УДК 681.3

ОПТИМИЗАЦИЯ МЕТОДИКИ МОДЕЛИРОВАНИЯ КЭША СЕТЕВЫХ ПРОЦЕССОРОВ

В.И.Грищенко, Ю.В.Ладыженский
Донецкий Национальный Технический Университет

Комплексные методики моделирования сетевых процессоров являются гибким инструментом для анализа их эффективности. В представленной работе предлагается подход, позволяющий минимизировать время, затрачиваемое на имитационную часть комплексной методики за счет использования аналитической модели промахов кэша.

1 Оптимизация комплексной методики моделирования

В работе [2] была предложена методика моделирования сетевых процессоров, объединяющая в себе имитационный и аналитический подходы (см. рис. 1).

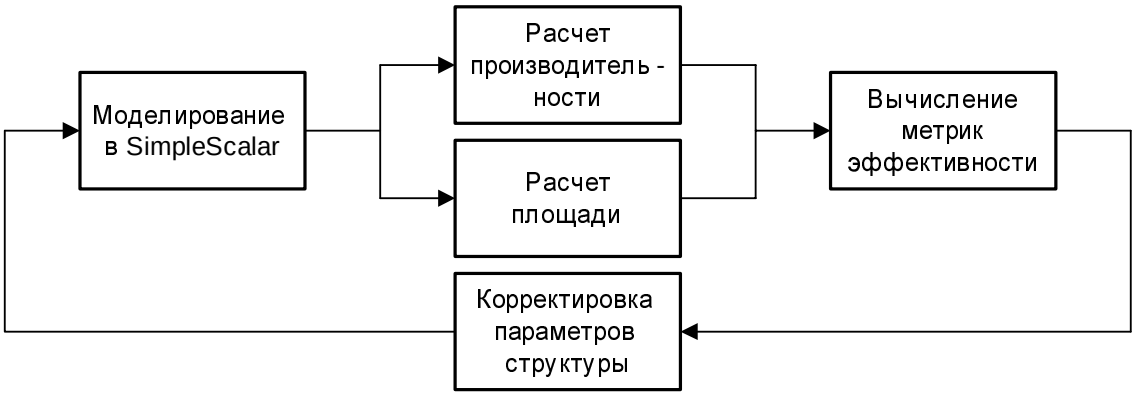

Рис. 1 — Структура моделирования сетевого процессора

Моделирование начинается с выбора параметров сетевого процессора: число вычислительных ядер, размеры кэшей данных и команд, количество аппаратно поддерживаемых потоков, используемый технологический процесс (для вычисления площади кристалла), выбор тестового приложения. На втором этапе собирается статистика о работе тестового приложения в системе имитационного моделирования SimpleScalar [6]. Система SimpleScalar широко используется в академических исследованиях для моделирования работы процессоров различной структуры и с разными наборами команд. Она позволяет гибко варьировать размеры кэшей, параметры АЛУ, блока предсказаний переходов, задержку доступа к памяти и другие характеристики процессора.

На основе полученных данных вычисляется производительность и площадь сетевого процессора, они используются для определения

степени эффективности моделируемой структуры. После чего корректируются параметры СП и весь цикл моделирования повторяется.

Проведенные эксперименты показывают, что более 90% времени моделирования занимает работа SimpleScalar. При этом для каждой конфигурации исследуемой структуры приходится менять параметры запуска SimpleScalar, что исключает возможность повторного использования полученных данных.

Из характеристик приложения, используемых в последующих расчетах, только две меняются в зависимости от размеров кэша: вероятности промахов кэшей данных и команд. Если же использовать SimpleScalar для получения характеристик приложения, не зависящих от размеров кэшей, но которые возможно использовать для расчета вероятностей промахов, то SimpleScalar может запускаться лишь единожды для каждого отдельного приложения, и в последствии влияние размеров кэшей на вероятность промахов будет рассчитываться на основе аналитической модели. Предположительно такой подход позволит уменьшить общее время моделирования без существенных потерь в точности получаемых результатов.

В работе [3] была предложена статистическая модель кэша StatCache, которая основывается на трассе обращений приложения к памяти и позволяет вычислить вероятность промаха для полностью ассоциативного кэша со случайной политикой замещения. В [4] было показано, что для большей части приложений сетевой обработки данных случайная политика замещения является одним из наиболее эффективных алгоритмов вытеснения записей в кэше, что позволяет пренебречь этим ограничением. Полностью ассоциативный кэш, имеет наибольшую эффективность [7], поэтому его использование в моделировании также является обоснованным.

Методика расчета StatCache основывается на понятии «расстояния повторного использования». Оперативная память компьютера делится на блоки, равные длине строки кэша. Для каждой операции обращения к памяти вычисляется номер блока, с которым она взаимодействует. Расстояние повторного использования определяется как количество инструкций доступа к ОЗУ между обращениями к одному и тому же блоку памяти (см. рис. 2).

