Эффективность реализации протокола маршрутизации AntNet на сетевых процессорах

Ладыженский Ю. В., Грищенко В.И.
Донецкий Национальный Технический Университет
ly@cs.dgtu.donetsk.ua, grishenko_v@pmi.dgtu.donetsk.ua

Abstract

U.V. Ladyzhensky, V.I. Grischenko. The Network Processor Implementation Effectiveness of AntNet Routing Algorithm. *To provide a scalability of computer networks new routing algorithms are required. Nowadays routers are using specialized processors (network processors or NP) to perform packet processing. In this paper an efficiency of executing AntNet routing algorithm on network processor is analyzed. This analysis estimate performance of NP depending to its cache sizes.*

Введение

С увеличением объемов информации, передаваемой через компьютерные сети, традиционные алгоритмы маршрутизации (RIP, OSPF) показали низкую способность к масштабированию [5]. В качестве альтернативного решения им на смену был предложен алгоритм AntNet [1,2]. В его основу легла имитация поведения колонии муравьев при поиске кратчайших путей к пище. Использование алгоритма AntNet в современных маршрутизаторах позволит увеличить пропускную способность сетей при тех же параметрах аппаратуры.

В представленной статье исследуется эффективность реализации алгоритма AntNet на сетевом процессоре (СП). Сетевые процессоры обеспечивают увеличение производительности маршрутизаторов за счет аппаратной реализации важнейших функций сетевой обработки данных (поиск в таблице маршрутизации, планирование очередей, расчет контрольных сумм и др.). Вместе с тем, СП предоставляют широкие возможности для программирования.

Анализ работы приложения на сетевом процессоре осуществляется с помощью системы моделирования SimpleScalar [6] и методики, предложенной в [3]. Такой подход позволяет не только определить ресурсоемкость алгоритма, но и выбрать оптимальную конфигурацию архитектуры для его работы.

1 Алгоритм маршрутизации AntNet

Процедура поиска кратчайших путей в алгоритме AntNet использует те же принципы, что и колония муравьев при поиске пищи [2]. Муравей на

своем пути оставляет след из активных веществ — феромонов. Феромоновый след во внешней среде существует некоторое время, и муравьи, которые будут идти следом, с больше вероятностью предпочтут то направление, по которому до них прошли другие муравьи. Эта вероятность будет тем больше, чем большее количество муравьев прошло по этому пути. Таким образом, через некоторое время большая часть муравьев будет использовать один, наиболее близкий к оптимальному, маршрут.

Роль муравьев в протоколе AntNet выполняют активные агенты. Активный агент — это специальный пакет, который несет с собой статистику о состоянии пройденных сетевых каналов. Муравьи делятся на два вида: F-муравьи и B-муравьи. F-муравьи собирают во внутренний стек статистику о состоянии сети. Они не изменяют таблицы маршрутизации. При прохождении маршрутизаторов F-муравьи испытывают те же задержки, что и обыкновенные пакеты данных, поэтому собранная ими информация в большой мере соответствует действительному состоянию сети. Каждый маршрутизатор, реализующий протокол AntNet, с заданной периодичностью рассылает F-муравьев в различные узлы сети, тем самым отслеживая ее состояние.

После того, как F-муравей достигнет адреса назначения, на его основе создается B-муравей. В него копируется накопленный стек статистики F-муравья, после чего F-муравей уничтожается. B-муравей отправляется обратно по маршруту F-муравья. Маршрутизаторы извлекают из проходящего B-муравья статистические данные и на их основе обновляют свои таблицы маршрутизации.

Таблица маршрутизации представляет собой матрицу, в которой строки соответствуют узлам сети, а столбцы — сетевым интерфейсам маршрутизатора. Например, если строка 1 описывает удаленный узел с IP-адресом 192.168.1.4, то в первом столбце будет храниться вероятность перехода пакета по первому сетевому интерфейсу, направленного в узел 192.168.1.4, во второй ячейке — по второму и т.д. Таким образом, число столбцов в таблице маршрутизации всегда равно количеству сетевых интерфейсов маршрутизатора. Сумма вероятностей по одной строке всегда равна единице.

