Введение в файловую систему Linux EXT4

Автор: Дэвид Бот

В предыдущих статьях о файловых системах Linux я написал введение в файловые системы Linux и о некоторых концепциях более высокого уровня, таких как все является файлом. Я хочу более подробно остановиться на специфике файловых систем EXT, но сначала давайте ответим на вопрос: «Что такое файловая система?» Файловая система - это все следующее:

- Хранение данных: основная функция любой файловой системы - быть структурированным местом для хранения и извлечения данных.
- Пространство имен: методология именования и организации, которая предоставляет правила именования и структурирования данных.
- Модель безопасности: схема определения прав доступа.
- API: системные вызовы функций для управления объектами файловой системы, такими как каталоги и файлы.

Реализация: программное обеспечение для реализации вышеуказанного.

его статья концентрируется на первом элементе в списке и исследует структуры метаданных, которые обеспечивают логическую основу для хранения данных в файловой системе EXT.

История файловой системы EXT

Хотя файловая система EXT написана для Linux, ее корни уходят в операционную систему Minix, а файловая система Minix, которая предшествовала Linux примерно на пять лет, была впервые выпущена в 1987 году. Понимание файловой системы EXT4 намного проще, если мы посмотрим на историю и технические аспекты. эволюция семейства файловых систем EXT от его корней Minix.

Minix

При написании исходного ядра Linux Линусу Торвальдсу нужна была файловая система, но тогда он не хотел ее писать. Поэтому он просто включил файловую систему Minix, написанную Эндрю С. Таненбаумом и являющуюся частью операционной системы Таненбаума Minix. Minix была Unix-подобной операционной системой, написанной для образовательных целей. Его код был свободно доступен и имел соответствующую лицензию, чтобы Торвальдс мог включить его в свою первую версию Linux.

Minix имеет следующие структуры, большинство из которых находится в разделе, где создается файловая система:

Загрузочный сектор в первом секторе жесткого диска, на котором он установлен. Загрузочный блок включает очень маленькую загрузочную запись и таблицу разделов.

Первый блок в каждом разделе - это суперблок, который содержит метаданные, которые определяют другие структуры файловой системы и находят их на физическом диске, назначенном разделу.

Блок битовой карты inode, который определяет, какие inode используются, а какие свободны.

Inodes, у которых есть собственное пространство на диске. Каждый индексный дескриптор содержит информацию об одном файле, включая расположение блоков данных, то есть зоны, принадлежащие файлу.

Растровое изображение зоны для отслеживания используемых и свободных зон данных. Зона данных, в которой фактически хранятся данные.

Для обоих типов растровых изображений один бит представляет одну конкретную зону данных или один конкретный индексный дескриптор. Если бит равен нулю, зона или индексный дескриптор свободны и доступны для использования, но, если бит равен единице, зона данных или индексный дескриптор уже используются.

Что такое индексный дескриптор? Сокращенно от index-node, inode - это 256-байтовый блок на диске, в котором хранятся данные о файле. Это включает размер файла; идентификаторы пользователей файла и владельцев группы; файловый режим (т.е. права доступа); и три отметки времени, указывающие время и дату: последний доступ к файлу, последнее изменение и данные в индексном узле были в последний раз изменены.

Inode также содержит данные, указывающие на расположение данных файла на жестком диске. В файловых системах Minix и EXT1-3 это список зон или блоков данных. Inodes файловой системы Minix поддерживает девять блоков данных, семь прямых и два косвенных. Если вы хотите узнать больше, в Википедии есть отличный PDF-файл с подробным описанием структуры файловой системы Minix и кратким обзором структуры указателя inode.

EXT

Исходная файловая система EXT (расширенная) была написана Реми Кардом и выпущена вместе с Linux в 1992 году, чтобы преодолеть некоторые ограничения размера файловой системы Minix. Основные структурные изменения коснулись метаданных файловой системы, которая была основана на файловой системе Unix (UFS), также известной как Berkeley Fast File System (FFS). Я нашел очень мало опубликованной информации о файловой системе

EXT, которую можно проверить, по-видимому, потому, что у нее были серьезные проблемы, и она была быстро заменена файловой системой EXT2.

