настойка файловой системы на максимальную производительность

крис касперски ака мыщъх, no email

LINUX (в отличии от Windows) поддерживает целый спектр файловых систем разного калибра и назначения: minix, ext2fs, ext3fs, ReiserFS, XFS, JFS, UFS, FFS… Какую файловую систему выбрать? Как правильно ее настроить? Стандартный выбор, предлагаемый составителями дистрибутива по умолчанию, не всегда оптимален и быстродействие системы можно значительно улучшить, если залезть вовнутрь и слегка ее подкрутить.

введение или железный дровосек на пеньке

Жесткий диск — хитрый зверь. Тихий как мыщъх, быстрый как леопард, надежный как сенбернар. Но процессор еще быстрее! И дисковая подсистема несмотря на все усилия инженеров по-прежнему остается слабейшим звеном, сдерживающим быстродействие всего компьютера в целом. А ведь объемы обрабатываемых данных все растут и растут….

Большинство материнских плат, выпушенных после 2000 года, несут на своем борту интегрированный RAID-контроллер, поддерживающий режимы RAID-0 ("stripe" mode – режим чередования при котором данные пишутся на несколько жестких дисков сразу) и RAID-1 ("mirror" mode — зеркальный режим при котором жесткие диски дублируют друг друга). Режим чередования значительно увеличивает производительность — два диска работают приблизительно в 1.5 раза быстрее, а четыре – в ~3.5 раза быстрее, чем один.

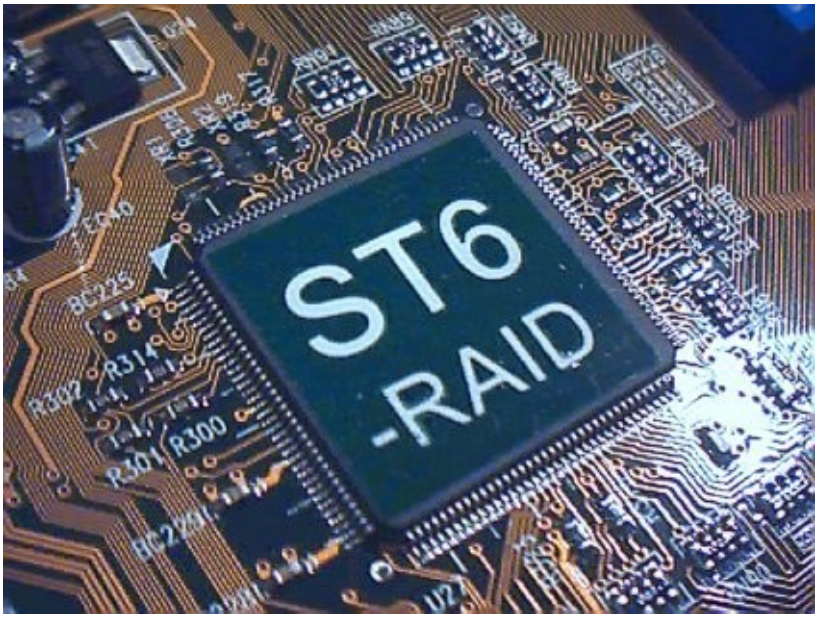

Рисунок 1 RAID-контроллер, интегрированный в материнскую плату

Обладатели ядра с версией 2.4 или более старшей могут использовать программный RAID-массив (software RAID), практически не уступающий по скорости аппаратному, но слегка напрягающий процессор. Более древние ядра (кстати говоря, уже практически вышедшие из употребления) скорее всего потребуют установки дополнительно программного обеспечения. Подобротнее об этом можно прочитать тут: http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html

Большинство руководств настоятельно рекомендуют подключать программный RAID к различным IDE-каналам, т. е. разводить диски по "своим" шлейфам. Проблема в том, что типичная материнская плата имеют всего два IDE-канала, а ведь помимо жестких дисков требуется как минимум один оптический привод! Для достижения наивысшей скорости приходится приобретать мать с несколькими IDE-каналами, что поделаешь — оптимизация

требует жертв! В частности, у EPOX 4PCA3+ этих каналов целых 6, но… она не всем по карману. В действительности же, совмещать два жестких диска на одном шлейфе — можно. Это совсем-совсем нестрашно. Они могут работать и параллельно. Ну… почти параллельно, на ~15% скорость все-таки упадет. Современные накопители освобождают шину на время выполнения медленных операций, но шина все-таки одна, а накопителей двое, вот им и приходится за нее сражаться. А вот жесткий диск с оптическим приводом на одном шлейфе лучше не совмещать, в некоторых случаях скорость падает в разы (попробуйте отключить у оптического привода режим DMA, возможно, это поможет винчестеру заработать быстрее).

