УДК 681.3

ГРИЩЕНКО ВИКТОР ИГОРЕВИЧ, ЛАДЫЖЕНСКИЙ ЮРИЙ ВАЛЕНТИНОВИЧ

Донецкий Национальный Технический Университет, Украина

ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ КЭША СЕТЕВЫХ ПРОЦЕССОРОВ

Рассмотрены аналитическая модель сетевых процессоров пакетной обработки данных (СП) и методика исследования влияния архитектуры кэша на производительность СП. Методом моделирования на тестовом наборе приложений получены оценки оптимальных объемов кэша команд и данных.
Ил. 7. Библиогр.: 9 назв.

Сетевой процессор, маршрутизатор, кэш, комплексная методика моделирования, тестовый набор приложений

Требования к пропускной способности компьютерных сетей постоянно растут. Увеличение общего числа пользователей и типов передаваемого трафика значительно повысило нагрузку на маршрутизаторы. Используемые ранее в маршрутизаторах процессоры общего назначения не в состоянии обрабатывать возросшие потоки данных. Поэтому для решения задач маршрутизации и управления сетями часто применяются специализированные сетевые процессоры пакетной обработки данных (сетевые процессоры, СП).

СП представляют собой устройства, ориентированные на выполнение узкого круга задач (Application Specific Instruction Processor, ASIP). Ключевые операции сетевой обработки (вычисление контрольных сумм, анализ заголовков пакетов) реализуются в СП аппаратно. Кроме того СП предоставляют широкие возможности для программирования сложных сетевых приложений на языке ассемблера и языке высокого уровня [1].

В настоящее время не существует общей методологии проектирования сетевых процессоров. Использование моделирования позволяет изучить свойства сетевых процессоров, прогнозировать их производительность при различных видах нагрузки, выбирать оптимальные архитектурные и структурные решения.

1 Модели сетевых процессоров

Можно указать три основных подхода к исследованию архитектуры сетевых процессоров: имитационные модели [2], аналитические модели, основанные на кривых прибытия и обслуживания [3] и комплексная методика моделирования сетевых процессоров (КМСП) [4]. Имитационная модель [2] реализует архитектуру процессора Intel IXP1200. Основным недостатком имитационного моделирования является сложность внесения изменений в исследуемую архитектуру. Обычно такая модель описывает одну архитектуру и ее модификация для исследования другого устройства по трудоемкости часто сопоставима с созданием новой модели.

Модель кривых [3] описывает сетевой процессор как совокупность обрабатывающих узлов. Возможности каждого узла ограничиваются снизу и сверху кривыми обслуживания. Входной поток ограничивается кривыми прибытия и по мере прохождения через устройство он уменьшается на величину полученного обслуживания. Такой подход позволяет анализировать сложные архитектуры процессоров и виды нагрузок на них. Модель кривых ориентирована на исследование различных структур СП. Анализ новых типов нагрузок в этой модели осложнен необходимостью построения кривых обслуживания для каждого вида внутреннего узла сетевого процессора.

В [4] представлена комплексная методика моделирования сетевых процессоров. КМСП включает имитационное моделирование работы приложения на эталонном процессоре и аналитический расчет характеристик всего устройства. Для исследования новых видов нагрузок в комплексной методике выполняется реализация алгоритма нового приложения на языке Си и последующее выполнение полученной программы на эталонном процессоре. При этом аналитическая модель архитектуры обеспечивает гибкость в выборе конфигурации самого устройства.

Главным отличием КМСП от модели кривых является возможность анализа не только производительности процессора, но и его других важных технических характеристик, таких как площадь и энергопотребление кристалла.

В данной статье за основу принята комплексная методика и выполнено ее развитие для анализа влияния структуры кэш-памяти на производительность и площадь сетевого процессора.

2 Методика исследования

Обобщенная архитектура сетевого процессора представлена на рис. 1. СП состоит из однотипных вычислительных ядер, сгруппированных по кластерам [4,5]. Ядра одного кластера разделяют общий интерфейс к внешней памяти. Каждый процессорный элемент аппаратно реализует многопоточность и имеет собственные кэши команд и данных. Предполагается, что переключение между контекстами потоков осуществляется с нулевой задержкой, т.е. как только один поток переходит в режим ожидания ответа от канала памяти, следующий тут же начинает работу. Параметрами кластерной архитектуры являются: число потоков, поддерживаемых вычислительными ядрами; объемы кэшей; число процессоров в кластере; общее число кластеров.