EXT2

Файловая система EXT2 была вполне успешной. Он использовался в дистрибутивах Linux в течение многих лет, и это была первая файловая система, с которой я столкнулся, когда начал использовать Red Hat Linux 5.0 примерно в 1997 году. Файловая система EXT2 имеет по существу те же структуры метаданных, что и файловая система EXT, однако EXT2 более продвинута. - с учетом того, что между структурами метаданных остается много места на диске для будущего использования.

Как и Minix, EXT2 имеет загрузочный сектор в первом секторе жесткого диска, на котором он установлен, который включает очень маленькую загрузочную запись и таблицу разделов. Затем после загрузочного сектора есть зарезервированное пространство, которое охватывает пространство между загрузочной записью и первым разделом на жестком диске, который обычно находится на границе следующего цилиндра. GRUB2 - и, возможно, GRUB1 - использует это пространство для части своего загрузочного кода.

Пространство в каждом разделе EXT2 разделено на группы цилиндров, что позволяет более детально управлять пространством данных. По моему опыту, размер группы обычно составляет около 8 МБ. На рисунке 1 ниже показана основная структура группы цилиндров. Единицей размещения данных в цилиндре является блок, который обычно имеет размер 4 КБ.

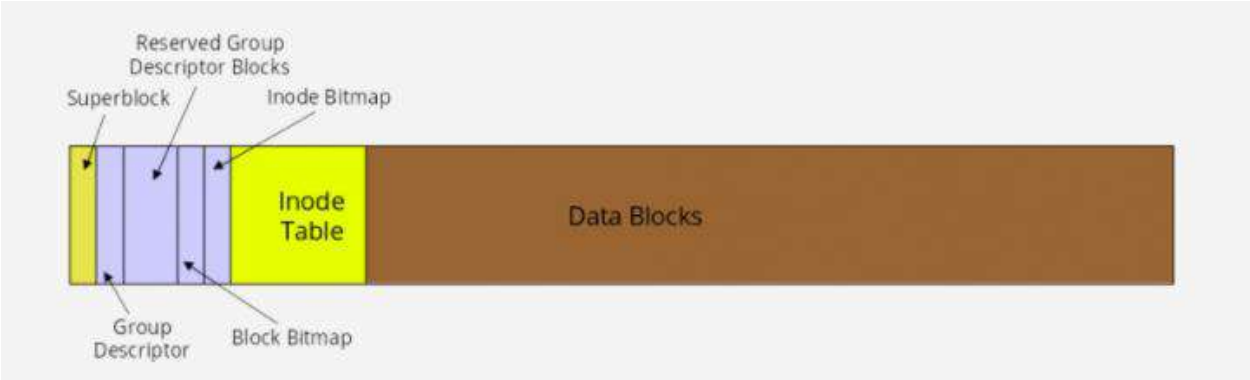

Рисунок 1: Структура группы цилиндров в файловых системах EXT

Первый блок в группе цилиндров - это суперблок, который содержит метаданные, которые определяют другие структуры файловой системы и находят их на физическом диске. Некоторые дополнительные группы в разделе будут иметь резервные суперблоки, но не все. Поврежденный суперблок можно заменить с помощью дисковой утилиты, такой как dd, для копирования содержимого резервного суперблока в первичный суперблок. Это случается не часто, но однажды, много лет назад, у меня был поврежден суперблок, и я смог восстановить его содержимое, используя один из резервных суперблоков. К счастью, я был предвиден и использовал команду dumpe2fs, чтобы выгрузить дескрипторную информацию о разделах моей системы.