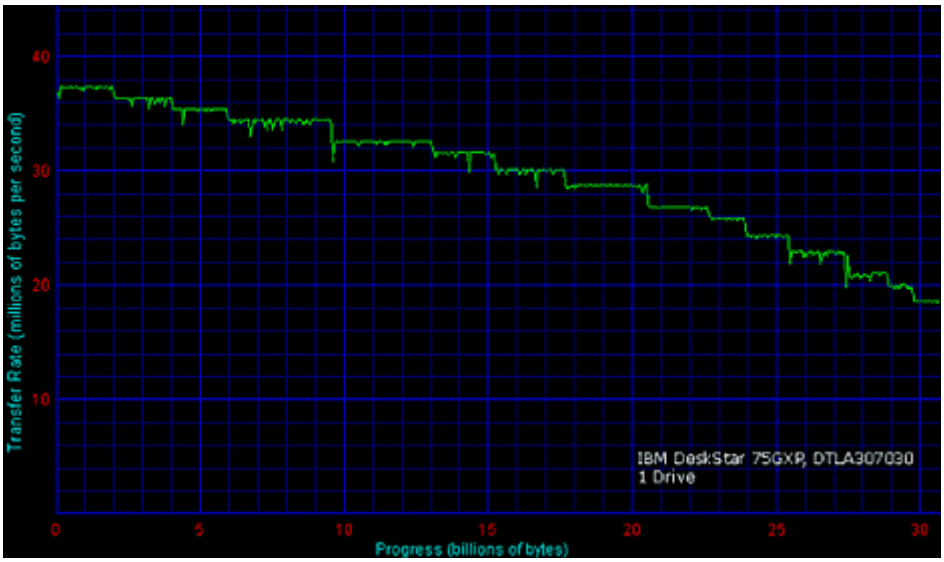

Рисунок 2 программный RAID. один диск — один канал

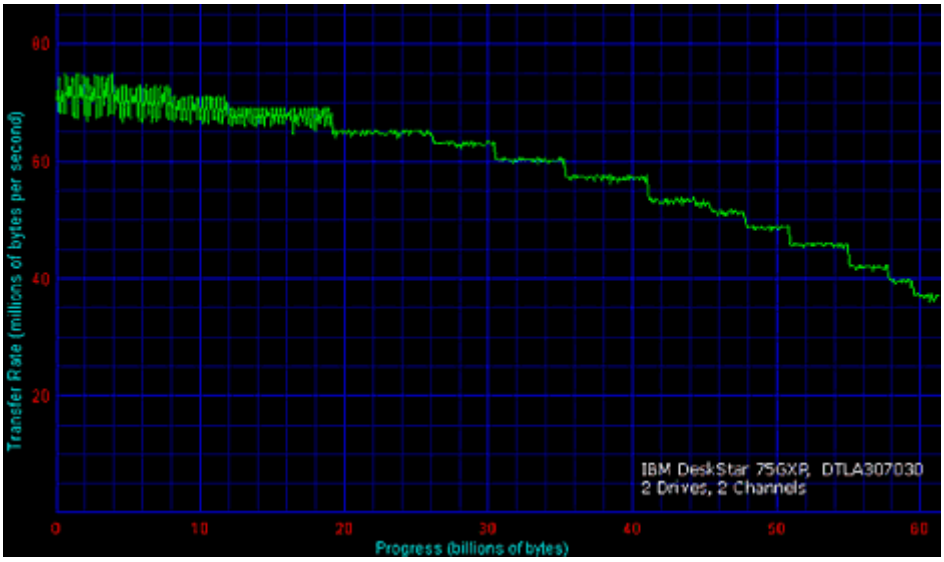

Рисунок 3 два диска — два канала

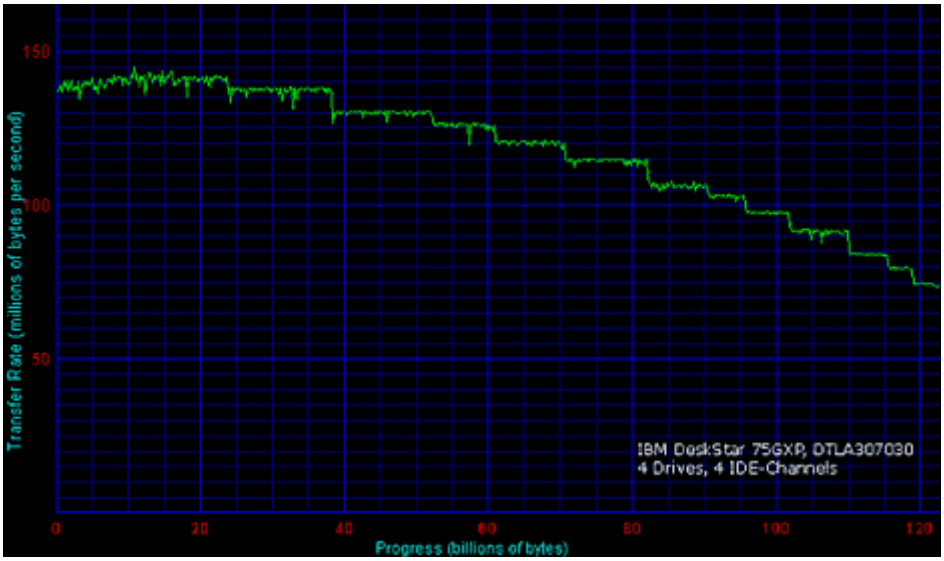

Рисунок 4 четыре диска — четыре канала

Дисковый массив, состоящий из 12 винтов, подключенных к EPOX 4PCA3+, работает со сверхзвуковой скоростью, но и шумит как самолет, не говоря уже о том, что приходится покупать мощный блок питания на 350 Ватт и ставить специальные фильтры на разветвлитель, чтобы подавлять помехи, к которым жесткие диски весьма чувствительны. Но выигрыш в скорости стоит того, особенно, если компьютер используется для занятий видеомонтажом или обработки изображений полиграфического качества. Но с таким потребностями лучше сразу обратится к SCSI-дискам. Мы же остановимся на IDE как на самом демократичном и дешевом интерфейсе.

