аналитическое сравнение XP и Vista – отличия ядер и ядерных компонентов

крис касперский a.k.a nezumi el raton, no-email

большинство статей, описывающих нововведения Висты концентрируются преимущественно на пользовательском интерфейсе и других сугубо прикладных аспектах, не обращая внимания на ядро, являющееся фундаментом ОС и в наибольшей степени определяющее устойчивость, безопасность, быстродействие системы. изменения ядра скрыты от взора пользователей, их нельзя обнаружить простым осмотром системы и без масштабных археологических раскопок тут не обойтись!

введение или немного ностальгических воспоминаний

Давным-давно, когда большинство пользователей сидело на Windows 98, Microsoft выпустила Windows 2000, выхода которой так давно ждал автор данной статьи, поскольку, NT 4.x (с которой он работал до этого) испытывала большие трудности с поддержкой нового железа, а вся линейка 9x вопреки заявлениям Microsoft, на самом деле была и остается построенной на ядре, доставшимся ей в "наследство" от Windows 3.x с "натянутым" поверх него API 32 — набором системных функций, позволяющих запускать 32-разрядные приложения.

Кстати говоря, аналогичный набор был выпущен и для Windows 3.x (там он назывался win32s), только вместо многих функций присутствовали "заглушки", не делающие ничего, кроме возращения сообщения об ошибке. Грубо говоря, Windows 9x — это Windows 3.x с интегрированным win32s, реализованным "должным образом". Тем не менее, множество системных функций в 9x по-прежнему представляют собой заглушки, отсутствует возможность работы с юникодом, поддержка консоли оставляет желать лучшего и часто падает из-за "ударов по памяти", и хотя Windows 9x представляет очень удачную операционную систему, возможно, даже самую удачную систему из всех, когда-либо выпущенных Microsoft, но… автору хотелось совсем другого: защищенного ядра с разграничением доступа и разделением пользователей, динамически загружаемых служб (позволяющих запускать легендарный отладчик Soft-Ice _без_ перезагрузки компьютера, как того требовала Windows 98)…

И вот наконец!!! Дотащив долгожданный диск дистрибутива с горой надписью Windows 2000 Professional, автор стал первым, кто перешел на нее в своем городе (вообще-то, не городе, а деревне, но и городе тоже). Теперь можно было программировать, не опасаясь "уронить" систему (в Windows 9x это легко, достаточно случайно захватить все ресурсы GDI и все… остается только давить на Reset). Конечно, при желании можно завесить и Windows 2000, но стоит различать _целенаправленно_ предпринятую атаку от непреднамеренной ошибки. Вещи эти очень разные и даже не перпендикулярные.

Вслед за автором потянулись и остальные… но! вместо радостных возгласов разразились глубоким разочарованием: "ну и чем эта Windows 2000 от Windows 98 отличается?!" — негодовали они — "тем, что появилась поддержка курсора с тенью?!". Автор от таких разговоров выпадал в осадок, просто не зная что ответить… тень от курсора он отключил сразу (точнее _не_ стал включать ее), но попытки объяснить людям, что Windows 2000 построена _совсем_ на другом ядре потерпели неудачу, особенно в свете того, что под ней не шли старые MS-DOS игры.

Осознав, что подавляющему большинству пользователей архитектурные тонкости ядра сугубо фиолетовы, Microsoft выпустила Windows XP со слегка доработанным ядром, фактически представляющем собой "работу над ошибками в Windows 2000" и радикально переделанным интерфейсом, с кодовым именем Luna, по поводу которого вспоминается один анекдот. Девушку спрашивают: "а вы бы могли полюбить радикала?!". "Ради чего?" — удивленно переспрашивает она. Так вот, с эстетической точки зрения Windows XP это… просто какой-то кошмар. Достаточно просто сравнить изображение "динамика" в Windows 9x/2000 и в XP: если в первой с первого взгляда ясно, что это такое, то во второй приходится пристально вглядываться в трехмерную фигуру громкоговорителя, изображенную в аксонометрической проекции. Разборчивость остальных иконок так же ухудшена. Некоторые операции уже невозможно выполнить с одной лишь клавиатуры без мыши (а количество пользователей Windows для которых клавиатура является _основным_ средством общения с операционной системой измеряется десятками, а то и сотнями тысяч!). К счастью, была предусмотрена

возможность отключения всех этих красивостей, но… старый интерфейс растворился в тумане и всех его возможностей было уже не вернуть (в частности исчезла поддержка "узора" рабочего стола и не нужно говорить, что она никому не нужна!!!).

Неудивительно, что выпустив Висту, Microsoft полностью переписала интерфейс (проходящий под кодовым именем Aero), уделив ему гораздо больше внимая чем, собственно, самому ядру. Пользователям интерфейс пришелся явно по вкусу и они даже согласились "проапргрейдить" свои компьютеры для установки Висты, причем количество "согласившихся" оказалось таково, что продажи железа по данным корпорации Dell подскочили до 178%, причем большинство из них покупали компьютер для дома.

А что на счет бизнеса? Офисный компьютер — это все-таки не роскошь, не произведение искусства и увлечение "красивостями" ему только вредит (вспомним строгий стиль первых IBM PC, предназначенных для бизнеса и сравним его с "Амигой", значительно превосходившей своего конкурента по возможностям, но именно из-за своего "игрового" имиджа так и не сумевшая пробиться на офисный рынок, подчиненный строгому стилю IBM PC).

Давайте, отбросив интерфейс, посмотрим что находится под ним и какие реальные преимущества дает Виста по сравнению с XP и насколько быстро они окупят переход на нее. Ядро — это царство абсолютной объективности. Здесь нет и не может быть места для "мнений". В отличии от "красоты" и "элегантности", которую каждый склонен оценивать по своему, технические характеристики ядра _однозначно_ определяют качество системы. А вот конкретную выгоду уже приходится определять индивидуально исходя из конкретных потребностей. Скажем, если перевести автомобиль на гусеничный ход, кто-то обрадуется, а кто-то, наоборот, завопит: верните мои колеса!!!

родословная висты

Корни Висты уходят вглубь многочисленных слоев абстракции и десятилетних наслоений кода, но ведут один вовсе не к XP, как это можно было бы подумать, а к… Windows 2003 Server, у которого разработчики Висты позаимствовали ядро, слегка его… даже не знаю как сказать: "испортив" или "доработав". ОК, выберем религиозно нейтральный и политически корректный вариант "внеся в него большое количество изменений", но как бы там ни было, собирая операционную систему для рабочих станций на серверном ядре, Microsoft упрощает себе жизнь (вместо двух разных ядер теперь будет одно), но совершает большую стратегическую ошибку: проигрывая в плане производительности на рабочих станциях, где серверная "оптимизация" превращается в "пессимизацию", поскольку, сервера и рабочие станции ориентированы на принципиально различные типы операций.