Ниже приведен частичный вывод команды dumpe2fs. Он показывает метаданные, содержащиеся в суперблоке, а также данные о каждой из первых двух групп цилиндров в файловой системе.

```
# dumpe2fs /dev/sda1
Filesystem volume name:   boot
Last mounted on:          /boot
Filesystem UUID:          79fc5ed8-5bbc-4dfe-8359-b7b36be6eed3
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent 64bit
flex_bg sparse_super large_file huge_file dir nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              122160
Block count:              488192
Reserved block count:     24409
Free blocks:              376512
Free inodes:              121690
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      238
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8144
Inode blocks per group:   509
Flex block group size:    16
Filesystem created:       Tue Feb  7 09:33:34 2017
Last mount time:          Sat Apr 29 21:42:01 2017
Last write time:          Sat Apr 29 21:42:01 2017
Mount count:              25
Maximum mount count:      -1
Last checked:             Tue Feb  7 09:33:34 2017
Check interval:           0 (<none>)
Lifetime writes:          594 MB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      c780bac9-d4bf-4f35-b695-0fe35e8d2d60
Journal backup:           inode blocks
```

```
Journal features:         journal_64bit
Journal size:             32M
Journal length:           8192
Journal sequence:         0x00000213
Journal start:            0


Group 0: (Blocks 0-32767)
 Primary superblock at 0, Group descriptors at 1-1
 Reserved GDT blocks at 2-239
 Block bitmap at 240 (+240)
 Inode bitmap at 255 (+255)
 Inode table at 270-778 (+270)
 24839 free blocks, 7676 free inodes, 16 directories
 Free blocks: 7929-32767
 Free inodes: 440, 470-8144
Group 1: (Blocks 32768-65535)
 Backup superblock at 32768, Group descriptors at 32769-32769
 Reserved GDT blocks at 32770-33007
 Block bitmap at 241 (bg #0 + 241)
 Inode bitmap at 256 (bg #0 + 256)
 Inode table at 779-1287 (bg #0 + 779)
 8668 free blocks, 8142 free inodes, 2 directories
 Free blocks: 33008-33283, 33332-33791, 33974-33975, 34023-34092, 34094-34104, 34526-34687, 34706-
34723, 34817-35374, 35421-35844, 35935-36355, 36357-36863, 38912-39935, 39940-40570, 42620-
42623, 42655, 42674-42687, 42721-42751, 42798-42815, 42847, 42875-42879, 42918-42943, 42975, 43000-
43007, 43519, 43559-44031, 44042-44543, 44545-45055, 45116-45567, 45601-45631, 45658-45663, 45689-
45695, 45736-45759, 45802-45823, 45857-45887, 45919, 45950-45951, 45972-45983, 46014-46015, 46057-
46079, 46112-46591, 46921-47103, 49152-49395, 50027-50355, 52237-52255, 52285-52287, 52323-
52351, 52383, 52450-52479, 52518-52543, 52584-52607, 52652-52671, 52734-52735, 52743-53247
 Free inodes: 8147-16288
Group 2: (Blocks 65536-98303)
 Block bitmap at 242 (bg #0 + 242)
 Inode bitmap at 257 (bg #0 + 257)
 Inode table at 1288-1796 (bg #0 + 1288)
 6326 free blocks, 8144 free inodes, 0 directories
 Free blocks: 67042-67583, 72201-72994, 80185-80349, 81191-81919, 90112-94207
 Free inodes: 16289-24432
Group 3: (Blocks 98304-131071)

<snip>
```

Каждая группа цилиндров имеет свой собственный битовый массив индексных дескрипторов, который используется для определения того, какие индексы используются, а какие свободны в этой группе. Inodes имеют собственное пространство в каждой группе. Каждый индексный дескриптор содержит информацию об одном файле, включая расположение блоков данных, принадлежащих этому файлу. Битовая карта блока отслеживает используемые и свободные блоки данных в файловой системе. Обратите внимание, что в выходных данных, показанных выше, содержится много данных о файловой системе. В очень больших файловых системах данные группы могут занимать сотни страниц. Метаданные группы включают список всех свободных блоков данных в группе.

В файловой системе EXT реализованы стратегии распределения данных, обеспечивающие минимальную фрагментацию файлов. Уменьшение фрагментации улучшило производительность файловой системы. Эти стратегии описаны ниже, в разделе EXT4.

Самая большая проблема с файловой системой EXT2, с которой я сталкивался в некоторых случаях, заключалась в том, что восстановление после сбоя могло занять много часов, потому что программе fsck (проверка файловой системы) потребовалось очень много времени, чтобы найти и исправить любые несоответствия в файловой системе. . Однажды на одном из моих компьютеров потребовалось более 28 часов, чтобы полностью восстановить диск при перезагрузке после сбоя, и это было тогда, когда размер дисков был меньше сотен мегабайт.

