виртуальные миры на стыке парадигмы эмуляции

крис касперски, no-email

виртуализаторы, доступные ранее только на майнфреймах, теперь стали доступными на PC, предоставив потребителям те же самые возможности, но за меньшие деньги. однако, все не так просто. в лагере разработчиков эмуляторов произошел раскол на методологической почве: одни производители пошли по пути чистой аппаратной виртуализации (например, Microsoft с ее Server 2008), другие же предпочти гибридные программно/аппаратные решения (яркий представитель которых — VM Ware). ну а на майнфреймы остались верны технологиям логического разбиения (login partition), так же называемых доменами (domain). естественно, каждая технология имеет свои сильные и слабые стороны, в которых мы сейчас и попробуем разобраться.

введение

На рынке толпиться огромное количество самых различных эмуляторов и виртуализаторов как бесплатных, так и коммерческих. Среди них есть узко специализированные продукты, так и пакеты общего назначения, предназначенные для решения широкого спектра задач. Часть эмуляторов требует специальной поддержки со стороны "железа" и работает на ограниченном контингенте архитектур, другая же довольствуется и PC.

Какой продукт выбрать? Какие перспективы он открывает? Станет ли наш бизнес от этого эффективнее, устойчивее, дешевле? А как насчет границ применимости технологий виртулизации, за которыми они становятся экономическими невыгодными, неудобными или просто бессмысленными?

Производители виртуализаторов атакуют нас техническими деталями реализации, жонглируют торговыми марками со звучными названиями, соревнующихся с конкурирующими решениями в крутости, но… к потребительским характеристиками все эти аспекты никак не относятся и тезис "как вы яхту назовете…" здесь не срабатывает. К сожалению, у потребителей нет другого выхода, кроме как доверять отделу маркетинга, ну и еще, быть может, статьям независимых авторов, большая часть из которых (и это ни для кого не секрет) просто добросовестно переписывает рекламные проспекты. Другие описывают свой личный опыт работы с конкретными эмуляторами и таким статьям действительно можно верить (если мы вообще верим печатному слову, не подкрепленному нецензурной бранью). Однако, мало кто работал более чем двумя-тремя эмуляторами сколь ни будь продолжительное время в "промышленном" масштабе. Варианты типа: установил, запустил, проигрался несколько дней и удалил, отметаются сразу, поскольку чтобы получить представление о продукте с ним нужно _работать_ а не играться.

Автор вынужден признаться, что из всех описываемых им эмуляторов, он работал только с VM Ware, а остальные… а вот остальные он потрошил в дизассемблере, ковырялся в исходных текстах, вдумчиво читал документацию и даже косвенно принимал участие в оптимизации некоторых виртуализаторов, что подтверждается наличием строки "thanks to nezumi", где nezumi — мой ник. Это я так шифруюсь, чтобы не выкупили. Однако, глубокое знание устройства эмуляторов и технических особенностей их реализации, не компенсирует отсутствие практического опыта работы с ними, а потому статья не дает никаких рекомендаций по выбору того или иного программного продукта, но подробно объясняет читателю что стоит за всеми этими торговыми марками и чем они (не)отличаются одна от другой.

области применения виртуализации

Возможности виртуализации практически безграничны, вернее, ограничены всего двумя факторами: нашей фантазией и аппаратной мощностью, а аппаратная мощность, как известно, это всего лишь вопрос денег. Для крупных компаний деньги — не проблема. Проблема — как вложить миллион долларов так, чтобы он превратился в два. Или если не в два, то хотя бы полтора. И виртуализация действительно позволяет это сделать!

Провайдеры, использующие виртуализаторы, уже начали предоставлять клиентам виртуальные машины, на которых они могут устанавливать любую операционную систему, которую им только заблагорассудиться, что выгодно отличается от традиционного хостинга, навязывающего клиенту конкретную операционную систему и конкретный набор серверных

приложений, а точнее, привязывающего его к нему, и смена хостера требует огромных затрат времени, в результате чего клиенты становятся заложниками своих провайдеров и даже если вы предлагаете лучшие условия, вами заинтересуются главным образом те клиенты, у которых еще нет сайта (а у кого его сейчас нет) или же те, у кого сайт настолько крошечный, что им вполне достаточно и бесплатного хостинга (как зарабатывать деньги на бесплатном хостинге — это вообще отдельный вопрос). Виртуализация позволяет клиентам "все свое всегда носить с собой", что создает благоприятную конкурентоспособную ситуацию.