Для сервера — это обслуживание множества пользователей, совершающих перекрывающиеся запросы к различным файлам, поддержка большого количества сетевых соединений и сервисов их обслуживающих. Пользователи рабочих станций (сколько бы окон они ни открывали) в каждый момент времени взаимодействуют только с одним приложением. Даже если в фоне играет WINAMP, скачивается несколько файлов и обсчитывается пара электронных таблиц, дисковые запросы на чтение/запись практически не перекрывают друг друга. Создание универсального ядра, одинаково хорошо работающего как на рабочих станциях, так и на серверах — вполне посильная задача, но взяв ядро от Server 2003, Microsoft пошла по пути усиления серверных возможностей в ущерб функциональности рабочих станций, что косвенно свидетельствует о ее дальнейших намерениях.

Вполне возможно, что Виста окажется "промежуточной" или же вовсе "тупиковой" операционной системой, вроде Windows Me, которая была выброшена на рынок только чтобы не пролететь на фондовом, и, написанная впопыхах, естественно, не "прижилась". Подозрения усиливает тот факт, что выход новой операционной системы назначен на 2009 год, т. е. к тому времени когда корпоративный пользователь только созреет к переходу на Висту, она уже "выйдет из моды" и на рынке появится… хм… очень хочется верить, что появится что-то действительно мощное, доброе и хорошее, хочется верить, что у Microsoft просто катастрофически не хватает времени и она выбрасывает Висту в том виде, в котором она есть, и система 2009 года не окажется ее "дочерью".

По слухам (и по данным личной переписки с сотрудниками Microsoft), разработчики Windows находятся на грани нервного срыва, не в силах поддерживать запутанную иерархию многочисленных компонентов и системных библиотек, сплетенных в огромный клубок и что они начинают постепенно приходить к мысли о том, что концепция построения UNIX – если не единственно правильная, то во всяком случае очень удобная. Ядро ничего не должно знать о

графической подсистеме, а графическая подсистема должна быть отделена от оконного менеджера, при этом браузер это все-таки не часть системы, а отдельное приложение, которого может вообще и не быть. Эх, мечты, мечты… Windows уже рушится под собственной тяжестью, но количество уровней абстракции только увеличивается. Ладно, оставим в стороне .NET (мы же договорились не обращать внимания на прикладной уровень) и погрузимся в ядро.

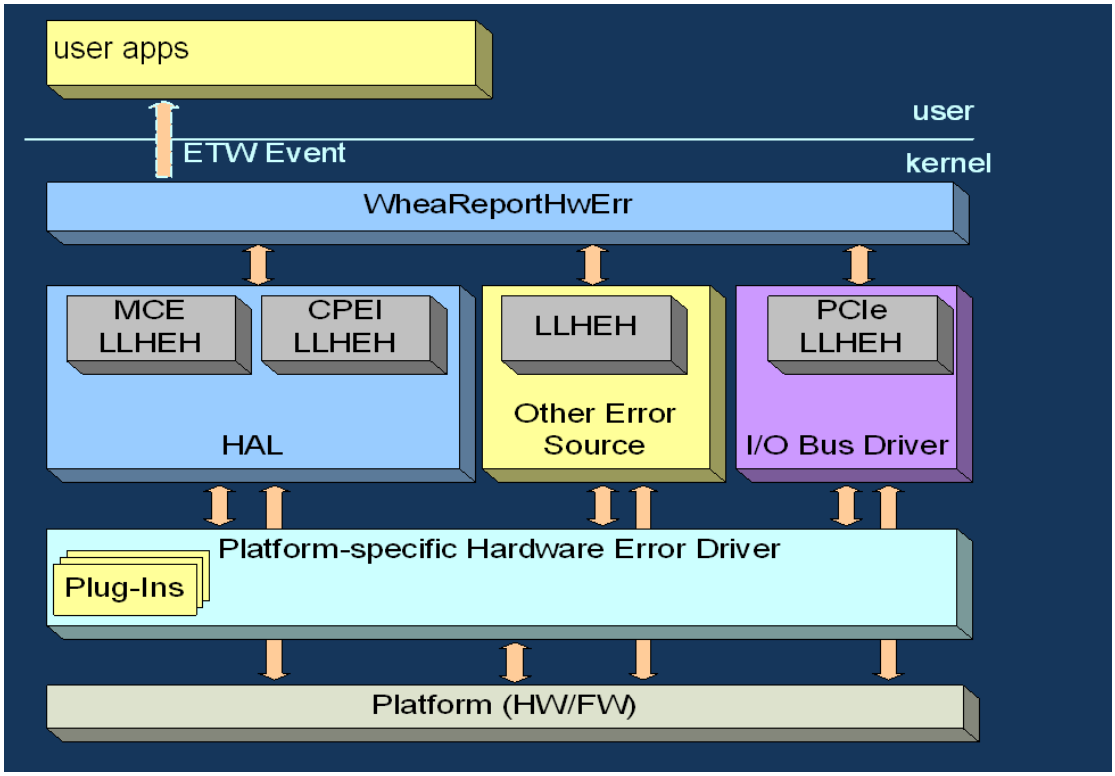

Рисунок 1 архитектура ядра Висты

изменения в менеджере памяти

Память — это основной ресурс, распределяемый операционной системой между приложениями, и в огромных количествах употребляемый ей самой в "служебных" целях, поэтому, неудачный выбор алгоритмов распределения чреват частыми обращениями к файлу подкачки, съедающими всю производительность и нивелирующими все остальные улучшения ядра. Microsoft непрерывно оптимизировала стратегию выделения памяти, переписывая огромные куски кода в каждой новой версии Windows.

Виста не стала исключением. Перечень усовершенствований (и других изменений) содержится в мультимедийной презентации http://go.microsoft.com/fwlink/?LinkId=67468, а так же следующей технической спецификации: download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc.

