchWeightMetadata.html.
3. Документация проекта LLVM-based partial evaluation (LLPE). — URL: http://llpe.org/docs.
4. Saeed Maleki, Yaoqing Gao, Maria J. Garzarán, Tommy Wong. David A. Padua, 2011. An Evaluation of Vectorizing Compilers. In PACT '11 Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques, 372-382.
5. Документация автовекторизации циклов в инфраструктуре LLVM. — URL: http://llvm.org/docs/Vectorizers.html.
6. Dorit Nuzman, Haifa, Israel Ira Rosen, Haifa, Israel Ayal Zaks, 2006. Auto-vectorization of interleaved data for SIMD. In PLDI '06 Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation, 132-143.
7. Полная справочная информация о языке внутреннего представления LLVM IR. — URL: http://llvm.org/docs/LangRef.html.