Естественно, раздача виртуальных машин требует наличия солидных машинных мощностей и в мелких масштабах совершенно невыгодно, поэтому, прежде чем раздавать виртуальные машины по выгодным тарифам следует посчитать стоимость железа с учетом времени его "амортизации". Порог "входимости" на данный рынок довольно высок и требуется весьма существенные вложения, в противном случае, окупаемость будет держаться на нуле, а клиенты, недовольные низкой производительностью виртуальных серверов, возвратятся назад к провайдерам с классической схемой хостига.

Другое направление использования виртуализации — создание гетерогенных сред, работающих на разнородных операционных системах. Действительно, если в компании используются программные продукты, работающие под UNIX и NT, то вместо приобретения двух машин, мы можем использовать одну! Во всяком случае так утверждает реклама. А подвох?! А вот и подвох — крупные компании крайне редко покупают программные продукты. Они вообще не знают что это такое, отдавая предпочтение производителям поставляющим (и обслуживающим) готовые решения — железо с предустановленным набором программ. И в этом есть свой резон. Практически все крупные производители серверов так же являются и производителями операционных систем, либо же адаптируют чужие операционные системы под свое железо. Конечно, клиент (который по определению всегда прав) может установить любую другую операционную систему (если только найдет для нее необходимые драйвера), но в таком случае он остается один на один со всеми проблемами, а поставщик забирает назад все данные им гарантии (что, в общем-то, логично и справедливо).

Поэтому, в данном случае, виртуализация реально работает лишь в low-end и… high-end сегментах рынка, а середина выпадает из области ее эффективного применения, возрождаясь лишь на самом конце high-end линейки, на котором находятся мощные машины и вычислительные центры, продающие машинное время (или же использующие его внутри фирмы — но это уже не суть важно). Виртуализация позволяет разделять ресурсы одного компьютера между несколькими операционными системами, балансируя нагрузку. В противном случае, для каждой операционной системы пришлось бы выделять свою машину, рассчитывая ее мощность с учетом популярности данной системы, и значительную часть времени компьютер с менее популярной системой работал бы в холостую, в то время как рядом стоящая машина изнемогала от перегрузки.

Еще виртуализацию предлагается использовать для построения отказоустойчивых систем, для которых неприемлема даже плановая остановка на короткое время. В классическом варианте эта проблема решается путем введения резервного сервера, берущего на себя обработку запросов, если основной сервер вдруг по каким-то причинам "упадет". Для крупных компаний покупка резервного сервера не представляет никакой проблемы и прибегать к виртуализации не возникает никакой необходимости. Напротив, в действительно критических инфрастуктурах все программно/аппаратное обеспечение должно быть сертифицировано и подчинено ряду требований, которому виртуализаторы, увы, не отвечают. Задумаемся, почему может "упасть" сервер? Возможных причин всего две — либо дефект железа, либо дефект программного обеспечения. Но если железо _уже_ неисправно и работает со сбоями, то виртуализатор _ничем_ не поможет и все виртуальные машины лягут одна за другой. Если же это дефект программного обеспечения… Хорошо, если это просто "блуждающая" ошибка, а если дыра?! Что если сервер атакует злобный хакер?! Кстати говоря, большинство виртуализаторов устанавливаются поверх операционной системы типа UNIX или Windows, а это значит, что они сами могут выступать в качестве объекта атаки! Таким образом, виртуализация увеличивает риск отказа сервера, особенно в свете того факта, что никакой виртуализатор не свободен от ошибок и вероятность краха операционной системы, запущенной под виртуализаторов существенно увеличивается даже без всякой хакерской атаки. Разумеется, это еще повод для отказа от виртуализаторов на low-end сегменте — действительно очень удобно держать резервную систему на одной машине с основной, не откладывая момент установки заплаток, требующих перезагрузки, а выполняя его в любое время, даже в самый пик нагрузки на сервер. Однако, обозначенное удобство только удобством и остается. Путь в сферу критических инфрастуктур ему закрыт.

