**РУКОВОДСТВО АДМИНИСТРАТОРА**

**RT.KEYVALUE**

2024

ОГЛАВЛЕНИЕ

[1 ОПИСАНИЕ RT.KEYVALUE 5](#_Toc86141357)

[2 Настройка RT.KeyValue 6](#_Toc86141358)

[2.1 Настройка файла cassandra.yaml 6](#_Toc86141359)

[2.2 Настройка файла cassandra-rackdc.properties 11](#_Toc86141360)

[2.2.1 GossipingPropertyFileSnitch 12](#_Toc86141361)

[2.2.2 AWS EC2 snitch 12](#_Toc86141362)

[2.2.3 Either snitch 12](#_Toc86141363)

[2.3 Настройка файла cassandra-env.sh 12](#_Toc86141364)

[2.4 Настройка файла cassandra-topologies.properties 15](#_Toc86141365)

[2.5 Настройка файла commitlog-archiving.properties 16](#_Toc86141366)

[2.5.1 Параметры 16](#_Toc86141367)

[2.6 Настройка файла logback.xml 17](#_Toc86141368)

[2.6.1 Параметры 17](#_Toc86141369)

[2.7 Настройка jvm-\* файлов 18](#_Toc86141370)

[3 Операции RT.KeyValue 19](#_Toc86141371)

[3.1 Аппаратное обеспечение 19](#_Toc86141372)

[3.1.1 Центральный процессор 19](#_Toc86141373)

[3.1.2 Объем оперативной памяти 19](#_Toc86141374)

[3.1.3 Диски 20](#_Toc86141375)

[3.1.4 Общие варианты облачных вычислений (Common Cloud Choices) 21](#_Toc86141376)

[3.2 Безопасность RT.KeyValue 21](#_Toc86141377)

[3.2.1 TSL / SSL шифрование 22](#_Toc86141378)

[3.2.2 Горячая перезагрузка SSL-сертификата (Hot Reloading) 22](#_Toc86141379)

[3.2.3 Роли 23](#_Toc86141380)

[3.2.4 Аутентификация 23](#_Toc86141381)

[3.2.5 Авторизация 25](#_Toc86141382)

[3.2.6 Кеширование 26](#_Toc86141383)

[3.2.7 JMX доступ 27](#_Toc86141384)

[3.3 Изменения топологии 31](#_Toc86141385)

[3.3.1 Начальная загрузка (Bootstrap) 31](#_Toc86141386)

[3.3.2 Удаление узлов 33](#_Toc86141387)

[3.3.3 Перемещение узлов 33](#_Toc86141388)

[3.3.4 Замена мертвого узла 33](#_Toc86141389)

[3.3.5 Мониторинг прогресса 34](#_Toc86141390)

[3.3.6 Очистка данных после перемещений диапазона 34](#_Toc86141391)

[3.4 Подсказки (Hints) 34](#_Toc86141392)

[3.4.1 Hinted Handoff 35](#_Toc86141393)

[3.4.2 Настройка подсказок 37](#_Toc86141394)

[3.4.3 Настройка подсказок во время выполнения с помощью nodetool 39](#_Toc86141395)

[3.4.4 Мониторинг доставки подсказок 40](#_Toc86141396)

[3.5 Восстановление 40](#_Toc86141397)

[3.5.1 Инкрементальное и полное восстановление 41](#_Toc86141398)

[3.5.2 Использование и лучшие практики 41](#_Toc86141399)

[3.5.3 Другие параметры 42](#_Toc86141400)

[3.5.4 Пример полного восстановления 42](#_Toc86141401)

[3.6 Восстановление при чтении (Read repair) 45](#_Toc86141402)

[3.6.1 Ожидание монотонного чтения кворума 45](#_Toc86141403)

[3.6.2 Конфигурация монотонного чтения на уровне таблицы 45](#_Toc86141404)

[3.6.3 Пример восстановления при чтении 46](#_Toc86141405)

[3.6.4 Уровень согласованности чтения и Восстановление при чтении 50](#_Toc86141406)

[3.6.5 Диагностические события для восстановления при чтении. 51](#_Toc86141407)

[3.6.6 Background восстановления при чтении 51](#_Toc86141408)

[3.7 Резервные копии (Backups) 52](#_Toc86141409)

[3.7.1 Типы резервных копий 52](#_Toc86141410)

[3.7.2 Структура каталога данных 53](#_Toc86141411)

[3.7.3 Снапшоты 56](#_Toc86141412)

[3.7.4 Инкрементальные резервные копии 62](#_Toc86141413)

[3.7.5 Восстановление из инкрементальных резервных копий и моментальных снапшотов 64](#_Toc86141414)

[3.8 Сжатие (compression) 65](#_Toc86141415)

[3.9 Уплотнение (Compaction) 66](#_Toc86141416)

[3.9.1 Стратегии 66](#_Toc86141417)

[3.9.2 Виды уплотнения 67](#_Toc86141418)

[3.9.3 Команды Compaction nodetool 68](#_Toc86141419)

[3.10 Мониторинг 68](#_Toc86141420)

[3.10.1 Типы метрик 68](#_Toc86141421)

[3.10.2 Табличные метрики 69](#_Toc86141422)

[3.10.3 Метрик репортер 69](#_Toc86141423)

[3.11 Массовая загрузка (Bulk loading) 70](#_Toc86141424)

[3.11.1 Инструменты для массовой загрузки 70](#_Toc86141425)

[3.11.2 Использование sstableloader 70](#_Toc86141426)

[3.11.3 Использование nodetool import 71](#_Toc86141427)

[3.11.4 Массовая загрузка внешних данных 71](#_Toc86141428)

[3.12 Change Data Capture 71](#_Toc86141429)

[3.12.1 Обзор 71](#_Toc86141430)

[3.12.2 Конфигурация 72](#_Toc86141431)

[3.12.3 Чтение сегментов CommitLog 73](#_Toc86141432)

[3.12.4 Предупреждение 73](#_Toc86141433)

[3.13 Фильтры Блума (Bloom filters) 73](#_Toc86141434)

[3.13.1 Изменение 74](#_Toc86141435)

[Приложение 1. Полное описание файла local\_system\_data\_file\_directory 75](#_Toc86141436)

[Приложение 2. содержимое logback.xml файла по умолчанию 76](#_Toc86141437)

ОПИСАНИЕ RT.KEYVALUE

**RT.KeyValue** – это NoSQL распределенная база данных, обладающая высокой масштабируемостью и доступностью без ущерба для производительности. Линейная масштабируемость и проверенная отказоустойчивость на стандартном оборудовании или облачной инфраструктуре делают её идеальной платформой для критически важных данных.

Данный документ описывет возможности взаимодейтсвия Администратора с настройкой, операциями, утилитами и архитектурой RT.KeyValue.

Настройка RT.KeyValue

В данном разделе описывается, как настроить RT.KeyValue.

Настройка файла cassandra.yaml

cluster\_name - имя кластера. Это в основном используется для предотвращения присоединения машин в одном логическом кластере к другому:

Значение по умолчанию (Default Value): 'Test Cluster'

num\_tokens - определяет количество токенов, случайным образом назначаемых этому узлу в кольце. Чем больше токенов относительно других узлов, тем большую долю данных будет хранить этот узел. Вероятно, вы хотите, чтобы все узлы имели одинаковое количество токенов, если возможности оборудования одинаковы.

Если вы не укажете это значение, RT.KeyValue будет использовать токен по умолчанию, равный 1, для совместимости с устаревшими версиями, и будет использовать initial\_token, как описано ниже.

Указание initial\_token переопределит этот параметр при первоначальном запуске узла, при последующих запусках этот параметр будет применяться, даже если установлен начальный токен.

Значение по умолчанию: 256

allocate\_tokens\_for\_keyspace. Запускает автоматическое выделение токенов num\_tokens для этого узла. Алгоритм распределения пытается выбрать токены таким образом, чтобы оптимизировать реплицированную нагрузку на узлы в центре обработки данных для фактора реплики.

**По умолчанию этот параметр закомментирован.**

Нагрузка, назначенная каждому узлу, будет почти пропорциональна его количеству vnodes.

Поддерживается только с Murmur3Partitioner.

Фактор репликации определяется с помощью стратегии репликации, используемой указанным пространством ключей.

Значение по умолчанию: KEYSPACE.

allocate\_tokens\_for\_local\_replication\_factor. Фактор реплики устанавливается явно, независимо от пространства ключей или центра обработки данных. Это фактор реплики в центре обработки данных, как и NTS.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: 3

initial\_token. initial\_token позволяет указывать токены вручную. Хотя вы можете использовать его с vnodes (num\_tokens> 1, выше) - в этом случае вы должны предоставить список, разделенный запятыми, он в основном используется при добавлении узлов в устаревшие кластеры, для которых не включены vnodes.

**По умолчанию этот параметр закомментирован.**

hinted\_handoff\_enabled. Может быть "true" или "false" для включения глобально.

Значение по умолчанию: true

hinted\_handoff\_disabled\_datacenters.

Если значение hinted\_handoff\_enabled равно true, создается черный список центров обработки данных, которые не будут выполнять подсказки (hinted) на передачу обслуживания.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: true

|  |
| --- |
| # - DC1  # - DC2 |

max\_hint\_window\_in\_ms - определяет максимальное количество времени, в течение которого мертвый хост будет генерировать подсказки.

Значение по умолчанию: 10800000 # 3 часа.

hinted\_handoff\_throttle\_in\_kb. Максимальное дросселирование в килобайтах (KBs) в секунду для каждого потока доставки. Уменьшеньшается пропорционально количеству узлов в кластере. (Если в кластере два узла, каждый поток доставки будет использовать максимальную скорость; если их три, каждый будет уменьшен до половины максимального значения, поскольку мы ожидаем, что два узла будут доставлять подсказки одновременно).

