ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ НА ТЕСТАХ С НЕРЕГУЛЯРНЫМ ДОСТУПОМ К ПАМЯТИ

А.А. Корж

Введение

Одной из проблем, снижающих производительность современных коммерческих процессоров на ряде вычислительных задач является так называемая «стена памяти». Принято говорить, что в последние несколько десятков лет пиковая производительность процессора росла гораздо быстрее, чем пропускная способность памяти. Расхождение в скорости процессора и его подсистемы памяти приводит к тому, что на ряде задач процессор не успевает получать данные из памяти, и по этой причине не может обеспечить полную загрузку своих функциональных устройств. На некоторых задачах процент реальной производительности от пиковой может быть ниже чем единицы процента. Для решения этой проблемы, а также конечно проблемы большой задержки выполнения операций, в последние годы использовались архитектуры с огромными размерами кэш-памяти. Использование кэшей основано на факте, что большая часть задач имеет довольно хорошую локализацию данных, таким образом данные с которыми работает задача будут как правило находиться в кэше, пропускная способность которого гораздо выше, чем системной памяти. Поначалу это давало отличные результаты, и размеры кэшей становились все больше и сложнее, придумали иерархическую структуру организации кэш-памяти. В настоящее время кэш третьего уровня используется почти во всех современных коммерческих процессорах. Однако, оказалось, что не все задачи эффективно используют кэш. Ряд задач оказалось довольно легко «оптимизировать», для других пришлось модифицировать алгоритмы, но все равно остались задачи, для которых кэш не помог преодолеть «стену памяти». Эти задачи обычно относят к классу DIS (Data Intensive Systems) [1]. Они имеют плохие показатели пространственно-временной локализации обращений к памяти и зачастую работают с мелкими блоками данных, меньшими, чем оптимальная длина DDR-транзакции в 8 слов. Сюда относятся, например, задачи оперирующие с разреженными матрицами огромного размера, с графами или огромными списками.

С другой стороны декларируемая пиковая пропускная способность современных подсистем памяти достигается (на практике не более чем 70-80 процентов) лишь при выполнении следующих условий: чтение данных подряд и большими блоками (как правило кратно 8-ми 64-битным словам). Для чтений длиной в одно слово, число транзакций в секунду такое же, а пропускная способность в 8 раз меньше пиковой. Также чтение данных должно происходить подряд из одной строчки памяти, иначе будет происходить постоянное переключение строк. То есть лучше читать подряд, чем «скакать» по памяти.

Следует упомянуть и другой подход к организации подсистемы памяти, получивший распространение в современных мультитредовых графических процессорах (GPU). В архитектуре GPU от компании Nvidia предпочли отказаться от подхода суперскалярных процессоров, при котором более половины кристалла отводится под иерархию кэш-памятей. Вместо кэша в освободившееся место на кристалле было размещено множество функциональных устройств, а также реализована простейшая аппаратная поддержка переключения тредов (в каждом ядре до 24-32 активных треда, называемых в CUDA warp, при общем количестве ядер до 32). Идея толерантности, на которую опирается данная архитектура, довольна проста: пока один тред ожидает данных, конвейер выдает команды с другого треда, который в силу своей SIMD-сущности может сразу загрузить все функциональные устройства ядра. Толерантность позволяет игнорировать проблему огромной задержки операций чтения из памяти. Из-за отсутствия кэш-памяти с огромной пропускной способностью, для преодоления «стены памяти» подсистема памяти была ускорена за счет использования до 8 DRAM-контроллеров. Это дает пиковую пропускную способность современных GPU на уровне 100 ГБ/с. Для сравнения, суперскалярные процессоры имели пиковую пропускную способность памяти около 12 ГБ/с (двухканальная DDR2 800), и только последние процессоры Intel имеют 32 ГБ/с (трехканальная DDR3 1333). Впрочем, по ряду причин такое преимущество в пропускной способности канала к памяти не сказалось на увеличении производительности задач DIS-класса. В частности, для получения большой пропускной способности требуется синхронная выдача тредом обращения к памяти шириной в 16 слов. Возможно это ограничение заложено искусственно, но в настоящее время автору не удалось получить высокую пропускную способность памяти GPU от Nvidia на нерегулярных шаблонах доступа к памяти.

