война миров: ext2fs и ext3fs
взгляд под необычным углом

крис касперски ака мыщъх, no-email

о преимуществах и недостатках файловой системы ext3fs написано много и по общему мнению она обеспечивает лучшую надежность за счет снижения производительности, однако, далеко не всегда ext3fs отстает от ext2fs и в некоторых случаях она даже ее обгоняет, причем значительно

alt: сравнивать файловые системы все равно что спорить о женщинах. одним нравится блондинки, другие без ума от толстушек, а суть… она определяется точкой схождения двух прямых — ее ног. очень трудно удержаться от соблазна и остаться непредвзятым. но все-таки попробуем…

введение

Истинный смысл не в тестах, не в графиках и не диаграммах, а в их физической интерпретации. Постановка эксперимента — это же не просто так! Чтобы получить достоверные, воспроизводимые и объективные результаты необходимо знать как устроена файловая система и какие шестеренки приводят ее в движение. Всегда можно подобрать такой набор текстов, на котором "хорошая" файловая система будет быстрее "плохой", а всех несогласных обозвать ламерами, ничего не смыслящими в тонких эффектах многозадачной операционной системы, многоуровневого кэша и т. д.

Попробуем сравнить файловые системы сразу по нескольким критерием: надежности, отказоустойчивости, производительности и т.д., чтобы каждый смог выбрать нужную. Вот с надежности мы, пожалуй, и начнем.

когда данные обращаются в прах

Файловые системы ext2fs и ext3fs очень похожи. Фактически, ext3fs это ext2fs с поддержкой журналирования, то есть транзакцией. Транзакциями называют групповые операции, выполняемые или невыполняемые как одна единая операция. Другими словами атомарно. Поясним это на следующем классическом примере перевода денег из банка А в банк Б. На низком уровне эта операция разбивается на две: снятие денег со счета и перевод. А если во время перевода произойдет сбой и выполнение программы прервется? Чтобы не оставить клиента без денег, необходимо предусмотреть автоматический "откат". Перевод либо выполняется, либо нет. Промежуточные состояния недопустимы. Во всяком случае в теории.

Вернемся к файловым системам. Почему в FAT16/32 постоянно образуются потерянные кластеры? Да потому, что она не поддерживает транзакций и многостадийные операции выполняются не атомарно! Вот, например, копирование файла. Система выделила дисковое пространство и только собиралась передать его файлу, как все повисло (варианты: монтер перерезал провода, юзер нажал на RESET) и один или несколько кластеров остались ничейными.

Журналируемые файловые системы (ext3fs, NTFS) в таких случаях делают автоматический откат при следующей загрузке и потери кластеров не происходит. Создание/удаление/переименования файла это атомарные операции, не допускающие промежуточных состояний. А вот с операциями перемещения все намного сложнее. Файловая система не позволяет перемещать файл между томами, вынуждая программу-оболочку делать это самостоятельно. В результате операция переноса разбивается на две: копирование файла-источника в файл-приемник и удаление источника. При этом может возникнуть такая нехорошая ситуация, когда файл-приемник не был записан на диск (система не успела сбросить кэш, например), но источник уже был удален. Вот такие они… транзакции. К тому же, поддержка транзакций не страхует от потери записываемых данных, поскольку файл журнала обновляется не мгновенно, а с некоторой задержкой. Транзакции бессильны противостоять физическим дефектам поверхности, некорректно работающему программному обеспечению и т. д.

Многие сравнивают ext2fs с FAT, а ext3fs с NTFS, но это неверно. По своей архитектуре ext2fs гораздо ближе к NTFS, чем к FAT. Грубо говоря, etx2fs это NTFS без транзакций. За счет

высокой степени избыточности (большого количества дублирующих друг друга структур), ext2fs весьма стоически относится к сбоям и потому за ее целостность можно не волноваться. После внезапного выключения питания она не упадет. Поддержка транзакций в ext3fs увеличивает надежность хранения данных, но не столь радикально, как некоторые пытаются доказать. При выборе режима "журналировать только метаданные" (data=writeback), все открытые на запись данные в момент исчезновения питания могут обнуляться или заполняться мусором. В режиме "журналировать все" (data=journal) все данные сначала пишутся в журнал и только затем переносятся в файл. Это значительно снижает производительность, но зато гарантирует непротиворечивость состояния данных и метаданных: файл либо записывается полностью, либо не записывается вообще. То есть, потеря информации при внезапном исчезновении питания или перезагрузке все-таки возможна.