Значение по умолчанию: 1024.

max\_hints\_delivery\_threads - количество потоков, с помощью которых доставляются подсказки.

Значение по умолчанию: 2

hints\_directory. Каталог, в котором RT.KeyValue должна хранить подсказки. Если не установлен, каталог по умолчанию - $ CASSANDRA\_HOME / data / hints.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: / var / lib / cassandra / hints

hints\_flush\_period\_in\_ms. Показывает, как часто подсказки должны сбрасываться из внутренних буферов на диск. **Не** запускает fsync.

Значение по умолчанию: 10000

max\_hints\_file\_size\_in\_mb. Максимальный размер одного файла подсказок в мегабайтах.

Значение по умолчанию: 128

hints\_compression. Сжатие для применения к файлам подсказок. Если этот параметр не указан, файлы подсказок будут записаны без сжатия. Поддерживаются компрессоры LZ4, Snappy и Deflate.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию (сложный вариант):

|  |
| --- |
| # - class\_name: LZ4Compressor  # parameters:  # - |

batchlog\_replay\_throttle\_in\_kb. Максимальное дросселирование в килобайтах в секунду. Это значение будет уменьшено пропорционально количеству узлов в кластере.

Значение по умолчанию: 1024.

authenticator. Бэкэнд аутентификации, реализующий IAuthenticator; используется для идентификации пользователей. Из коробки RT.KeyValue предоставляет org.apache.cassandra.auth. {AllowAllAuthenticator, PasswordAuthenticator}.

* AllowAllAuthenticator не выполняет никаких проверок - установите его, чтобы отключить аутентификацию.
* PasswordAuthenticator полагается на пары имени пользователя и пароля для аутентификации пользователей. Он хранит имена пользователей и хешированные пароли в таблице system\_auth.roles. Увеличьте коэффициент репликации пространства ключей system\_auth, если вы используете этот аутентификатор. При использовании PasswordAuthenticator также необходимо использовать CassandraRoleManager (см. Ниже)

Значение по умолчанию: AllowAllAuthenticator.

authorizer. Бэкэнд авторизации, реализующий IAuthorizer; используется для ограничения доступа / предоставления разрешений. Из коробки RT.KeyValue предоставляет org.apache.cassandra.auth. {AllowAllAuthorizer, CassandraAuthorizer}.

* AllowAllAuthorizer разрешает любое действие любому пользователю - установите его, чтобы отключить авторизацию.
* CassandraAuthorizer хранит разрешения в таблице system\_auth.role\_permissions. Увеличьте коэффициент репликации пространства ключей system\_auth, если вы используете этот авторизатор.

Значение по умолчанию: AllowAllAuthorizer.

role\_manager. Часть серверной части аутентификации и авторизации, реализующая IRoleManager; используется для поддержки грантов и членства между ролями. По умолчанию RT.KeyValue предоставляет org.apache.cassandra.auth.CassandraRoleManager, который хранит информацию о ролях в пространстве ключей system\_auth. Большинство функций IRoleManager требуют аутентифицированного входа в систему, поэтому, если настроенный IAuthenticator фактически не реализует аутентификацию, большая часть этих функций будет недоступна.

* CassandraRoleManager хранит данные о ролях в пространстве ключей system\_auth. Увеличьте коэффициент репликации пространства ключей system\_auth, если вы используете этот менеджер ролей.

Значение по умолчанию: CassandraRoleManager.

network\_authorizer. Бэкэнд сетевой авторизации, реализующий INetworkAuthorizer; используется для ограничения доступа пользователей к определенным контроллерам домена. Из коробки RT.KeyValue предоставляет org.apache.cassandra.auth. {AllowAllNetworkAuthorizer, CassandraNetworkAuthorizer}.

* AllowAllNetworkAuthorizer разрешает доступ к любому дата-центру любому пользователю - отключите авторизацию.
* CassandraNetworkAuthorizer хранит разрешения в таблице system\_auth.network\_permissions. Увеличьте коэффициент репликации пространства ключей system\_auth, если вы используете этот авторизатор.

Значение по умолчанию: AllowAllNetworkAuthorizer.

roles\_validity\_in\_ms. Срок действия кеша ролей (получение предоставленных ролей может быть дорогостоящей операцией в зависимости от диспетчера ролей, CassandraRoleManager является одним из примеров). Предоставленные роли кэшируются для аутентифицированных сеансов в AuthenticatedUser и после указанного здесь периода становятся доступными для (асинхронной) перезагрузки. По умолчанию 2000, 0 - полное отключение кеширования. Будет автоматически отключен для AllowAllAuthenticator.

Значение по умолчанию: 2000.

roles\_update\_interval\_in\_ms. Интервал обновления кеша ролей (если включен). По истечении этого интервала записи кэша становятся доступными для обновления. При следующем доступе планируется асинхронная перезагрузка, и до ее завершения возвращается старое значение. По умолчанию используется то же значение, что и role\_validity\_in\_ms.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: 2000

permissions\_validity\_in\_ms. Срок действия кеша разрешений (получение разрешений может быть дорогостоящей операцией в зависимости от авторизатора, CassandraAuthorizer является одним из примеров). По умолчанию - 2000, 0 - отключить. Будет автоматически отключен для AllowAllAuthorizer.

Значение по умолчанию: 2000

permissions\_update\_interval\_in\_ms. Интервал обновления для кеша разрешений (если включен). По истечении этого интервала записи кэша становятся доступными для обновления. При следующем доступе планируется асинхронная перезагрузка, и до ее завершения возвращается старое значение. Если permissions\_validity\_in\_ms не равно нулю, то это тоже должно быть. По умолчанию используется то же значение, что и permissions\_validity\_in\_ms.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: 2000

credentials\_validity\_in\_ms. Срок действия кеша учетных данных. Этот кеш тесно связан с предоставленной реализацией PasswordAuthenticator IAuthenticator. Если настроена другая реализация IAuthenticator, этот кеш не будет использоваться автоматически, поэтому следующие параметры не будут иметь никакого эффекта. Обратите внимание, что учетные данные кэшируются в зашифрованном виде, поэтому, хотя активация этого кеша может уменьшить количество запросов к базовой таблице, это может не привести к значительному сокращению задержки отдельных попыток аутентификации. По умолчанию 2000, 0 для отключения кэширования учетных данных.

Значение по умолчанию: 2000

credentials\_update\_interval\_in\_ms. Интервал обновления для кеша учетных данных (если включен). По истечении этого интервала записи кэша становятся доступными для обновления. При следующем доступе планируется асинхронная перезагрузка, и до ее завершения возвращается старое значение. По умолчанию используется то же значение, что и credentials\_validity\_in\_ms.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию: 2000

partitioner. Разделитель отвечает за распределение групп строк (по ключу раздела) по узлам кластера. Разделитель НЕ может быть изменен без перезагрузки всех данных. Если вы добавляете узлы или обновляете, вам следует установить для этого тот же разделитель, который вы используете в данный момент.

Разделителем по умолчанию является Murmur3Partitioner. Старые средства разделения, такие как RandomPartitioner, ByteOrderedPartitioner и OrderPreservingPartitioner, были включены только для обратной совместимости. Для новых кластеров НЕ следует изменять это значение.

Значение по умолчанию: org.apache.cassandra.dht.Murmur3Partitioner

data\_file\_directories. Каталоги, в которых RT.KeyValue должна хранить данные на диске. Если указано несколько каталогов, RT.KeyValue равномерно распределит данные по ним, разделив диапазоны токенов. Если не установлен, каталог по умолчанию - $ CASSANDRA\_HOME / data / data.

**По умолчанию этот параметр закомментирован.**

Значение по умолчанию (сложный вариант):

|  |
| --- |
| # - /var/lib/cassandra/data |

local\_system\_data\_file\_directory. Каталог, в котором RT.KeyValue должна хранить данные пространств ключей локальной системы.

**По умолчанию этот параметр закомментирован.**

По умолчанию RT.KeyValue будет хранить данные локальных системных пространств ключей (за исключением system.batches, system.paxos, system.compaction\_history, system.prepared\_statements и system.repair) в первом из каталогов данных, указанные в data\_file\_directories.

Такой подход гарантирует, что в случае потери одного из других дисков RT.KeyValue сможет продолжить работу. Для дополнительной безопасности этот параметр позволяет хранить эти данные в другом каталоге, что обеспечивает избыточность.

Полное описание файла local\_system\_data\_file\_directory представлено в **Приложении 1.**

Настройка файла cassandra-rackdc.properties

Некоторые параметры snitch используют файл cassandra-rackdc.properties, чтобы определить, каким центрам обработки данных (datacenters) и узлам кластера принадлежат. Информация о топологии сети позволяет эффективно маршрутизировать запросы и равномерно распределять реплики. Здесь можно настроить следующие снитчи:

* GossipingPropertyFileSnitch
* Однорегиональный снитч AWS EC2
* Многорегиональный снитч AWS EC2

GossipingPropertyFileSnitch рекомендуется для производства. Этот snitch использует информацию о центре обработки данных и стойке, настроенную в файле cassandra-rackdc.properties локального узла, и передает информацию другим узлам с помощью сплетен (gossip). Это снитч по умолчанию и настройки в этом файле свойств включены.

Снитчи AWS EC2 настроены для кластеров в AWS. Этот снитч использует параметры cassandra-rackdc.properties для обозначения одного из двух соглашений об именах центров обработки данных и стоек AWS EC2:

* legacy: имя центра обработки данных - это часть имени зоны доступности, предшествующая последнему знаку «-», когда зона заканчивается на -1, и включает номер, если не -1. Имя стойки - это часть имени зоны доступности после последнего знака «-».

Примеры: us-west-1a ⇒ dc: us-west, rack: 1a; us-west-2b ⇒ dc: us-west-2, стойка: 2b;

* standard: имя центра обработки данных - это стандартное название региона AWS, включая номер. Название стойки - это регион и буква зоны доступности.