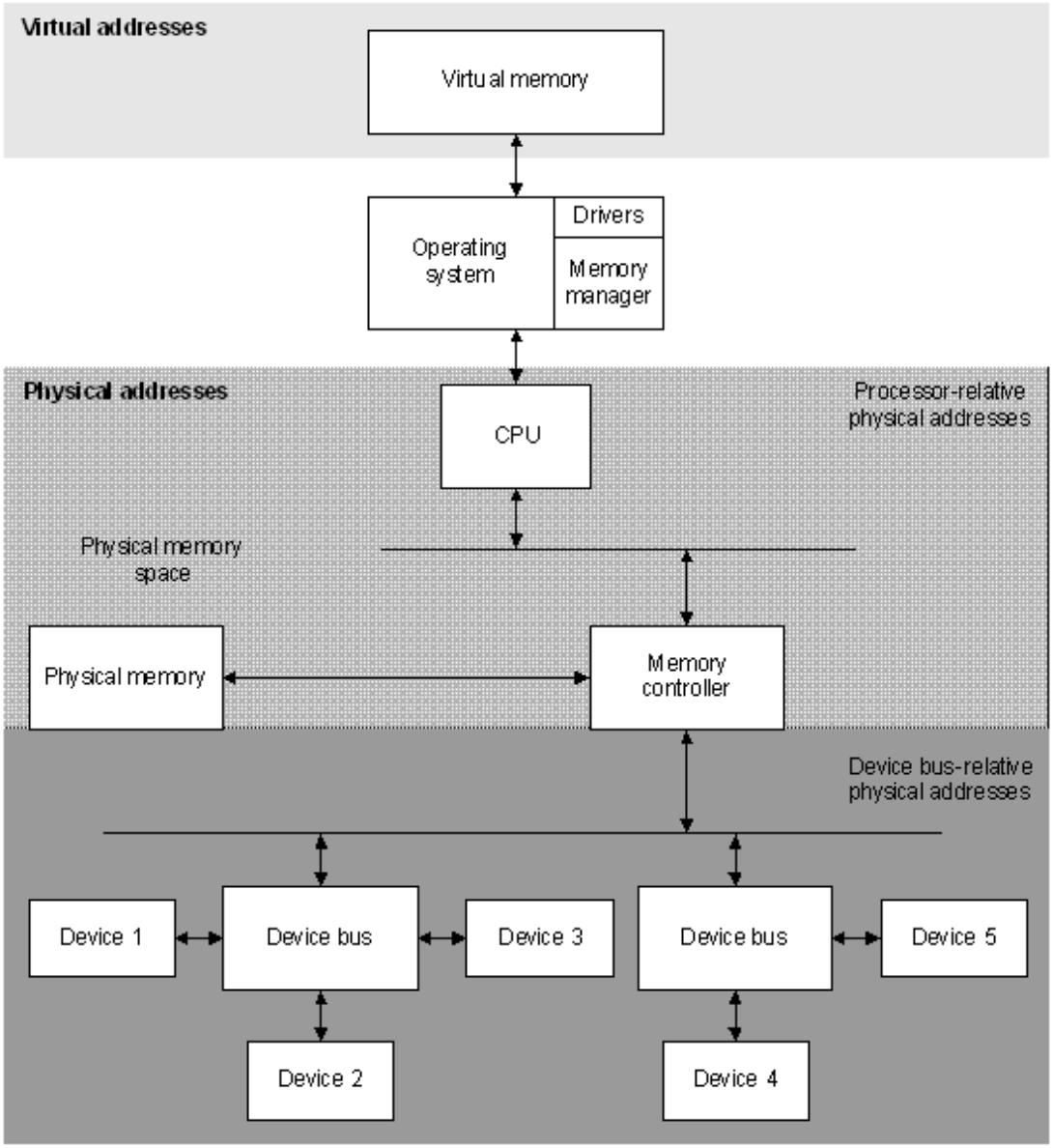

Рисунок 2 архитектура виртуальной памяти Висты

Начнем с поддержки NUMA-памяти, впервые появившейся в Server 2003 SP1 и отсутствующую в более ранних версиях Windows (в том числе и в XP). Что это такое?! Аббревиатура NUMA расшифровывается как Non-Uniform Memory Architecture (Архитектура с Неоднородной Памятью) и охватывает как многопроцессорные машины, так и целые кластеры.

В операционных системах без поддержки NUMA планировщик ставит все потоки в очередь и гоняет их по кругу, при этом в различные моменты времени каждый поток выполняется то на одном, то на другом процессоре, что снижает производительность за счет накладных расходов на обеспечение когерентности (согласованности) кэш-памяти всех уровней. Для обхода этой проблемы в Server 2003 SP1 были реализованы новые API-функций семейства ExNuma, например: VirtualAllocExNuma(..., Node), MapViewOfFileExNuma(..., Node) и CreateFileMappingExNuma(..., Node), явным образом указывающие системе, что поток предпочтительнее всего запускать на том процессоре, на котором и произошло выделение памяти. Двуядерные и HT-процессоры, разделяющие общий кэш между всеми потоками, к этому нововведению абсолютно равнодушны и выигрыш в производительности достигается только на машинах, снабженных двумя (и более) _физическими_ процессорами, да и то при условии, что приложение использует новые вызовы API, а такие приложения появятся не скоро… Тоже самое относится и к поддержке расширения AWE, преодолевающему барьер в 4 Гбайта физической памяти на x86 системах. Материнские платы, поддерживающие свыше 4 Гбайт памяти, относятся к серверному классу и стоят весьма недешево. Конечно, неуклонное

снижение цены на вычислительную технику "опустит" такие платы в бюджетный класс уже через несколько лет, но никакой гарантии, что действительно произойдет, у нас нет.

Поддержка больших объемов памяти повлекла за собой необходимость пересмотра стратегии выделения адресного пространства и переходу от статической модели (выделяющей все адресное пространство при старте системе) к динамическому (выделяющую адресное пространство по мере его потребления). Если раньше каталог виртуальных страниц инициализировался на ранних стадиях загрузки оси, резервируя 1,5 Мбайта на x86 системах, 3 Мбайта, на x86 системах с поддержкой PAE (Page Address Extension) и 2,5 Гбайта на x86-64 и IA64, то сейчас выделение памяти и построение страничного каталога происходят по мере необходимости (on-demand), что слегка ускоряет загрузку, но ощутимо замедляет работу приложений, "пожирающих" память. И, если на серверах, работающих на 64-битных процессорах или процессорах с поддержкой AWE, этот шаг еще хоть как-то оправдан (действительно, глупо инициализировать все адресное пространство, не будучи уверенным — понадобиться ли оно или нет), то рабочие станции однозначно оказываются в явном проигрыше.