А вот восстанавливать данные на ext3fs значительно труднее, чем на ext2fs поскольку перед удалением файла список принадлежащих ему блоков тщательно вычищается и undelete сделать уже невозможным. Причем, это не баг, а "так задумано". На портале www.opennet.ru лежит FAQ по файловой системе ext3fs (http://www.opennet.ru/base/faq/ext3_faq.txt.html), которое со ссылкой на Andreas Dilger'a (одного из разработчиков), говорит следующее "*Для проверки возможности безопасного продолжения разлинковки (unlink) после падения файловая система ext3 обнуляет указатели на блоки в inode'ах, а ext2 просто помечает эти блоки как неиспользуемые, inode'ы -- как удаленные, оставляя указатели нетронутыми. Единственное, что вам остается делать, — вызывать grep для нахождения частей удаленных файлов и надеяться на лучшее*". На самом деле, не все так безнадежно. Да, указатели на DIRECT блоки гибнуть безвозвратно (и зачем их обнулять? как будто бы было нельзя пойти другим путем), но содержимое блоком косвенной адресации остается нетронутым и хвост файла восстанавливается элементарно. Собирать по кускам приходится только его начало. Подобротнее об этом можно прочитать в моей книге "Техника восстановления данных" (название рабочее), которая выйдет в ближайшее время, ну а пока доступна только "облегченная" версия, живущая на мыщъхином ftp.

Другая серьезная проблема — целостность журнала и агрессивный характер fsck, неадекватным образом реагирующий на некоторые виды повреждений. В последнее время появилось множество сообщений о некачественных SATA-контроллерах, приводящих к различным сбоям, затрагивающим журнал и метаданные. Основная структура тома остается практически неповрежденной (так маленькая трещинка) и ручным восстановлением его еще можно спасти, но запуск fsck грохает раздел окончательно, причем, ext3fs страдает намного сильнее, чем ext2fs. Вероятно, так происходит потому, что в журнале оказывается мусор, а fsck пытается его интерпретировать "правильным" образом, вот и… Конечно, поклонники ext3fs могут сказать, что не фиг гонять ее на кривом железе, купите себе нормальный SCSI-контроллер и выкиньте этот галимый ATA. Все это верно и совершенно правильно, но все-таки, от сбоев железа никто не застрахован, поэтому при обкатке нового оборудования все-таки лучше использовать ext2fs и только затем переходить на ext3fs. К тому же, прежде чем позволять fsck лечить диск, настоятельно рекомендуется запустить дисковый редактор lde (сокращение от LINUX DISK EDITOR) и посмотреть, что именно случилось с данными и, может быть, проще все восстановить вручную? (Описание приемов работы с lde можно найти в заначках мыщъх'а, лежащих по адресу http://kpnc.opennet.ru/recover.zip)

Так что вопрос надежности остается открытым и во многих случаях ext2fs все-таки оказывается более предпочтительной.

быстродействие или черепаха приходит первой

Считается, что в общем случае журналируемые файловые системы проигрывают нежурналируемым в производительности, однако, "общий случай" понятие растяжимое. Результаты тестов варьируются в очень широких пределах и без хорошей травы истины здесь не найдешь. Но зачем нам трава, когда у нас имеется гораздо более могучее оружие — логика.

Исходя из самых общих соображений, на операциях чтения обе файловые системы должны обеспечить идентичный уровень производительности, поскольку при чтении данных никакого обращения к журналу не происходит. В первом приближении это действительно так, в чем нас убеждают данные независимых тестеров, работающих на однодисковых десктопах (например: http://staff.osuosl.org/~kveton/fs/page2.php). Отсюда вывод: если операции чтения доминируют над операциями записи, разница в производительности между ext2fs и ext3fs становится практически незаметной, а на дисках, смонтированных "только на чтение" и вовсе равной нулю. На домашних компьютерах это действительно так.