Этапы исследования свойств архитектур сетевых процессоров представлены на рис. 2. В отличие от подхода [5] в исследуемой модели оптимизация СП ведется только по производительности и площади, как наиболее важных его характеристик.

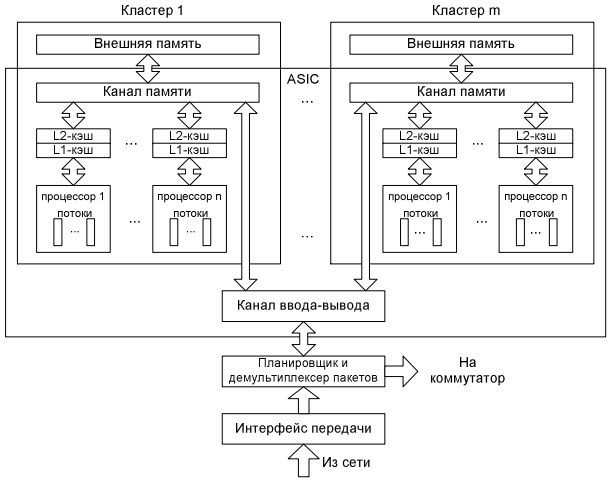

Рис. 1. Обобщенная архитектура сетевого процессора

Анализ архитектуры сетевого процессора состоит из двух частей: имитационное моделирование работы приложения на эталонном процессоре и аналитический расчет характеристик архитектуры. Работа приложения анализируется с использованием системы SimpleScalar [6], которая позволяет моделировать работу программ на RISC-процессоре. Для анализа эффективности работы СП определяется архитектура процессора (в нашем случае параметры кэша) и выбирается тестовое приложение, которое будет на нем выполняться.

В качестве тестового набора приложений в исследовании использовался пакет CommBench [7]. Приложения тестового пакета включают многие задачи, выполняемые на современных маршрутизаторах. Логически эти программы можно разделить на две группы: приложения, обрабатывающие только заголовки пакетов и приложения, анализирующие данные, передаваемые по сети. К первой группе относятся задачи фрагментации (FRAG), маршрутизации (RTR) и планирования очередей (DRR). Во вторую группу попали программы работы с медиаданными (на примере JPEG), сжатием потока (ZIP), выполнение избыточного кодирования (REED) и

шифрование информации (CAST). Использование CommBench в качестве тестового набора задач не обязательно. SimpleScalar позволяет анализировать работу любого приложения, написанного на языке Си и скомпилированного с помощью поставляемого вместе с ней кросскомпилятора.

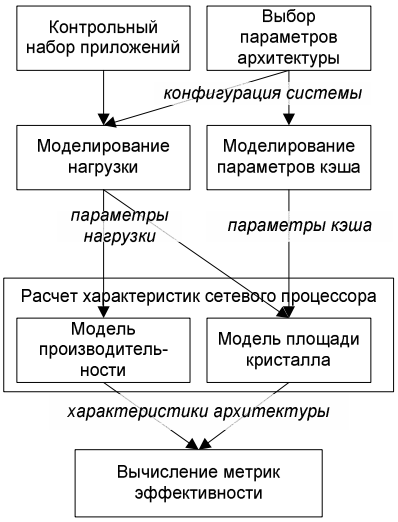

Рис. 2. Среда моделирования

Моделирование в SimpleScalar позволяет узнать динамические характеристики кэша для заданного приложения: число обращений, вероятность промахов и др. Для вычисления площади процессора необходимо знать площадь каждой отдельной его компоненты и площадь кэшей в том числе. Площадь кэшей существенно зависит от его размеров и структуры. В предоставленном исследовании характеристики кэшей определялись с использованием системы моделирования CACTI [8]. Система CACTI позволяет получать детальные физические характеристики кэш-памяти, но для расчета площади устройства достаточно знать только общую площадь кэша.

Перед использованием CACTI и SimpleScalar должны быть определены параметры исследуемой модели. На этом этапе достаточно знания только параметров кэша, остальные параметры используются при расчете общих характеристик архитектуры. В проведенном исследовании сетевой процессор использует кэш данных и кэш инструкций, которые описываются тремя параметрами: размер, длина строки кэша и его ассоциативность. В представленной работе рассматривается только влияние кэша первого уровня.

После выполнения имитационного моделирования, полученные параметры используются для расчета характеристик сетевого процессора. Характеристики системы рассчитываются в два этапа: расчет производительности процессора (IPS) и расчет площади кристалла (A).

