ядра, процессоры, кластеры и все-все-все

крис касперски, no-email

спектр вычислительных машин, представленных на рынке, довольно широк и простилается от многоядерных процессоров до кластеров (причем кластеры не образуют монолитный блок, а делятся на типы и подтипы). большинство многопроцессорных систем предназначено для решения узкого круга задач и разница между ними не только количественна (стоимость/производительность), но и качественная (отказоустойчивость, распределенность в пространстве). многопроцессорных систем общего назначения не существует, даже если они и позиционируются производителем в качестве таковых. не верьте! производители заблуждаются. факты и только факты!

введение

Компьютер — это не роскошь, а средство обработки информации. Производительность — комплексная велича и выражать ее в скалярных единицах некорректно. Давайте попробуем сравнить "производительность" КАМАЗа с "Формулой-1". Какой результат мы ожидаем получить? И какую (практическую) пользу сможем из него извлечь? Очевидно, что существуют разные классы машин и даже в рамках одного класса (легковые автомобили) сравнивать внедорожник с каким-нибудь "мини" совершенно бессмысленно и еще более бессмысленно пытаться добраться до Хацепетовки на Лимизине.

Среди однопроцессорных систем достаточно много машин универсального назначения, которые могут работать и как офисный компьютер, и как сервер начального уровня, но вот многопроцессорные машины в силу фундаментальных ограничений, присущих параллельным системам, обладают ярко выраженной харизмой, т. е. уникальным набором технических характеристик, отличающих их от всех остальных.

Эффективность многопроцессорной системы в первую очередь зависит от адекватности соответствия поставленной задаче выбранной машинной архитектуре. К сожалению, производители "железа", в стремлении привлечь как можно больше клиентов, используют все уловки, чтобы "впарить" им более дорогое решение, даже если оно наименее адекватное. Ведь оценить эффективность использования приобретенной системы рядовой потребитель все равно не сможет. Исключения составляют случаи, когда приобретаются несколько систем различной архитектуры для решения схожих или идентичных задач. Вот тогда-то и наступает прозрение, но вложенные деньги, увы, уже не вернуть назад.

Какие типы многопроцессорных систем существуют в природе? И чем они отличаются друг от друга?!

Рисунок 0 фото автора

многоядерные процессоры

Считается, что многоядерные процессоры обязаны своим появлениям на свет тому факту, что бесконечно увеличивать тактовую частоту невозможно, а повышать производительность как-то надо. На счет частоты — все верно. Производители уже подошли к тому барьеру, который им в ближайшие годы не преодолеть, но это не только не единственный, но и не основной мотив.

От чего зависит себестоимость процессора? При прочих равных — от площади кристалла, определяемой в свою очередь количеством транзисторов, львиная доля которых расходуется на кэш второго уровня, от размера которого производительность зависит даже больше, чем от тактовой частоты и который "съедает" практически всю площадь кристалла. Добавление еще одного вычислительного ядра (или даже нескольких ядер) практически не увеличивает площади кристалла, а, значит, позволяет продавать два процессора в одном корпусе по себестоимости одного.