На серверах и мощных рабочих станциях ситуация совсем иная. Там стоят целые массивы дисков, один из которых выделяется для хранения журнала. Этот трюк значительно увеличивает производительность при записи, но снижает эффективную пропускную способность на операциях чтения, поскольку один из дисков массива остается незадействованным. Взгляните на результаты тестирования сервера Adaptec DuraStor 6220SS, внутри которого стоит RAID уровня 5 (см. рис 1), приведенные в статье "JOURNALING ON RAID" (http://linuxgazette.net/102/piszcz.html). На данной аппаратной конфигурации ext3fs с одним потоком данных читает чуть ли не в два раза медленнее! На 16 потоках разрыв немного сглаживается, но все равно остается довольно значительным. Вывод: если операции чтения доминируют над описаниями записи, то на серверах ext2fs рулит однозначно, тем более что риск потери данных в этом случае равен нулю, ведь запись на диск не производится!

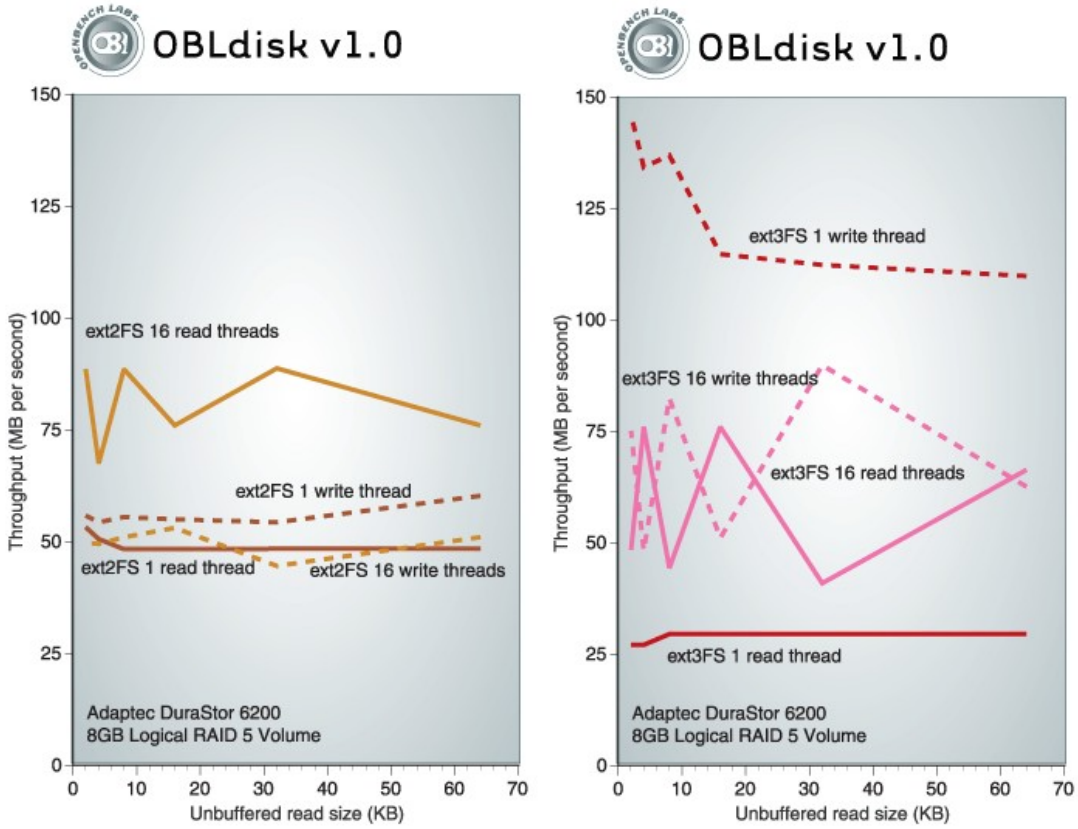

Рисунок 1 производительность сервера Adaptec DuraStor 6220SS на операциях записи/чтения под различными файловыми системами (по данным http://www.open-mag.com/features/Vol_18/filesystems/filesystems.htm)

Кстати, о записи. С записью дела обстоят намного хуже. Исходя из самых общих рассуждений, журналирование съедает время и ощутимо снижает производительность, что подтверждается всеми независимыми тестерами. Запись на ext3fs отстает от ext2fs приблизительно на 50%, а операции удаления большого количества объектов (файлов, директорий) на ext3fs тормозят в десятки раз! На основании чего делается вполне очевидный вывод: etx3fs — довольно медленная файловая система, оправдывающая свое существование только повышенным уровнем надежности (см. рис. 2).