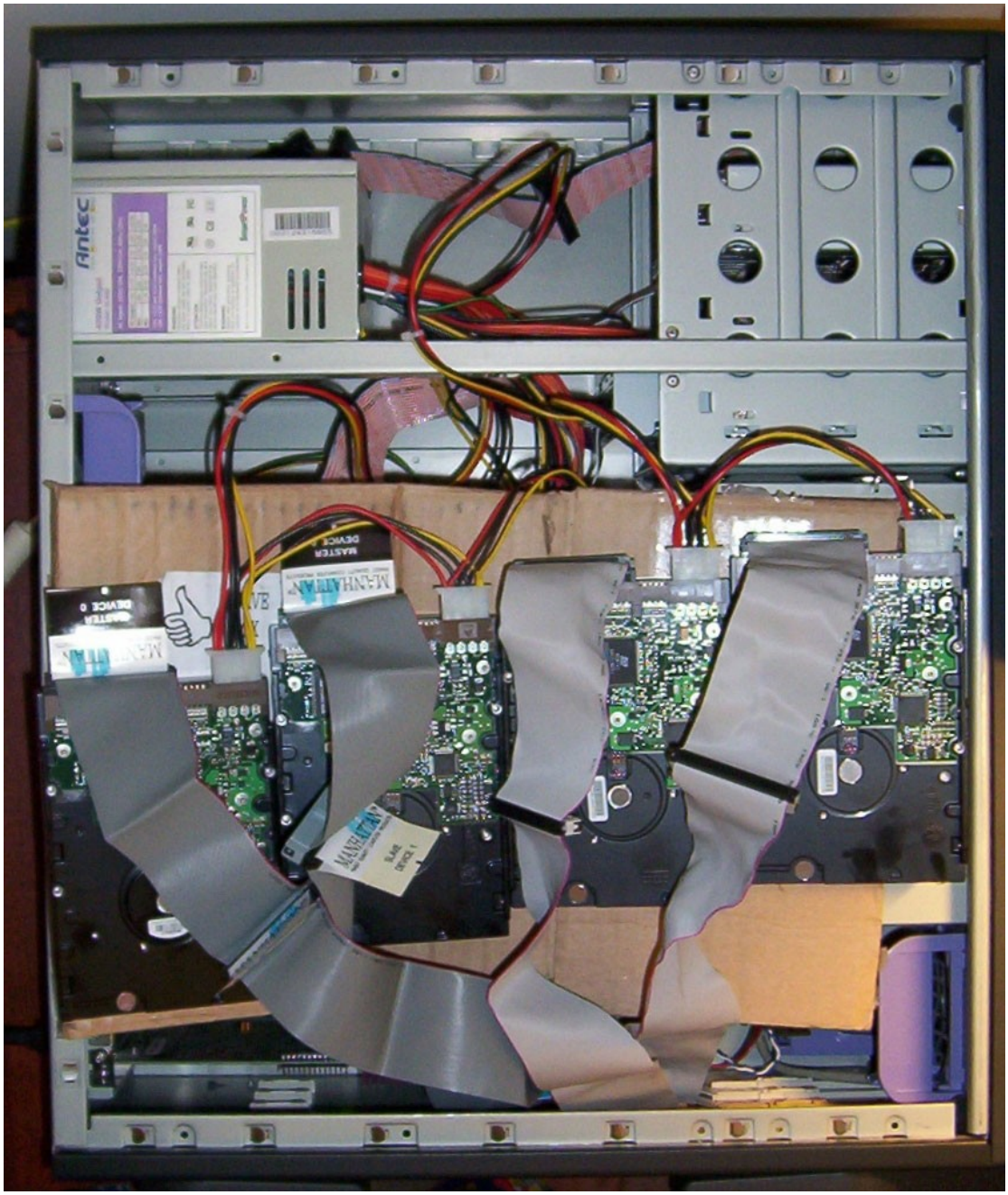

Рисунок 5 массив из четырех дисков — проблемы монтажа

hdparm – крутим, вертим, сверлим, клеим

Для достижения наивысшей производительности каждый жесткий диск, установленный в систему, должен быть настроен в соответствии со своим предназначением. Стандартные настройки, принимаемые ядром по умолчанию, ориентированы на абстрактного среднестатистического пользователя и редко совпадают с конкретными требованиями. Учет преобладающего типа запросов к дисковой подсистеме значительно повышает быстродействие (в некоторых случаях чуть ли не на порядок), хотя это оружие работает и в обратном направлении. Бестолковая настройка сваливает производительность в глубокую яму, из которой, впрочем, всегда можно выбраться применив настройки по умолчанию.