Примеры: us-west-1a ⇒ dc: us-west-1, стойка: us-west-1a; us-west-2b ⇒ dc: us-west-2, стойка: us-west-2b;

Любой снитч может использовать локальный или внутренний IP-адрес, когда несколько центров обработки данных не обмениваются данными.

GossipingPropertyFileSnitch

dc

dc - название центра обработки данных. Значение чувствительно к регистру.

Значение по умолчанию: DC1

rack

rack – обозначение стойки. Значение чувствительно к регистру.

Значение по умолчанию: RAC1

AWS EC2 snitch

**По умолчанию этот параметр закомментирован.**

ec2\_naming\_scheme - соглашение об именах центров обработки данных и стоек. Возможные варианты: legacy или standard (по умолчанию).

Значение по умолчанию: standard

Either snitch

**По умолчанию этот параметр закомментирован.**

perfer\_local. Возможность использовать локальный или внутренний IP-адрес, когда связь не осуществляется между разными центрами обработки данных.

Значение по умолчанию: true

Настройка файла cassandra-env.sh

Файл сценария (script) bash cassandra-env.sh может использоваться для передачи дополнительных параметров виртуальной машине Java (JVM), таких как максимальный и минимальный размер кучи (heap), вместо их установки в среде. Если настройки JVM статичны и их не нужно вычислять на основе характеристик узла, вместо них следует использовать файлы cassandra-jvm-options. Например, обычно вычисляемые значения - это размеры кучи с использованием системных значений.

Например, добавьте VM\_OPTS="$JVM\_OPTS -Dcassandra.load\_ring\_state=false" в файл cassandra\_env.sh и для запуска запустите cassandra из командной строки. Параметр устанавливается из файла cassandra-env.sh и эквивалентен запуску RT.KeyValue с параметром командной строки cassandra -Dcassandra.load\_ring\_state=false.

Параметр -D указывает параметры запуска как в командной строке, так и в файле cassandra-env.sh. Доступны следующие варианты:

1. cassandra.auto\_bootstrap=false. Облегчает установку auto\_bootstrap на false при начальной настройке кластера. В следующий раз, когда вы запустите кластер, вам не нужно будет изменять файл cassandra.yaml на каждом узле, чтобы вернуться к значению true, по умолчанию.
2. cassandra.available\_processors=<number\_of\_processors>. При развертывании с несколькими экземплярами несколько экземпляров RT.KeyValue будут независимо предполагать, что для них доступны все процессоры центрального процессора. Этот параметр позволяет указать меньший набор процессоров.
3. cassandra.boot\_without\_jna=true. Если JNA не инициализируется, RT.KeyValue не загружается. Используйте эту команду для загрузки Cassandra без JNA.
4. cassandra.config=<directory>. Расположение каталога cassandra.yaml file. Расположение по умолчанию зависит от типа установки.
5. cassandra.ignore\_dynamic\_snitch\_severity=true|false Установка для этого свойства в значение true заставляет динамический снитч игнорировать индикатор серьезности сплетен (gossip) при оценке узлов. Изучите обнаружение и восстановление после сбоев, а также динамический снитч для получения дополнительной информации.

По умолчанию: false.

1. cassandra.initial\_token=<token>. Используйте, когда виртуальные узлы (vnodes) не используются. Устанавливает начальный токен разделителя для узла при первом запуске узла. Примечание: настоятельно рекомендуется использовать виртуальные узлы, поскольку они автоматически выбирают токены.

По умолчанию: disabled.

1. cassandra.join\_ring=true|false. Установите значение false, чтобы запустить RT.KeyValue на узле, но не присоединить узел к кластеру. Вы можете использовать nodetool join и JMX-вызов, чтобы впоследствии присоединиться к кольцу.

По умолчанию: true

1. cassandra.load\_ring\_state=true|false. Установите значение false, чтобы очистить все сплетни для узла при перезапуске.

По умолчанию: true

1. cassandra.metricsReporterConfigFile=<filename>. Включите подключаемый репортер метрик. Изучите подключаемые отчеты по метрикам для получения дополнительной информации.
2. cassandra.partitioner=<partitioner>. Установите разделитель (partitioner).

По умолчанию: org.apache.cassandra.dht.Murmur3Partitioner

1. cassandra.prepared\_statements\_cache\_size\_in\_bytes=<cache\_size>. Установите размер кеша для подготовленных операторов.
2. cassandra.replace\_address=<listen\_address of dead node>|<broadcast\_address of dead node>. Чтобы заменить умерший узел (dead node), перезапустите новый узел на его месте, указав listen\_address или broadcast\_address, который принимает новый узел. Новый узел не должен иметь никаких данных в своем каталоге данных, в том же состоянии, что и до начальной загрузки. Примечание. По умолчанию для параметра broadcast\_address используется listen\_address, за исключением случаев использования Ec2MultiRegionSnitch.
3. cassandra.replayList=<table>. Разрешить восстановление определенных таблиц из заархивированного журнала фиксации.
4. cassandra.ring\_delay\_ms=<number\_of\_ms>. Определяет время, в течение которого узел ожидает ответа от других узлов, прежде чем формально присоединиться к кольцу.

По умолчанию: 1000 мс

1. cassandra.native\_transport\_port=<port>. Установите порт, на котором собственный транспорт CQL прослушивает клиентов.

По умолчанию: 9042

1. cassandra.rpc\_port=<port>. Установите порт для службы Thrift RPC, который используется для клиентских подключений.

По умолчанию: 9160

1. cassandra.storage\_port=<port>. Установите порт для связи между узлами.

По умолчанию: 7000

1. cassandra.ssl\_storage\_port=<port>. Установите порт SSL для зашифрованной связи.

По умолчанию: 7001

1. cassandra.start\_native\_transport=true|false. Включите или отключите собственный транспортный сервер. См. start\_native\_transport в cassandra.yaml.

По умолчанию: true

1. cassandra.start\_rpc=true|false. Включите или отключите сервер Thrift RPC.

По умолчанию: true

1. cassandra.triggers\_dir=<directory>. Установите местоположение по умолчанию для триггерных JAR-файлов.

По умолчанию: conf / triggers

1. cassandra.write\_survey=true. Для тестирования новых стратегий уплотнения и сжатия. Это позволяет вам экспериментировать с различными стратегиями и тестировать различия в производительности записи, не влияя на производственную рабочую нагрузку.
2. consistent.rangemovement=true|false. Значение true позволяет RT.KeyValue безопасно выполнять загрузку без нарушения согласованности. Значение false отключает это.

Настройка файла cassandra-topologies.properties

Параметр snitch PropertyFileSnitch использует файл конфигурации cassandra-topologies.properties, чтобы определить, каким центрам обработки данных и стойкам принадлежат узлы кластера. Если используются другие snitches, необходимо использовать: ref: cassandra\_rackdc. Снитч определяет топологию сети (близость стойки и центра обработки данных), чтобы запросы маршрутизировались эффективно и позволяли базе данных равномерно распределять реплики.

Включите каждый узел кластера в файл свойств, задав имена центров обработки данных, как в определении пространства ключей. Имена центра обработки данных и стойки чувствительны к регистру.

Файл cassandra-topologies.properties необходимо копировать идентично на каждый узел кластера.

В следующем примере используются 3 центра обработки данных:

|  |
| --- |
| # datacenter One  175.56.12.105=DC1:RAC1  175.50.13.200=DC1:RAC1  175.54.35.197=DC1:RAC1  120.53.24.101=DC1:RAC2  120.55.16.200=DC1:RAC2  120.57.102.103=DC1:RAC2  # datacenter Two  110.56.12.120=DC2:RAC1  110.50.13.201=DC2:RAC1  110.54.35.184=DC2:RAC1  50.33.23.120=DC2:RAC2  50.45.14.220=DC2:RAC2  50.17.10.203=DC2:RAC2  # datacenter Three  172.106.12.120=DC3:RAC1  172.106.12.121=DC3:RAC1  172.106.12.122=DC3:RAC1  # default for unknown nodes  default =DC3:RAC1 |

Настройка файла commitlog-archiving.properties

В файле конфигурации commitlog-archiving.properties можно дополнительно установить команды, которые выполняются при архивировании или восстановлении сегмента журнала фиксации.

Параметры

archive\_command=<command> - Одна команда может быть вставлена с аргументами% path и% name.

% path - это полный путь к архиву сегмента журнала фиксации.

% name - это имя файла журнала фиксации.

STDOUT, STDIN или несколько команд не могут быть выполнены. Если требуется несколько команд, добавьте указатель на сценарий в этом параметре.

Пример: archive\_command=/bin/ln %path /backup/%name

Значение по умолчанию: пусто (blank).

restore\_command=<command> - Одна команда может быть вставлена с аргументами %from и %to.

%from - это полный путь к заархивированному сегменту журнала фиксации с использованием указанных каталогов восстановления.

%to определяет каталог для расположения текущего журнала фиксации.

Пример: restore\_command = / bin / cp –f %from %to

Значение по умолчанию: пусто (blank).

restore\_directories=<directory> - Определяет каталог для сканирования файлов восстановления.

Значение по умолчанию: пусто (blank).

restore\_point\_in\_time=<timestamp> - Восстановление мутаций (Restore mutations), созданные до этой отметки времени включительно, в GMT в формате yyyy:MM:dd HH:mm:ss. Восстановление будет продолжаться в течение всего сегмента, когда будет обнаружена первая предоставленная клиентом метка времени, превышающая это время, но будут применены только мутации, меньшие или равные этой метке времени.

Пример: 2020:04:31 20:43:12

Значение по умолчанию: пусто (blank).

precision=<timestamp\_precision> - Точность метки времени, используемой во вставках. Выбор обычно составляет МИЛЛИСЕКУНДЫ или МИКРОСЕКУНДЫ.