2 Структура маршрутизатора

Структура исследуемой модели маршрутизатора представлена на рис. 1. Маршрутизатор имеет n+1 сетевой интерфейс, где n – число соседних маршрутизаторов и один интерфейс для внутренней сети. Все адресованные во внутреннюю сеть пакеты автоматически отправляются на нулевой интерфейс. В случае, если для пакета не находится соответствующая запись в таблице маршрутизации, он отправляется по первому интерфейсу (маршрут по умолчанию).

Маршрутизатор описывается множеством параметров:

- IP-адрес;
- маска сети;
- количество сетевых интерфейсов;
- массив IP-адресов соседей (индексы массива являются номерами сетевых интерфейсов);
- массив средних времен доставки пакетов к узлам сети;
- таймер запуска муравьев;
- таблица маршрутизации.

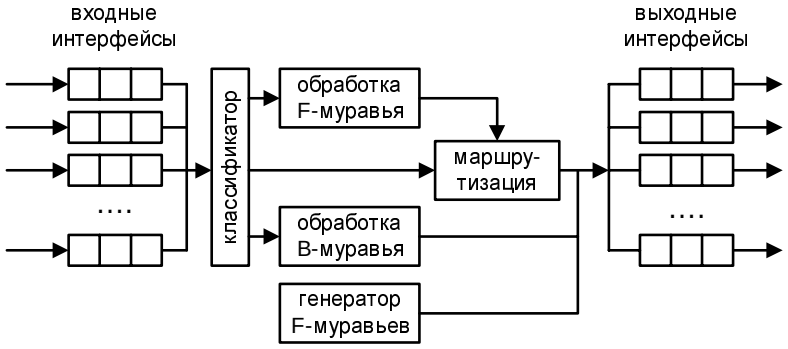


Рисунок 1 – Структура исследуемой модели маршрутизатора

В начале работы конфигурация читается из специализированного файла и на ее основе создается маршрутизатор с заданными параметрами. В процессе работы маршрутизатор последовательно извлекает пакеты из файла потока и они сразу попадают на классификатор. Классификатор делит пакеты на три группы: пакеты данных, F-муравьи и B-муравьи. В зависимости от типа пакеты обрабатываются различным образом. Пакет данных передается на блок маршрутизации. Для этого типа пакетов маршрутизация включает поиск строки в таблице маршрутов, соответствующей адресу назначения пакета, и выбор из этой строки элемента с максимальной вероятностью перехода.

Для F-муравья вычисляется время, которое ему потребовалось на переход от предыдущего муршрутизатора. Вычисленная величина помещается в стек статистики пакета. Затем адрес назначения муравья сравнивается с адресом маршрутизатора, если они равны, то создается B-муравей, в него помещается вся статистика из стека F-муравья, и он передается в выходную очередь того сетевого интерфейса, по которому прибыл пакет. Если пакет должен быть направлен на другой маршрутизатор, то он передается в блок маршрутизации, где определяется необходимый выходной интерфейс. Однако, в отличие от пакета данных, после нахождения соответствующей строки таблицы маршрутизации, выбор ячейки производится не по максимальному значению, а случайным образом с учетом установленных вероятностей перехода.

При обработке B-муравья осуществляется обновление коэффициентов таблицы маршрутизации. Обрабатываемый B-муравей в первую очередь проверяется на наличие IP-адреса маршрутизатора в стеке статистики пакета. Если его там нет, делается вывод о том, что пакет «заблудился» и его данные нельзя использовать; пакет уничтожается. Просматривая стек статистики муравья, маршрутизатор обновляет таблицу маршрутизации с помощью функции корректировки. Выбор функции корректировки значительно влияет на эффективность алгоритма. В представленном исследовании использовалась формула (1) [9].