Всем этим ведает консольная утилита **hdparm**, входящая в комплект штатной поставки большинства (если не всех) LINUX'ов и работающая из-под root'а. В случае чего, взять ее можно здесь: http://metalab.unc.edu/pub/Linux/system/hardware/hdparm-3.6.tar.gz. Формат ее вызова следующий:

```
hdparm опция1 опция2 … опцияN /dev/жесткий_диск
```

Листинг 1 вызов hdparm из командной строки

Жестким дискам с IDE-интерфейсом обычно присваиваются имена hda (первый жесткий диск), hdb (второй жесткий диск), hdc и так далее. SCSI диски, соответственно, именуются sda, sdb, sdc, только hdparm с ними, увы, не работает. Строго говоря, hdparm настраивает параметры не одного лишь жесткого диска, но так же его контроллера и отчасти драйвера, но это слишком длинно писать, так что не будем углубляться в терминологические тонкости, а сразу перейдем к конкретным параметрам.

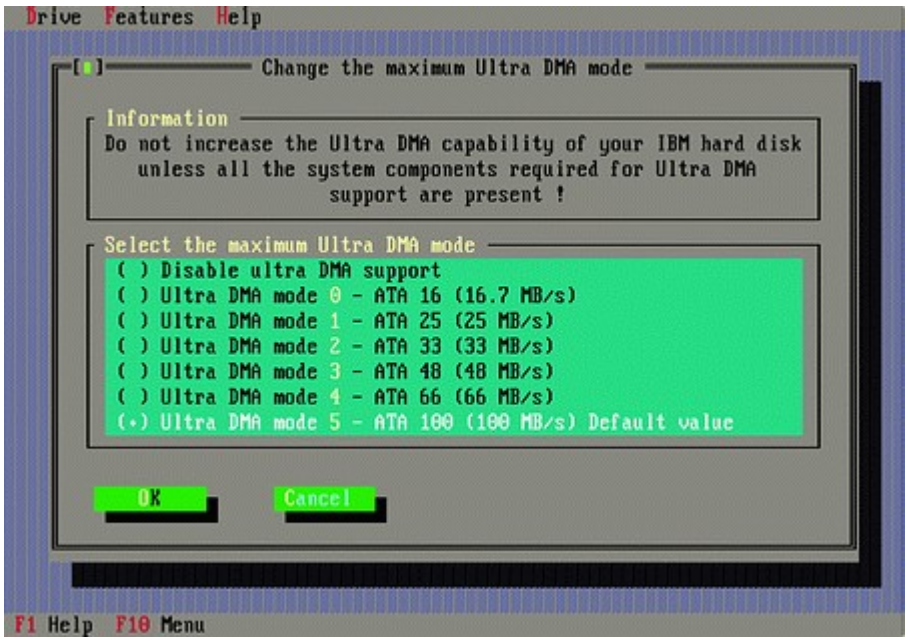

Рисунок 6 hdparm в интерактивной оболочке

Ключ –a устанавливает количество секторов "опережающего чтения", которые будут автоматически прочитаны контроллером в надежде, что они все-таки пригодятся пользователю. По умолчанию, ядро читает 8 секторов (4 Кбайта). При последовательном чтении больших слабофрагментированных файлов это значение рекомендуется увеличить в несколько раз, а при хаотичном доступе, работе с мелкими или сильнофрагментированными файлами — уменьшить до 1 – 2 секторов. Ключ –P задействует механизм аппаратной предвыборки, сообщая приводу сколько секторов ему необходимо прочитать. Грубо говоря, это тоже самое, что и –a, только намного круче. Однако, не все приводы поддерживают аппаратную предвыборку.

Ключ –m специфицирует количество секторов, обрабатываемых приводом за одну операцию обмена (так называемый multiple sector I/O или block mode). В зависимости от конструктивных особенностей жесткого диска он может обрабатывать от 2 до 64 (и больше) секторов за раз. Конкретное значение можно узнать с помощью ключа –i (оно находится в графе MaxMultSect). В общем случае, скорость обработки данных прямо пропорциональна кол-ву секторов, однако некоторые приводы (например, WD Caviars) при больших значениях –m начинают жутко тормозить. Выяснить практическое положение дел помогает ключ –t, измеряющий пропускную способность дисковой подсистемы в режиме чтения. ***Внимание! Запредельные значения –m могут привести к повреждениям данных, поэтому, не рискуйте без необходимости!***

Ключ –M отвечает за настройку шумовых характеристик накопителя (Automatic Acoustic Management или сокращенно AAM). Значение 128 соответствует наиболее тихому режиму, 254 — наиболее быстрому. Промежуточные значения в общем случае неопределенны (некоторые накопители их поддерживают, некоторые нет). Следует сказать, что значение 128 не только уменьшает шум, но и способствует меньшому износу накопителя, однако, падение производительности может быть очень и очень значительным, поэтому трудно посоветовать какое именно значение выбрать.

Ключ –c управляет режимом передачи данных. Параметр 0 – 16-битная передача, 1 — 32-битная передача, 3 — 32-битная передача со специальным синхросигналом. По умолчанию ядро использует параметр 3 (возможно, не для всех ядер), как наиболее надежный, но и менее производительный чем 1. Большинство современных чипсетов вполне нормально работают с параметром 1, так что излишняя осторожность тут ни к чему.