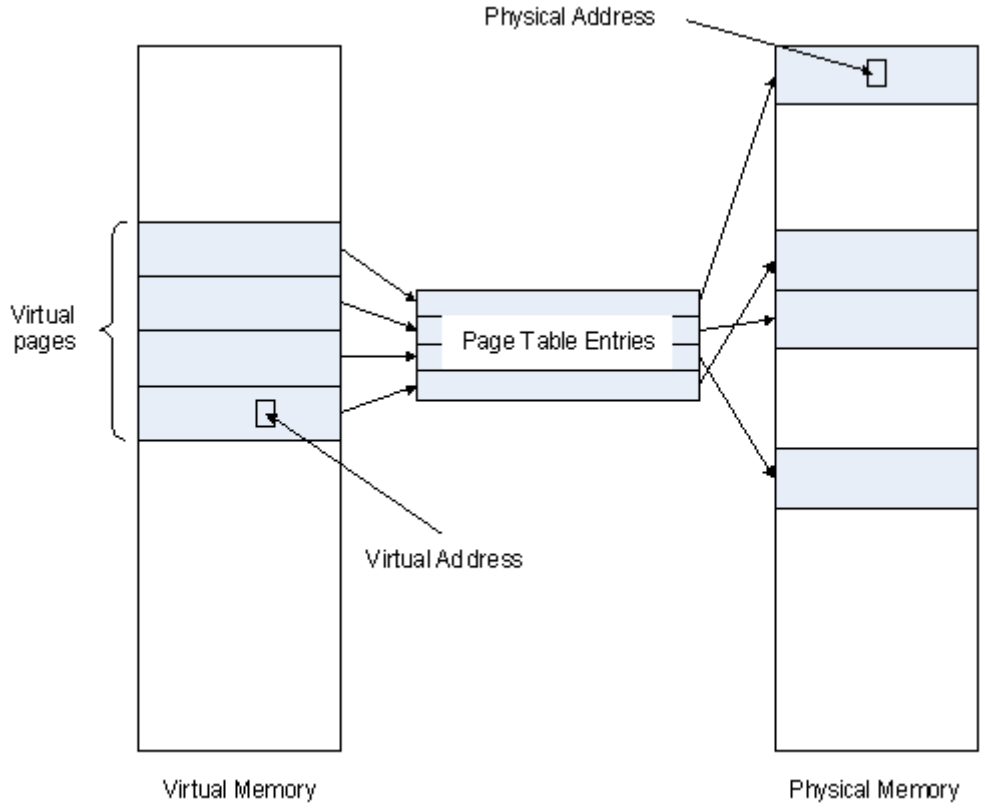

Рисунок 3 страничный каталог (Page Table) хранит соответствие между физическими адресами (physical address) и страницами виртуальной памяти (virtual memory)

Параллельное "обнуление" (zeroing) страниц, появившееся в Server 2003 SP1, так же относится к кластерам и не дает никакого выигрыша даже на многопроцессорных системах, поскольку физическая память — одна. Поддержка новых типов проекций и, в частности, чередующихся виртуальных адресных дескрипторов — Rotate Virtual Address Descriptors (или сокращенно VADs) позволит видео-драйверам полнее использовать возможности шины AGP, быстрее отображая видеопамять на адресное пространство прикладных приложений, с выбором одного из следующих типов проекций: cached, non-cached, write-combined AGP или video-RAM mappings, что несомненно порадует любителей трехмерных игр, но никак не отразится на судьбе офисных и корпоративных пользователей.

версия операционной системы	редакция	объем виртуальной памяти	макс. объем физической памяти
Microsoft Windows Server™ 2003 SP 1/Vista	Standard	4 GB	4 GB
	Web	4 GB	2 GB
	Enterprise	4 GB	64 GB (если железо поддерживает PAE)
	Enterprise (64-bit)	16 terabytes	1 terabyte
	Datacenter	4 GB	128 GB (если железо поддерживает PAE)
	Datacenter (64-bit)	16 terabytes	1 terabyte
Windows Server 2003	Standard	4 GB	4 GB
	Web	4 GB	2 GB
	Enterprise	4 GB	32 GB (если железо поддерживает PAE)
	Enterprise (64-bit)	16 terabytes	64 GB
	Datacenter	4 GB	128 GB (если железо поддерживает PAE)
	Datacenter (64-bit)	16 terabytes	512 GB
Windows XP	Home	4 GB	4 GB
	Professional	4 GB	4 GB
	64-bit Edition Version 2003	16 terabytes	128 GB
Windows 2000	Professional	4 GB	4 GB
	Server	4 GB	4 GB
	Advanced Server	4 GB	8 GB
	Datacenter Server	4 GB	32 GB (если железо поддерживает PAE)

Таблица 1 количество виртуальной и физической памяти, поддерживаемое разными версиями Windows

изменения в менеджере подкачки

Разработчики Висты переписали значительную часть кода, отвечающего за подкачку страниц памяти с диска, пересмотрев базовые стратегические алгоритмы, и оптимизировав его работу (точнее, попытавшись оптимизировать), но… это если смотреть на ситуацию глазами Microsoft. С точки же зрения пользователей, _гораздо_ большей оптимизации можно добиться установив на XP порядка 1 Гбайта памяти и… отключив файл подкачки. Подавляющему большинству приложений такого количество памяти окажется вполне достаточно.

Виста потребляет намного больше памяти и для отключения файла подкачки потребуется как минимум 4 Гбайта физической памяти, но даже такого количества некоторым .NET приложениям может не хватить и, как показывает практика, при активной работе системы, им действительно ее не хватает и без файла подкачки уже не обойтись. Чтобы хоть как-то скомпенсировать разрыв в производительности и не уронить Висту в глазах пользователей XP, Microsoft была вынуждена прибегнуть к многочисленным ухищрениям, забыв о том, что оптимизация никогда не дается даром и выигрывая в одном мы неизбежно теряем что-то другое.

Главное архитектурное изменение заключается в том, что страницы виртуальной памяти, ранее объединенные в связанный список (linked list), теперь реализованы как сбалансированные AVL-деревья, и это при том, что каждому студенту известно, что деревья дают выигрыш только при обработке действительно больших объемов данных (которым в данным случае является не размер памяти, а количество страниц). Сколь ни будь заметный выигрыш достигается только при интенсивном использовании десятков гигабайт виртуальной памяти, что способно удовлетворить аппетит даже очень мощного сервера или рабочей станции, обрабатывающей цифровое видео огромного разрешения, занимающейся трехмерным моделированием или решающей другие "прожорливые" в плане памяти задачи, но… тогда уже имеет смысл приобретать не PC, а нечто вроде Silicon Graphics. Про офисные приложения мы вообще молчим. Даже если их переписать на .NET они все равно не догонят свои потребности в памяти до таких безумных пределов.

Другое интересное нововведение — уменьшение фрагментации файла подкачки (как внешней — на диске, так и внутренней — соседние виртуальные страницы выгружаются рядом) снижает эффективность использования дискового пространства, увеличивает время позиционирования головок жесткого диска при считывании несмежных страниц и в ряде случаев несколько ухудшает производительность, хотя в общем случае быстродействие системы возрастает, причем иногда _очень_ значительно, чуть ли не в несколько раз. Расплачиваться за это приходится размером файла подкачки, впрочем, учитывая, что жесткие диски на 500 Гбайт и даже 750 Гбайт уже не редкость, о размере можно смело забыть.