$$P_{тек} = P_{тек} + (1 - r') \cdot (1 - P_{тек}), \quad (1)$$

где, $P_{тек}$ - текущая вероятность перехода по интерфейсу, с которого пришел B-муравей; r' - коэффициент, зависящий от времени, затраченное пакетом на путь от текущего узла к узлу назначения. Коэффициент r' вычисляется по формуле (2):

$$r' = \begin{cases} \frac{T}{c\mu} & \text{если} \frac{T}{c\mu} < 1 \\ 1 & \text{иначе} \end{cases}, \quad (2)$$

где, T, время перехода к узлу назначения, взятое из стека пакета; μ - среднее время, которое затрачивали предыдущие пакеты на этот переход; c - масштабирующий коэффициент, принимается равным 2 [1].

После корректировки маршрутной таблицы производится проверка на равенство IP-адреса текущего маршрутизатора и адреса назначения муравья. При выполнении этого условия пакет удаляется, иначе он отправляется на узел, указанный следующим в стеке статистики муравья.

Кроме обработки данных, маршрутизатор, реализующий алгоритм AntNet должен периодически отправлять в сеть F-муравьев. На рисунке 1 за это отвечает блок «генератор F-муравьев». При генерации муравья адрес назначения выбирается случайным образом из множества адресов известных узлов-маршрутизаторов. Также случайным образом выбирается интерфейс, по которому будет отправлен муравей.

3 Методика анализа производительности СП

Основу применяемого метода исследования составляет эмуляция выполнения приложения на эталонном процессорном ядре и последующий расчет общей производительности сетевого процессора [3, 4]. Обобщенная структура СП приведена на рисунке 2. Устройство состоит из однотипных вычислительных ядер, сгруппированных по кластерам. 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.

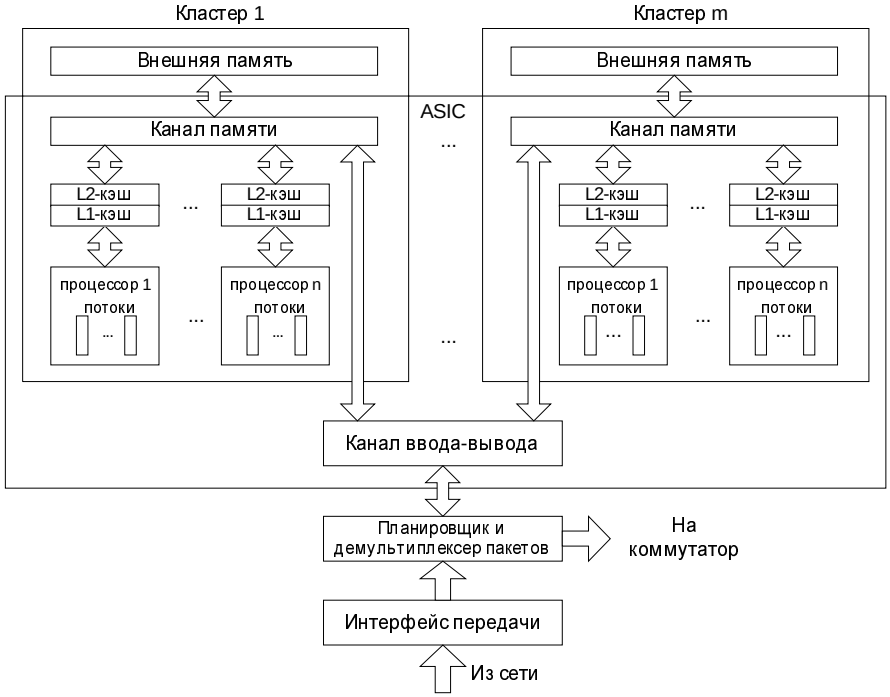


Рисунок 2 – Обобщенная структура сетевого процессора

Производительность сетевого процессора оценивается суммой производительностей его вычислительных ядер:

$$IPS = \sum_{j=1}^{m} \sum_{k=1}^{n} \rho_{p_{j,k}} \cdot f_{p_{j,k}} \qquad (2)$$

где $\rho_{p_{j,k}}$ - производительность ядра (j, k), команд/такт, $f_{p_{j,k}}$ - тактовая частота процессора, такт/секунда, m – число кластеров в СП, n – количество процессорных ядер в каждом кластере. Для удобства представления результатов, в статье производительность указывается в миллионах операций в секунду (MIPS).