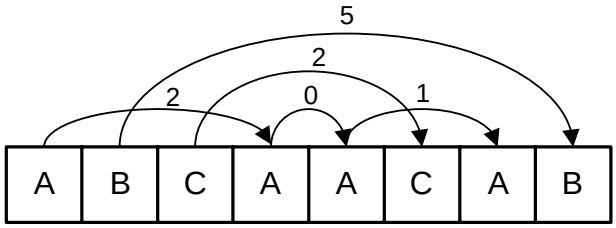

Рис. 2 — Расстояние повторного использования для блоков памяти A, B и C

Учитывая, что рассматривается полностью ассоциативный кэш со случайной политикой замещения, то вероятность того, что запись не будет замещена после первого промаха равна 1–1/L, где L — число строк в кэше. Для n промахов вероятность того, что запись останется в кэше равна $(1–1/L)^n$. Таким образом, вероятность того, что строка кэша будет замещена после n промахов равна:

$$f(n) = 1 - (1 - 1/L)^n \quad (1)$$

Для каждого обращения к памяти определим случайную величину X_i, которая равна 1, если произошел промах и 0 иначе. Также определим вероятность промаха при i-м обращении к памяти p_i, которое выражается как математическое ожидание величины X_i: $p_i=M(X_i)$. Пусть A(i) — расстояние повторного использования для i-го обращения, тогда предыдущий доступ к этому же блоку оперативной памяти осуществлялся на (i – A(i) – 1)-м обращении. Общее число промахов кэша на этом интервале равно:

$$M\left(X_{i-A(i)} + X_{i-A(i)+1} + \cdots + X_{i-1}\right) = p_{i-A(i)} + p_{i-A(i)+1} + \cdots + p_{i-1} \quad (2)$$

Используя функцию 1, можно оценить вероятность промаха p_i как:

$$p_i \approx f\left(p_{i-A(i)} + p_{i-A(i)+1} + \cdots + p_{i-1}\right) \quad (3)$$

Просуммировав обе части выражения от 1 до N получим:

$$\sum_{i=1}^{N} p_i \approx \sum_{i=1}^{N} f\left(p_{i-A(i)} + p_{i-A(i)+1} + \cdots + p_{i-1}\right) \quad (4)$$

Выражение 4 определяет взаимодействие между всеми p_i, но содержит слишком много неизвестных, что делает невозможным вычисление общей вероятности промаха для всего журнала обращений. Поэтому предположим, что общая вероятность промаха R не изменяется на протяжении всей работы приложения. Отсюда следует, что:

$$p_i + p_{i+1} + \cdots + p_{i+j-1} = j \cdot R \quad (5)$$

где, $1 \le i << (i+j) \le N$, а N — общее число обращений к памяти. Заменим j на A(i):

$$p_{i-A(i)} + p_{i-A(i)+1} + \cdots + p_{i-1} = A(i) \cdot R \quad (6)$$

Подставив уравнение 6 в выражение 4 получим:

$$N \cdot R \approx \sum_{i=1}^{N} f\left(A(i) \cdot R\right) \quad (7)$$

Рассмотрим элементы суммы в выражении 7. Число таких элементов со значениями A(i), равными некоторой константе K, эквивалентно числу обращений к памяти с расстоянием повторного использования K, что, в свою очередь, равно h(K), где h — гистограмма расстояний повторного использования, построенная по журналу обращений к памяти. Таким образом, выражение 7 может быть записано как:

$$RN \approx h(1)f(R) + h(2)f(2R) + h(3)f(3R) + \cdots \quad (8)$$

Формула 8 хорошо подходит для численного решения и используется для нахождения вероятности промаха кэша для всего приложения.

В выражении 8 не учитываются холодные промахи, которые происходят каждый раз, когда программа впервые обращается к какой-либо области памяти. В работе [5] предлагается модификация модели StatCache, принимающая во внимание такие промахи. Для этого, при анализе потока обращений к памяти, подсчитывается количество холодных обращений N_{cold} (первых обращений к каждому блоку памяти), и вычисляется вероятность возникновения холодного промаха:

$$M_{cold} = \frac{N_{cold}}{N} \quad (9)$$

При этом формула 8 приобретает вид:

$$RN \approx NM_{cold} + h(1)f(R) + h(2)f(2R) + h(3)f(3R) + \cdots \quad (10)$$

Корни уравнения 10 находятся численными методами. Полученное значение вероятности промаха R используется для расчета характеристик сетевого процессора.

2 Модификация SimpleScalar

Система имитационного моделирования SimpleScalar предоставляет возможность генерации журнала изменений состояния конвейера процессора. Из журнала конвейера доступна информация о последовательности и адресах выполняемых команд, но отсутствует информация об эффективных адресах используемых блоков памяти, есть лишь имена базовых регистров и регистров смещения, в которых эти адреса хранятся.