Наконец, если раньше файл подкачки, реестр и дисковый кэш представляли собой совершенно независимые системные компоненты, теперь между ними протянулась нить интеграции, согласующая операции ввода/вывода, что самым благотворным образом сказывается на производительности, но… реально выигрыш ощущается опять-таки на серверах, а на рабочих станциях разницу в быстродействии приходится измерять профилировщиком, или хронометрировать с таймеров в руках.

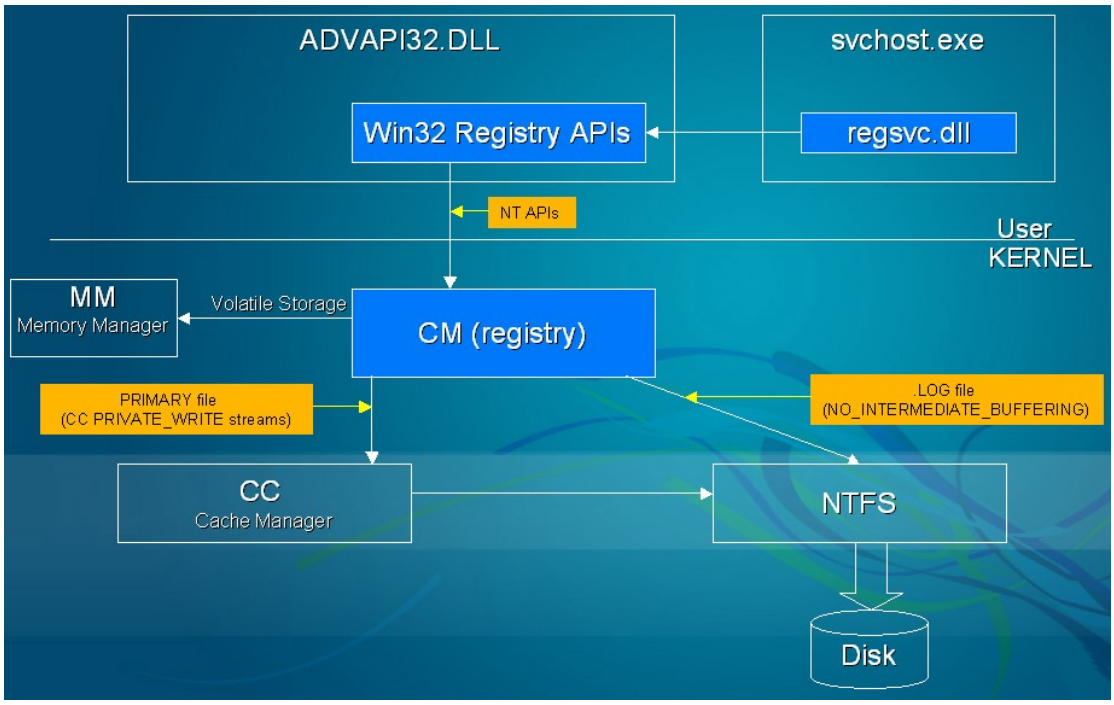

Рисунок 4 интеграция кэш-подсистемы с файлом подкачки и файлом реестра

изменения в менеджере кучи

Куча, (она же heap, она же динамическая память) представляет собой слой абстракции над менеджером памяти, позволяющий программисту выделять куски памяти заданного размера, а по завершению работы с ним — освобождать, возвращая обратно в общий пул памяти. Динамическая память, активно используемая в Си, в языках последующих поколений (Си++, Java, .NET) стала своеобразным фундаментом, используемом как явно, так и неявно. Кажется очевидным, что на неудачно построенном менеджере кучи далеко не уедешь и его оптимизация способна принести большую отдачу, но в действительности это не более, чем распространенное заблуждение.

Прикладные программы используют прямые вызовы функций менеджера кучи только в исключительных случаях, поскольку поверх него натянут еще один слой абстракции, создаваемый языком программирования, а точнее поставляемой вместе с ним библиотекой времени исполнения (она же RTL). Другими словами, каждый язык имеет свой собственный менеджер кучи, запрашивающий у системного менеджера большие куски памяти, а затем "нарезающий" ее мелкими порциями. Реальная производительность определяется именно стратегией встроенного менеджера кучи и слабо зависит от системного.