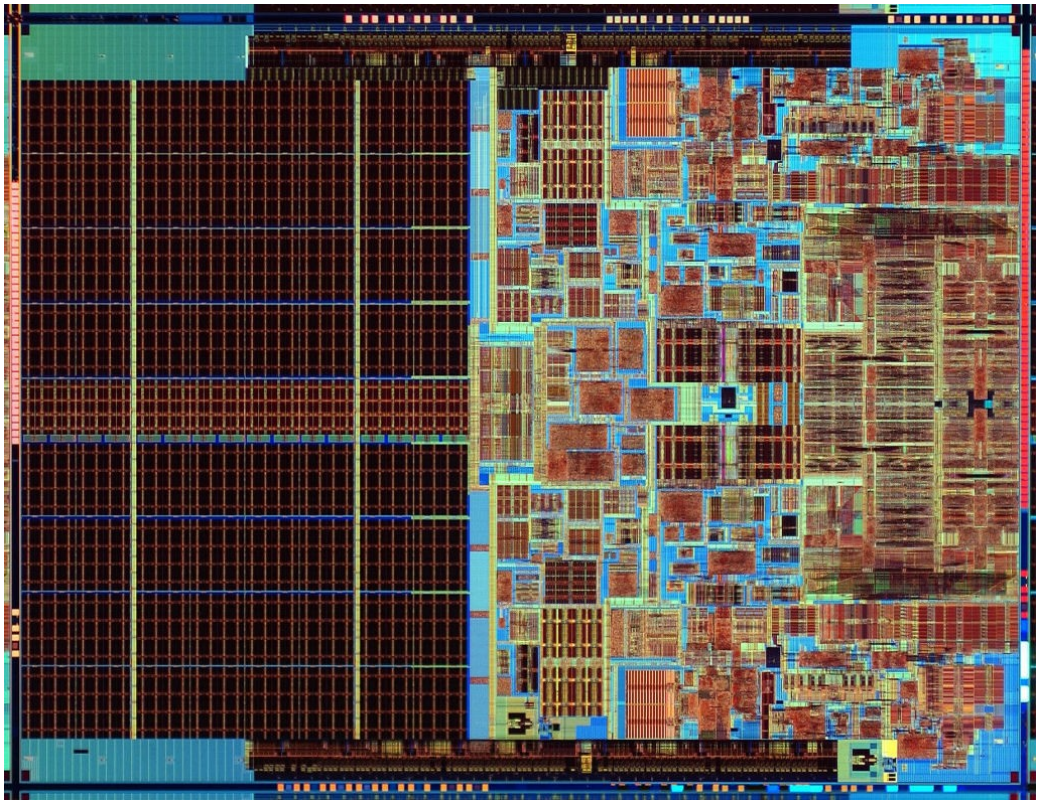

Рисунок 1 "обнаженный" Intel Core 2 Duo

А как на счет производительности? Вот тут-то и начинается самое интересное. Сравнивая быстродействие двуядерных процессоров с двумя одноядерными, различные группы "тестеров" получают сильно неодинаковые результаты, заставляющие усомниться в корректности тестов и беспристрастности самих экспериментаторов.

На самом деле никакого всемирного заговора тут нет. Возьмем задачу, хорошо поддающуюся распараллеливанию (ибо, если задачу вообще не удается распараллелить, то скорость ее выполнения от числа процессоров _никак_ не зависит), состоящую из двух подзадач, обрабатывающих огромные объемы данных в блочном режиме, запущенной на операционной системе, поддерживающей привязку потоков к "своим" процессам (например, Microsoft Server 2008). Поскольку, суммарный размер кэш-памяти у двух одноядерных процессоров вдвое больше, чем у одного двуядерного, то, производительность двуядерного процессора составит всего 50%.

Если же две подзадачи интенсивно обмениваются обрабатываемыми данными, то за счет совмещенного кэша второго уровня (исключающего межпроцессорный перегон данных), результат получится диаметрально противоположным, причем, поскольку пропускная способность системной шины много меньше пропускной способности кэш-памяти, быстродействие двухпроцессорной системы рискует составить 30% или даже 10% от быстродействия двуядерного процессора.

А вот при потоковой обработке данных (т. е. когда к каждой загруженной ячейке процессор обращается всего лишь раз), размер кэша уже не имеет решающего значения и производительность обоих систем будет близкой или полностью идентичной.

На практике _как_ _правило_ встречаются задачи, требующие интенсивного межпроцессорного взаимодействия и потому в _общем_ _случае_ приобщение многоядерной однопроцессорной машины намного более предпочтительно, чем многопроцессорной системы (да и стоит последняя обычно существенно дороже).

Недостаток многоядерных процессоров лишь один — они не масштабируются. То есть, практически не масштабируются. В принципе, если у нас установлен двуядерный процессор, мы можем вставить четырех ядерный (если материнская плата его поддерживает), но что делать, если для достижения заданной производительности необходимо 32 процессора?! Так что от истинно многопроцессорных систем все равно никуда не уйти, однако, следует помнить, что четырехпроцессорная машина с одноядерными процессорами в общем случае менее производительна, чем однопроцессорная с четырех ядерным процессором.