EXT3

Файловая система EXT3 имела единственную цель - преодолеть огромное количество времени, которое требовалось программе fsck для полного восстановления структуры диска, поврежденной в результате неправильного завершения работы, произошедшего во время операции обновления файла. Единственным дополнением к файловой системе EXT был журнал, который заранее записывает изменения, которые будут внесены в файловую систему. В остальном структура диска такая же, как и в EXT2.

Вместо записи данных в области данных диска напрямую, как в предыдущих версиях, журнал в EXT3 записывает данные файла вместе с его метаданными в указанную область на диске. Как только данные надежно помещены на жесткий диск, их можно объединить или добавить к целевому файлу с почти нулевой вероятностью потери данных. Поскольку эти данные фиксируются в области данных на диске, журнал обновляется, чтобы файловая система оставалась в согласованном состоянии в случае сбоя системы до того, как все данные в журнале будут зафиксированы. При следующей загрузке файловая система будет проверена на наличие несоответствий, и данные, оставшиеся в журнале, будут переданы в области данных на диске для завершения обновления целевого файла.

Ведение журнала снижает производительность записи данных, однако для журнала доступны три варианта, которые позволяют пользователю выбирать между производительностью и целостностью и безопасностью данных. Лично я предпочитаю безопасность, потому что мои среды не требуют интенсивной записи на диск.

Функция ведения журнала сокращает время, необходимое для проверки жесткого диска на наличие несоответствий после сбоя, с часов (или даже дней) до простых минут, самое большее. За эти годы у меня было много проблем, из-за которых мои системы ломались. Подробности можно было бы заполнить в другой статье, но достаточно сказать, что большинство из них были нанесены самим, например, выдернули вилку из розетки. К счастью, журнальные файловые системы EXT сократили время восстановления при

загрузке до двух или трех минут. Кроме того, у меня никогда не было проблем с потерей данных с тех пор, как я начал использовать EXT3 с журналированием.

Функцию ведения журнала EXT3 можно отключить, и тогда она будет работать как файловая система EXT2. Сам журнал все еще существует, он пуст и не используется. Просто перемонтируйте раздел с помощью команды mount, используя параметр type для указания EXT2. Вы можете сделать это из командной строки, в зависимости от того, с какой файловой системой вы работаете, но вы можете изменить спецификатор типа в файле / etc / fstab и затем перезагрузиться. Я настоятельно не рекомендую монтировать файловую систему EXT3 как EXT2 из-за дополнительной возможности потери данных и увеличения времени восстановления.

Существующую файловую систему EXT2 можно обновить до EXT3, добавив журнал, используя следующую команду.

```
tune2fs -j /dev/sda1
```

Где / dev / sda1 - идентификатор диска и раздела. Обязательно измените спецификатор типа файла в / etc / fstab и перемонтируйте раздел или перезагрузите систему, чтобы изменения вступили в силу.

EXT4

Файловая система EXT4 в первую очередь улучшает производительность, надежность и емкость. Для повышения надежности были добавлены метаданные и контрольные суммы журнала. Чтобы соответствовать различным критически важным требованиям, временные метки файловой системы были улучшены с добавлением интервалов до наносекунд. Добавление двух старших битов в поле отметки времени откладывает проблему 2038 года до 2446 года - по крайней мере, для файловых систем EXT4.

В EXT4 распределение данных было изменено с фиксированных блоков на экстенты. Экстент описывается его начальным и конечным местом на жестком диске. Это позволяет описывать очень длинные, физически смежные файлы в одной записи указателя inode, что может значительно уменьшить количество указателей, необходимых для описания расположения всех данных в больших файлах. В EXT4 реализованы другие стратегии распределения для дальнейшего уменьшения фрагментации.

EXT4 уменьшает фрагментацию, разбрасывая вновь созданные файлы по диску, чтобы они не собирались в одном месте в начале диска, как это делали многие ранние файловые системы ПК. Алгоритмы распределения файлов пытаются распределить файлы как можно более равномерно по

группам цилиндров и, когда необходима фрагментация, сохранять прерывистые экстенты файла как можно ближе к другим в том же файле, чтобы максимально уменьшить задержку поиска головы и вращения. насколько возможно. Дополнительные стратегии используются для предварительного выделения дополнительного дискового пространства при создании нового файла или при расширении существующего файла. Это помогает гарантировать, что расширение файла автоматически не приведет к его фрагментации. Новые файлы никогда не размещаются сразу после существующих файлов, что также предотвращает фрагментацию существующих файлов.