Производительность отдельного многопоточного процессорного ядра с кэшами данных и команд определяется по формуле [7]:

$$\rho_p(u) = 1 - \frac{1}{\sum_{i=0}^{u} \left(\frac{1}{p_{miss} \tau_{mem}} \right)^i \frac{u!}{(u-i)!}} \qquad (3)$$

где ρ_p - загруженность вычислительного ядра, u - число аппаратно поддерживаемых потоков, p_{miss} - совокупная 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 ($\tau_{transmit}$):

$$\tau_{mem} = \tau_Q + \tau_{DRAM} + \tau_{transmit} \quad (4)$$

Интенсивность промахов кэшей зависит от приложения, выполняемого на СП:

$$p_{miss} = mi_c + (f_{load} + f_{store}) \cdot md_c \quad (5)$$

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.

Данные об интенсивности промахов кэшей получаются с помощью имитационного моделирования работы вычислительного ядра на системе SimpleScalar [6]. Система SimpleScalar моделирует выполнение программ на RISC-процессоре, она позволяет определить частоту обращений к ОЗУ, интенсивность промахов кэша и другие характеристики приложения.

4 Результаты моделирования

В представленном исследовании рассматривалась работа приложения AntNet на однопрецессорном однопоточном СП. Размеры кэшей данных и команд варьировались от 1 до 1024 КБ. При моделировании использовалась следующие параметры конфигурации маршрутизатора: 200 сетевых интерфейсов, маршрутная таблица содержит 3200 записей, IP-адрес: 194.156.45.1/24. Все соседние маршрутизаторы (соединенные с сетевыми интерфейсами моделируемого устройства) входят в сеть 194.0.0.0, длина префикса их IP-адресов определяется случайным образом по равномерному закону и принадлежит промежутку [8, 24].

При заполнении таблицы маршрутизации IP-адреса известных узлов генерируются случайным образом и при генерации их масок используется тот же принцип, что и при генерации масок соседних маршрутизаторов. Сгенерированные данные добавляются в маршрутную таблицу после адресов соседних маршрутизаторов.

На рисунке 3 показана зависимость интенсивности промахов кэшей от их размеров. Из графиков видно, что алгоритм AntNet не требователен к объему кэша команд. Динамика изменения интенсивности промахов кэша данных показывает, что для исследуемого алгоритма объем кэша команд не имеет большого значение и уже при 8 КБ интенсивность промахов стремится к нулю.