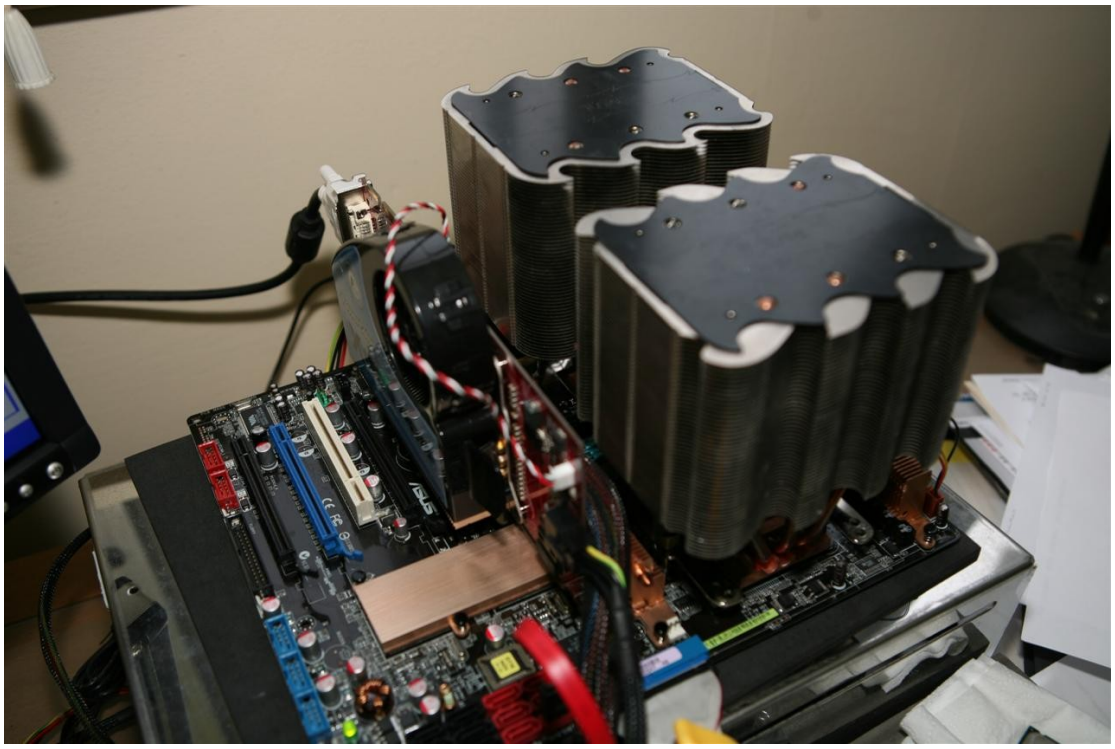

Рисунок 2 материнская плата с двумя двуядерными процессорами AMD Dual Core

А что на счет пары двуядерных процессоров? Ответ опять-таки зависит от специфики решаемой задачи. В некоторых случаях мы не заметим разницы по сравнению с четырех ядерным процессором, в некоторых — получим выигрыш или проигрыш. Имея многопроцессорную материнскую плату, поддерживающую многоядерные процессоры, мы можем приспособить ее для решения более широкого круга задач, чем однопроцессорную многоядерную или же многопроцессорную, но поддерживающую только одноядерные процессоры.

Бытует мнение, что от установки дополнительных процессоров хуже не стает. Производительность быть может и не возрастет, но ведь и не упадет же. Да как бы не как!!! Рассмотрим двухпроцессорную систему с двумя двуядерными процессорами, решающую задачу, характеризирующуюся интенсивным межпроцессорным обменом данных. Пока у нас стоял только один двуядерный процессор, данные гонялись внутри кристалла и все работало очень быстро. Но вот появился еще один процессор и, если программное обеспечение, решило задействовать и его, то система вынуждена прокачивать огромные объемы данных по системной шине, передавая их от одного физического процессора к другому и обратно. Как результат — мы имеем драматическое падение производительности, но стоит только выдернуть один процессор, как все будет ОК.

Проблема в том, что многое программное обеспечение еще не умеет отличать физические процессоры от "виртуальных", предполагая, что все процессоры равноправны, а в системе состоящей из нескольких многоядерных процессоров это не так. Выход из ситуации? Обычного программное обеспечение, сконструированное с учетом работы в многопроцессорной среде, имеет опцию, позволяющую задавать кол-во используемых процессоров (или кол-во вычислительных потоков, что в данном контексте одно и тоже). Устанавливаем кол-во процессов равному кол-во ядер на одном процессоре и тогда операционная система типа Microsoft Server 2008 приложит все силы к тому, чтобы потоки данного приложения выполнялись внутри одного процессора и как можно реже выходили за его пределы, высвобождая оставшиеся процессоры для других приложений.

Разумеется, не стоит слепо следовать этому решению, поскольку оно подходит далеко не для всех задач и узнать каким образом нам лучше всего оптимизировать многопроцессорную систему можно только изучив специфику используемых приложений. Обычно на сайте производителя (или в технической документации) содержатся советы по конфигурации. Если же многоядерные процессоры там явно не упоминаются, то это означает одно из двух: приложению действительно все равно какой это процессор — виртуальный или физический, поскольку

межпроцессорный обмен между ними отсутствует, а кэш загружен лишь на доли процента (что характерно для приложений с потоковой обработкой данных), либо же приложение писалось еще в те далекие времена, когда многоядерных процессоров просто не существовало и с тех пор никак не обновлялась разработчиками, никакие тесты не проводились и производитель сам не знает какая конфигурация окажется оптимальна, но тщательно скрывает свою некомпетентность, прячась за обтекаемыми фразами или же раздавая советы наугад.

Таковы реалии нашей жизни, увы.

кластеры

Два и более компьютеров, объединенный в единую вычислительную систему называют кластером (от английского "cluster" – скопление), а входящие в ее состав компьютеры узлами (или, по-английски, нодами — nodes). Другими словами, всякий кластер переставляет собой разновидность параллельной или распределенной системы, состоящей из нескольких связанных между собой компьютеров, используемых как единый вычислительный ресурс.

Узлы кластеры могут строиться как по одно- так и по многопроцессорной (многоядерной) схеме, а в качестве соединительных артерий используется локальная сеть, хотя в некоторых системах применяются более скоростные устройства для передачи данных. Большинство кластеров работают с разделяемой дисковой памятью, на которой хранятся обрабатываемые данные. Внутренние жесткие диски либо вообще отсутствуют, либо предназначены для сугубо утилитарных (системных) функций, например, Microsoft Server до сих пор не способен грузиться с read-only носителей и без винчестера здесь никак не обойтись, хотя для UNIX-систем это давно перестало быть проблемой. Они способы загружаться по сети, полностью размещаясь в оперативной памяти.