Ключ –d1 активирует, а –d0 дезактивирует режим DMA, значительно увеличивающий производительность и радикально снижающий нагрузку на процессор. Однако, так бывает далеко не всегда. IDE-устройства, висящие на одной шине, могут конфликтовать между собой и тогда хотя бы одно из них должно быть принудительно переведено в режим PIO. Выяснить как обстоят дела в данном конкретном случае помогает ключ –T, измеряющий скорость передачи данных. Ключ –d1 обычно используется совместно с ключом –Xnnn, форсирующим конкретный режим PIO или DMA. Режиму PIOn соответствует значение (n + 8), т.е. –X9 задает PIO1, а –X12 – PIO4. Режиму DMAn соответствует значение (n+32), например –X34 для DMA2, а Ultra DMA – (n+64), например, -X69 для UDMA5, который обеспечивает наивысшую производительность, однако, поддерживается не всеми жесткими дисками и чипсетами. Узнать список поддерживаемых режимов можно с помощью ключа –i. По умолчанию ядро выбирает не слишком агрессивные режимы передачи данных, оставляя солидный запас производительности за спиной. Однако, переход на высшие UDMA режимы чреват разрушением всего дискового тома, поэтому обязательно зарезервируйте его содержимое перед началом экспериментов!

Рисунок 7 чипсеты от VIA всегда славились кривой поддержкой высших UltraDMA режимов, на которых происходило разрушение данных

Для сохранения установок необходимо дать команду "`hdparm -k 1 /dev/hdx`", в противном случае они будут утеряны при первом же сбросе IDE-контроллера или перезапуске машины.

выбор файловой системы

Существует два типа файловых систем — журналируемые (journaling) и нет. К первым относятся ext3fs, ReiserFS, XFS, а последним — minix, ext2fs и UFS. Журналирумые файловые системы намного легче переносят зависание системы и отключение питания во время

интенсивных дисковых операций, автоматически возвращая файловую систему в стабильное состояние, однако, от других типов разрушений (отказ контроллера, дефекты поверхности, вирусное нашествие) оно никак не спасает, а вот производительность падает изрядно.

Для домашних компьютеров и большинства рабочих станций журналирование на хрен не нужно и надежности файловой системы ext2fs вполне достаточно, особенно если компьютер оборудован UPS'ом. В ответственных случаях используйте ext3fs или ReiserFS. По тестам (ну типа там сферический конь в вакууме), ReiserFS в среднем вдвое, а на операциях записи в 35 раз быстрее, чем ext3fs, что особенно хорошо заметно на мелких файлах. В реальной же жизни часто все бывает наоборот. Высокая латентность ReiserFS (т. е. промежуток между подачей запроса и получением ответа) вкупе с агрессивной загрузкой процессора заметно отстает от ext3fs, что особенно хорошо заметно на мелких файлах (да-да, на тех самых, на которых нам обещали выигрыш!). Подробнее об этом можно прочитать здесь http://kerneltrap.org/node/view/3466.

Журналирование можно значительно ускорить, если размесить журнал на отдельном носителе. Такой журнал называется внешним (external). Подключить его можно командной "tune2fs –J device=external_journal" (где external_journal имя раздела соответствующего устройства), причем внешний журнал должен быть предварительно создан командой "mke2fs –O journal_dev external_journal". Команда "tune2fs –J size=journal_size" управляет размером журнала. Чем меньше размер журнала, тем ниже производительность. Предельно допустимый размер составляет 102.400 блоков или ~25 Мбайт (точное значение зависит от размера блока, о котором мы еще поговорим).

По умолчанию ext3fs журналирует только метаданные (т. е. служебные данные файла, такие например, как INODE), записывая их на диск только после того, как будет обновлен журнал. Для увеличения быстродействия можно задействовать "разупорядоченный" режим, в котором метаданные записываются одновременно с обновлением журнала, что соответствует команде: "mount /dev/hdx /data –o data=writeback". Естественно, надежность файловой системы при этом снижается. При желании можно журналировать все данные (команда "mount /dev/hdx /data –o data= jourmal"), после чего никакие зависания или отказы питания нам будут не страшны, правда о производительности придется забыть.

При создании новой файловой системы важно выбрать правильный размер блока (в терминологии MS-DOS/Windows – кластера). На ext2fs, ext3fs это осуществляется командой "mke2fs -b block-size", на XFS – " mkfs.xfs -b size=block-size" и "newfs -b block-size" на UFS. Чем больше блок, тем ниже фрагментация, но и выше дисковые потери за счет грануляции дискового пространства. Некоторые файловые системы (например, UFS) поддерживают фрагменты (fragments) — порции данных внутри блоков, позволяющие задействовать свободное пространство в "хвостах" блоков, благодаря чему использование блоков большого размера уже не приводит ни к каким потерям. Файловая система ReiserFS в отличии от остальных, не нарезает диск на ломтики фиксированного размера, а динамически выделяет требуемый блок данных, забивая диск файлами под завязку. В среднем это на 6% увеличивает доступный объем, однако, приводит к чрезмерной фрагментации, "съедающий" всю производительность. Рекомендуется использовать максимально доступный размер блока (4Кбайта для ext2fs и ext3fs, 16 Кбайт для UFS и 64 Кбайта для XFS, файловые системы ReiserFS и JFS не поддерживают этой опции) и задействовать максимальное количество фрагментов на блок (в UFS – 8).