Помимо фактического расположения данных на диске, EXT4 использует функциональные стратегии, такие как отложенное выделение, чтобы файловая система могла собирать все данные, записываемые на диск, перед тем, как выделить для него место. Это может повысить вероятность того, что пространство данных будет непрерывным.

Старые файловые системы EXT, такие как EXT2 и EXT3, можно смонтировать как EXT4, чтобы добиться небольшого увеличения производительности. К сожалению, для этого необходимо отключить некоторые важные новые функции EXT4, поэтому я не рекомендую этого делать.

EXT4 является файловой системой по умолчанию для Fedora, начиная с Fedora 14. Файловая система EXT3 может быть обновлена до EXT4 с помощью процедуры, описанной в документации Fedora, однако ее производительность по-прежнему будет ухудшаться из-за остаточных структур метаданных EXT3. Лучший метод обновления EXT4 с EXT3 - это создать резервную копию всех данных в разделе целевой файловой системы, использовать команду mkfs для записи пустой файловой системы EXT4 в раздел, а затем восстановить все данные из резервной копии.

Inode

Inode, описанный ранее, является ключевым компонентом метаданных в файловых системах EXT. На рисунке 2 показана взаимосвязь между индексным дескриптором и данными, хранящимися на жестком диске. Эта диаграмма представляет собой каталог и индексный дескриптор для одного файла, который в этом случае может быть сильно фрагментирован. Файловые системы EXT активно работают над уменьшением фрагментации, поэтому очень маловероятно, что вы когда-нибудь увидите файл с таким количеством косвенных блоков данных или экстентов. Фактически, как вы увидите ниже, фрагментация файловых систем EXT чрезвычайно мала, поэтому большинство inodes будут использовать только один или два прямых

указателя данных и ни один из косвенных указателей.