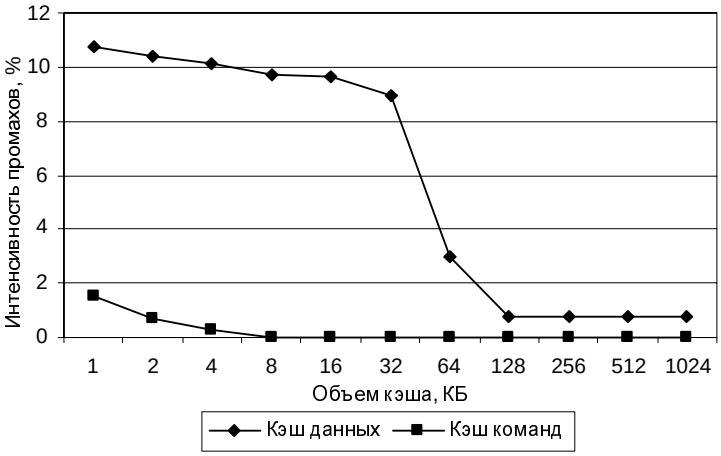
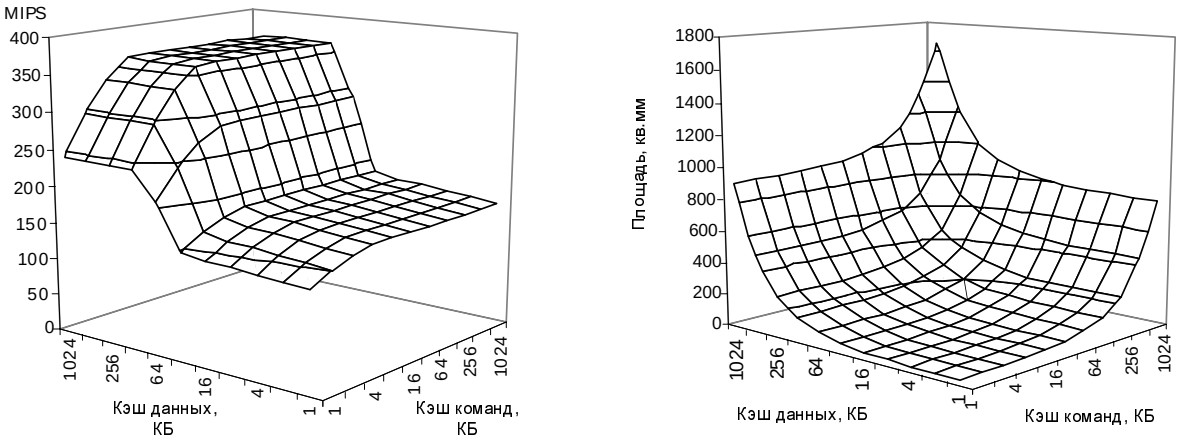


Рисунок 3 – Зависимость интенсивность промахов кэшей от их размеров



а) б)

Рисунок 4 – Зависимость производительности процессора (а) и его площади (б) от размеров кэшей

В гораздо большей мере алгоритм AntNet использует кэш данных. Как видно из рисунка 3, интенсивность промахов кэша данных значительно снижается только начиная с 64 КБ и достигает минимума на 128 КБ. Это показывает, что алгоритм AntNet интенсивно использует оперативную память и малые объемы кэша данных не позволяет качественно обрабатывать эти запросы.

Рисунок 4.а демонстрирует зависимость производительности процессора, измеряемой в миллионах операций в секунду (MIPS), от размеров кэшей. Здесь полностью подтверждаются данные, представленные на рисунке 3. При увеличении объема кэша данных от 32 до 128 КБ наблюдается более чем двукратный рост производительности устройства, в то же время использование кэша инструкций размером более 16 КБ не дает преимуществ. Но увеличение площади кристалла при

увеличении объемов кэша (см. рис. 4.б) может существенно снизить выигрыш от увеличения производительности.