Другая важная опция определяет режим хеширования директорий. Для ускорения работы с директориями, содержащими большое количество файлов и подкаталогов, директория должна быть организована в виде двоичного дерева. В ext2fs и ext3fs это осуществляется командой "mke2fs –O dir_index", а в ReiserFS — "mkreiserfs –h HASH", где HASH – один из следующих типов хэш-таблицы: r5, rupasov или tea. По умолчанию выбирается r5, который наилучшим образом подходит для большинства файловых операций, тем не менее некоторые приложения (например Squid Web Proxy-сервер) настоятельно рекомендует использовать rupasov-хэш, в противном случае за быстродействие никто не ручается. С другой стороны, r5 и rupasov очень медленно работают с директориями, содержащими несколько миллионов файлов и здесь лучше подходит tea, а на директориях из нескольких десятков файлов все три алгоритма хеширования проигрывают стандартному нехешируемому plain-алгоритму. К сожалению, опция хеширования носит глобальный характер — нельзя одни директории хешировать, а другие нет.

Файловая система XFS — единственная из всех, кто позволяет задавать размер INODE вручную. Обычно в INODE хранятся служебные данные файла (атрибуты, порядок размещения блоков на диске), но если файл целиком умещается в INODE система сохраняет его именно там! Дополнительное дисковое пространство уже не выделяется, что избавляет головку винчестера от лишних перемещений, в результате чего время доступа к файлу существенно сокращается.

Точно так же поступают ReiserFS, NTFS и некоторые другие файловые системы, однако, размер INODE они менять не в состоянии, а жаль! Если мы планируем работать с большим количеством мелких файлов, размер INDOE желательно увеличить, что положительно скажется как на производительности, так и на доступном дисковом пространстве. При работе с большими файлами размер INODE лучше, наоборот, сократить, в противном случае потери дискового пространства будет довольно значительными. Выбор предпочтительного размера INODE осуществляется командной "mkfs.xfs –i size=value". Минимальный размер составляет 512 байт, максимальный — 2048.

Рисунок 8 Файловая система ReiserFS собственной персоной — разработка, спонстируемая фирмой Novell.

заключение

Windows предоставляет минимум рычагов управления для настройки дисковой подсистемы и угробить свои данные под ее управлением довольно затруднительно. LINUX же позволяет крутить вся и все! Как следствие — малейшая оплошность приводит к катастрофическим разрушениям. И винить в этом некого — нечего было браться за штурвал, не выучив мануал, как правило написанный на английском языке. Но даже мануал не поможет определить какие именно режимы поддерживаются вашим оборудованием, а какие нет (может, у вас кабель перекручен или разъем барахлит, а на высокосортных режимах это сражу же скажется!). Настройка дисковой подсистемы на максимальную производительность — это огромный риск! Никогда не экспериментируйте, не зарезервировав всех данных!

>>> врезка фрагментация

В процессе работы с диском его фрагментация неизбежно увеличивается. Больше всего от этого страдают ext2fs/ext3fs и ReiserFS. На UFS и XFS за счет поддержи блоков большого размера падение производительности уже не так заметно. Утверждение, что файловые системы LINUX якобы не подвержены фрагментации — нелепый миф, который может быть легко опровергнут любым опытным пользователем.

При последовательной записи на диск нескольких файлов, система их размещает один за другим, так что первый файл "упирается" во второй. Свободного места для "карьерного" роста уже нет (короткий "хвост" в конце блока не считается) и система вынуждена выделять блоки где-то за концом следующего файла. Если же их там нет, свободные блоки ищутся вначале диска, в результате чего файл как бы "размазывается" по поверхности. Или вот другой случай. Мы записали пять файлов по 100 блоков каждый и затем удалили первый, третий и пятый файлы, освободив 300 блоков в трех фрагментах. При записи 300-блочного файла, система сначала попытается отыскать непрерывный регион свободного пространства, но если его не окажется, будет вынуждена "размазывать" файл по поверхности. Чтобы исправить ситуацию, необходимо собрать все свободные блоки, объединив их в один непрерывный фрагмент, то есть дефрагментировать раздел.

Из бесплатных дефрагментаторов лучшим на мой взгляд является стандартный defrag, входящий в штатный комплект поставки большинства LINUX'ов. Если же в вашем дистрибьютиве его нет, исходные тексты дефргаментатора можно утянуть отсюда: ftp://metalab.unc.edu/pub/Linux/system/filesystems/defrag-0.70.tar.gz

Фирма OO-Software, известная своим одноименным дефрагментатором для NT, выпустила замечательный консольный дефрагмантатор для LINUX, в настоящее время находящийся в стадии бета-тестрирования и распространяющийся на бесплатной основе. Так что качайте его пока дают, а скачать его можно отсюда: http://www.oo-software.com/cgi-bin/download/download-e.pl?product=OODLXBIN

Регулярная дефрагментация это хороший способ противостоять растущему падению производительности файловой системы.

Рисунок 9 вот что значит, фрагментация!