Производительность сетевого процессора оценивается суммой производительностей его вычислительных ядер:

$$IPS_{NP} = \sum_{j=1}^{m}\sum_{k=1}^{n} \rho_{p_{j,k}} \cdot f_{p_{j,k}}, \qquad (1)$$

где $\rho_{p_{j,k}}$ - производительность процессора (j, k), команд/такт, $f_{p_{j,k}}$ - тактовая частота процессора, такт/секунда.

Производительность отдельного многопоточного процессорного ядра с кэшами команд и данных определяется по формуле [9]:

$$\rho_p(t) = 1 - \frac{1}{\sum_{i=0}^{t}\left(\frac{1}{p_{miss}\tau_{mem}}\right)^i \frac{t!}{(t-i)!}}, \qquad (2)$$

где ρ_p - загруженность процессора, t - число потоков, поддерживаемых процессором, p_{miss} - интенсивность промахов кэша, τ_{mem} - время доступа к памяти.

Время доступа к памяти учитывает время ожидания запроса в очереди (τ_Q), время физического отклика памяти (τ_{DRAM}) и время передачи данных через кэш ($\tau_{transmit}$):

$$\tau_{mem} = \tau_Q + \tau_{DRAM} + \tau_{transmit} \qquad (3)$$

Интенсивность промахов кэшей зависит от приложения, выполняемого на СП:

$$p_{miss} = mi_c + (f_{load} + f_{store}) \cdot md_c \qquad (4)$$

где

mi_c и md_c - интенсивности промахов кэшей команд и данных соответственно, f_{load} и f_{store} - частота появления в коде программы команд чтения и записи в память.

Площадь кристалла, необходимая для размещения заданной конфигурации процессора определяется как сумма площадей всех его компонент:

$$A_{NP} = A_{IO} + m \cdot \left(A_{mc} + n \cdot \left(A_{p,t} + A_{c_i} + A_{c_d}\right)\right) \quad (5)$$

где A_{NP} - площадь сетевого процессора, A_{mc} - площадь канала памяти, включая контакты, $A_{p,t}$ - площадь одного процессорного ядра, A_{c_i} и A_{c_d} - площадь кэша команд и данных соответственно.

Полученные значения в дальнейшем могут быть использованы как самостоятельно, так и для составления сводных метрик.

3 Результаты моделирования

На рис. 3 и 4 показана зависимость комплексной метрики по производительности и площади устройства (IPS/A) от размеров кэшей команд и данных для задач фрагментации (FRAG) и планирования очередей (DRR) соответственно.

Из графиков видно, что оптимальными для этих задач являются размеры кэша 2 КБ как для команд так и для данных. Это объясняется тем, что обе задачи имеют невысокую алгоритмическую сложность и не требуют большого числа обращений к оперативной памяти.

Для программы маршрутизации (см. рис. 5) пик производительности достигается при кэше команд равном 32 КБ и кэше данных — 4 КБ. В отличие от других приложений маршрутизация является более сложной задачей и часто обращается к внешней памяти при поиске в таблице маршрутов. На большую сложность этой задачи указывает и абсолютное значение метрики: для фрагментации пакетов оптимизация кэшей дает производительность в 18,76 MIPS (миллионов инструкций в секунду), а для маршрутизации максимальная производительность достигает лишь 3,94 MIPS при оптимальных кэшах команд и данных.

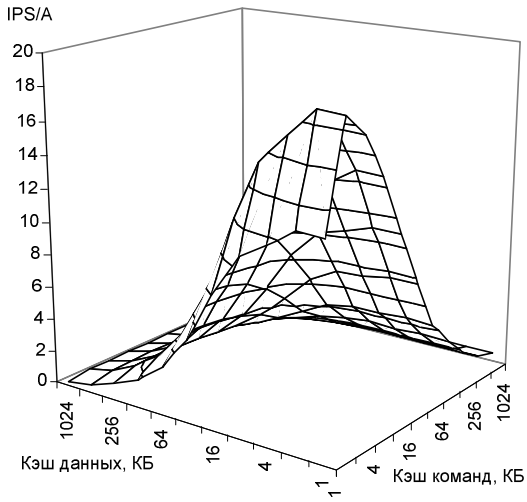

Рис. 3. Зависимость метрики IPS/A от размеров кэшей для задачи фрагментации (FRAG)