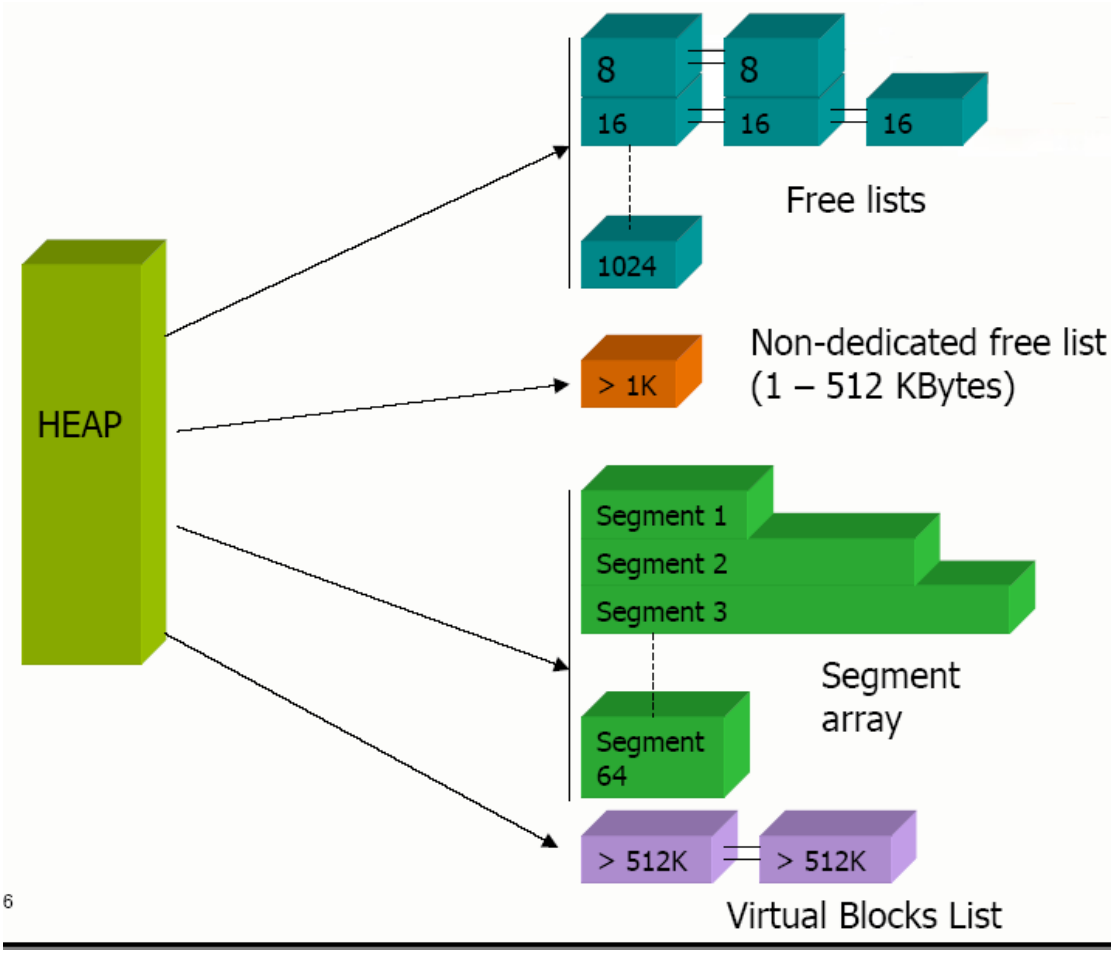

Рисунок 5 организация динамической памяти в Висте

Тем не менее, не стоит игнорировать работу, проделанную Microsoft, и в первую очередь следует отметить уменьшением фрагментации динамической памяти (the low-fragmentation heap — LFH), а так же новыми эвристическими алгоритмами, автоматически подстраивающимися под характер запрос на выделение памяти.

Вопреки расхожему заблуждению, фрагментация кучи приводит отнюдь не к падению производительности, а к _невозможности_ выделения непрерывного блока требуемых размеров, хотя по "кускам" свободной памяти может быть в сотни раз больше. Как следствие, при достижении определенной степени фрагментации, часть приложений отказывает в работе и их приходится перезапускать. Если этим приложением окажется база данных или HTTP-сервер, то такое положение дел может привести к большим неприятностям, особенно если программист забыл обработать ситуацию невозможности выделения памяти и вместо корректного завершения работы мы получаем крах с потерей всех не сохраненных данных. Впрочем, как показывает практика, подобных приложений практически не встречается и если приложение требует периодического перезапуска, то с вероятностью близкой к единице, можно утверждать, что проблема кроется совсем не в фрагментации, а… в утечке памяти! Ошибки проектирования приводят к тому, что выделенная однажды память не освобождается, продолжая выделяться вновь и вновь, вплоть до полного исчерпания всех системных ресурсов. Справиться с утечками операционная система не в состоянии, поскольку не существует никаких формальных признаков, по которым было бы можно отличить реально используемый блок памяти от блока который просто "забыли" освободить.

С адаптивными алгоритмами выделения памяти (еще одна инновация Microsoft) все одновременно просто и сложно. Если они "угадают" характер поступления запросов, мы получим увеличение производительности (под час весьма значительное), но всегда существует угроза, что они сработают с точностью до наоборот!

Но как бы там ни было, максимальное время выделения/освобождения памяти напрямую зависит от так называемого lookup-алгоритма, естественно, переписанному в новой версии Windows и теперь его эффективность составляет не O(n), а O(1). В переводе с

математического языка на человеческий: теперь время работы алгоритма представляет собой константу, не зависящую ни от каких посторонних обстоятельств (например, количества выделенных блоков), что в каком-то смысле очень даже хорошо. Плохо только то, что при небольшом количестве блоков (вполне типичным для офисных приложений) прежний алгоритм показывает лучшую эффективность, то есть выигрыш опять-таки достигается только на серверах, ну или высокопроизводительных рабочих станциях, решающих серьезные задачи, к которым расчет зарплаты в Excel'е ну никак не относится.