>>> врезка обновлять или не обновлять

Некоторые приложения, в частности уже упомянутый (например Squid Web Proxy-сервер) требуют особой настройки файловой системы "под себя". Для увеличения быстродействия рекомендуется отключить обновления времени последнего доступа к файлу ("mount —o noatime"). Наибольший прирост производительности наблюдается на UFS, которая в отличии от подавляющего большинства остальных файловых систем, не откладывает обновление INODE в долгий ящик (lazy write), а делает это сразу же после его изменения (write through). На ext3fs в силу ее журналирующей природы, обновление atime вносит столь незначительный вклад в общее быстродействие, что никакой разницы просто нет. оппонентов

>>> врезка покажи мне свой хвост и я скажу кто ты

По умолчанию ReiserFS сохраняет короткие файлы (и файловые хвосты) на листьях двоичных деревьев. В большинстве случаев это многократно увеличивает производительность, особенно если свободное дисковое пространство далеко от исчерпания (см .рис. 10, 11). Тем не менее, при работе с некоторыми приложениями, "хвосты" лучше отключить. При работе с огромным количеством мелких файлов, которые постепенно растут, системе приходится перестраивать большое количество структур данных, "гоняя" растущие хвосты между блоками и деревьями, в результате чего производительность держится на уровне плинтуса. Команда "mount –o notail" отключает "паковку" хвостом и коротких файлов, а повторное монтирование с настройками по умолчанию включает ее обратно, однако, следует помнить, что уже "упакованые"/"распакованные" хвосты останутся на своем месте вплоть до модификации "своего" файла.

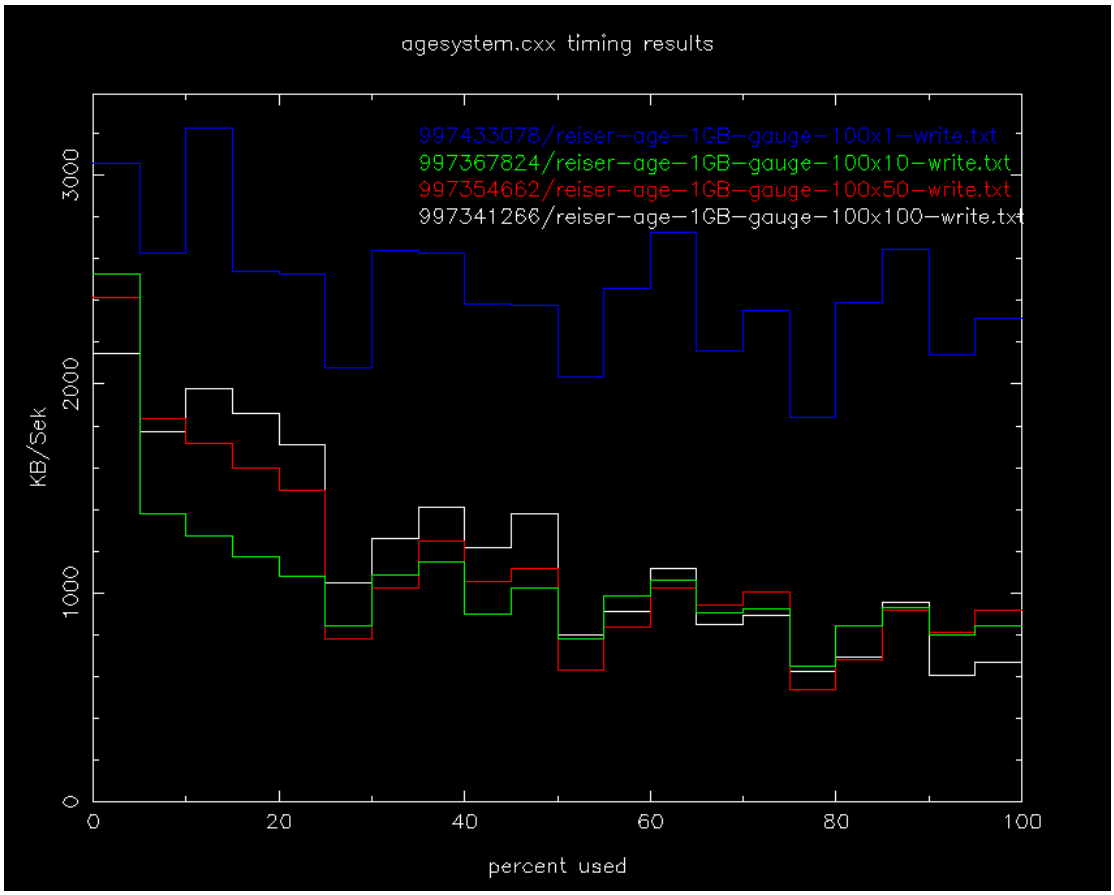

Рисунок 10 производительность файловой системы ReiserFS на операциях записи в зависимости от объема свободного пространства (паковка хвостов включена)