Постановка задачи

Целью данной работы является исследование зависимости производительности задач DIS-класса от устройства подсистемы памяти. Очевидно, что в отличие от задач близких к LINPACK, производительность на которых измеряется количеством выполненных операций с плавающей точкой двойной точности разделить на время, DIS-задачи бессмысленно измерять операциями с плавающей точкой — например в некоторых задачах их может и не быть вообще. В DIS-задачах производительность ограничивается скоростью работы подсистемы памяти, поэтому число операций с плавающей точкой объективной характеристикой системы, не зависящей от

задачи не является: в задаче в которой на такое же количество чтений выполняется в два раза больше операций, FLOPS будет в два раза больше на той же системе. Для оценки производительности систем поэтому предложена другая метрика, аналогичная Флопсам для вычислительных задач. Известный тест RandomAccess, имеющий худший шаблон нерегулярного доступа к памяти, измеряет производительность суперкомпьютера в GUPS (Giga Updates Per Second). Суть теста описана в [2] и заключается в том, что нужно выполнить большое число апдейтов по случайным адресам, где под апдейтом понимается чтение слова, изменение его значения и запись, причем работа идет с максимально доступной областью памяти, таким образом и временная и пространственная локализация данного теста минимальны.

Теоретическая оценка GUPS

Первоначально можно предположить, что пиковое значение GUPS ограничивается возможностями контроллера памяти. Например, для процессоров с двухканальным контроллером памяти DDR2 800, получается 200 миллионов транзакций (1-64 байта) в секунду. Один апдейт содержит две операции: чтение и запись. Получаем оценку в 100 MUPS. Ознакомившись с принципами устройства памяти и учтя случайный характер теста очевидно, что после каждой операции контроллеру придется выдавать команду закрытия строки памяти и активации другой строки. Данные потери зависят от различных параметров памяти, таких как tCAS, tRAS, и.т.п., но обычно составляют приблизительно 40-50 процентов. Таким образом, пиковая скорость вышеприведенного контроллера по нашей оценке составляет 50 MUPS. Здесь следует упомянуть, что мы не учитываем потери от переключений с чтения на запись и обратно, так как современные контроллеры умеют переставлять запросы, собирая вместе записи и чтения.

Другим узким местом может стать ограниченная возможность ядер процессора выдавать операции чтения: ограничивает размер буфера кэш-промахов. При этом каждое обращение выполняется несколько сотен тактов, таким образом пиковый темп выдачи чтений будет ограничен задержкой разделенной на размер буфера кэш-промахов. Однако современные процессоры содержат несколько ядер, что позволяет увеличить совместный темп выдачи. Вопрос для исследования заключается в том, хватит ли темпа выдачи, чтобы полностью загрузить контроллер или буфер кэш-промахов окажется «узким горлом»?

Проведенные эксперименты

Сначала были проведены эксперименты с помощью реализации теста Randomaccess входящей в пакет HPC Challenge. Результаты оказались значительно хуже, чем теоретическая оценка полученная выше. Поэтому, для исследований был написан собственная реализация теста RandomAccess использующая ту же последовательность адресов, что в оригинальной реализации. Новая реализация позволила нам явно управлять запускаемыми тредами, привязывать их к ядрам и к процессорам на NUMA-системах, управлять размером области памяти и ее размещением в случае NUMA-систем. А также было сделано три версии теста, одна выполняет апдейты (чтение плюс последующая запись), вторая выполняет только чтения, третья — только записи.

Использование версии выполняющей записи по случайным адресам должно было показать возможности контроллера, так как запись не является информационно блокирующей операцией такой, как чтение и поэтому темп выдачи записей ядром потенциально неограничен и на практике значительно выше, чем у чтений. Однако учитывая наличие кэш-памяти, все записи инициируют чтение кэш-строчки из памяти, после чего запись выполняется в кэш. Сброс кэш-строки происходит по мере вытеснения новыми запросами из-за политики кэширования Write-Back. Таким образом, вместо запрошенной записи 8 байт, контроллер выполняет две транзакции: чтение и запись 64 байт, что эквивалентно действиям выполняемым при апдейте. Поэтому третий вариант теста был модифицирован следующим образом: запись одного слова выполнялась ассемблерной вставкой с помощью SSE-инструкции non-temporal move MOVNTQ, которая выполняла запись минуя иерарию кэшей.