Раз уж мы затронули low-end сегмент, будет нелишним упомянуть, что виртуальные машины идеально подходят для экспериментов с различным программам обеспечением. Допустим, захотели мы перелезть с Windows на Linux (или с Linux на BSD), но не уверены, что при этом предусмотрели все потенциальные проблемы, которые нас ждут. Так почему бы не развернуть систему сначала на виртуальных машинах, соединенных виртуальной сетью и не протестировать ее некоторое время? И только убедившись в правильности выбранного пути переносить его на живое железо.

Кстати говоря, перед установкой свежих заплаток (равно как и обновления программного обеспечения) многие администраторы уже взяли себе за правило предварительно их проверять на копии сервера, установленного на виртуальной машине, поскольку нередки случаи, когда установленная заплатка приводит к серьезным проблемам, включая разрушение данных или тяжелые конфликты с жизненно необходимыми приложениями.

А вот другой прием — устанавливаем на виртуальную машину незалатанную операционную систему, даем ей выход во внешнюю сеть и отлавливаем всех хакеров, набросившихся на "легкую" (с их точки зрения) добычу. Другими словами, мы организуем honey-pot, достающийся нам практически даром. Если бы не виртуализация нам пришлось бы приобретать отдельную машину.

Компании, занимающие разработкой или поддержкой программного обеспечения так же не мыслят свою жизнь без виртуальных машин, позволяющих держать целый зоопарк различных операционных систем на одном компьютере.

типы виртуализаторов

Классификация виртуализаторов весьма условна и простилается от полностью программных, до чисто аппаратных, а между ними находится огромное количество гибридных моделей. На самом деле, чисто аппаратных виртуализаторов не существует в природе и всем им требуется программная поддержка. Весь вопрос в том: какие операции реализуются программно, а какие аппаратно и чего это стоит в смысле денег и машинных ресурсов.

Чисто программная виртуализация необычайно гибка и позволяет в частности, эмулировать многопроцессорную машину на однопроцессорной, причем, сам эмулятор может быть портирован под любую архитектуру, поскольку использует базовый набор арифметическо-логических инструкций, который имеется на любом, даже самом примитивном процессоре. Примером чисто программных виртуализаторов является BOCHS, популярный в среде разработчиков, но не имеющий никаких перспектив на "промышленных" рынках, поскольку программная эмуляция требует немыслимых процессорных затрат и для продакшен-машины экономически выгоднее купить "живое" железо, чем виртуализовать ее на программном эмуляторе.

Динамические эмуляторы работают по гибридной схеме: часть инструкций выполняется непосредственно на самом процессоре, а часть эмулируется. Процессор, поддерживающий разделение привилегий, идеально подходит для динамической виртуализации, поскольку, если поместить привилегированный код операционной системы на непривилегированный уровень, то при каждом обращении к привилегированным ресурсам (например, портам ввода/вывода), процессор будет генерировать исключение, перехватываемое виртуализатором, эмулирующим ее выполнение, в результате чего накладные расходы на эмуляцию окажутся незначительными (привилегированные инструкции составляют менее процента от всех остальных).

Однако, далеко не все процессоры спроектированы правильно. В частности, популярная архитектура x86 поддерживает аж целых четыре уровня привилегий (когда на остальных платформах их обычно два или от силы три), но при этом содержит ряд инструкций, позволяющих читать содержимое привилегированных ресурсов с наименее привилегированного кольца. Разработчики процессора, вероятно, думали, что чтение это не опасная операция. Это верно. Не опасно. Нарушить работу операционной системы прикладной код не сможет, но и отследить обращение к привилегированным ресурсам мы не сможем тоже. А это значит, что прежде чем передавать эмулируемому коду управление на выполнение, эмулятор должен убедиться, что в нем отсутствуют "нехорошие" инструкции, то есть вручную декодировать его. Декодирование же требует времени и съедает всю производительность. С другой стороны, поскольку львиную долю своего времени программы проводят в циклах, которые эмулятору достаточно просмотреть один раз, мы получаем двух трехкратный выигрыш в производительности по сравнению с чисто программными эмуляторами.