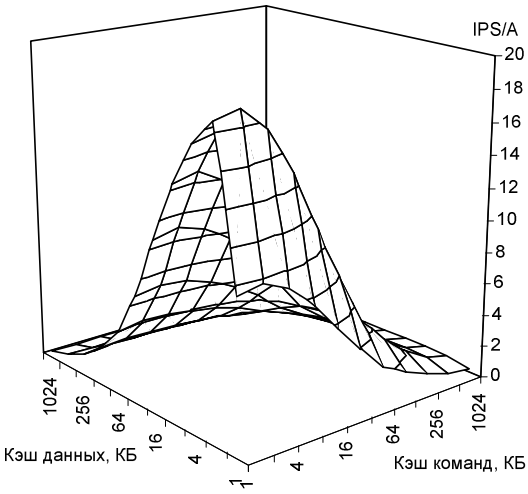

Рис. 4. Зависимость метрики IPS/A от размеров кэшей для задачи планирования очередей (DRR)

Таким образом, для задач обработки заголовков пакетов достаточно кэша данных объемом не более 4 КБ, а кэша инструкций - объемом 32 КБ. Дальнейшее увеличение промежуточной памяти не будет давать должного роста производительности относительно увеличения площади кристалла. Использование меньших кэшей снижает эффективность маршрутизации пакетов, выполняемой на сетевых процессорах.

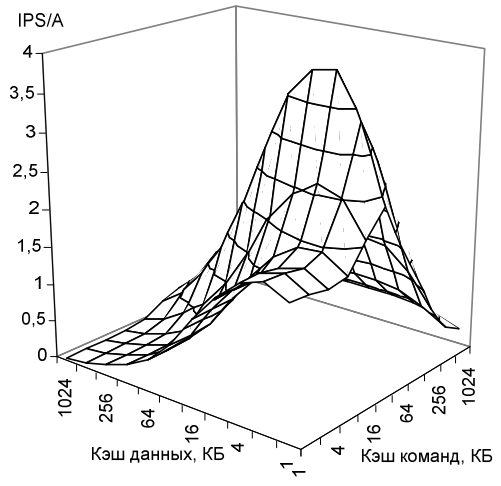

Рис. 5. Зависимость метрики IPS/A от размеров кэшей для задачи маршрутизации (RTR)

Каждое приложение из набора CommBench, работающее с данными в пакетах, состоит из двух программ: кодировки и декодировки данных. Например, в приложении JPEG программа кодировки преобразовывает BMP-картинку в формат JPEG, а декодировка выполняет обратную операцию. Часть задач второго типа достигает оптимума на кэшах размером до 4 КБ (ZIP, REED и кодирование JPEG), другая (CAST и декодирование JPEG) более требовательна к ресурсам. На рисунке 6 показана метрика для кодирования ZIP (сжатие данных). Здесь оптимум достигается при размерах кэшей 2 и 4 КБ для команд и данных соответственно. Так же, как и с первой группой задач, дальнейшее увеличение объема кэш-памяти уменьшает эффективность архитектуры, постепенно устремляя ее к нулю. Распаковка данных для алгоритма ZIP дает аналогичные результаты. Задача избыточного кодирования (REED) требует еще меньших объемов промежуточной памяти: 1КБ кэш команд и 4 КБ кэш данных. Обе этих задачи оказались симметричными с точки зрения ресурсоемкости, для них график зависимости метрики IPS/A от размеров кэша выглядит одинаково как для кодирования, так и для декодирования.

Единственным явно несимметричным приложением из исследуемых оказался алгоритм JPEG, с помощью которого моделировалась обработка медиа-данных на маршрутизаторе. Сжатие по алгоритму JPEG дает оптимальные размеры кэша команд -1 КБ и кэша данных — 2КБ. Но для распаковки JPEG оптимальными оказываются значения 32КБ и 16 КБ соответственно.

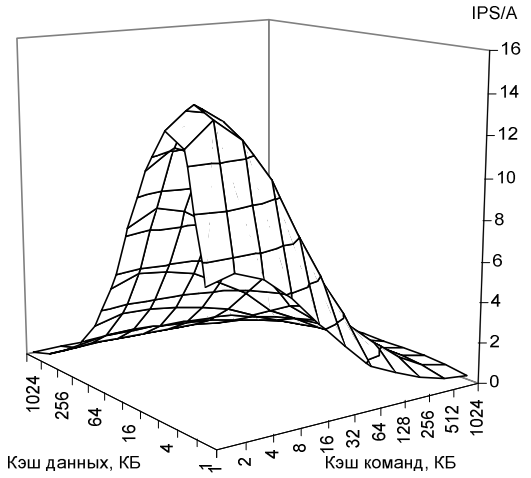

Рис. 6. Зависимость метрики IPS/A от размеров кэшей для задачи сжатия данных (ZIP)