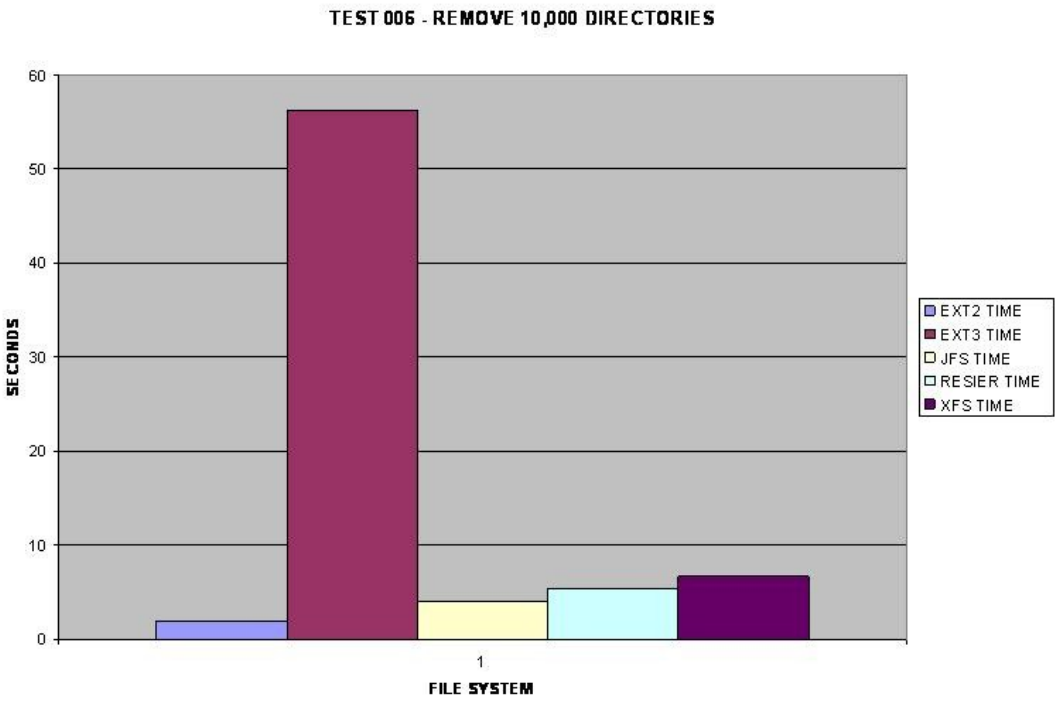

Рисунок 2 время удаления 10.000 каталогов под различными файловыми системами (по данным http://linuxgazette.net/102/piszcz.html)

На самом деле, это утверждение неверно. Запись в журнал может происходить одновременно с обновлением данных/метаданных, достаточно расположить их на различных жестких дисках. Чисто интуитивно это должно вплотную приблизить ext3fs к ext2fs. Любой здравомыслящий человек подтвердит, что в таких условиях ext3fs будет не медленнее, но ему и в голову не придет, что она может оказаться… быстрее, причем значительно быстрее! Вернемся к рис. 1. Сервер Adaptec DuraStor 6220SS пишет на ext3fs с одним потоком в три раза быстрее, чем на ext2fs! На 16 потоках разрыв естественно сокращается, но ext3fs по-прежнему остается впереди. Выходит, что с точки зрения производительности, на серверах и мощных рабочих станциях, ориентированных на запись выгоднее держать ext3fs, а read-only тома всегда размечать под ext2fs? Довольно неожиданных для некоторых администраторов ход. Да может этот Adaptec DuraStor 6220SS специально "Заточен" под ext3fs, а результаты эксперимента фальсифицированы?

Хорошо. Давайте обратимся к базам данных. Возьмем, например, Oracle и посмотрим, на какие файловые системы его рекомендуется ставить. На сайте компании (www.oracle.com/technology/oramag/webcolumns/2002/techarticles/scalzo_linux02.htm) приводятся крайне интересные результаты, которым можно верить, поскольку заниматься пропагандой ext3fs Oracl'у не резон. Мы видим (см. рис. 3, 4), что на всех операциях, которые только можно выполнить над базой, ext3fs обеспечивает вдове большую производительность.