Значение по умолчанию: МИКРОСЕКУНДЫ.

Настройка файла logback.xml

В файле конфигурации logback.xml можно дополнительно установить уровни ведения журнала для журналов, записываемых в system.log и debug.log. Уровни ведения журнала также можно установить с помощью nodetool setlogginglevels.

Параметры

appender name="<appender\_choice>"…​</appender>. Укажите тип журнала и настройки. Возможные имена приложений: SYSTEMLOG, DEBUGLOG, ASYNCDEBUGLOG и STDOUT. SYSTEMLOG обеспечивает синхронную запись сообщений WARN и ERROR в указанный файл. DEBUGLOG и ASYNCDEBUGLOG гарантируют, что сообщения DEBUG записываются либо синхронно, либо асинхронно, соответственно, в указанный файл. STDOUT записывает все сообщения в консоль в удобочитаемом формате.

Пример: <appender name = "SYSTEMLOG" class = "ch.qos.logback.core.rolling.RollingFileAppender">

<file> <filename> </file>. Укажите имя файла для журнала.

Пример: <file> $ \ {cassandra.logdir} /system.log </file>.

<level> <log\_level> </level>. Укажите уровень для журнала. Часть фильтра. Уровни: ALL, TRACE, DEBUG, INFO, WARN, ERROR, OFF. TRACE создает наиболее подробный журнал, ERROR - наименьший.

|  |
| --- |
| **Примечание.** |
| Повышение уровня ведения журнала может привести к тяжелому выходу журнала в кластере с умеренным трафиком. Вы можете использовать команду nodetool getlogginglevels, чтобы увидеть текущую конфигурацию ведения журнала. |

По умолчанию: INFO

Пример: <level> INFO </level>

<rollingPolicy class="<rolling\_policy\_choice>" <fileNamePattern><pattern\_info></fileNamePattern> …​ </rollingPolicy>. Укажите политику перехода журналов в архив.

Пример: <RollingPolicy class = "ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">

<fileNamePattern> <pattern\_info> </fileNamePattern>. Укажите информацию о шаблоне для переноса журнала в архив. Часть скользящей политики.

Пример: <fileNamePattern> $ \ {cassandra.logdir} /system.log.%d {yyyy-MM-dd}.% I.zip </fileNamePattern>

<maxFileSize> <size> </maxFileSize>. Укажите максимальный размер файла для запуска прокатки журнала. Часть скользящей политики.

Пример: <maxFileSize> 50 МБ </maxFileSize>

<maxHistory> <number\_of\_days> </maxHistory>. Укажите максимальную историю в днях для запуска прокрутки журнала. Часть скользящей политики.

Пример: <maxHistory> 7 </maxHistory>

<encoder> <pattern>…​</pattern> </encoder>. Укажите формат сообщения. Часть скользящей политики.

Example: <encoder> <pattern>%-5level [%thread] %date{ISO8601} %F:%L - %msg%n</pattern> </encoder>

В **Приложении 2**. представлено содержимое logback.xml файла по умолчанию.

Настройка jvm-\* файлов

В RT.KeyValue включено несколько файлов для конфигурации JVM. Файл jvm-server.options и соответствующие файлы jvm8-server.options и jvm11-server.options являются основным файлом для настроек, которые влияют на работу RT.KeyValue JVM на узлах кластера. Файл включает параметры запуска, общие настройки JVM, такие как сборка мусора и настройки кучи. jvm-clients.options и соответствующие файлы jvm8-clients.options и jvm11-clients.options можно использовать для настройки параметров JVM для таких клиентов, как nodetool и инструменты sstable.

Операции RT.KeyValue

В данном разделе описываются операции RT.KeyValue.

Аппаратное обеспечение

Как и у большинства баз данных, пропускная способность RT.KeyValue увеличивается за счет большего количества ядер ЦП, большего объема оперативной памяти и более быстрых дисков. В то время как RT.KeyValue может работать на небольших серверах для сред тестирования или разработки, минимальный производственный сервер требует не менее 2 ядер и не менее 8 ГБ ОЗУ. Типичные производственные серверы имеют 8 или более ядер и не менее 32 ГБ оперативной памяти.

Центральный процессор

RT.KeyValue работает одновременно с множеством единовременных запросов (как на чтение, так и на запись), используя несколько потоков, работающих на максимально возможном количестве ядер ЦП. Путь записи RT.KeyValue имеет тенденцию быть значительно оптимизированным (запись в журнал фиксации, а затем вставка данных в таблицу памяти), поэтому запись, в частности, обычно связана с процессором. Следовательно, добавление дополнительных ядер ЦП часто увеличивает пропускную способность как чтения, так и записи.

Объем оперативной памяти

RT.KeyValue работает в виртуальной машине Java, которая предварительно выделяет кучу (heap) фиксированного размера (системный параметр Java Xmx). Помимо кучи (heap), RT.KeyValue будет использовать значительные объемы оперативной памяти для метаданных сжатия, фильтров Блума, кешей строк, ключей и счетчиков, а также кеша страниц в процессе. Наконец, RT.KeyValue воспользуется преимуществами кеширования страниц операционной системы, сохраняя файлы недавно использованных частей в ОЗУ для быстрого повторного использования.

Для оптимальной производительности операторы должны тестировать и настраивать свои кластеры в соответствии с их индивидуальной рабочей нагрузкой. Однако основные рекомендации предполагают:

* Всегда следует использовать ОЗУ с ECC, поскольку в RT.KeyValue мало внутренних средств защиты от повреждения битового уровня.
* Куча (heap) RT.KeyValue должна быть не менее 2 ГБ и не более 50% оперативной памяти вашей системы.
* Кучи (heap) размером менее 12ГБ должны учитывать сборку мусора ParNew / ConcurrentMarkSweep.
* Для Кучи размером более 12 ГБ следует учитывать:
* Куча 16ГБ с 8-10ГБ нового поколения, коэффициент выживаемости 4-6 и максимальный порог владения 6
* G1GC

Диски

RT.KeyValue сохраняет данные на диск для двух очень разных целей. Первый - в журнал фиксации, когда делается новая запись, чтобы ее можно было воспроизвести после сбоя или выключения системы. Второй - в каталог данных, когда пороговые значения превышены, и memtables сбрасываются на диск как SSTables.

Журналы фиксации получают каждую запись, сделанную в узел RT.KeyValue, и могут блокировать клиентские операции, но они всегда читаются только при запуске узла. С другой стороны, запись в SSTable (файл данных) происходит асинхронно, но считывается для удовлетворения запросов клиентов. SSTables также периодически объединяются и перезаписываются в процессе, называемом уплотнением. Данные, хранящиеся в каталоге commitlog, - это данные, которые не были постоянно сохранены в каталогах данных SSTable - они будут периодически очищаться после того, как они будут сброшены в файлы данных SSTable.

RT.KeyValue очень хорошо работает как на вращающихся жестких дисках, так и на твердотельных дисках. В обоих случаях отсортированные неизменяемые таблицы SSTables RT.KeyValue допускают линейное чтение, небольшое количество поисков и небольшое количество перезаписей, максимизируя пропускную способность для жестких дисков и срок службы твердотельных накопителей, избегая усиления записи. Однако при использовании вращающихся дисков важно, чтобы журнал фиксации (commitlog\_directory) находился на одном физическом диске (не просто в разделе, а на физическом диске), а файлы данных (data\_file\_directories) были настроены на отдельный физический диск. Отделив журнал фиксации от каталога данных, записи могут выиграть от последовательных добавлений в журнал фиксации без необходимости искать вокруг диска при чтении данных запроса из различных таблиц SSTable на диске.

В большинстве случаев RT.KeyValue предназначена для обеспечения избыточности через несколько независимых недорогих серверов. По этой причине использование NFS или SAN для каталогов данных является антипаттерном, и его обычно следует избегать. Точно так же серверы с несколькими дисками часто лучше обслуживаются с помощью RAID0 или JBOD, чем RAID1 или RAID5 - репликация, предоставляемая RT.KeyValue, устраняет необходимость репликации на уровне диска, поэтому обычно рекомендуется, чтобы операторы использовали дополнительную пропускную способность RAID0, а не защита от сбоев с помощью RAID1 или RAID5

Общие варианты облачных вычислений (Common Cloud Choices)

Многие крупные пользователи RT.KeyValue работают в различных облаках. Пользователи должны выбрать оборудование, аналогичное тому, что потребуется в физическом пространстве. В EC2 популярные варианты включают:

* инстансы i2, которые обеспечивают как высокое соотношение ОЗУ: ЦП, так и локальные временные твердотельные накопители
* инстансы i3 с дисками NVMe
* EBS работает нормально, если вам нужно легкое резервное копирование и замена
* инстансы m4.2xlarge / c4.4xlarge, которые предоставляют современные процессоры, расширенные возможности работы в сети и хорошо работают с хранилищем EBS GP2 (SSD)

Как правило, производительность диска и сети увеличивается с увеличением размера и поколения экземпляров, поэтому новые поколения экземпляров и более крупные типы экземпляров в каждом семействе часто работают лучше, чем их меньшие или более старые альтернативы.

Безопасность RT.KeyValue

Функции безопасности, предоставляемые RT.KeyValue, состоят из трех основных компонентов:

* шифрование TLS / SSL для связи между клиентом и узлами;
* проверка подлинности клиента;
* авторизация.

По умолчанию эти функции отключены, так как RT.KeyValue настроена так, чтобы ее легко находили другие члены кластера. Другими словами, готовая установка RT.KeyValue представляет собой большую поверхность атаки для злоумышленника. Включение аутентификации для клиентов, использующих двоичный протокол, недостаточно для защиты кластера. Злоумышленники, имеющие доступ к межузловым коммуникациям и портам JMX, по-прежнему могут:

* Создавайте межузловые сообщения для вставки пользователей в схему аутентификации.
* Создавайте межузловые сообщения для усечения или удаления схемы.
* Используйте такие инструменты, как sstableloader, для перезаписи таблиц system\_auth.
* Подключитесь к кластеру напрямую для захвата трафика записи.