Кластеры выгодно отличаются от многопроцессорных компьютеров практически неограниченной масштабируемостью и намного более высокой отказоустойчивостью — в правильно сконструированной системе выход одного из узлов никак не отражается на остальных. Производительность кластера, конечно, снижается, но система продолжает функционировать даже при выходе значительного количества узлов, которые, кстати говоря, могут быть как сосредоточены в одном месте, так и географически разобщены.

Выделяют следующие типы кластеров: а) отказоустойчивые кластеры (High-availability clusters, HA), б) кластеры с балансировкой нагрузки (Load balancing clusters, LBC); в) высокопроизводительные кластеры (High-performance clusters, HPC); г) распределенные кластеры (Grid-системы);

Рисунок 3 кластер, установленный в университете McGill

отказоустойчивые кластеры

Отказоустойчивые кластеры (High-availability clusters, HA), они же кластеры высокой доступности, они же failover-системы главным образом применяются в критических инфраструктурах, к которым относятся базы данных, коммерческие web-сайты и многое другое.

Минимальный HA-кластер состоит из двух узлов, дублирующих друг друга и при выходе одного узла из строя, его функции берет на себя другой. Естественно, узлы HA-кластеров должны заботиться о поддержке когерентности, т. е. находится в согласованном состоянии, что технически очень-очень сложно, а практически вообще невозможно. Допустим, оба узла разделяют один и тот же жесткий диск (а, точнее, RAID-массив, поскольку дисковая подсистема так же должна быть построена по отказоустойчивой схеме).

Если узлы кластеры кэшируют дисковые операции (а они их кэшируют), то узел, считавший и модифицирующий дисковые данные, имеет шансы "лечь" раньше, чем он "сбросит" дисковый кэш или пошлет соседнему узлу сигнал о том, что эти данные уже более не когерентны.

Выход состоит в использовании транзакций — групповых операций чтения/обработки/записи данных, выполняемых как единичная операция, т. е. атомарно. Транзакция либо выполняется целиком, либо не выполняется вообще. Никакое промежуточное состояние не допускается. Если транзакция уже начала выполняется и тут произошел сбой, то система выполняет операцию "отката", возвращая данные в предыдущее состояние, в котором они были до модификации. Естественно, поддержка транзакций не проходит даром и производительность такой системы снижается (на некоторых задачах — весьма существенно),

но расплата за снижение производительности намного меньше убытков (потерянной выгоды) от простоя системы, поэтому, избыточность — это must have.

На рис. 5 показана схема WEB-сервера, построенного по кластерной отказоустойчивой схеме.