После внесения данной модификации, скорость теста случайной записи значительно выросла и стала выше скорости случайного чтения. Однако, было замечено, что с ростом размера используемой памяти скорость записи начинает падать. При отсутствии влияния кэша это было странным. После профилировки теста с использованием аппаратных счетчиков процессора и библиотеки oprofile, было обнаружено, что число промахов TLB практически равно числу выполняемых записей. Оказалось, что объем TLB-кэша современных процессоров составляет не более 1000 ячеек, что в случае использования страниц размером 4КБ покрывает не более 4МБ памяти. Таким образом, при случайной работе с массивом размером в несколько гигабайт, вероятность TLB-промаха близка к 100%.

Чтобы уменьшить влияние промахов, при выделении памяти были использованы страницы размером 2МБ. Несмотря на то, что результаты улучшились в несколько раз, аппаратные счетчики профилирования показывали, что для процессора Intel Q6600 TLB-кэш 2МБ страниц покрывает массив размером в 64MB. Для массива 128MB число промахов составило 50%, для 256MB — 75%, и.т.п. Для массива в несколько гигабайт число промахов практически не изменилось, хотя производительность и выросла в несколько раз. Для объяснения этого факта был проверен аппаратный счетчик PAGE_WALKS показывающий сколько тактов процессора было потрачено на выбор дескриптора из многоуровневой таблицы страниц. Данное время

оказалось на два порядка больше для случае с 4 КБ страницами. Это объясняется двумя причинами: большая иерархия таблиц трансляции, а также тот факт, что суммарный объем всех таблиц страниц занимает в памяти несколько мегабайт, не помещаясь в кэш память процессора. А это значит, что на каждую операцию чтения мы получали несколько дополнительных чтений дескрипторов таблиц из памяти, таким образом, контроллер памяти становился еще более узким местом. Дополнительно следует отметить, что в процессорах AMD имеется возможность использования страниц размером в 1 ГБ, в данном случае промахов TLB не происходит, однако на результатах теста RandomAccess это никоим образом не сказалось. По-видимому, страниц в 2МБ достаточно, чтобы при наличии трех-четырех ядер упереться в узкое место - контроллер памяти.

Еще один интересный момент был обнаружен на тесте случайная запись. Упомянутый выше прирост при введении записи с помощью команды MOVNTQ был достигнут только на процессоре Intel Clovertown, а на процессорах AMD Barcelona и Intel Nehalem EX использование команды MOVNTQ давало на порядок худший результат. Для эксперимента, тест был модифицирован таким образом, что вместо записи блока в 8 байт выполнялась запись выровненного блока в 64 байта с помощью 8-ми команд MOVNTQ (или четырех команд MOVNTPS), после которых шла инструкция SFENCE. Результат данного теста оказался выше, (причем измеряемый в транзакциях в секунду) чем при записях по 8 байт. Отсюда можно сделать вывод, что современные процессоры и их подсистема памяти не оптимизированы под транзакции по одному слову, а предпочитает транзакции размером в кэш строку 64 байта.

Кроме теста RandomAccess исследовался тест APEX-поверхность [1], показывающий производительность подсистемы памяти в зависимости от пространственно-временной локализации. В крайнем случае и пространственной локализации (обращения размером в одно слово) и временной локализации (адрес полностью случаен) APEX повторяет шаблон теста RandomAccess. Отличия следующие: использование дополнительного массива со случайными индексами, выполнение чтений, а не апдейтов, развертка цикла на 4 итерации в оригинальном исполнении теста. Влияние дополнительного массива незначительно, так как его размер 1024 элемента и он прекрасно умещается в кэш-памяти. Исполнение только чтений снижает нагрузку на контроллер памяти в два раза, так как при вытеснении кэш-строки не происходит ее запись в память, как это происходило в случае с апдейтом. Развертка вносит небольшой интересный эффект на некоторых процессорах, как будет видно далее.