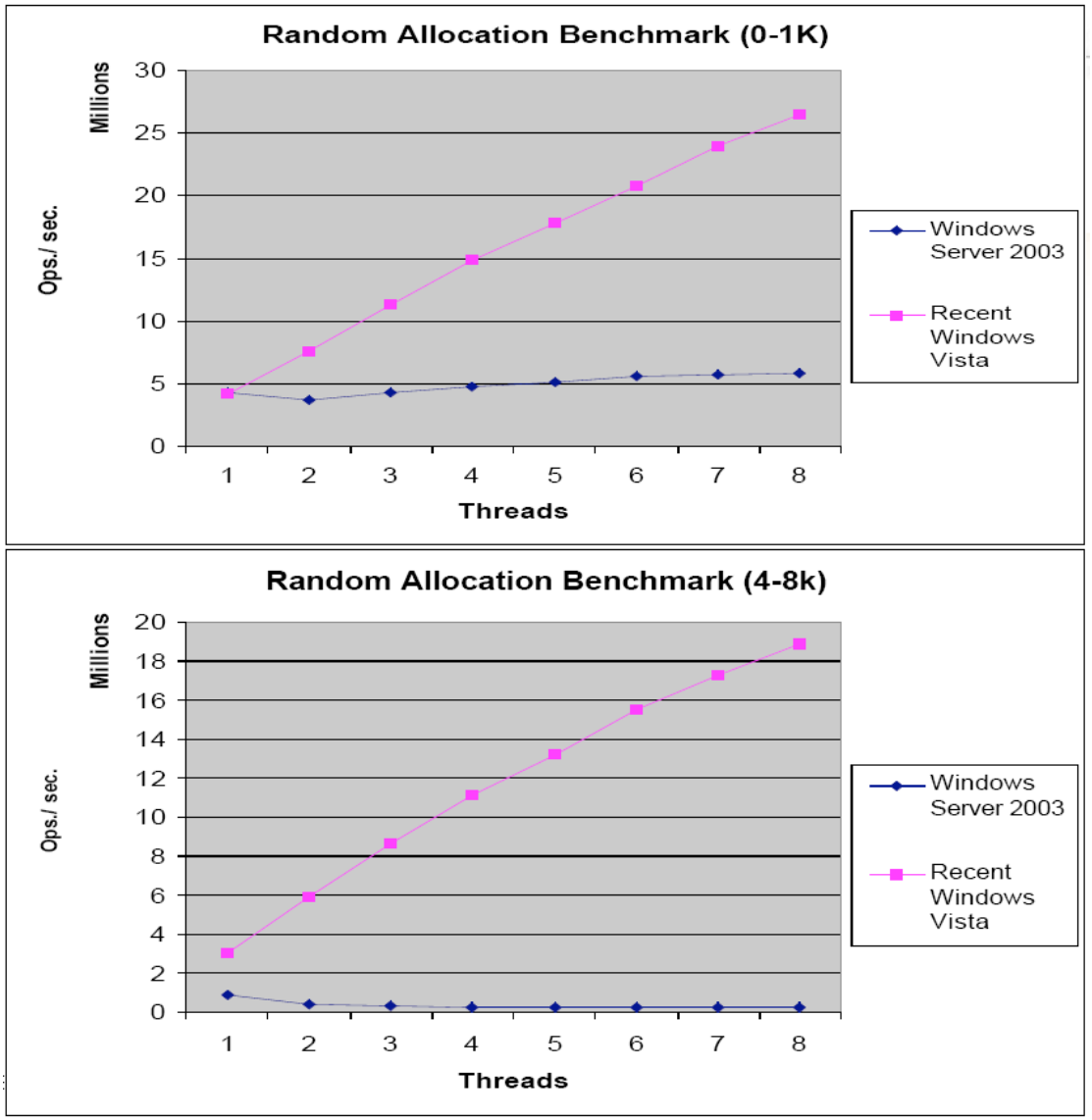

Рисунок 6 сравнение производительности менеджера кучи Висты с Server 2003

изменения в менеджере ввода/вывода

Ввод/вывод — традиционно самая узкая часть в системе, своеобразное "бутылочное горлышко", сводящее на нет высокое быстродействие процессора и огромную пропускную способность подсистемы памяти. Все упирается в диск. Производительность винчестеров, конечно, растет, но объемы обрабатываемых данных растут еще быстрее, причем, зачастую к диску совершается более одного запроса одновременно (особенно на серверах), но не все запросы равнозначны по своей ценности. Windows изначально поддерживала приоритеты процессов (которыми можно манипулировать как через API-функции, так и с помощью "Диспетчера Задач", но вот приоритеты потоков в ней все это время отсутствовали. И вот наконец….

…появился приотизированный ввода/вывод! И теперь потоки данных с более высоким приоритетом вытесняют потоки с более низким приоритетом. Какую отдачу это дает нам в практическом плане? Хм, хороший вопрос, сейчас постараемся подобрать под него подходящий пример. Ну вот допустим, нам необходимо скопировать большое количество файлов в фоновом режиме, при этом мы хотим, чтобы система реагировала на открытие текстовых файлов и электронных таблиц практически мгновенно (XP в этом случае заметно тормозит). А еще примеры будут? Увы! В повседневной работе, офисному компьютеру приотиризированный ввод/вывод практически совсем не нужен. Вот на серверах — ситуация совсем иная и разница в производительности ощущается даже на домашнем http/ftp, который теперь можно перевести в фоновой режим, чтобы при большом наплыве пользователей, Виста не "проседала" под нагрузкой, как это делает XP, вынуждая владельца сервера устанавливать излишне жесткие лимиты на количество одновременно устанавливаемых соединений.

Вместе с появлением приоритизированного ввода/вывода так же изменилась и политика сброса дисковых буферов. Если файлы, отображаемые в память в XP сбрасываются на диск маленькими кусочками, размер которых не превышает 64 Кбайт, то Виста может зараз сохранять до 4 Гбайт данных. Какой выигрыш это дает? Поскольку большинство приложений работает с файлами через ReadFile/WriteFile, без проецирования в память — ровным счетом никакого. Единственное, что ускоряется — так это работа с файлом подкачки, поскольку система проецирует его в память, но реальное ускорение удается обнаружить лишь на быстрых дисках и при интенсивном вводе/выводе, что опять-таки на рабочих станциях не так уж часто и встречается.

стабильность, безопасность, надежность

Хорошая новость — Microsoft перенесла часть драйверов с ядерного уровня на прикладной (точнее, предоставила программистам набор функций, способный реализовать некоторые типы драйверов, главным образом относящиеся к USB-устройствам, на прикладном уровне). И что же здесь хорошего?

Все дело в том, что Windows, в отличии от Linux, крайне болезненно реагирует на исключения возникающие в драйверах ядерного уровня, отвечая на это аварийным остановом системы, сопровождаемым знаменитым Голубым Экраном Смерти, даже тогда, когда для этого нет никаких причин. Допустим, драйвер обратился к странице памяти, которой нет. Ясно же, что остальные компоненты системы от этого никак не страдают и достаточно просто остановить "неправильный" драйвер или даже перезапустить его! Linux именно так и поступает. Естественно, если ошибка обнаружится в одном из критических драйверов (например, драйвере файловой системы), то останов "неправильного" драйвера с высокой степенью вероятности приведет к неработоспособности всей системы, но ошибки такого типа крайне редки, а вот ошибки в драйверах сторонних разработчиков — вполне обычное явление.

В идеале, компании Microsoft следовало бы доработать ядро на манер Linux'a, но она как всегда пошла своим путем. Ошибки, вызываемые драйверами прикладного уровня, вызывают критические исключения, ведущие к аварийному завершению (перезапуску) драйвера, но безвредные для всей системы в целом. Так что такой шаг можно только приветствовать, однако, не стоит надеяться, что установка Висты автоматически перетянет ранее написанные драйвера на прикладной уровень! Это произойдет не раньше, чем программисты выпустят обновленные версии, использующие данную возможность, а для этого разработчикам придется: а) внимательно прочитать документацию; б) научиться писать драйвера по-новому; в) переписать уже существующий отлаженный код. Вопрос: сколько программистов реально займутся этим?! Ответ: старые драйвера переписывать никто не станет, и даже драйвера для новых устройств еще долгое время будут создаваться по прежней схеме, поэтому, Виста и дальше продолжить нас радовать Голубыми Экранами Смерти. Конечно, в перспективе, драйвера прикладного уровня станут вполне обычным явлением, но… не быстрее было бы просто доработать ядро, останавливая систему только в критических ситуациях?!

Плохая новость — разработчики Висты полностью переписали TCP/IP стек, добавив в него поддержку IPv6 и… повторив _все_ ошибки, допущенные при разработке старого стека, частично позаимствованного еще из BSD. Подробный анализ "дыр" нового сетевого стека можно найти в статье: "Windows Vista Network Attack Surface Analysis: A Broad Overview", возникшей в результате исследований первых beta-версий, корпорацией Symantec: www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf. Конечно, сейчас все эти ошибки уже исправлены, но сам факт их наличия говорит о том, что квалификация программистов, вовлеченных в это дело, оставляет желать лучшего, стек писался кое-как, а

тестировался еще хуже, то есть вообще не тестировался, поскольку бы в противном случае, сотрудники Symantec'а не обнаружили бы столько "тупых" ошибок. А сколько еще дыр предстоит открыть? Переводить критические сервера на Висту (точнее Server Longhorn) это верное самоубийство! Как минимум, должно пройти несколько лет и выйти не меньше двух-трех Service Pack'ов, но даже тогда переписанный сетевой стек остается источником "сюрпризов" в виде хакерских атак с полным захватом управления удаленным узлом.