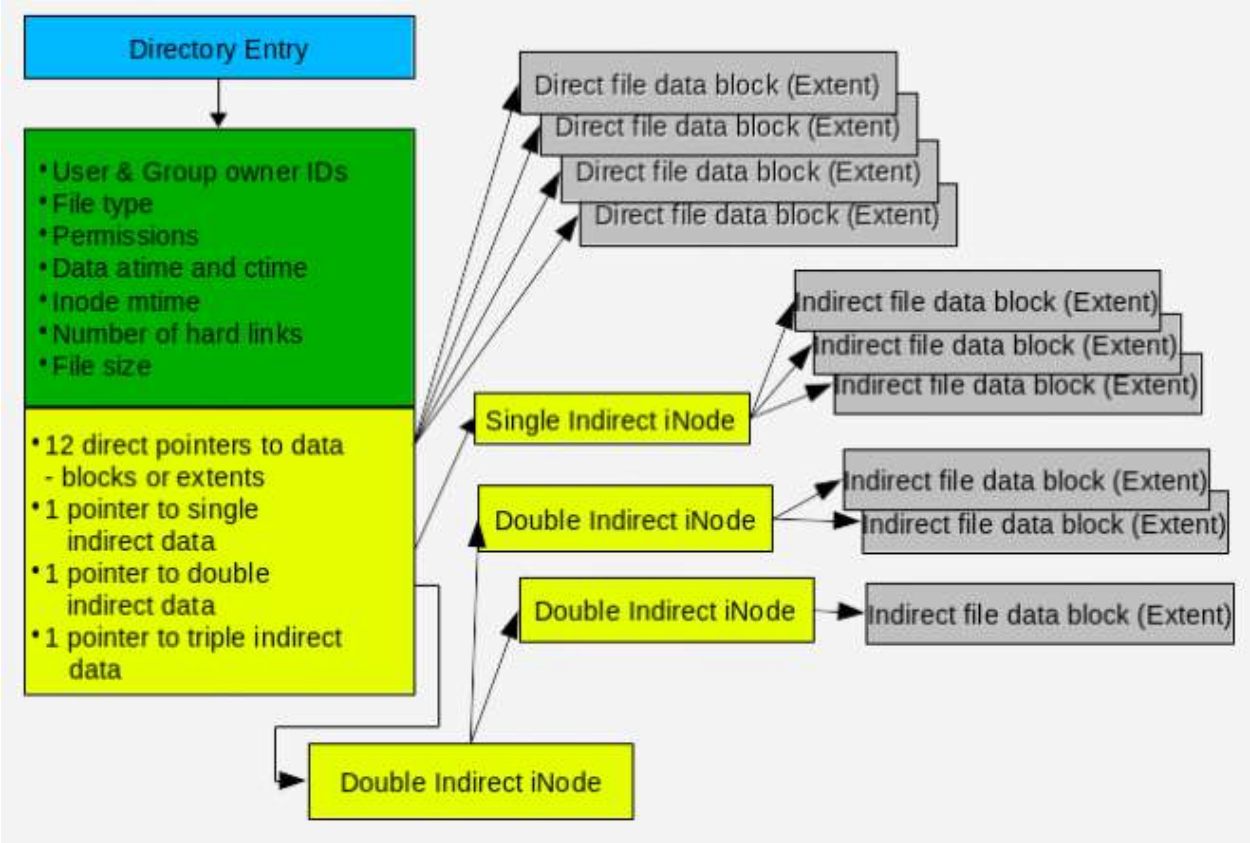

Рисунок 2: inode хранит информацию о каждом файле и позволяет файловой системе EXT находить все данные, принадлежащие ему.

Inode не содержит имени файла. Доступ к файлу осуществляется через запись каталога, которая сама по себе является именем файла и содержит указатель на индексный дескриптор. Значением этого указателя является номер inode. Каждый индексный дескриптор в файловой системе имеет уникальный идентификационный номер, но индексные дескрипторы в других файловых системах на том же компьютере (и даже на том же жестком диске) могут иметь тот же номер. Это имеет значение для ссылок, и это обсуждение выходит за рамки данной статьи.

Inode содержит метаданные о файле, включая его тип и разрешения, а также его размер. Inode также содержит пространство для 15 указателей, которые описывают расположение и длину блоков данных или экстентов в части данных группы цилиндров. Двенадцать указателей обеспечивают прямой доступ к экстентам данных и должны быть достаточными для обработки большинства файлов. Однако для файлов со значительной фрагментацией необходимо иметь некоторые дополнительные возможности в виде косвенных узлов. Технически это не inodes, поэтому для удобства я использую термин «узел».

Косвенный узел - это обычный блок данных в файловой системе, который используется только для описания данных, а не для хранения метаданных, поэтому может поддерживаться более 15 записей. Например, размер блока 4K может поддерживать 512 4-байтовых непрямых узлов, что позволяет использовать 12 (прямые) + 512 (косвенные) = 524 экстентов для одного файла. Также поддерживается двойная и тройная косвенная поддержка узлов, но большинство из нас вряд ли столкнутся с файлами, требующими такого количества экстентов.

Фрагментация данных

Для многих старых файловых систем ПК, таких как FAT (и все ее варианты) и NTFS, фрагментация была серьезной проблемой, приводящей к снижению производительности диска. Дефрагментация сама по себе превратилась в отрасль с различными брендами программного обеспечения для дефрагментации, которое варьировалось от очень эффективного до незначительного.

Расширенные файловые системы Linux используют стратегии распределения данных, которые помогают минимизировать фрагментацию файлов на жестком диске и уменьшить последствия фрагментации, когда она действительно происходит. Вы можете использовать команду fsck в файловых системах EXT, чтобы проверить общую фрагментацию файловой системы. В следующем примере проверяется домашний каталог моей основной рабочей станции, который был фрагментирован всего на 1,5%. Обязательно используйте параметр -n, потому что он не позволяет fsck выполнять какие-либо действия с проверяемой файловой системой.

```
fsck -fn /dev/mapper/vg_01-home
```

Однажды я провел несколько теоретических расчетов, чтобы определить, может ли дефрагментация диска привести к какому-либо заметному повышению производительности. Хотя я сделал некоторые предположения, данные о производительности диска, которые я использовал, были взяты с нового жесткого диска Western Digital емкостью 300 ГБ с временем поиска 2,0 мс. Количество файлов в этом примере было фактическим количеством, существовавшим в файловой системе в день, когда я производил расчет. Я действительно предполагал, что каждый день будет обрабатываться довольно большое количество фрагментированных файлов (20%).