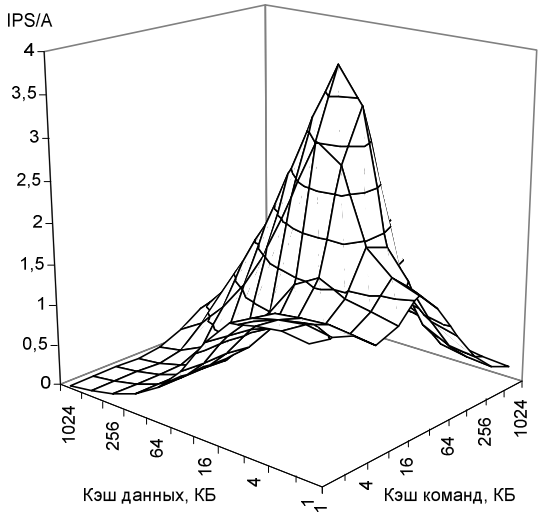

Рис. 7. Зависимость метрики IPS/A от размеров кэшей для задачи дешифровки данных (CAST)

Наиболее ресурсоемкой задачей оказалось шифрование информации (CAST). Для этого приложения требуется кэш инструкций в 64 КБ для кодировки и 32 КБ для дешифровки данных. В обоих случа-

ях оптимальным оказался кэш данных, равный 16 КБ. График зависимости для кодирования приведен на рис. 7. Из рисунка видно, что рост эффективности для кэша команд начинается на отметке 16 КБ, достигает пика в 64 и затем резко падает. Аналогично, для кэша данных, рост эффективности начинается при объеме 4 КБ, растет до 16, затем уменьшается с большой интенсивностью.

Выводы

Проведенное исследование показывает, что для типичных задач, выполняемых на маршрутизаторе, оптимальными является размер кэша команд, равный 64 КБ и кэш данных объемом 16 КБ. Этих характеристик будет достаточно как для выполнения задач обработки заголовков сетевых пакетов, так и для работы с передаваемыми данными. Если проектируемый сетевой процессор ориентирован только для решения классических задач маршрутизатора (маршрутизация, фрагментация пакетов и т.п.), то достаточными будет кэш команд размером 16 КБ и кэш инструкций объемом 4 КБ.

В дальнейшем планируется исследовать сетевые процессоры с двухуровневыми кэшами по критериям производительности и стоимости. Также планируется рассмотреть другие модели кэширования и сравнить результаты их использования для сетевых процессоров.

Литература

1. M. Gries, C. Kulkarni, C. Sauer, K. Keutzer: Exploring Trade-offs in Performance and Programmability of Processing Element Topologies for Network Processors, Network Processor Design: Issues and Practices, volume 2, Editors: P. Crowley, M. Franklin, H. Hadimioglu, P. Onufryk, Morgan Kaufmann Publishers, Nov. 2003
2. G. Memik and W. H. Mangione-Smith. NEPAL: A framework for efficiently structuring applications for network processors. InProc. Of Network Processor Workshop in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9), Feb. 2003.
3. Lothar Thiele, Samarjit Chakraborty, Matthias Gries, and Simon Kunzli, “Design space exploration of network processor architectures,” in Proc. of First Network Processor Workshop (NP-1) in conjunction with Eighth International Symposium on High Performance Computer Architecture (HPCA-8), Cambridge, MA, Feb. 2002, pp. 30–41.
4. 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.
5. Ладыженский Ю.В., Грищенко В.И. Моделирование сетевых процессоров пакетной обработки данных. ИНТЕРНЕТ-ОБРАЗОВАНИЕ-НАУКА, пятая международная конференция ИОН-2006. 10-14 октября, 2006. Сборник материалов конференции. Том 2. – Винница: УНИВЕРСУМ-Винница, 2006, стр. 417-422.
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. Tilman Wolf and Mark A. Franklin, “CommBench - a telecommunications benchmark for network processors,” in Proc. of IEEEInternational Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, Apr. 2000, pp. 154–162.
8. Premkishore Shivakumar and Norman P. Jouppi, “CACTI 3.0: An integrated cache timing, power and area model,” Tech. Rep. WRL Research Report 2001/2, Western Research Laboratory, Palo Alto, CA, Aug. 2001.