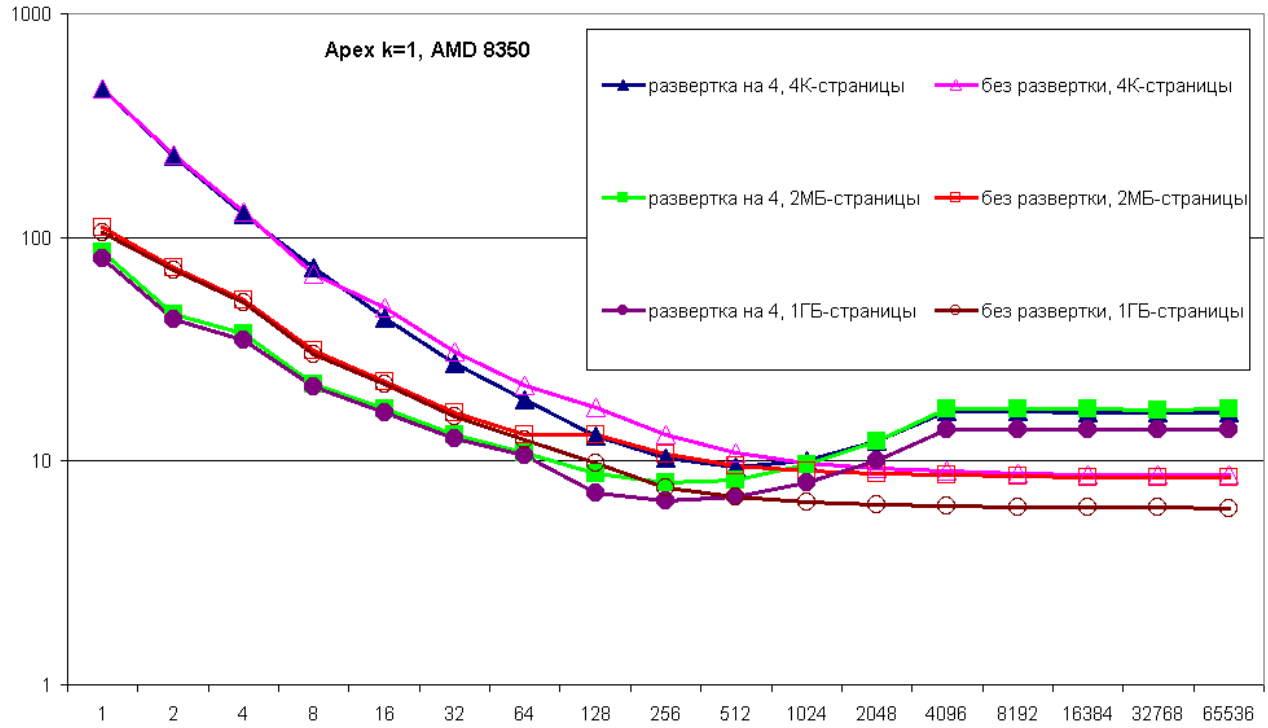

Рис.1 Срез APEX-поверхности при k=1 для одного ядра Intel Q6600. Изображена зависимость темпа исполнения обращений к памяти (в тактах процессора) в зависимости от размера блока L (в 64-битных словах)

Для нас интерес представлял край APEX поверхности при худшей временной локализации (k=1). Единственным параметром является размер блока L, измеряемый в 64-битных словах от 1 до 65536. В зависимости от него откладывался так называемый темп выполнения операций в тактах процессора равный времени выполнения N операций чтения деленому на N, то есть фактически величина обратная к пропускной способности. На Рис.1 и Рис.2 показаны результаты теста APEX поверхность для процессоров Intel Q6600 и AMD Barcelona. На графике относящемся к процессору AMD видно седло, так что начиная с некоторого момента наступает ухудшение темпа выдачи. Данное седло довольно отчетливо видно и на всей поверхности в

том числе и для более старых суперскалярных процессоров [1], кроме моделей последних лет от компании Intel. Алексеем Соколовым была высказана мысль, что седло связано с раскруткой цикла в реализации теста. Автором данная идея была проверена путем сравнения результатов оригинальной версии и модифицированной версией теста, из которой раскрутка была убрана. Как мы видим, седло, проявлявшееся на процессоре AMD, исчезло. После изучения спецификации оказалось, что кэш первого уровня данного процессора имеет двухканальную ассоциативность, и в случае развертки на 4 при большом размере блока возрастает число промахов в кэш первого уровня, так как одни и те же ячейки разных блоков постоянно выбивают друг-друга из кэша. Современные процессоры Intel имеют 8-16 канальную ассоциативность кэша первого уровня, поэтому для них данное седло не наблюдается.