Правильная настройка всех трех компонентов защиты должна сводить на нет эти векторы. Поэтому понимание функций безопасности RT.KeyValue имеет решающее значение для настройки кластера в соответствии с вашими потребностями в безопасности.

TSL / SSL шифрование

RT.KeyValue обеспечивает безопасную связь между клиентским компьютером и кластером базы данных, а также между узлами внутри кластера. Включение шифрования гарантирует, что данные в полете не будут скомпрометированы и будут безопасно передаваться. Параметры шифрования "клиент-узел" и "узел-узел" управляются отдельно и могут быть настроены независимо.

В обоих случаях используются значения JVM по умолчанию для поддерживаемых протоколов и наборов шифров, когда включено шифрование. Их можно переопределить с помощью настроек в cassandra.yaml, но это не рекомендуется, если не существует политик, которые диктуют определенные настройки, или необходимости отключать уязвимые шифры или протоколы в случаях, когда JVM не может быть обновлен.

Параметры, совместимые с FIPS, могут быть настроены на уровне JVM и не должны включать изменение параметров шифрования в cassandra.yaml.

Горячая перезагрузка SSL-сертификата (Hot Reloading)

Если в RT.KeyValue включена поддержка SSL / TLS, узел периодически опрашивает доверенные хранилища и хранилища ключей, указанные в cassandra.yaml. Когда файлы будут обновлены, RT.KeyValue перезагрузит их и будет использовать для последующих подключений. Обратите внимание, что пароли Trust & Key Store являются частью yaml, поэтому в обновленных файлах также должны использоваться те же пароли. Интервал опроса по умолчанию составляет 10 минут.

Горячая перезагрузка сертификата также может быть запущена с помощью команды nodetool reloadssl. Используйте это, если хотите, чтобы RT.KeyValue сразу заметила измененные сертификаты.

Межузловое шифрование (inter-node)

Настройки для управления межузловым шифрованием находятся в cassandra.yaml в разделе server\_encryption\_options. Чтобы включить межузловое шифрование, измените параметр internode\_encryption со значения по умолчанию none на одно значение из: rack, dc или all.

Шифрование от клиента к узлу (client to node)

Настройки для управления шифрованием от клиента к узлу находятся в cassandra.yaml в разделе client\_encryption\_options. Здесь есть два основных переключателя для включения шифрования: включено (enabled) и необязательно (optional).

* Если ни один из них не установлен в значение true, клиентские соединения полностью незашифрованы.
* Если для параметра enabled установлено значение true, а для параметра optional установлено значение false, все клиентские соединения должны быть защищены.
* Если для обоих параметров установлено значение true, поддерживаются как зашифрованные, так и незашифрованные соединения с использованием одного и того же порта. Клиентские подключения, использующие шифрование с этой конфигурацией, будут автоматически обнаруживаться и обрабатываться сервером.

В качестве альтернативы необязательной (optional) настройке отдельные порты также могут быть настроены для безопасных и незащищенных соединений, если этого требуют операционные требования. Для этого установите для optional значение false и используйте параметр native\_transport\_port\_ssl в cassandra.yaml, чтобы указать порт, который будет использоваться для безопасного взаимодействия с клиентом.

Роли

RT.KeyValue использует роли базы данных, которые могут представлять либо одного пользователя, либо группу пользователей, как при аутентификации, так и при управлении разрешениями. Управление ролями - это точка расширения в RT.KeyValue, которую можно настроить с помощью параметра role\_manager в cassandra.yaml. По умолчанию используется CassandraRoleManager, реализация, которая хранит информацию о ролях в таблицах пространства ключей system\_auth.

Аутентификация

Аутентификация подключается в RT.KeyValue и настраивается с помощью параметра аутентификатора (authenticator) в cassandra.yaml. RT.KeyValue поставляется с двумя опциями, включенными в дистрибутив по умолчанию.

По умолчанию RT.KeyValue настроен с AllowAllAuthenticator, который не выполняет проверки аутентификации и, следовательно, не требует учетных данных. Он используется для полного отключения аутентификации. Обратите внимание, что аутентификация является необходимым условием подсистемы разрешений RT.KeyValue, поэтому, если аутентификация отключена, то фактически таковы и разрешения.

Распространение по умолчанию также включает PasswordAuthenticator, который хранит зашифрованные учетные данные в системной таблице. Это можно использовать для включения простой аутентификации по имени пользователя и паролю.

Включение аутентификации по паролю

Перед включением проверки подлинности клиента в кластере клиентские приложения должны быть предварительно настроены с использованием предполагаемых учетных данных. Когда соединение инициируется, сервер будет запрашивать учетные данные только после включения аутентификации, поэтому предварительная настройка конфигурации на стороне клиента безопасна. Напротив, как только сервер включит аутентификацию, любая попытка подключения без надлежащих учетных данных будет отклонена, что может вызвать проблемы с доступностью для клиентских приложений. После того, как клиенты настроены и готовы к включению проверки подлинности, выполните эту процедуру, чтобы включить ее в кластере.

Выберите один узел в кластере, на котором нужно выполнить начальную настройку. В идеале ни один клиент не должен подключаться к этому узлу во время процесса установки, поэтому вы можете удалить его из конфигурации клиента, заблокировать на сетевом уровне или, возможно, добавить для этой цели новый временный узел в кластер. На этом узле выполните следующие шаги:

1. Откройте сеанс cqlsh и измените коэффициент репликации пространства ключей system\_auth. По умолчанию в этом пространстве ключей используется SimpleReplicationStrategy и коэффициент replication\_factor, равный 1. Рекомендуется изменить это значение для любого нетривиального развертывания, чтобы гарантировать, что даже если узлы станут недоступными, вход в систему будет по-прежнему возможен. Лучше всего настроить коэффициент репликации от 3 до 5 для каждого контроллера домена.

|  |
| --- |
| ALTER KEYSPACE system\_auth WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2': 3}; |

1. Отредактируйте cassandra.yaml, чтобы изменить параметр аутентификатора следующим образом:

|  |
| --- |
| authenticator: PasswordAuthenticator |

1. Перезагрузите узел.
2. Откройте новый сеанс cqlsh, используя учетные данные суперпользователя по умолчанию:

|  |
| --- |
| $ cqlsh -u cassandra -p cassandra |

1. Во время входа в систему учетные данные суперпользователя по умолчанию считываются с уровнем согласованности QUORUM, тогда как учетные данные для всех других пользователей (включая суперпользователей) считываются с LOCAL\_ONE. В интересах производительности и доступности, а также безопасности операторы должны создать еще одного суперпользователя и отключить пользователя по умолчанию. Этот шаг не является обязательным, но настоятельно рекомендуется. Войдя в систему как суперпользователь по умолчанию, создайте другую роль суперпользователя, которую можно использовать для начальной загрузки дальнейшей конфигурации.

|  |
| --- |
| $ # create a new superuser  CREATE ROLE dba WITH SUPERUSER = true AND LOGIN = true AND PASSWORD = 'super'; |

1. Начните новый сеанс cqlsh, на этот раз войдя в систему как new\_superuser и отключив суперпользователя по умолчанию.

|  |
| --- |
| ALTER ROLE cassandra WITH SUPERUSER = false AND LOGIN = false; |

1. Наконец, настройте роли и учетные данные для пользователей приложения с помощью операторов CREATE ROLE <create-role-statement>.

В конце этих шагов один узел настроен на использование аутентификации по паролю. Чтобы развернуть это в кластере, повторите шаги 2 и 3 на каждом узле кластера. После перезапуска всех узлов, аутентификация будет полностью включена во всем кластере.

Обратите внимание, что использование PasswordAuthenticator также требует использования CassandraRoleManager <operation-roles>.

Смотрите также: setting-credentials-for-internal-authentication, CREATE ROLE <create-role-statement>, ALTER ROLE <alter-role-statement>, ALTER KEYSPACE <alter-keyspace-statement>  и GRANT PERMISSION <grant-permission-statement>.

Авторизация

Авторизация подключается в RT.KeyValue и настраивается с помощью настройки авторизатора (authorizer) в cassandra.yaml. RT.KeyValue поставляется с двумя опциями, включенными в дистрибутив по умолчанию.

По умолчанию RT.KeyValue настроена с помощью AllowAllAuthorizer, который не выполняет никаких проверок и поэтому фактически предоставляет все разрешения для всех ролей. Это необходимо использовать, если настроенным аутентификатором является AllowAllAuthenticator.

В дистрибутив, по умолчанию, также входит CassandraAuthorizer, который реализует полную функциональность управления разрешениями и хранит свои данные в системных таблицах RT.KeyValue.

Включение внутренней авторизации

Разрешения моделируются как белый список с предположением по умолчанию, что данная роль не имеет доступа к каким-либо ресурсам базы данных. Следствием этого является то, что после включения авторизации на узле все запросы будут отклоняться до тех пор, пока не будут предоставлены необходимые разрешения. По этой причине настоятельно рекомендуется выполнить первоначальную настройку на узле, который не обрабатывает клиентские запросы.