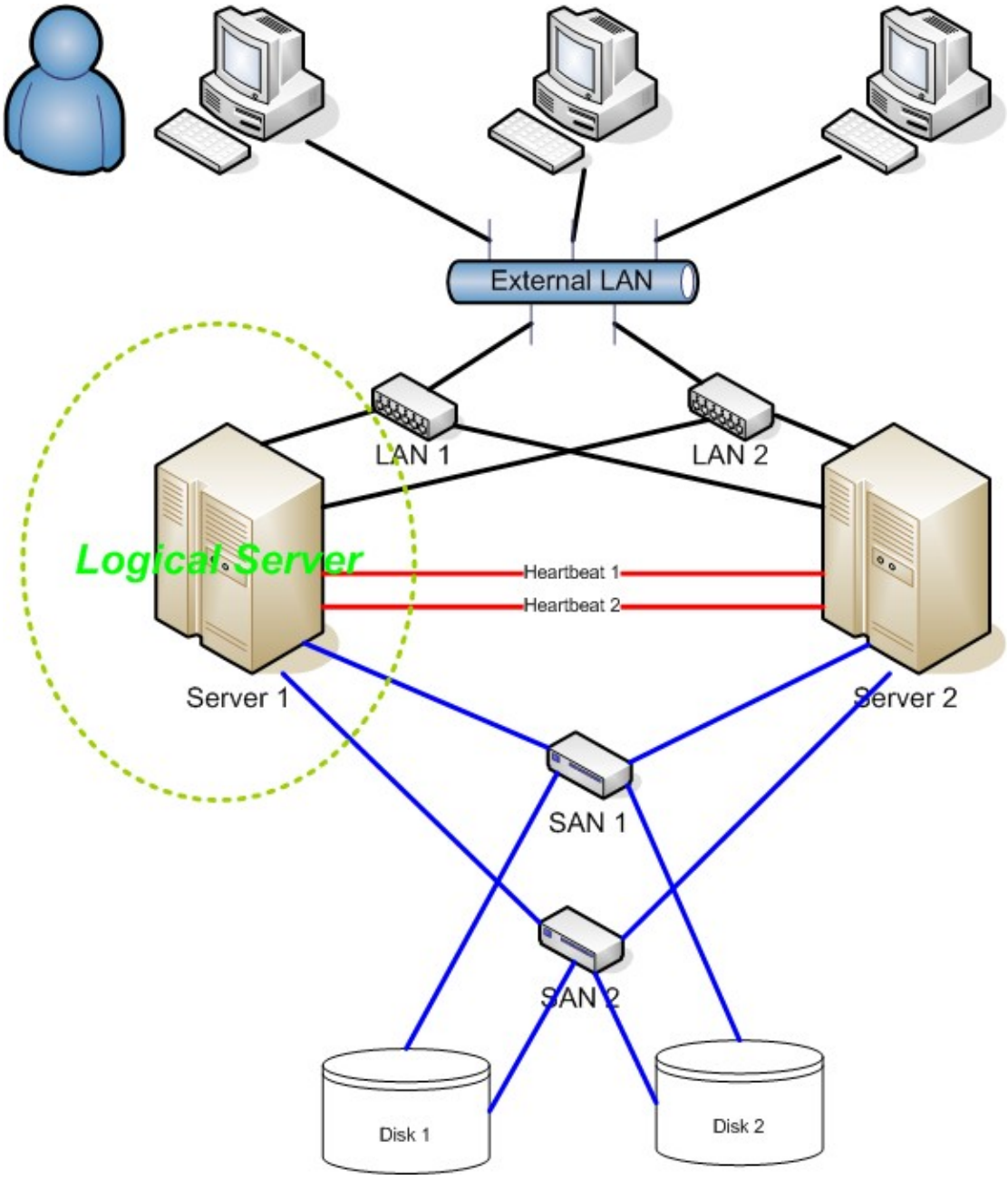

Рисунок 4 схема HA-кластера, состоящего из двух узлов и обрабатывающего сетевые запросы

Практически все современные операционные системы поддерживают HA-кластеризацию. Microsoft Windows (там эта технология называется **Microsoft Cluster Server**, или, сокращенно, MSCS) и в настоящий момент она поддерживается Server 2003 и Server 2008 (http://www.microsoft.com/windowsserver2003/ccs/default.aspx).

Linux так же поддерживает HA-кластеры, но требует установки специального программного обеспечения (бесплатного, разумеется), которое можно скачать с главной страницы проекта Linux-HA (http://www.linux-ha.org), и которое помимо, собственно, Linux'а поддерживает FreeBSD, OpenBSD, Solaris, Mac OS X и некоторые другие популярные операционные системы.

Компании Hewlett-Packard и Digital Equipment Corporation совместными усилиями создали специализированную операционную систему, ориентированную на работу в кластерной среде. Распространяемая по проприетарной лицензии с закрытыми исходными тестами, она, тем не менее носит гордое имя OpenVMS (http://www.hp.com/go/openvms) и в настоящее время поддерживает процессоры семейства VAX, Alpha и Itanium.

Остальные компании (IBM, Novell), так же предлагают собственные кластерные расширения для своих операционных систем, так что жаловаться на скудность выбора не приходится.

Компания VM-Ware предлагает систему поддержки виртуальных HA-кластеров (VM-Ware HA), построенных на базе одного физического узла — http://www.vmware.com/products/vi/vc/ha.html), что существенно снижает стоимость владения, но вот что касается устойчивости… От аппаратных отказов оборудования VM-Ware HA, естественно, не страхует, а чисто программная избыточность защищает лишь от сбоев самого программного обеспечения в _гостевых_ операционных системах. Сбой базовой операционной системы приводит к краху всех "гостевых" узлов кластера.

Как показывает практика, аппаратные сбои превалируют над программными и если 90% программных сбоев "лечатся" простой перезагрузкой, отнимающей минимум времени, то отказ "железа" влечет за собой необходимость его ремонта. Самое смешное, что за счет огромных накладных расходов на виртуализацию, стоимость двух физических узлов, обеспечивающих туже самую производительность, что и пара VM-Ware HA, находится приблизительно на одном уровне.

Впрочем, физические узлы, обеспечивая аппаратную избыточность, чрезвычайно уязвимы перед целенаправленными атаками на программное обеспечение. Если хакер "положит" один узел, то он положит и другой (а так же третий, четвертый…). Следовательно, без HA-системы предъявляются высокие требования к безопасности и, как показывает статистика, хакерские атаки случаются намного чаще, чем естественные отказы железа/ПО.