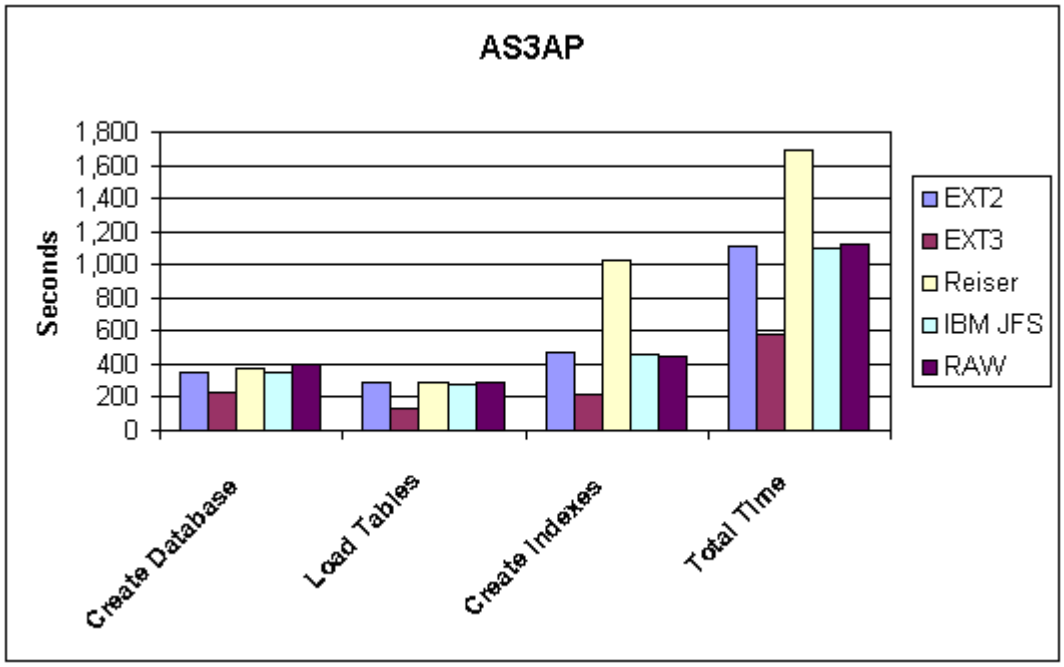

Рисунок 3 результаты теста AS3AP, имитирующего работу с базой данных (по данным специалистов компании Oracle)