Также были проведены исследования NUMA-систем, имеющих несколько процессоров, а стало быть и контроллеров памяти. Подсистема памяти таких процессоров начинает зависеть от пропускной способности используемого межсокетного интерконнекта (Hypertransport для AMD и QuickPath для Intel). Подробные результаты исследования данных систем выходят за рамки данной статьи, скажем лишь, что узким местом на данном тесте становились линки межсокетного интерконнекта. С помощью библиотеки numactl имелась возможность привязывать треды и память к определенным ядрам/процессорам и за счет этого измерять различные компоненты подсистемы памяти. Доступны режимы измерения: треды и память находятся в одном сокете, треды в одном сокете, а память в другом, треды в одном сокете, а память равномерно распределена по нескольким сокетам, треды в нескольких сокетах, а память в одном, и треды и память распределены по всей системе(полная загрузка системы). Для полной загрузки системы были получены следующие результаты: 0.124 GUPS для 4-х процессоров AMD Barcelona с памятью DDR2 667, 0.247 GUPS для двух процессоров Intel Nehalem EX с памятью DDR3 1333. Для сравнения, на GPU G80 автору удалось получить результат всего в 0.09 GUPS при сравнимой с двумя Nehalem пиковой пропускной способности памяти.

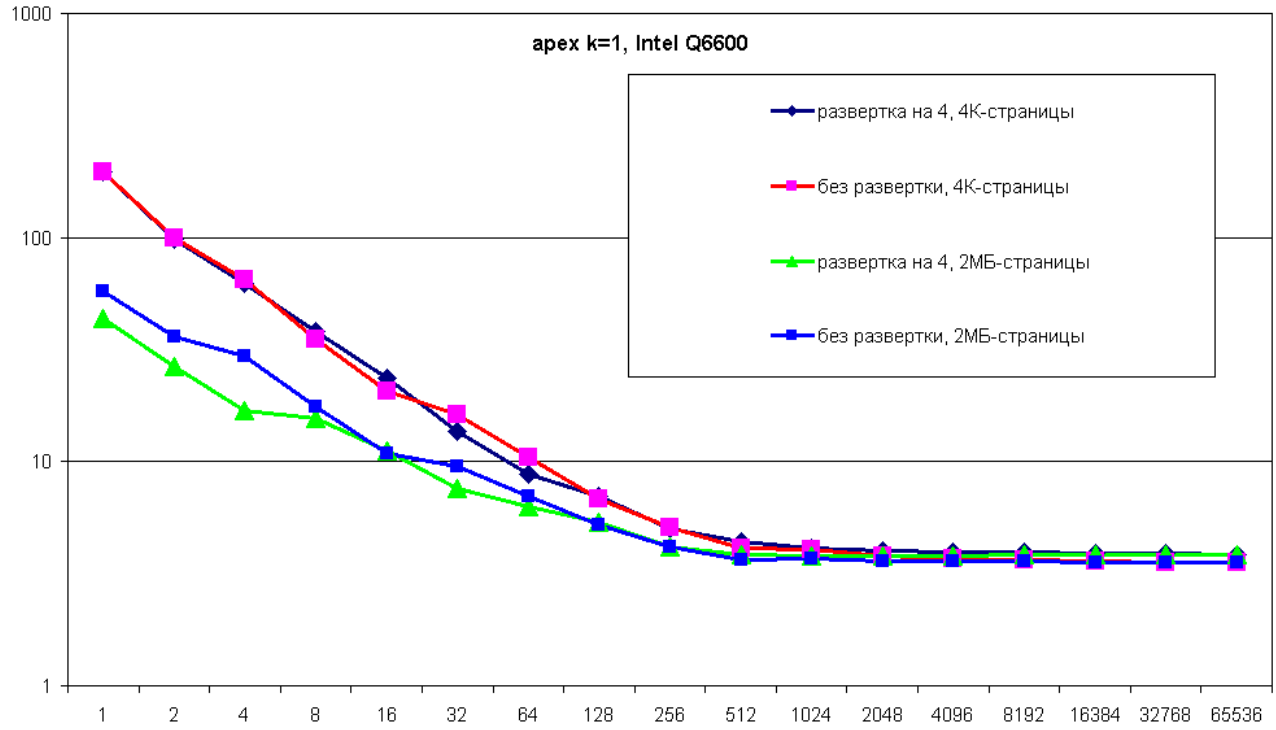

Рис.2 Срез APEX-поверхности при k=1 для одного ядра AMD Barcelona 8350.

