УДК 519.685.3

Методы сравнения и выбор стратегии компиляции процедур

О.А. Четверина

ЗАО «МЦСТ»

В повышении производительности современных вычислительных систем важное место занимает оптимизация кода. Для некоторых архитектур, к примеру VLIW, качество планирования кода имеет решающее значение с точки зрения времени исполнения задачи. Поэтому для повышения производительности кода на отдельных задачах используется тонкая настройка оптимизаций посредством регулирующих опций. Сложность использования такого подхода заключается в том, что он требует глубокого понимания процесса оптимизации и потому недоступен большинству пользователей. Кроме того, он не позволяет произвести настройку оптимизаций для отдельных процедур. В различных исследованиях предпринимаются попытки создания систем, позволяющих автоматически произвести попроцедурную настройку компиляции. На данный момент наибольшее развитие получили итерационные системы, производящие многократный запуск кода, и итерационные системы, оценивающие время исполнения по результатам планирования в процессе компиляции [1],[2]. Использование обоих подходов позволяет ускорить производительность в случае, если тренировочные данные или статическая оценка профиля исполнения в достаточной мере соответствует рабочему использованию задачи. Их общим недостатком является существенное увеличение времени, затрачиваемое на получение исполняемого кода [3]. Представленная работа ставит задачу предварительного выбора оптимальных настроек оптимизаций для процедуры, решение которой позволяет ускорить исполнение без замедления компиляции.

Для разработки системы статического попроцедурного выбора последовательности оптимизирующих преобразования необходима тренировочная база процедур и соответствующих им времен компиляции и исполнения. В качестве тренировочного кода был использован пакет задач spec2000. Для оценки времени исполнения использовались два подхода — предсказание времени работы по результатам планирования и профиль в результате реального исполнения. У обоих методов оценки выявились положительные и отрицательные стороны.

Плюсы использования профиля реального исполнения:

1. Существенно улучшается оценка соотношения времени исполнения процедур при отсутствии тренировочного запуска.
2. Избегаются ошибки, связанные с неточностью коррекции тренировочных значений счетчиков линейных участков после применения преобразований

3. Есть возможность учитывать возникающие в процессе исполнения блокировки по чтениям и записям или блокировки из-за отсутствия кода. Стоит отметить, что оптимизация работы памяти является одной из самых важных задачи компилятора. Так, проведенный замер пакета spec2000 показал, что на некоторых задачах блокировки по чтениям составляют 67% всего времени исполнения.

К минусам использования профиля реального исполнения стоит отнести:

1. Эффекты памяти, связанные с изменяющейся работой окружающих процедур, и краевые эффекты, связанные с памятью.
2. Погрешность замеров времени исполнения процедур и уменьшение тренировочной базы.

Было выявлено, что использование предсказания времени исполнения помогает при формировании набора последовательностей оптимизаций, в то время как для настройки автоматической системы выбора предпочтительней пользоваться профилем реального исполнения. Стоит также отметить, что даже при наличии тренировочного профиля не всегда можно полностью полагаться на него для выбора набора оптимизаций. Так, было показано, что использование минимальной оптимизационной последовательности для компиляции нулевых по тренировочному профилю процедур дает почти 20% замедление CFP задач пакета spec2000.

Попроцедурное использование разработанных последовательностей оптимизаций позволяет получить ускорение исполнения на 8,5% при одновременном ускорении компиляции на 17% в среднем для задач пакета spec2000. Описанные механизмы были исследованы и реализованы на базе оптимизирующего компилятора для процессоров семейства «Эльбрус».

Литература

1. *Triantafyllis S., Vachharajani M., Vachharajani N., August D.I.*, Compiler optimization-space exploration – Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization – March 23-26, 2003
2. *Haneda M., Knijnenburg P.M.W., Wijshoff H.A.G.,* Generating new general compiler optimization settings – Proceedings of the 19th annual international conference on Supercomputing – June 20-22, 2005
3. *Kulkarni P.A., Jantz M.R., Whalley D.B.*, Improving both the performance benefits and speed of optimization phase sequence searches – LCTES '10 Proceedings of the ACM SIGPLAN/SIGBED 2010 conference on Languages, compilers, and tools for embedded systems – April 2010