Далее предполагается, что аутентификация уже была включена посредством процесса, описанного в разделе аутентификация по паролю. Выполните следующие действия, чтобы включить внутреннюю авторизацию в кластере:

1. На выбранном узле отредактируйте cassandra.yaml, чтобы изменить параметр авторизатора (authorizer) следующим образом:

|  |
| --- |
| authorizer: CassandraAuthorizer |

1. Перезагрузите узел.
2. Откройте новый сеанс cqlsh, используя учетные данные роли с учетными данными суперпользователя:

|  |
| --- |
| $ cqlsh -u dba -p super |

1. Настройте соответствующие права доступа для своих клиентов с помощью операторов GRANT PERMISSION. На других узлах, пока конфигурация не будет обновлена и узел не будет перезапущен, это не повлияет на работу клиентов, что позволит избежать сбоев в работе клиентов.

|  |
| --- |
| GRANT SELECT ON ks.t1 TO db\_user; |

1. После предоставления всех необходимых разрешений повторите шаги 1 и 2 для каждого узла по очереди. После перезапуска каждого узла и повторного подключения клиентов начнется принудительное применение предоставленных разрешений.

Смотрите также: GRANT PERMISSION <grant-permission-statement>, GRANT ALL <grant-all> и REVOKE PERMISSION <revoke-permission-statement>.

Кеширование

Включение аутентификации и авторизации создает дополнительную нагрузку на кластер из-за частого чтения из таблиц system\_auth. Кроме того, эти операции чтения находятся на критическом пути многих клиентских операций и могут серьезно повлиять на качество обслуживания. Чтобы смягчить это, данные аутентификации, такие как учетные данные, разрешения и сведения о ролях, кэшируются на настраиваемый период. Кеширование можно настроить (и даже отключить) из cassandra.yaml или с помощью клиента JMX. Интерфейс JMX также поддерживает аннулирование различных кешей, но любые изменения, сделанные через JMX, не являются постоянными и будут повторно считаны из cassandra.yaml при перезапуске узла.

Каждый кеш имеет 3 параметра, которые можно установить:

* Срок действия

Управляет истечением срока хранения записей в кэше. По истечении этого периода записи становятся недействительными и удаляются из кеша.

* Частота обновления

Управляет скоростью, с которой выполняются фоновые чтения, чтобы улавливать любые изменения в базовых данных. Пока выполняются эти асинхронные обновления, кеши будут продолжать обслуживать (возможно) устаревшие данные. Как правило, это будет более короткое время, чем период действия.

* Максимальное количество записей

Управляет верхней границей размера кеша.

Именование для этих опций в cassandra.yaml следует соглашению:

* <type>\_validity\_in\_ms
* <type>\_update\_interval\_in\_ms
* <type>\_cache\_max\_entries

Где <type>- одно из учетных данных (credentials), разрешений (permissions) или ролей (roles).

Как уже упоминалось, они также доступны через JMX в mbeans в домене org.apache.cassandra.auth.

JMX доступ

Управление доступом для клиентов JMX настраивается отдельно от управления доступом для CQL. И для аутентификации, и для авторизации доступны два провайдера; первый основан на стандартной безопасности JMX, а второй более тесно интегрируется с собственной подсистемой аутентификации RT.KeyValue.

Настройки по умолчанию для RT.KeyValue делают JMX доступным только с localhost. Чтобы включить удаленные соединения JMX, отредактируйте cassandra-env.sh (или cassandra-env.ps1 в Windows), чтобы изменить параметр LOCAL\_JMX на no. В стандартной конфигурации, когда включены удаленные JMX-соединения, также включается стандартная аутентификация JMX <standard-jmx-auth> (standard JMX authentication <standard-jmx-auth>).

Обратите внимание, что по умолчанию только локальные соединения не подлежат аутентификации, но это можно включить.

При включении удаленных подключений рекомендуется также использовать подключения SSL <jmx-with-ssl>.

Наконец, после включения аутентификации и / или SSL убедитесь, что инструменты, использующие JMX, такие как nodetool <nodetool>, правильно настроены и работают должным образом.

Стандартная JMX Аутентификация

Пользователи, которым разрешено подключаться к серверу JMX, указываются в простом текстовом файле. Расположение этого файла задается в cassandra-env.sh строкой:

|  |
| --- |
| JVM\_OPTS="$JVM\_OPTS -Dcom.sun.management.jmxremote.password.file=/etc/cassandra/jmxremote.password" |

Отредактируйте файл паролей, чтобы добавить пары имя пользователя / пароль:

|  |
| --- |
| jmx\_user jmx\_password |

Защитите файл учетных данных, чтобы его мог прочитать только пользователь, запустивший процесс RT.KeyValue:

|  |
| --- |
| $ chown cassandra:cassandra /etc/cassandra/jmxremote.password  $ chmod 400 /etc/cassandra/jmxremote.password |

При желании можно включить контроль доступа, чтобы ограничить объем того, что определенные пользователи могут делать через JMX. Обратите внимание, что в данном контексте это довольно грубый инструмент, поскольку для большинства операционных инструментов в RT.KeyValue требуется полный доступ для чтения / записи. Чтобы настроить файл простого доступа, раскомментируйте эту строку в cassandra-env.sh:

|  |
| --- |
| #JVM\_OPTS="$JVM\_OPTS -Dcom.sun.management.jmxremote.access.file=/etc/cassandra/jmxremote.access" |

Затем отредактируйте файл доступа, чтобы предоставить пользователю JMX разрешение на чтение и запись:

|  |
| --- |
| jmx\_user readwrite |

Чтобы получить новые настройки, необходимо перезапустить RT.KeyValue.

Интегрирования Аутентификация RT.KeyValue

Альтернативой стандартной JMX-аутентификации является использование собственных провайдеров аутентификации и / или авторизации eCassandra для JMX-клиентов. Это потенциально более гибкий и безопасный вариант, но с одним серьезным недостатком. А именно, что он недоступен до тех пор, пока узел не присоединится к кольцу, потому что подсистема аутентификации не полностью настроена до этого момента. Однако для целей мониторинга часто бывает критичным для целей мониторинга иметь доступ JMX, особенно во время начальной загрузки. Поэтому рекомендуется, где это возможно, использовать только локальную аутентификацию JMX во время начальной загрузки, а затем, если требуется удаленное подключение, переключиться на встроенную аутентификацию после того, как узел присоединится к кольцу и начальная настройка будет завершена.

С этой опцией те же роли базы данных, которые используются для аутентификации CQL, могут использоваться для управления доступом к JMX, поэтому обновлениями можно управлять централизованно, используя только cqlsh. Более того, точный контроль над тем, какие именно операции разрешены на определенных MBean-компонентах, можно получить с помощью GRANT PERMISSION <grant-permission-statement>.

Чтобы включить встроенную аутентификацию, отредактируйте cassandra-env.sh, чтобы раскомментировать эти строки:

|  |
| --- |
| #JVM\_OPTS="$JVM\_OPTS -Dcassandra.jmx.remote.login.config=CassandraLogin"  #JVM\_OPTS="$JVM\_OPTS -Djava.security.auth.login.config=$CASSANDRA\_HOME/conf/cassandra-jaas.config" |

И отключите стандартную аутентификацию JMX, прокомментировав эту строку:

|  |
| --- |
| JVM\_OPTS="$JVM\_OPTS -Dcom.sun.management.jmxremote.password.file=/etc/cassandra/jmxremote.password" |

Чтобы включить интегрированную авторизацию, раскомментируйте эту строку:

|  |
| --- |
| #JVM\_OPTS="$JVM\_OPTS -Dcassandra.jmx.authorizer=org.apache.cassandra.auth.jmx.AuthorizationProxy" |

Убедитесь, что стандартный контроль доступа отключен, закомментировав эту строку:

|  |
| --- |
| #JVM\_OPTS="$JVM\_OPTS -Dcom.sun.management.jmxremote.access.file=/etc/cassandra/jmxremote.access" |

При включенной интегрированной аутентификации и авторизации операторы могут определять определенные роли и предоставлять им доступ к определенным ресурсам JMX, которые им нужны. Например, роль с необходимыми разрешениями для использования таких инструментов, как jconsole или jmc в режиме только для чтения, будет определяться как:

|  |
| --- |
| CREATE ROLE jmx WITH LOGIN = false;  GRANT SELECT ON ALL MBEANS TO jmx;  GRANT DESCRIBE ON ALL MBEANS TO jmx;  GRANT EXECUTE ON MBEAN 'java.lang:type=Threading' TO jmx;  GRANT EXECUTE ON MBEAN 'com.sun.management:type=HotSpotDiagnostic' TO jmx;  # Grant the role with necessary permissions to use nodetool commands (including nodetool status) in read-only mode  GRANT EXECUTE ON MBEAN 'org.apache.cassandra.db:type=EndpointSnitchInfo' TO jmx;  GRANT EXECUTE ON MBEAN 'org.apache.cassandra.db:type=StorageService' TO jmx;  # Grant the jmx role to one with login permissions so that it can access the JMX tooling  CREATE ROLE ks\_user WITH PASSWORD = 'password' AND LOGIN = true AND SUPERUSER = false;  GRANT jmx TO ks\_user; |

Также поддерживается детальный контроль доступа к отдельным MBean-компонентам:

|  |
| --- |
| GRANT EXECUTE ON MBEAN 'org.apache.cassandra.db:type=Tables,keyspace=test\_keyspace,table=t1' TO ks\_user;  GRANT EXECUTE ON MBEAN 'org.apache.cassandra.db:type=Tables,keyspace=test\_keyspace,table=\*' TO ks\_owner; |

Это позволяет роли ks\_user вызывать методы в MBean, представляющие одну таблицу в пространстве test\_keyspace, предоставляя при этом одинаковые разрешения для всех MBEANS уровня таблицы в этом пространстве ключей роли ks\_owner.

Добавление / удаление ролей и предоставление / отзыв разрешений обрабатываются динамически после завершения начальной настройки, поэтому дальнейшие перезапуски не требуются, если разрешения изменены.