Пример "честного" динамического эмулятора подобного типа — QEMU. В чем заключается его честность? А в том, что он действует строго по вышеописанной схеме, и как не старается разбежаться — все никак не может взлететь. Какая нам разница — замедляется ли быстродействие процессора в сто или пятьдесят раз?! Большой удачей для QEMU считается запуск игрушки времен ранней молодости MS-DOS на современных процессорах…

А вот VM Ware прячет тузы в рукавах, учитывая особенности конкретных операционных систем и избегая проверок на наличие привилегированных команд там, где их не может быть, плюс использует другие приемы оптимизации, радикально увеличивающие производительность, но… существует тысяча и один способ развалить VM Ware и операционные системы о которых он не знает под ним просто не выполняются. Тоже самое относится и к "нестандартным" действиям, о которых VM Ware не подозревает. В частности, UNIX-системы позволяют просматривать адресное пространство портов ввода/вывода как обычный файл, попытка чтения которого под VM Ware приводит к смерти последнего. Но это, в сущности мелочи. VM Ware вполне пригоден для решения огромного количества практических задач и обладает вполне удовлетворительными техническими характеристиками, пригодными для развертывания продакшен-систем. В частности, даже на Pentium-III 733 MHz уже можно держать несколько SOHO-серверов, вращающихся на Windows 2000/Server 2003/FreeBSD и довольно комфортно с ними работать. Падение производительности по сравнению с живой машиной, конечно, весьма существенно (приблизительно от двух до десяти раз в зависимости от рода выполняемых операций) и дальнейшая оптимизация уже требует аппаратной поддержки.

И такая поддержка действительно появилась! Первым на x86 платформе ее реализовала корпорация AMD в мае 2006 года на процессорах Opteron (которые на самом деле не x86, а x86-64, но это уже детали), а потом и на Athlon 64/Turion 64, обозначив данную технологию кодовым именем "**Pacifica**". У Intel так же имеется аналогичная технология, только реализованная чуть-чуть иначе и названная, естественно, по другому. На Itanium'ах это "**Silvervale**", а на Pentium'ах — "**Vanderpool**". Однако, во избежании терминологической путаницы оба имени превращены в обезличивающую официальную аббревиатуру **VT-X** (Virtualization Technology X — X-Технология Виртуализации).