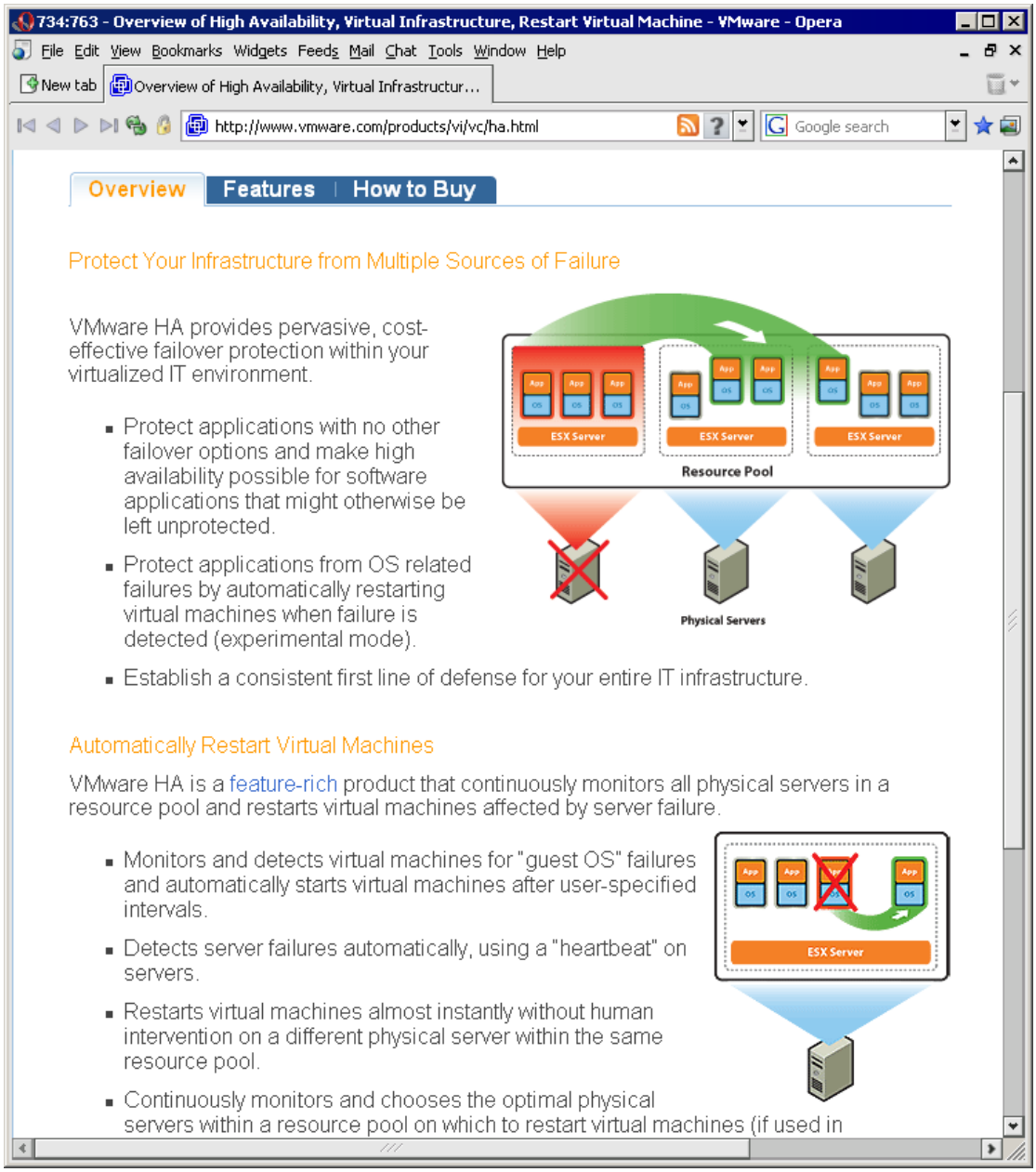

Рисунок 5 виртуальный кластер высокой доступности, построенный на базе VM-Ware HA с использованием всего одного физического сервера

кластеры с балансировкой нагрузки

Кластеры с балансировкой нагрузки (load balancing clusters или LBC), внешне очень похожи на HA-кластеры, однако, если LBC-кластеры автоматически являются и HA-кластерами, обеспечивая высокий уровень доступа, то обратное утверждение неверно и HA-кластеры не могут выступать в роли LBC.

Идея кластеров с балансировкой нагрузки очень проста. Если у нас имеется HA-кластер, узлы которого "тупо" дублируют друг друга (разделяя общую внешнюю память), то почему бы не распределить поступающие запросы равномерно между всеми узлами кластера? Пользователь, заходящий на WEB (например), автоматически перенаправляется на наименее загруженный узел, что позволяет всем узлам работать параллельно и легко масштабировать мощность кластера. Выход одного узла из строя не приведет к падению всей системы. Производительность кластера, конечно, уменьшится, но с этим можно жить!

Специально для этой цели DNS-сервера поддерживают технологию "Round robin DNS", связывающую с одним доменным именем список IP-адресов, соответствующих "своим" узлам кластера, которые перебираются в кольцевом порядке. Степень загрузки узлов при этом не

учитывается, поскольку внешний DNS-сервер ничего не может знать о ней, однако, внутренний маршутизатор (которому, кстати говоря, достаточно всего одного IP) может собирать данные со счетчиков производительности, выбирая наименее загруженный узел кластера.

Аналогичная технология используется для брандмауэров, установленных на магистральных каналов, антивирусов, спам-фильтров и многих других задач.

Единственной операционной системой, поддерживающей кластеризацию с балансировкой нагрузки, была и остается OpenBSD с установкой специального программного обеспечения (http://www.openbsd.org/cgi-bin/cvsweb/src/usr.sbin/relayd/), все остальные требуют аппаратной поддержки со стороны маршрутизаторов, что, впрочем, не является проблемой, поскольку, все крупные производители (3Com, Cisco) ее поддерживают.