Total files	271,794
% fragmentation	5.00%
Discontinuities	13,590
% fragmented files touched per day	20% (assume)
Number of additional seeks	2,718
Average seek time	10.90 ms
Total additional seek time per day	29.63 sec
	0.49 min
Track-to-track seek time	2.00 ms
Total additional seek time per day	5.44 sec
	0.091 min

Таблица 1: Теоретическое влияние фрагментации на производительность диска

Я выполнил два расчета общего дополнительного времени поиска в день, один на основе времени поиска от трека к треку, что является более вероятным сценарием для большинства файлов из-за стратегии распределения файлов EXT, и один для среднего времени поиска , что, как я предполагал, будет справедливым худшим сценарием.

Как видно из таблицы 1, влияние фрагментации на современную файловую систему EXT с жестким диском даже скромной производительности будет минимальным и незначительным для подавляющего большинства приложений. Вы можете вставить числа из вашей среды в вашу собственную аналогичную электронную таблицу, чтобы увидеть, что вы можете ожидать от воздействия на производительность. Этот тип расчета, скорее всего, не будет отражать фактическую производительность, но он может дать некоторое представление о фрагментации и ее теоретическом влиянии на систему.

Большинство моих разделов фрагментированы примерно на 1,5% или 1,6%; У меня есть одна, на 3,3% фрагментированная, но это большая файловая система 128 ГБ с менее чем 100 очень большими файлами ISO-образов; Мне приходилось расширять раздел несколько раз за эти годы, так как он был переполнен.

Это не означает, что некоторые среды приложений не требуют большей уверенности в еще меньшей фрагментации. Файловая система EXT может быть настроена с осторожностью опытным администратором, который может настроить параметры для компенсации определенных типов рабочей нагрузки. Это можно сделать при создании файловой системы или позже с помощью команды tune2fs. Результаты каждого изменения настройки должны быть протестированы, тщательно записаны и проанализированы, чтобы гарантировать оптимальную производительность для целевой среды. В худшем случае, когда производительность не может быть улучшена до желаемого уровня, доступны другие типы файловых систем, которые могут быть более подходящими для конкретной рабочей нагрузки. И помните, что обычно в одной хост-системе смешиваются типы файловых систем, чтобы соответствовать нагрузке на каждую файловую систему.

Из-за низкой степени фрагментации в большинстве файловых систем EXT дефрагментация не требуется. В любом случае безопасного инструмента дефрагментации для файловых систем EXT не существует. Есть несколько инструментов, которые позволяют вам проверять фрагментацию отдельного файла или фрагментацию оставшегося свободного места в файловой системе. Есть один инструмент, e4defrag, который дефрагментирует файл, каталог или файловую систему настолько, насколько позволяет оставшееся свободное пространство. Как следует из названия, он работает только с файлами в файловой системе EXT4 и имеет некоторые ограничения.

Если возникает необходимость выполнить полную дефрагментацию файловой системы EXT, есть только один метод, который будет работать надежно. Вы должны переместить все файлы из файловой системы для дефрагментации, убедившись, что они будут удалены после безопасного копирования в другое место. Если возможно, вы могли бы увеличить размер файловой системы, чтобы уменьшить фрагментацию в будущем. Затем скопируйте файлы обратно в целевую файловую систему. Даже это не гарантирует, что все файлы будут полностью дефрагментированы.

Выводы

Файловые системы EXT используются по умолчанию для многих дистрибутивов Linux более 20 лет. Они предлагают стабильность, высокую емкость, надежность и производительность при минимальном обслуживании.

Я пробовал другие файловые системы, но всегда возвращаюсь к EXT. Все места, где я работал с Linux, использовали файловые системы EXT и находили их подходящими для всех основных нагрузок, используемых на них. Без сомнения, файловая система EXT4 должна использоваться для большинства систем Linux, если нет веских причин для использования другой файловой системы.