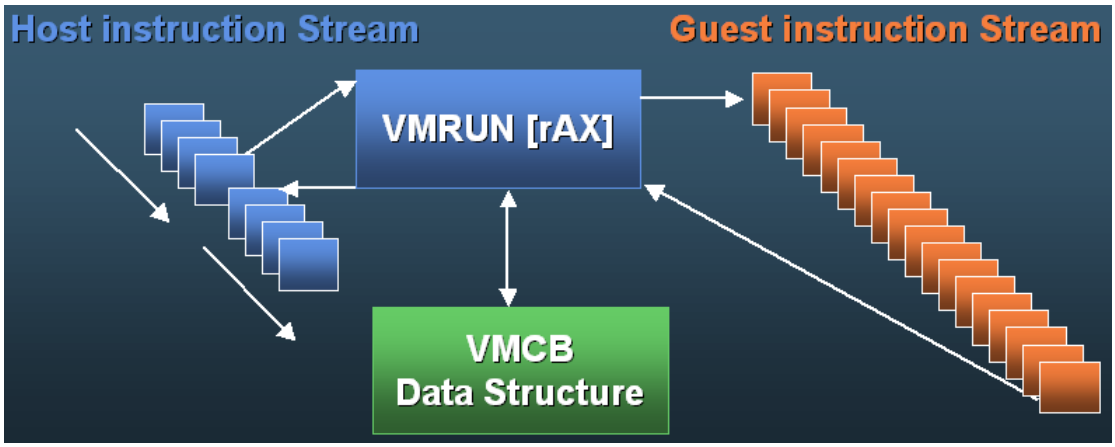

Рисунок 1 технология аппаратной виртуализации Pacifica, реализованная в процессорах фирмы AMD: при выполнении машинной команды VMRUM, процессор создает новую SVM (Secure Virtual Machine – Защищенную Виртуальную Машину), управляемую гипервизором (hyper-visor) и структурой данных под названием Virtual Memory Control Block (Блок Управления Виртуальной Памятью) или, сокращенно VMCB

В чем же заключается суть аппаратной виртуализации? Грубо говоря, к процессору добавили еще одно кольцо защиты, которое, условно можно назвать "мину первым кольцом", поскольку ранее самым привилегированным было нулевое кольцо, а наименее привилегированным — третье. И что же мы получили в итоге? То же самое, что в процессорах Motorola 680x0 (появившихся одновременно с x86) было реализовано изначально, а именно — хорошо продуманное разделение привилегий, благодаря которому стало возможно реализовать эффективный динамический виртуализатор.

Важно понять, что ни "Pacifica", ни "Silvervale"/"Vanderpool" не являются законченными аппаратными виртуализаторами и требуют программной поддержки — специального слоя, который у AMD называется "гипервизором", а у Intel – монитором

виртуальных машин, но суть от этого не меняется. Главное, что производительность виртуализатора вплотную приближается к быстродействию "живого" железа и специальной "заточки" под конкретные операционные системы уже не требуется.

Аппаратную виртуализацию (которая, как мы уже поняли в действительности, является венцом динамической эмуляции) поддерживает большое количество программного обеспечения. Самый известный некоммерческий проект — XEN, коммерческий — Server 2008. А вот создатели VM Ware отказались от поддержки аппаратной виртуализации. Во всяком случае пока… И их можно понять, поскольку аппаратная виртуализация фактически зачеркнула всю их работу и ядро VM Ware необходимо переписывать заново, а это весьма проблемно. Но ведь надо же как-то удерживать рынок, вот потому в VM Ware стали появился функции, отсутствующие в XEN'e и Server'e 2008. В частности, механизм, Record/Replay позволяет записывать и восстанавливать состояние виртуальных машин, что существенно упрощает борьбу с ошибками. Достаточно всего лишь один раз воспроизвести сбой и потом его можно повторять сколько угодно раз! Список эмулируемого виртуального оборудования у VM Ware намного богаче, чем у Server 2008, да и управляемость у него получше будет и потому Server 2008 реально выигрывает только когда самым критичным фактором является производительность. Если же у нас имеется солидный запас неистраченной аппаратной мощности, то использование VM Ware становится более предпочтительным, в силу более богатого функционала (естественно, если этот функционал нам не требуется, то можно обойтись и Server 2008 — однозначные рекомендации дать трудно).

Но x86/x86-64 это далеко не high-end! High-end это мир майнфрймов, это IBM LPAR, HP Domains, Sun domains и другие виртуализаторы, устроенные по принципу все тех же хорошо продуманных честных динамический виртуализаторов. В частности, у IBM LPAR слой программной поддержки так же называется гипервизором (как и у AMD), встроенном в операционные системы IBM i5/OS и IBM AIX 5L. Логическое разбиение сервера на операционные системы (термин, используемый IBM) тождественно "доменам" от Sun и HP, которые в свою очередь тождественны Pacific'e и Silvervale/Vanderpool'y. Естественно, майнфреймы будут помощнее, да и управляемость у них получше. В частности, квотирование процессорного времени реализовано на аппаратном уровне, что существенно упрощает балансировку сервера, в то время как ни VM Ware, ни XEN, ни Server 2008 не предоставляют никаких рычагов для разделения машинного времени между гостевыми операционными системами и его приходится осуществлять различными обходными методами, работающими только при условии "честного использования" (fair use) системных ресурсов. То есть, если виртуализатор управляет нашими собственными системами — все ок, но если же мы начинаем сдавать виртуальные машины в аренду, то любой злоумышленник может вызвать перегруз процессора и максимум, что можно сделать в такой ситуации — пристрелить "нехорошую" виртуальную машину, но корректно урезать ее паек невозможно. А вот майнфреймы справляются с этим без труда, поскольку изначально проектировались с учетом разделения системных ресурсов в "нечестной" конкурентной среде.