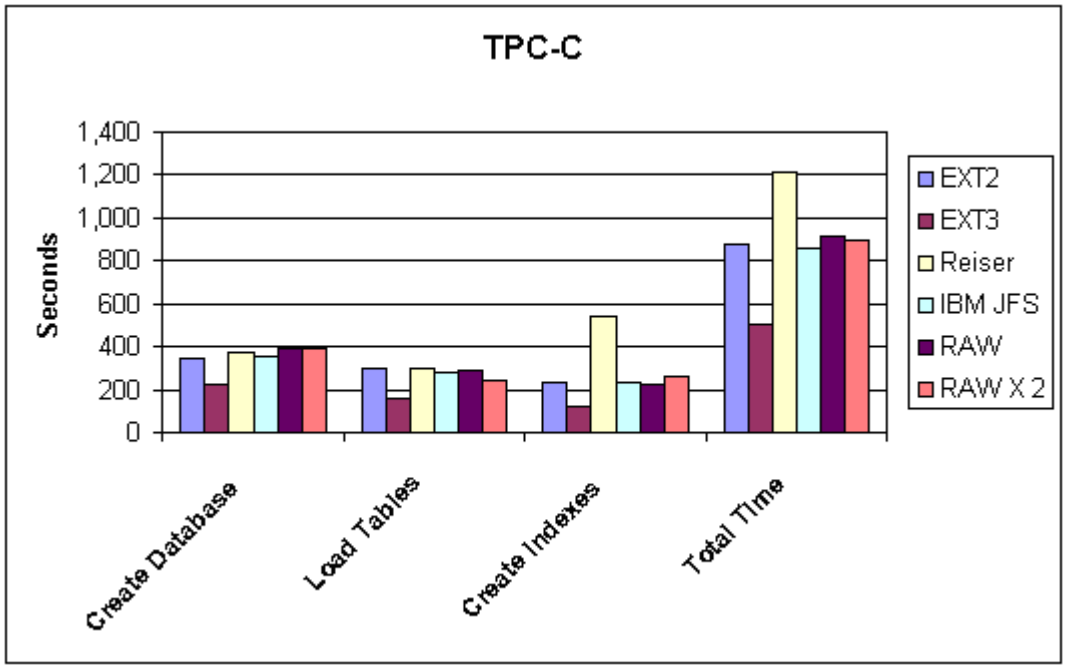

Рисунок 4 результаты теста TPC-C, имитирующего работу с базой данных (по данным специалистов компании Oracle)

Как же такое может быть?! Неужели наличие журнала увеличивает быстродействие? На самом деле, журнал тут совсем не при чем и быстродействие он только съедает. Просто в ext3fs слегка доработан механизм кэширования и внесен ряд других изменений, о которых умалчивает документация, но зато результат на лицо.

Аналогичным образом обстоят дела и с MySQL, однако, мне не удалось найти "официальных" результатов тестов, а протестировать базу данных в домашних условиях довольно затруднительно, так что не обессудьте или найдите себе другого мыщъх'а порасторопнее и побогаче.

>>> врезка раз — фрагмент, два — фрагмент

Сравнивать производительность файловых систем можно только при идентичных условиях, в частности, одинаковом уровне фрагментации. Коллектив японского агентства IPA (Information-Technology Promotion Agency) выпустил специальную утилиту davtools, визуализирующую состояние диска так же как это делал древний Norton Speed Disk, бесплатную версию которой вместе с исходными тестами можно скачать с http://davtools.sourceforge.net.