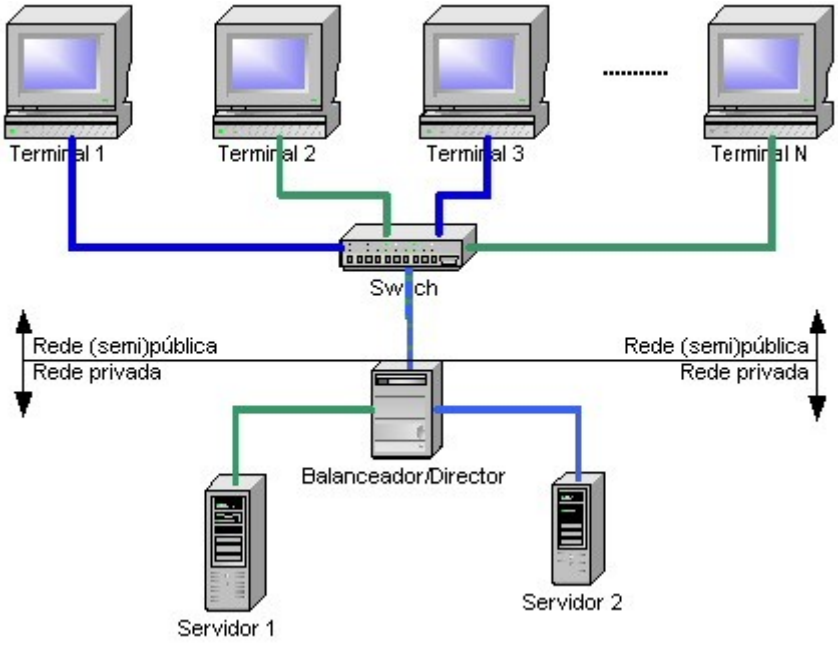

Рисунок 6 схема LBC-кластера, состоящего из двух узлов и обрабатывающего сетевые запросы

высокопроизводительные кластеры

Идеологически и архитектурно высокопроизводительные кластеры (High-performance clusters, или, сокращенно HPC) ближе всех стоят к многопроцессорным машинам, только вместо системной шины, соединяющей процессоры, у них локальная сеть, пропускная способность которой в разы ниже, а потому HPC-кластеры эффективны только на задачах, где "межусобный" обмен данными слаб или вовсе отсутствует (впрочем, с появлением гигабитного Ethernet'а сфера применения HPC-кластеров существенно расширилась).

Большинство HPC-кластеров строятся по Beowulf-технологии (http://www.beowulf.org/), разработанном в научно-космическом центре NASA в середине 1994 года, объединив в единую вычислительную систему кластер из 16-узлов, собранных на базе 486DX4/100MHz процессоров, с одним мегабайтом оперативной памяти и тремя 10 МБитными Ethernet-адаптерами на каждом узле. Чуть позже по той же самой схеме был собран суперкомпьютер Avalon, построенный на базе 68 процессоров DEC Alpha/533MHz., управляемых операционной системой Linux (причем, в дальнейшем количество процессоров неоднократно увеличивалось).

HPC-кластеры по своей производительности вплотную приблизились к суперкомпьютерам, а по соотношению цена/производительность даже обогнали их (причем весьма значительно). Однако, следует помнить, что HPC-кластеры эффективны лишь для решения сравнительно узкого круга параллельных задач, не требующих интенсивного межпроцессорного обмена.

Начиная с ядра 2.2.x, Linux поддерживает HPC-кластеризацию, даже созданы специальные узкоспециализированные дистрибутивы, однако, их использование не является обязательным, т. к. RedHat и Fedora уже включают в себя все необходимое.

Помимо операционной системы, нужны еще и компиляторы, обеспечивающие высокую степень параллелизма. Бесплатный GCC, к сожалению, недостаточно эффективен и рекомендуется остановить свой выбор на коммерческих компилятор от Intel, а лучше — компиляторы компании The Portland Group, Inc. (более известной под аббревиатурой PGI). И Intel, и PGI поддерживают языки Fortran/Си/Си++ и операционные системы Linux/xBSD/Windows. Приложения, распространяемые вместе с исходными текстами, выгодно отличаются тем, что могут быть перекомпилированы любым компилятором, что позволяет потребителям повышать их производительность.

Наконец, необходимы библиотеки параллельного программирования, например, пакет MPICH (http://www.mcs.anl.gov/research/projects/mpich2/), отвечающий за передачу меж узловых сообщений, которая в данном случае (как и следует из названия пакета) осуществляется по протоколу MPI, но это уже — дебри технических деталей, в которые конечному потребителю можно и не вникать.

Главное это то, что HPC-кластеры позволяют собирать суперкомпьютеры на базе обычных ПК, которые _уже_ имеются на фирме, эффективно используя существующие вычислительные мощности, вместо приобретения дорогостоящей специализированной техники.

Рисунок 7 HPC-кластер, собранный на базе обыкновенных ПК

распределенные кластеры

Распределенные кластеры (grid-ситемы) представляют собой разновидность HPC-кластеров, только соединенных не локальной сетью, а взаимодействующих через Интернет и, как правило, географически очень разобщенных.