JMX с SSL

Конфигурация JMX SSL контролируется рядом системных свойств, некоторые из которых являются необязательными. Чтобы включить SSL, отредактируйте соответствующие строки в cassandra-env.sh (или cassandra-env.ps1 в Windows), чтобы раскомментировать и установить требуемые значения этих свойств:

* com.sun.management.jmxremote.ssl

установите значение true, чтобы включить SSL

* com.sun.management.jmxremote.ssl.need.client.auth

установите значение true, чтобы включить проверку сертификатов клиентов

* com.sun.management.jmxremote.registry.ssl

включает сокеты SSL для реестра RMI, из которых клиенты получают заглушку коннектора JMX

* com.sun.management.jmxremote.ssl.enabled.protocols

по умолчанию будут использоваться протоколы, поддерживаемые JVM, переопределенные списком, разделенным запятыми. Обратите внимание, что обычно в этом нет необходимости, и использование значений по умолчанию является предпочтительным вариантом.

* com.sun.management.jmxremote.ssl.enabled.cipher.suites

по умолчанию будут использоваться комплекты шифров, поддерживаемые JVM, переопределенные списком, разделенным запятыми. Обратите внимание, что обычно в этом нет необходимости, и использование значений по умолчанию является предпочтительным вариантом.

* javax.net.ssl.keyStore

установить путь в локальной файловой системе хранилища ключей, содержащий закрытые ключи сервера и открытые сертификаты

* javax.net.ssl.keyStorePassword

установить пароль для файла хранилища ключей

* javax.net.ssl.trustStore

если требуется проверка сертификатов клиентов, используйте это свойство, чтобы указать путь к хранилищу доверенных сертификатов, содержащему общедоступные сертификаты доверенных клиентов

* javax.net.ssl.trustStorePassword

установить пароль для файла склада доверенных сертификатов

Изменения топологии

Добавление, замена, перемещение и удаление узлов.

Начальная загрузка (Bootstrap)

Добавление новых узлов называется «начальной загрузкой». Параметр num\_tokens будет определять количество виртуальных узлов (токенов), которым присоединяющийся узел будет назначен во время начальной загрузки. Токены определяют разделы кольца (диапазоны токенов), за которые будет отвечать узел.

Распределение токенов

С алгоритмом распределения токенов по умолчанию, новый узел выберет случайные токены num\_tokens, за которые будет отвечать. Поскольку токены распределяются случайным образом, распределение нагрузки улучшается с увеличением количества виртуальных узлов, но это также увеличивает накладные расходы на управление токенами. По умолчанию 256 виртуальных узлов должны обеспечивать разумный баланс нагрузки с приемлемыми накладными расходами.

Так же существует алгоритм распределения токенов на основе нагрузки существующих виртуальных узлов для заданного пространства ключей, что позволиляет улучшить распределение нагрузки с меньшим количеством токенов. Чтобы использовать этот подход, новый узел должен быть запущен с параметром JVM -Dcassandra.allocate\_tokens\_for\_keyspace=<keyspace>, где <keyspace> - это пространство ключей, из которого алгоритм может найти информацию о загрузке для оптимизации назначения токенов.

Назначение токена вручную

Вы можете указать список токенов, разделенных запятыми, вручную с помощью параметра initial\_token cassandra.yaml, и если он указан, RT.KeyValue пропустит процесс выделения токенов. Это может быть полезно при назначении токенов с помощью внешнего инструмента или при восстановлении узла с его предыдущими токенами.

Диапазон потоковой передачи (Range streaming)

После распределения токенов присоединяющийся узел выберет текущие реплики диапазонов токенов, за которые он станет отвечать для потоковой передачи данных. По умолчанию он будет передаваться из первичной реплики каждого диапазона токенов, чтобы гарантировать, что данные в новом узле будут соответствовать текущему состоянию.

В случае недоступности реплики согласованный процесс начальной загрузки завершится ошибкой. Чтобы переопределить это поведение и потенциально пропустить данные из недоступной реплики, установите флаг JVM -Dcassandra.consistent.rangemovement=false.

Возобновление неудачной / зависшей начальной загрузки

Если процесс начальной загрузки завершился неудачно, можно возобновить загрузку из предыдущего сохраненного состояния, вызвав возобновление начальной загрузки nodetool (nodetool bootstrap resume). Если по какой-то причине начальная загрузка зависает или останавливается, ее также можно возобновить, просто перезапустив узел. Чтобы очистить состояние начальной загрузки и начать заново, вы можете установить флаг запуска JVM -Dcassandra.reset\_bootstrap\_progress=true.

Ручная загрузка

Можно полностью пропустить процесс начальной загрузки и сразу же присоединиться к кольцу, установив скрытый параметр auto\_bootstrap: false. Это может быть полезно при восстановлении узла из резервной копии или создании нового дата-центра.

Удаление узлов

Вы можете вывести узел из кластера с помощью nodetool decommission из эксплуатации на действующий узел или nodetool removenod (на любом другом компьютере), чтобы удалить мертвый. Это назначит диапазоны, за которые отвечал старый узел, другим узлам и реплицирует туда соответствующие данные. Если используется вывод из эксплуатации, данные будут передаваться из списанного узла. Если используется removenode, данные будут передаваться из оставшихся реплик.

Никакие данные не удаляются автоматически из узла, который выводится из эксплуатации, поэтому, если вы хотите вернуть узел в эксплуатацию с другим токеном в кольце, его следует удалить вручную.

Перемещение узлов

Когда num\_tokens: 1, можно перемещать позицию узла в кольце с помощью nodetool move. Перенос удобен и более эффективен, чем вывод из эксплуатации + самозагрузка. После перемещения узла следует запустить nodetool cleanup, чтобы удалить все ненужные данные.

Замена мертвого узла

Чтобы заменить мертвый узел, запустите RT.KeyValue с флагом запуска JVM -Dcassandra.replace\_address\_first\_boot=<dead\_node\_ip>. Как только это свойство включено, узел переходит в состоянии гибернации, в течение которого все остальные узлы будут видеть, что этот узел находится в состоянии DOWN (DN), однако этот узел будет видеть себя как UP (UN). Точное состояние замены можно найти в nodetool netstats.

Теперь заменяющий узел начнет загрузку данных с остальных узлов кластера. Заменяющий узел будет получать записи только на этапе начальной загрузки, если он имеет IP-адрес, отличный от заменяемого узла.

|  |
| --- |
| **Примечание.** |
| Если применим любой из следующих случаев, НЕОБХОДИМО выполнить восстановление, чтобы снова сделать замененный узел согласованным, поскольку он пропустил текущие записи во время / до начальной загрузки. Временные рамки замены относятся к периоду с момента первоначальной смерти узла до момента завершения процесса замены новым узлом.  Узел не работает дольше, чем max\_hint\_window\_in\_ms, прежде чем его заменят.  Вы заменяете, используя тот же IP-адрес, что и мертвый узел, и замена занимает больше времени, чем max\_hint\_window\_in\_ms. |

Мониторинг прогресса

Прогресс загрузки, замены, перемещения и удаления можно отслеживать с помощью nodetool netstats, который показывает ход потоковых операций.

Очистка данных после перемещений диапазона

В качестве меры безопасности, RT.KeyValue не удаляет автоматически данные с узлов, которые «теряют» часть своего диапазона токенов из-за операции перемещения диапазона (начальной загрузки, перемещения, замены). Запустите очистку (nodetool cleanup) на узлах, которые потеряли диапазоны для присоединяющегося узла, когда вы убедитесь, что новый узел включен и работает. Если вы этого не сделаете, старые данные все равно будут учитываться при подсчете нагрузки на этот узел.

Подсказки (Hints)

Хинтинг - это метод восстановления данных, применяемый во время операций записи. Когда узлы реплик недоступны для принятия мутации либо из-за сбоя, либо из-за обычного планового обслуживания, координаторы, пытающиеся записать в эти реплики, сохраняют временные подсказки в своей локальной файловой системе для последующего применения к недоступной реплике. Подсказки (hints, хинты) - важный способ помочь сократить продолжительность несогласованности данных. Координаторы быстро воспроизводят подсказки после того, как недоступные реплики возвращаются в кольцо. Однако подсказки - это лучший способ и не гарантируют конечной согласованности, как антиэнтропийное восстановление (anti-entropy repair <repair>).

Подсказки полезны из-за того, как RT.KeyValue реплицирует данные для обеспечения отказоустойчивости, высокой доступности и долговечности. RT.KeyValue разделяет данные по кластеру (partitions data across the cluster <consistent-hashing-token-ring>), используя согласованное хеширование, а затем реплицирует ключи на несколько узлов в хэш-кольце. Чтобы гарантировать доступность, все реплики ключа могут принимать мутации без согласия, но это означает, что некоторые реплики могут принимать мутацию, а другие - нет. Когда это происходит, вносится несоответствие.

Подсказки - это один из трех способов, в дополнение к восстановлению чтения и полному / инкрементальному антиэнтропийному восстановлению, с помощью которых RT.KeyValue реализует гарантию конечной согласованности, что все обновления в конечном итоге будут получены всеми репликами. Подсказки, такие как чтение-восстановление, являются оптимальным вариантом и не являются альтернативой выполнению полного восстановления, но они действительно помогают сократить продолжительность несоответствия между репликами на практике.

Hinted Handoff

Hinted Handoff - это процесс, с помощью которого RT.KeyValue применяет подсказки к недоступным узлам.

Например, на **Рисунке 1**. предположим, что мутация должна быть произведена на уровне согласованности LOCAL\_QUORUM (Consistency Level LOCAL\_QUORUM) против пространства ключей с фактором репликации (Replication Factor) 3. Обычно клиент отправляет мутацию одному координатору, который затем отправляет мутацию всем трем репликам, и когда две из трех реплики подтверждают мутацию, когда координатор успешно отвечает клиенту. Однако, если узел-реплика недоступен, координатор сохраняет подсказку локально в файловой системе для последующего применения. Новые подсказки будут сохраняться до max\_hint\_window\_in\_ms простоя (по умолчанию 3 часа). Если недоступная реплика возвращается в кластер до истечения периода времени, координатор применяет к реплике любые ожидающие хинтинговые мутации, чтобы гарантировать сохранение согласованности в конечном итоге.