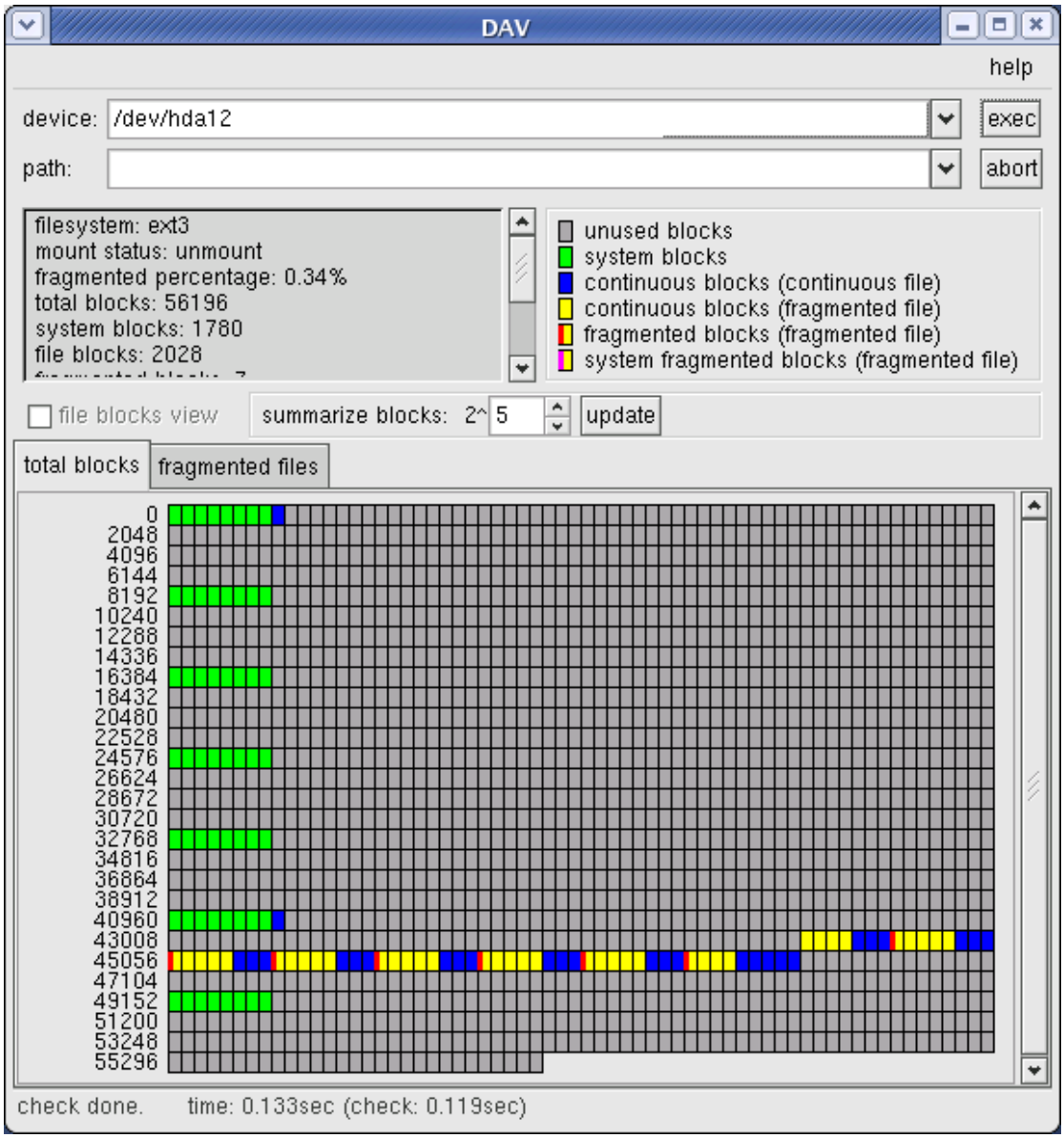

Рисунок 5 утилита davtools, визуализирующая фрагментацию всего раздела целиком

Выяснилось, что ext2fs/ext3fs разделы довольно сильно фрагментируются с течением времени (см. рис. 5), а некоторые очень сильно (см. рис. 6), что опровергает тезис о совершенстве ext2fs/ext3fs и ее не подвластности фрагментации. Фрагментации подвластны все (исключая естественно те системы, что поддерживают фоновую дефрагментацию, как это сделано, например, в UFS).