Честные Grid-ситемы (например, SETI@HOME) предлагают пользователю установить специальное клиентское программное обеспечение, которое в фоновом режиме (или в моменты бездействия машины) занимается вычислениями, отсылая их центральному серверу и получающему очередной блок данных. Достоинства Grid-ситем — их дешевизна и высокая

отказоустойчивость (если одновременно "лягут" компьютеры, расположенные в Европе, Америки, Азии, Японии и Африке, то, значит, наступил конец света и никакие вычисления более не понадобятся).

Grid-ситемамы легко взламывают шифры, перед которыми пасуют суперкомпьютеры. Куда там IBM eServer Blue Gene Solution с его жалкими 212.992 процессорами, когда мы можем объединить в кластер _миллионы_ ПК!!! Правда, для этого необходимо как-то мотивировать пользователей, что весьма затруднительно, но всякая крупная компания, имеющая филиалы в разных странах, может задействовать их с пользой для дела (тем более, что общее число компьютеров достаточно велико и намного превышает количество процессоров в IBM eServer Blue Gene Solution).

Нечестные Grid-ситемы (известные как boot-net'ы) собираются путем несанкционированного внедрения в компьютер зловредного кода, проникающего как через дыры в программном обеспечении, так и через пресловутый человеческий фактор (рассылка исполняемых файлов по e-mail).

Популярность Grid-ситем (как честных, так и нет) неуклонно растет и составляет существенную конкуренцию традиционным HPC-кластерам, поскольку пропускная способность современных Интернет-каналов уже не та, что была лет пять назад и передача огромных объемов данных теперь не представляет ни технических, ни экономических проблем. Безлимитным DSL сегодня никого не удивишь, а завтра это будет вообще норма, стирающая различия между локальными и глобальными сетями.

Вспомним, что первые HPC-кластеры строились (и успешно функционировали) на базе 10 Мбитного Ethernet'а, что вполне по зубам DSL-модемам.

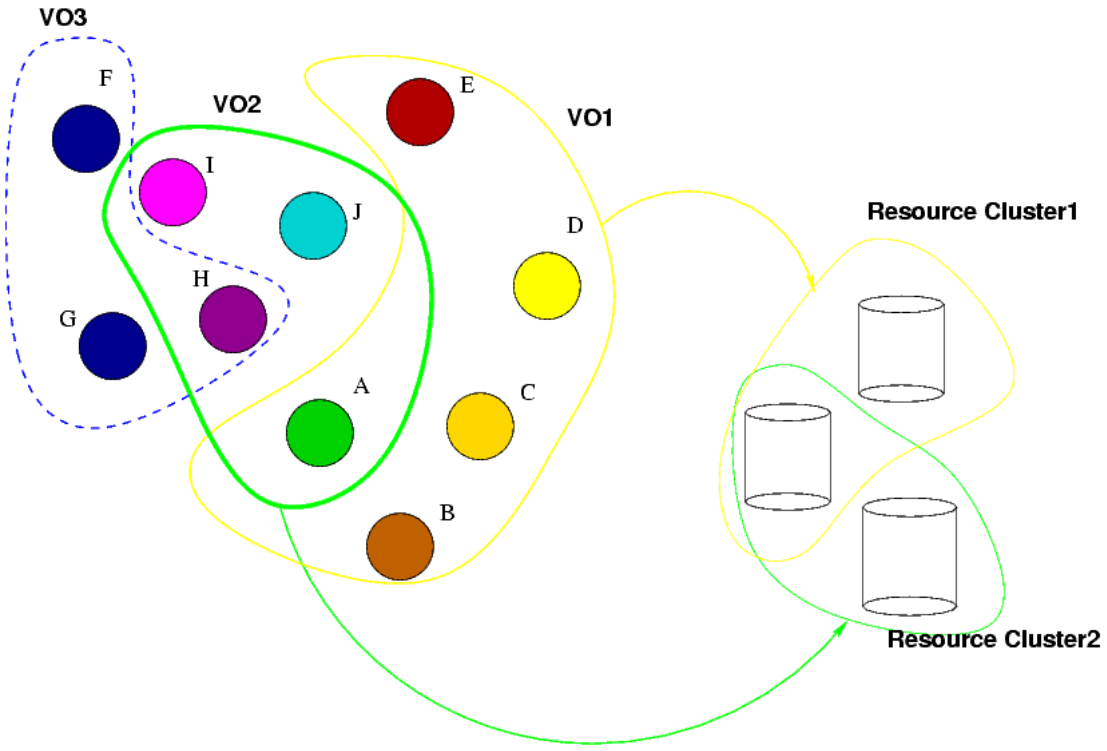

Рисунок 8 упрощенная схема Grid-системы

заключение

Так какой же тип кластеров является наилучшим? Ответа нет. У каждой архитектуры своя ниша, свои достоинства и недостатки, а потому пытаться решать все задачи на кластере одного типа — заведомо проигрышный вариант.

Наибольшей (пиковой) производительностью обладают HPC-кластеры, наилучшим соотношением цена/производительность — Grid-ситемы, ну а HA- и LBC-кластерам нет равных по массовости и популярности, поскольку, компаний, которым нужно обрабатывать WEB-запросы, намного больше тех, кого интересуют "настоящие" вычисления.