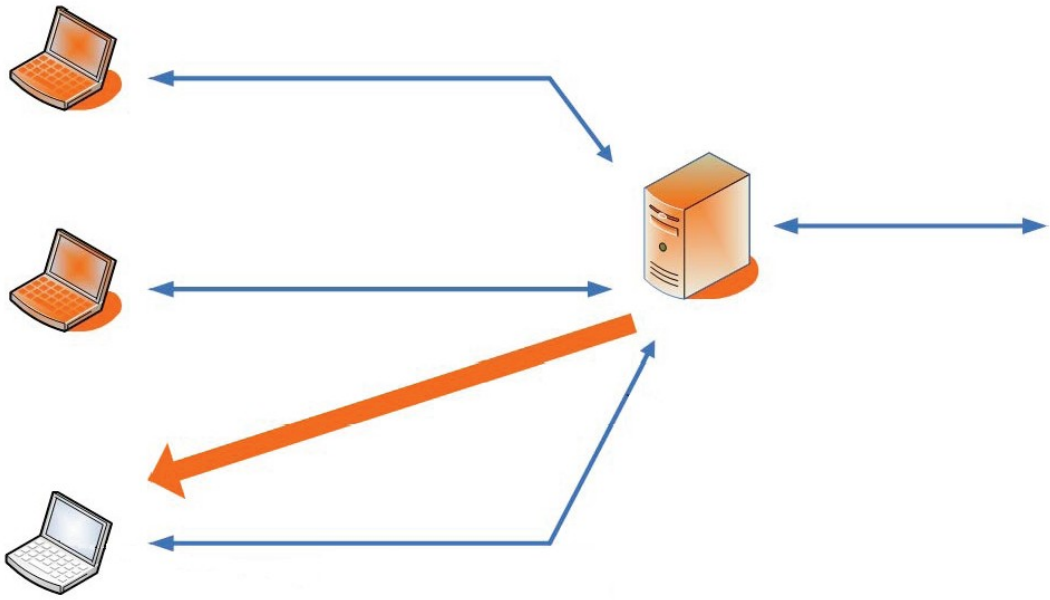

Рисунок 7 переписанный с нуля сетевой стек делает Висту весьма небезопасной системой в плане хакерских атак

Кстати, растет количество сообщений о нестабильной работе и зависаниях Висты. На самом деле, на нормальном желе Виста просто так не зависает, но где вы видели это нормальное железо?! Предъявляя более высокие требования (по сравнению с XP) к аппаратуре, Виста гоняет аппаратуру и в хвост и в гриву. Очень часто зависания удается устранить, установив дополнительное охлаждение на серверный мост чипсета (охлаждение памяти и силовых элементов стабилизаторов так же не помешает), но в некоторых случаях приходится менять блок питания вместе с материнской платой целиком, выбирая качественные модели по соответствующей цене. Другая возможная причина — конфликт с драйверами, которые еще не были протестированы производителем под Висту, так что с подбором железа придется повозиться!

Вообще, с поддержкой нового железа в 64-битной версии Висты дела обстоят далеко не самым лучшим образом. По не совсем понятным соображениям, Microsoft ввела _обязательную_ политику цифровой подписи драйверов. Не подписанный драйвер невозможно загрузить даже имея права администратора!!! Причем, процедура выдачи цифровой подписи довольно запутана и далеко не бесплатна. Если даже крупнейшие разработчики (типа Matrox или NVIDIA) испытывают большие проблемы с ее получением (и новейшие версии драйверов обычно некоторое время остаются неподписанными), то что же тогда говорить о мелких? Не стоит так же забывать о "псевдо-драйверах" — никаким оборудованием они не управляют и оформлены в виде драйвера только затем, чтобы получить доступ к необходимым ядерным функциям, недоступным с прикладного уровня. На этом принципе, в частности, построены антивирусы, персональные брандмауэры и многие другие программы, перехватывающие определенные системные функции и следящие за всеми интересующими их событиями (например, попытка запуска исполняемого файла).

Теперь всему этому придет конец. Microsoft в целях повышения защищенности системы запретила модификацию ядра и большинства прилегающих к нему компонентов. Мотив — уж слишком много программистов (и хакеров) стало решать свои задачи путем "доработки" ядра "напильником" и далеко не всегда эта доработка выполняется правильно с учетом всех архитектурных особенностей операционной системы. Как следствие — на пользователя обрушивается шквал Экранов Голубой Смерти, в которых пользователь винит Microsoft, которой надоело за чужие грехи отдуваться и она пошла по пути наименьшего сопротивления,

то есть по пути запрета. Специальный компонент ядра — Patch Guard периодически сканирует систему на проверку целостности и при обнаружении вторжения аварийно завершает ее выполнение.

Что ж, логика Microsoft вполне понятна, только ведь программисты отнюдь не от хорошей жизни полезли править ядро. Если бы поставленные задачи решались легальными средствами, то никакой бы модификации вообще не понадобилось!!! Формально, Microsoft предоставляет набор функций, позволяющих антивирусу или брандмауэру перехватывать определенные системные события, но… эти функции только для честных приложений, а нечестное может отнять у антивируса управление так что тот об этом даже не узнает! Скажите, вам нужен брандмауэр, контролирующий трафик честных приложений, но не защищающий даже от простейших атак?!

Таким образом, мы будем вынуждены либо использовать ненадежные защитные пакеты от сторонних поставщиков, либо ограничиться тем, что уже встроено в ядро самой Microsoft. Действительно, какое-то подобие брандмауэра там есть, но… во-первых, не всех он устраивает, а, во-вторых, хакеры уже давно научились его обходить.

Самое забавное, что Patch-Guard очень легко обмануть или даже вообще "оглушить", действуя хакерскими методами, так что на судьбе вирусов, червей и root-kit'ов запрет на модификацию ядра никак не отразится, а вот честные разработчики оказываются в очень неприятном положении. Они не могут отключать Patch-Guard, и не только потому, что законность такого действия вызывает сомнения. Легальных способов отключения — нет и не будет, а нелегальные — не надежды и могут вызывать различные проблемы у конечных пользователей. Но если хакерам на это наплевать, то уважающие себя разработчики таким путем идти не могут. Фактически их просто выжимают с рынка в пользу решений от Microsoft, качество которых в отсутствии конкуренции лучше не станет.

Подчеркнем, что Patch-Guard существует только в 64-битная редакции Висты. 32-битную Microsoft решила не трогать, поскольку такое бы решение вызвало бы неработоспособность огромного количества уже написанного программного обеспечения и никому из пользователей Виста оказалась бы не нужна. А вот под 64-битные платформы программного обеспечения существует не так уж и много и требование обратной совместимости не стоит.

заключение

Отличия Висы от XP носят довольно радикальный характер, но увы… этот характер совсем не в пользу рабочих станций (а на сервера Висту ставить из-за переписанного сетевого стека слишком опасно). Переход на Висту влечет за собой большие расходы (в первую очередь на качественное железо), но ничего не дает в замен. А интерфейс… хм, ну интерфейс. Ну, красивый. Только ведь нам не любоваться надо, а работать!