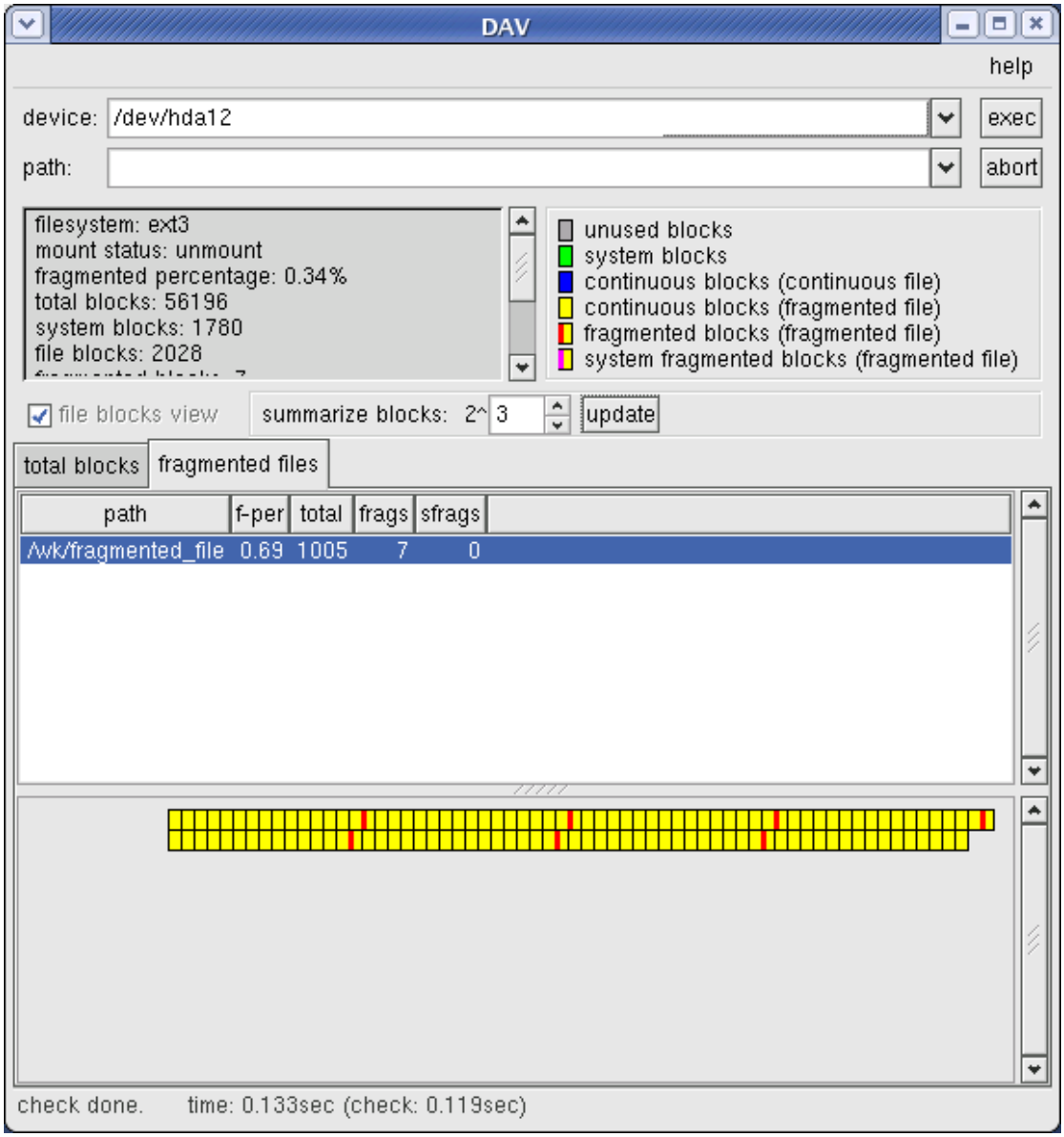

Рисунок 6 утилита davtools, визуализирующая фрагментацию выбранного списка файлов

Некоторые "специалисты" утверждают, что, дескать, это в отличии от MS-DOS и Windows, в LINUX'e фрагментация влияет на производительность намного более сложным образом. Допустим, два файла читаются одновременно. В отсутствии фрагментации головке придется совершать попеременные броски, мотаясь между двумя файлами, что будет нехорошо. Если же разбить файлы на блоки, чередующиеся друг за другом, движения головки примут характер прямолинейной последовательности и несмотря на сильную фрагментацию скорость чтения значительно возрастет.

Естественно, фрагментация бывает разной, однако, наивно думать, что оптимальная "раскладка" образуется естественным путем. В условиях "дикой природы" система раскидывает файлы по всему оперативному периметру и головке приходится совершать очень большие поползновения, чтобы собрать их воедино. Ни о каком последовательном чтении не приходится и говорить! А дефрагментаторов под ext2fs/ext3fs нет. Во всяком случае хороших, которые можно было бы рекомендовать.

В этом смысле, ext2fs чуть-чуть более предпочтительна, чем ext3fs, поскольку журнал последней зачастую оказывается сильно фрагментирован, что (учитывая интенсивность его использования) приводит к значительным тормозам, если, конечно, журнал не размещен на отдельном разделе.

заключение

Выбор оптимальной файловой системы — очень сложна и неочевидная задача. Теория не всегда соответствует практике и приходится быть готовым ко всяческим неожиданностям. Всегда прислушивайтесь к советам производителей оборудования и создателей программ. Как правильно, все необходимые тесты они уже провели или даже заоптимизировали свой продукт под определенную файловую систему с конкретными настройками. Универсальных советов, пригодных для всех, здесь, увы, дать невозможно, поэтому мы ограничимся только самыми общими рекомендациями.

На домашних компьютерах, оборудованных UPS'ом, лучшим выбором будет ext2fs, устанавливаемая большинством дистрибутивов по умолчанию. Если же "упсы" нету, а отключения электричества (зависания, перезагрузки) — обычное дело, ставьте ext3fs и выбирайте максимальный уровень журналирования. Для поддержания производительности в "тонусе" размещайте файл журнала на отдельном жестком диске, подключенном к своему IDE-каналу (впрочем, это не обязательно, современные IDE-устройства нормально делят одну шину друг с другом, если, конечно, не вздумают конфликтовать).

На серверах и рабочих станциях, снабженных RAID-массивами, ставьте ext2fs, в том случае если они ориентированы на чтение, и ext3fs — если на запись. Наибольший выигрыш при работе с ext3fs достигается при работе с базами данных, однако, тут все зависит от рода запросов и типа самой базы. Достоверный ответ может дать только эксперимент.