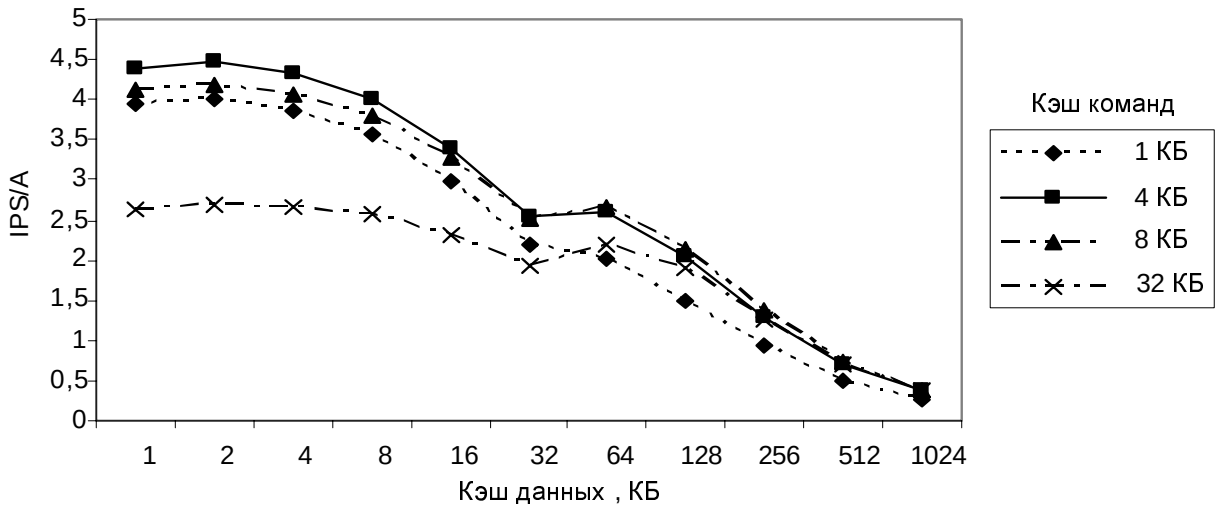


Рисунок 5 – Зависимость комплексной метрики MIPS/A от размеров кэшей данных и команд

На рисунке 5 показана зависимость комплексной метрики IPS/A от размеров кэшей. Метрика IPS/A показывает эффективность процессора с точки зрения производительности и площади кристалла. Различными линиями показаны кривые для разных размеров кэша команд. Наибольшей эффективности структура достигает при кэше команд объемом 4 КБ и кэше данных — 2КБ. Таким образом, хотя увеличение объема кэша данных до 128 КБ даст двукратное увеличение производительности, рост площади кристалла снижает это преимущество. При проектировании устройства может оказаться, что площадь кристалла не имеет критического значения и использование большого кэша данных будет невысокой ценой за большую производительность.

Выводы

Исследованы свойства реализации алгоритма маршрутизации AntNet на одноядерном однопоточном сетевом процессоре. Определено, что при такой конфигурации наивысшую эффективность имеет структура с объемом кэша команд, равным 4КБ, и кэшем данных размером 2КБ. Показано, что увеличение кэша данных до 128 КБ дает двукратный рост производительности по сравнению с оптимальным значением, что можно использовать в случае, если высокая производительность процессора имеет больший приоритет по сравнению с его геометрическими размерами.

В дальнейших работах планируется исследовать эффективность выполнения приложения AntNet на многоядерном и многопоточном СП. Кроме того, планируется рассмотреть ресурсоемкость модифицированной

версии алгоритма, представленной в работе [5].

Литература

1. Di Caro G. Dorigo. M. AntNet: Distributed Stigmergetic Control for Communications Networks: Journal of Artifical Intelligence Research. – 1998. - №9 – pp. 317-365.
2. Ладыженский Ю.В., Мирецкая В.А. Исследование муравьиных алгоритмов маршрутизации в компьютерных сетях. ИНТЕРНЕТ-ОБРАЗОВАНИЕ-НАУКА, пятая международная конференция ИОН-2006. 10-14 октября, 2006. Сборник материалов конференции. Том 2. – Винница: УНИВЕРСУМ-Винница, 2006, стр. 375-377.
3. Tilman Wolf, Mark Franklin, “Performance Models for Network Processor Design” IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 6, pp. 548-561, Jun., 2006.
4. 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.
5. Ладыженский Ю.В., Мирецкая В.А. Исследование динамических алгоритмов маршрутизации в компьютерных сетях. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей Міжнародної науково-практичної конференції м.Запоріжжя. 13-15 квітня 2006 року / Під заг. ред. Д.М. Пізи. – Запоріжжя: ЗНТУ, 2006, стр. 163-165.
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. Anant Agarwal, “Performance tradeoffs in multithreaded processors,” IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 5, pp. 525–539, Sept. 1992.
8. Цилькер Б. Я., Орлов С. А. Организация ЭВМ и систем - СПб: Питер, 2006. - 668 с.
9. Benjamin Baran, Ruben Sosa. AntNet Routing Algorithm for Data Networks based on Mobile Agents. Inteligencia Artifical, Revista Iberoamericana de Inteligencia Artifical. No 12 (2001), pp 75-84.