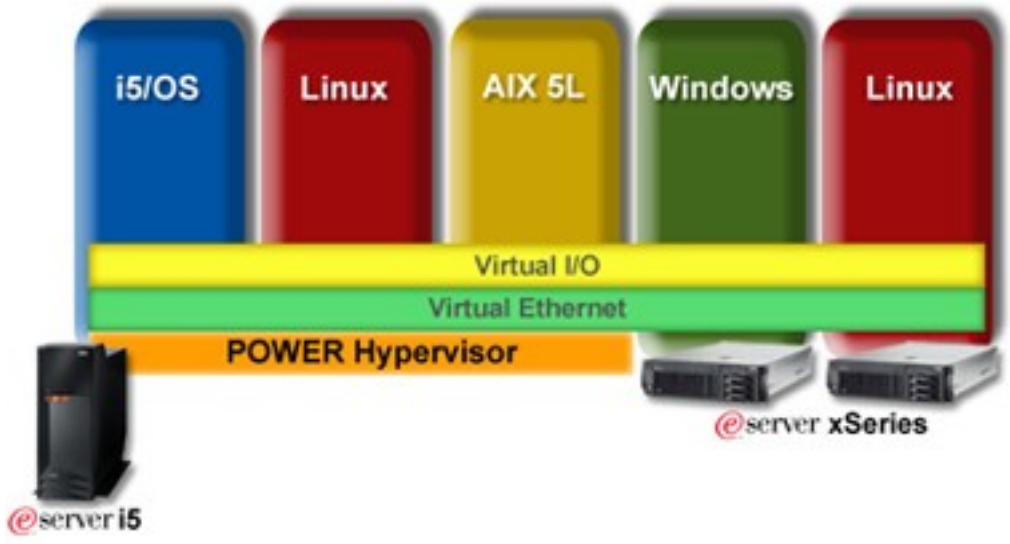

Рисунок 2 архитектура IBM LPAR

заключение

Различий между "большим железом" майнфреймов и архитектурой, выросшей из мелкобюджетных 8086 процессоров, становится все меньше и меньше. С точки зрения потребителя такое положение можно только приветствовать, поскольку даже большие компании умеют считать деньги и даже если у нас есть миллиард долларов, то это не означает, что нам необходимо покупать малиновый пиджак и заказывать майнфрейм с "шестисотой" осью и считать на ней электронные таблицы в Excel, чувствуя себя невероятно крутым.

С другой стороны, x86 как была "подзаборной" архитектурой, так ей и остается. Производительность, функциональность, дешевизна — все это у нее есть, а вот единой целостности — как не было, так и нет. Бремя обратной совместимости дает о себе знать, заставляя расплачиваться за ошибки проектных решений десятилетний давности, точнее даже не ошибки, а жестокий компромисс между "хочется создать" и "необходимо уложиться в заданную сумму". Майнфреймы, изначально спроектированные для тех, у кого есть деньги, на такой компромисс никогда не шли (точнее шли, но с гораздо меньшим ущербом для своей архитектуры), а потому работают по принципу: купил, воткнул, забыл.

И хотя имеется множество примеров создания действительно крупных вычислительных систем на базе x86 (некоторые из которых даже попадают в TOP самых мощных суперкомпьютеров мира), это еще ни о чем не говорит. Сколько денег потрачено на их создание, введение в эксплуатацию и поддержку? Естественно, если проект держится на энтузиазме его основателей и человеко-часы не идут в зачет, то, да, безусловно, архитектура x86 выигрывает. Но если же мы говорим о бизнесе и оплачиваем труд технического персонала, то майнфрймы становятся более выгодными.