Для получения информации об используемых блоках памяти авторы дополнили функциональность SimpleScalar двумя новыми возможностями: ведение журналов обращений к памяти за данными и инструкциями с выводом физических адресов, к которым происходит обращение. Оба журнала имеют одинаковый формат. Каждая запись состоит из трех полей: тип обращения (принимает значения «l» - чтение или «s» - запись), размер запрашиваемого блока в байтах и адрес в ОЗУ, по которому производится обращение, записанный в десятичной системе счисления.

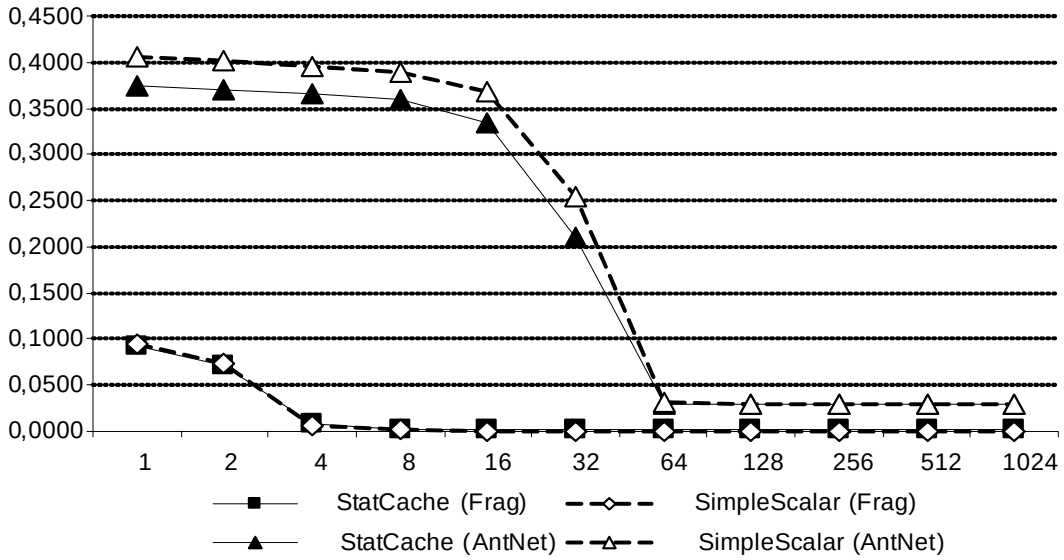

Рис. 3 — Вероятность промаха кэша данных для приложений Frag и Antnet

Журналы, полученные единожды, могут впоследствии многократно использоваться для анализа вероятности промахов кэшей разных размеров и обладающих различными длинами строк. Таким образом, исключается необходимость выполнения имитационного моделирования для каждой исследуемой структуры процессора.

3 Результаты моделирования

Описанная методика использовалась при анализе приложений сетевой обработки данных. На рисунке 3 представлены результаты моделирования вероятности промахов кэша данных для задач фрагментации (Frag) и маршрутизации с активными агентами (AntNet). Из графиков видно, что результаты, полученные с помощью модели StatCache в значительной мере соответствуют результатам имитационного моделирования. Это означает, что использование предложенного подхода для исследования СП является оправданным.

Выводы

Модифицированная методика обладает достаточной точностью результатов и, вместе с тем, оставляет широкие возможности по уменьшению времени проведения экспериментов. Рассмотренная модель промахов кэшей имеет некоторые ограничения, но ими можно пренебречь в силу особенностей функционирования сетевых процессоров.

Литература

1. Ладыженский Ю.В., Грищенко В.И. Моделирование работы приложений на сетевых процессорах. Матеріали міжнародної науково-практичної конференції „Моделювання та комп’ютерна графіка 2007”, м. Донецьк, 10-12 жовтня.

2. Tilman Wolf, Mark A. Franklin, "Performance Models for Network Processor Design," IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 6, pp. 548-561, Jun., 2006.

3. E. Berg , E. Hagersten, StatCache: a probabilistic approach to efficient and accurate data locality analysis, Proceedings of the 2004 IEEE International Symposium on Performance Analysis of Systems and Software, p.20-27, March 10-12, 2004

4. Ладыженский Ю.В., Грищенко В.И. Влияние политики замещения записей в кэше на производительность сетевого процессора. //Наукові праці Донецького національного технічного університету. Серія: обчислювальна техніка та автоматизація. – Вип. 12 (118). – Донецьк. – 2007. С. 114-119.

5. E. Berg, H. Zeffer and E. Hagersten, "A Statistical Multiprocessor Cache Model," Proc. of Int'l Symp. on Performance Analysis of Systems and Software, March 2006.

6. Doug Burger and Todd M. Austin, “The SimpleScalar tool set, version 2.0,” Tech. Rep. 1342, Department of Computer Science, University of Wisconsin in Madison, June 1997.

7. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем - СПб: Питер, 2006. - 668 с.

Получено 28.05.09