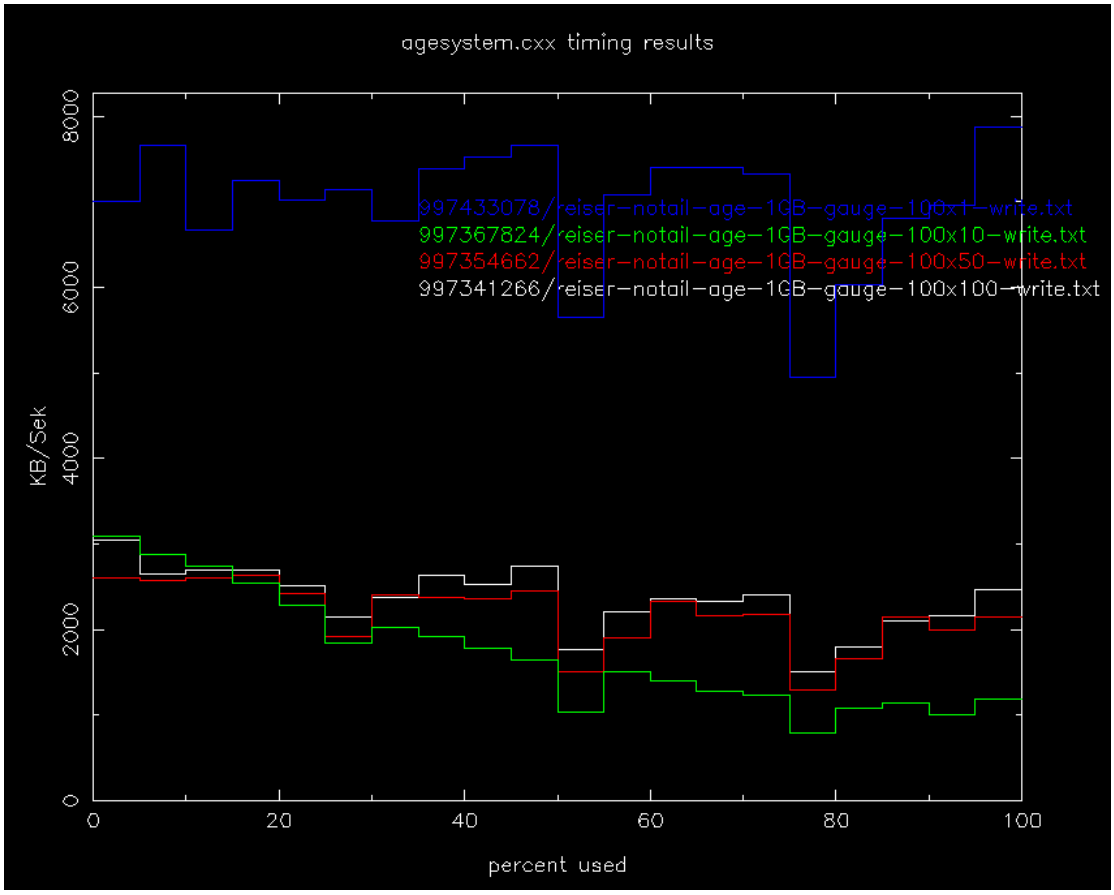

Рисунок 11 производительность файловой системы ReiserFS на операциях записи в зависимости от объема свободного пространства (паковка хвостов выключена)

>>> врезка внимание!

Помните, что mke2fs это деструктивная команда, разрушающая всю файловую систему целиком! Грубо говоря, это format.com под LINIX.

>>> врезка ссылки по теме

- The Software-RAID HOWTO
  - руководство по созданию программных RAID'ов под LINUX'ом (на английском языке): http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html
- Тонкая настройка IDE дисков в Linux с помощью hdparm
  - отличная статья на русском языке www.opennet.ru/base/sys/htparm_tune.txt.html;
- JFS for Linux
  - домашняя страничка проекта JFS — исходные тексты, документация, технология и т.д. (на английском языке): http://jfs.sourceforge.net/
- ReiserFS
  - домашняя страничка проекта ReiserFS (на английском языке): http://www.namesys.com;
- Работа с дисками и файловыми системами в FreeBSD
  - отличный faq на русском языке: www3.opennet.ru/base/sys/freebsd_fs_mount.txt.html
- Understanding Filesystem Performance for Data Mining Applications
  - сравнение производительности различных файловых систем под LINUX с советами по их "тонкой" настройке (на английском языке):http://www.cs.rpi.edu/~szymansk/papers/hpdm03.pdf
- Linux Filesystem Performance Comparison for OLTP

- еще одна статья по сравнению производительности файловых систем под LINUIX (на английском языке): http://otn.oracle.com/tech/linux/pdf/Linux-FS-Performance-Comparison.pdf;

- Journaling file systems
  - журналируемые файловые системы и все, что с ними связано (на английском языке): awlinux1.alphaworks.ibm.com/developerworks/linux390/perf/tuning_res_journaling.shtml;
- Linux: Low Latency and Filesystems
  - обсуждение преимуществ и недостатков ReiserFS (на английском языке): http://kerneltrap.org/node/view/3466;
- ext3 or reiserfs? hans reiser says red hat's move is understandable.
  - еще одно сравнение ext3fs и ReiserFS (на английском языке) http://www.linuxplanet.com/linuxplanet/reports/3726/1/;
- Optimizing Linux filesystems
  - отличная статья про оптимизацию файловых систем под LINUX (на английском языке): http://www.newsforge.com/article.pl?sid=03/10/07/1943256
- Journaling-Filesystem Fragmentation Project
  - исследовательская работа по фрагментации файловых систем и ее влиянию на производительность (на английском языке) http://www.informatik.uni-frankfurt.de/~loizides/reiserfs/agesystem.html;
- HDD REPAIR FORUMS
  - форум по тестированию жестких дисков и восстановлению данных (на русском языке): http://mhddsoftware.com/forum/
- Filesystem defragmenter for Linux filesystems
  - исходные тексты стандартного деферагментатора: ftp://metalab.unc.edu/pub/Linux/system/filesystems/defrag-0.70.tar.gz
- O&O Defrag Linux BETA - 1.0.4761
  - бета-версия хорошего коммерческого дефргаментатора: http://www.oo-software.com/cgi-bin/download/download-e.pl?product=OODLXBIN