Выводы

Результат теста GUPS современных многоядерных процессоров зависит в первую очередь от используемого типа памяти. Современные многоядерные процессоры позволяют загрузить свою подсистему памяти на шаблоне RandomAccess, причем для насыщения одного контроллера достаточно трех-четырех ядер. Теоретическая оценка возможностей современных контроллеров равна количеству транзакций в секунду разделить на 4 для апдейтов и 2 для чтений. Например, для двухканального контроллера DDR2 800 имеем 100 млн транзакций/с для каждого канала, итого 50 MUPS. Реально достигается 44 MUPS или 89 миллионов чтений в секунду на процессоре Intel Q6600 и 49 MUPS для процессоров AMD Barcelona. Для трехканального контроллера DDR3 1333 имеем 166 млн транзакций на канал, то есть 125 MUPS и 250 млн чтений в секунду. Реально достигается 120 MUPS и 240 млн чтений с секунду на процессоре Intel Nehalem EX. Причем важно отметить, что данный результат достигается при использовании лишь одного треда из двух в каждом из четырех

ядер, и не становится лучше в случае использования всех 8-ми тредов. Это значит, что технологию SMT при программировании DIS-задач можно использовать следующим образом: один тред подкачивает нерегулярно размещенные данные в памяти, в то время как другой занимается вычислениями и не простаивает на доступе к данным.

Исходя из указанных цифр, преимущество подсистемы памяти последнего поколения процессоров Intel Nehalem над собственными процессорами прошлого поколения и процессорами фирмы AMD составляет несколько раз не только по пиковой пропускной способности, но и по эффективности нерегулярного доступа.

Механизм страничной виртуальной памяти может в несколько раз понизить производительность в случае использования страниц стандартного размера 4 КБ. Для массивов большого размера, доступ к которым будет иметь плохую пространственно-временную локализацию рекомендуется выделять память страницами размера 2МБ или даже 1 ГБ, если это позволяет аппаратура. В целом, наблюдается нецелесообразность использования механизмов защиты и виртуализации памяти на вычислительном узле кластера, на котором в один момент времени выполнятся не более одной задачи.

Современные процессоры в NUMA-системах не оптимизированы на выполнение записей одиночными словами, поэтому целесообразно применять грануляцию и выравнивание данных по кэш-строкам. Целесообразно организовывать алгоритмы таким образом, чтобы нерегулярным было чтение, а запись осуществлялась по последовательным адресам.

Результат GUPS для NUMA-систем зависит от скорости межпроцессорного интерконнекта. Результат GUPS многоузловых кластеров с распределенной общей памятью зависит только от применяемой коммуникационной сети и сетевого интерфейса коммуникационной сети. Если подсистема памяти одного узла позволяет выполнять порядка сотни миллионов апдейтов, то коммуникационная сеть Infiniband QDR при использовании MPI ограничена одним миллионом 8-байтовых сообщений в секунду. В случае использования API нижнего уровня IB-Verbs получается менее 5 миллионов сообщений в секунду.

В связи с этим, при разработке СКСН Ангара ведущейся в НИЦЭВТ, большое внимание уделяется именно разработке коммуникационной сети, а самое главное интерфейсу процессора с сетью. Реализованный в макете коммуникационной сети 3D-тор, изготовленной в НИЦЭВТ, с маршрутизатором в ПЛИС, интерфейс с поддержкой глобально адресуемой памяти уже сейчас позволяет достигать скорости 8 миллионов сообщений в секунду. При реализации в СБИС, ожидается увеличение данного показателя еще в 4-8 раз.

ЛИТЕРАТУРА:

1. Л. Эйсымонт, А. Семенов, А. Корж, А. Фролов «Программа создания перспективных суперкомпьютеров», Открытые Системы, 2007, №9, http://osp.ru/os/2007/09/4566841/
2. Д. Волков, А. Фролов «Оценка быстродействия нерегулярного доступа к памяти» // Открытые Системы, 2008, №1, http://osp.ru/os/2008/01/4836914/