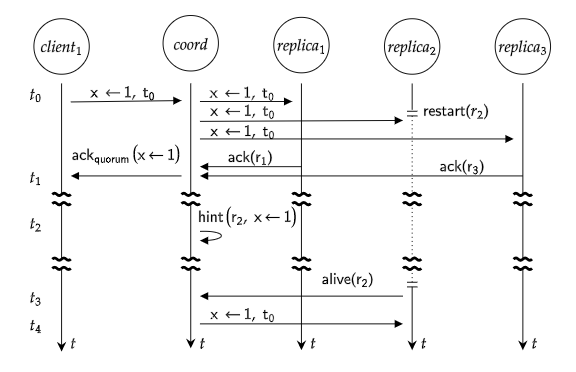


Рисунок 1. Пример согласованнного изменения (мутации)

* (t0): запись отправляется клиентом, и координатор отправляет ее трем репликам. К сожалению, replica\_2 перезапускается и не может получить мутацию.
* (t1): клиент получает подтверждение кворума от координатора. На этом этапе клиент считает, что запись долговечна и видна для чтения (что так и есть).
* (t2): после тайм-аута записи (по умолчанию 2 секунды) координатор решает, что replica\_2 недоступна, и сохраняет подсказку на свой локальный диск.
* (t3): Позже, когда replica\_2 начинает резервное копирование, она отправляет сообщение сплетни (gossip) всем узлам, включая координатора.
* (t4): координатор воспроизводит подсказки, включая пропущенную мутацию для replica\_2.

Если узел не вернется вовремя, реплика назначения будет постоянно не синхронизирована до тех пор, пока либо чтение-восстановление, либо полное / инкрементное антиэнтропийное восстановление не распространят мутацию.

Применение подсказок

Подсказки передаются массово, по сегментам, на целевой узел реплики, и целевой узел воспроизводит их локально. После того, как целевой узел воспроизвел сегмент, он удаляет сегмент и получает следующий сегмент. Это продолжается до тех пор, пока не исчезнут все подсказки.

Хранение подсказок на диске

Подсказки хранятся в неструктурированных (flat) файлах в каталоге $CASSANDRA\_HOME/data/hints узла координатора. Подсказка включает идентификатор подсказки, целевой узел реплики, на котором должна храниться мутация, сериализованную мутацию (хранящуюся в виде большого двоичного объекта), которая не может быть доставлена на узел реплики, временную метку мутации и версию RT.KeyValue для сериализации мутации. По умолчанию подсказки сжимаются с помощью LZ4Compressor. К одному файлу подсказок добавляется несколько подсказок.

Поскольку подсказки содержат исходную неизмененную метку времени мутации, приложение подсказки идемпотентно и не может перезаписать будущую мутацию.

Подсказки для просроченных запросов на запись

Подсказки также сохраняются для запросов на запись, срок действия которых истек. Параметр write\_request\_timeout\_in\_ms в cassandra.yaml настраивает тайм-аут для запросов на запись.

|  |
| --- |
| write\_request\_timeout\_in\_ms: 2000 |

Координатор ожидает завершения настроенного периода времени для запросов на запись, после чего он отключится и сгенерирует подсказку для запроса с истекшим временем ожидания. Наименьшее допустимое значение для write\_request\_timeout\_in\_ms составляет 10 мс.

Настройка подсказок

Подсказки включены по умолчанию, поскольку они имеют решающее значение для согласованности данных. В файле конфигурации cassandra.yaml есть несколько настроек для настройки подсказок, представленных в **Таблице 1**.

**Таблица 1. Настройки подсказок**

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Описание** | **Значение по умолчанию** |
| hinted\_handoff\_enabled | Включает / отключает hinted handoff | true |
| hinted\_handoff\_disabled\_datacenters | Список центров обработки данных, которые не выполняют hinted handoff передачу обслуживания, даже если передача обслуживания разрешена иным образом. Пример:  hinted\_handoff\_disabled\_datacenters:  - DC1  - DC2 | unset |
| max\_hint\_window\_in\_ms | Определяет максимальное количество времени (мс), в течение которого узел должен генерировать подсказки после сбоя. | 10800000 (3 часа) |
| hinted\_handoff\_throttle\_in\_kb | Максимальное дросселирование передачи в килобайтах в секунду для каждого потока доставки. Это будет уменьшено пропорционально количеству узлов в кластере. (Если в кластере два узла, каждый поток доставки будет использовать максимальную скорость; если их 3, каждый будет работать до половины максимального значения, поскольку ожидается, что два узла будут доставлять подсказки одновременно) | 1024 |
| max\_hints\_delivery\_threads | Количество потоков, с помощью которых доставляются подсказки; Рассмотрите возможность увеличения этого числа, когда у вас есть развертывания с несколькими постоянными токами, поскольку hinted handoff передача обслуживания между центрами обработки данных имеет тенденцию быть медленнее. | 2 |
| hints\_directory | Каталог, в котором RT.KeyValue хранит подсказки. | $CASSANDRA\_HOME/data/hints |
| hints\_flush\_period\_in\_ms | Как часто подсказки должны сбрасываться из внутренних буферов на диск. Не запускает fsync. | 10000 |
| max\_hints\_file\_size\_in\_mb | Максимальный размер одного файла подсказок в мегабайтах. | 128 |
| hints\_compression | Сжатие применимое к файлам подсказок. Если этот параметр не указан, файлы подсказок будут записаны без сжатия. | Поддерживаются компрессоры LZ4, Snappy и Deflate. |

Настройка подсказок во время выполнения с помощью nodetool

Команды nodetool переопределяют соответствующие настройки, если они есть в cassandra.yaml для узла, выполняющего команду. В **Таблице 2**. указаны команды nodetool для настройки подсказок или получения информации, связанной с подсказками.

Таблица 2. Команды Nodetoll для подсказок

|  |  |
| --- | --- |
| **Команды** | **Описание** |
| nodetool disablehandoff | Отключает хранение и доставку подсказок |
| nodetool disablehintsfordc | Отключает хранение и доставку подсказок в центр обработки данных |
| nodetool enablehandoff | Повторно включает будущие подсказки для хранения и доставки на текущий узел |
| nodetool enablehintsfordc | Включает подсказки для центра обработки данных, который ранее был отключен |
| nodetool getmaxhintwindow | Выводит максимальное окно подсказки в мс. |
| nodetool handoffwindow | Выводит текущее окно hinted handoff |
| nodetool pausehandoff | Приостанавливает процесс доставки подсказок |
| nodetool resumehandoff | Возобновляет процесс доставки подсказок |
| nodetool sethintedhandoffthrottlekb | Устанавливает дросселирование hintedhdnoff передачи в килобайтах в секунду для каждого потока доставки |
| nodetool setmaxhintwindow | Устанавливает указанное максимальное окно подсказки в мс |
| nodetool statushandoff | Статус сохранения будущих подсказок на текущем узле |
| nodetool truncatehints | Обрезает все подсказки на локальном узле или обрезает подсказки для указанных конечных точек |

Ускорение воспроизведения подсказок во время выполнения

Дроссель передачи обслуживания по умолчанию 1024 kbps является консервативным для большинства современных сетей, и вполне возможно, что при простом перезапуске узла вы можете накопить много гигабайтных подсказок, воспроизведение которых может занять несколько часов. Например, если вы загружаете 100 Mbps данных на узел, один перезапуск продолжительностью 10 минут создаст 10 minutes \* (100 megabit / second) ~= 7 GiB данных, которые при (1024 KiB / second) будут занимать 7.5 GiB / (1024 KiB / second) = 2.03 hours воспроизведения. Точная математика зависит от стратегии балансировки нагрузки (циклический перебор лучше, чем осведомленность о токенах), количества токенов на узел (больше токенов лучше, чем меньше) и, естественно, скорости записи кластера, но независимо от того, вы можете захотеть увеличить дроссель во время выполнения.

Если вы оказались в такой ситуации, вы можете подумать о динамическом подъёме hinted\_handoff\_throttle с помощью команды nodetool sethintedhandoffthrottlekb.

Разрешение узлу дольше оставаться в нерабочем состоянии во время выполнения (Allow a Node to be Down Longer at Runtime)

Иногда узел может быть недоступен дольше обычного max\_hint\_window\_in\_ms (по умолчанию - три часа), но оборудование и сами данные по-прежнему будут доступны. В таком случае вы можете рассмотреть возможность динамического повышения max\_hint\_window\_in\_ms с помощью команды nodetool setmaxhintwindow. Это проинструктирует RT.KeyValue продолжать удерживать подсказки для неработающей конечной точки в течение более длительного периода времени.

Эта команда должна применяться ко всем узлам кластера, которые могут содержать подсказки. При необходимости настройку можно применить постоянно, установив параметр max\_hint\_window\_in\_ms в cassandra.yaml с последующим непрерывным перезапуском.

Мониторинг доставки подсказок

В RT.KeyValue добавлены гистограммы, позволяющие понять, сколько времени требуется для доставки подсказок, что полезно для операторов, чтобы лучше идентифицировать проблемы.

Также доступны метрики для отслеживания метрик Hinted Handoff <handoff-metrics> и Hints Service <hintsservice-metrics>.

Восстановление

RT.KeyValue остается доступной, несмотря на то, что один из ее узлов не работает или недоступен. Однако, когда узел не работает или недоступен, ему необходимо в конечном итоге обнаружить пропущенные записи. Подсказки пытаются проинформировать узел о пропущенных записях, хоть это лучшая попытка, и нет гарантии, что они сообщат узлу о 100% пропущенных операций записи. Эти несоответствия могут в конечном итоге привести к потере данных при замене узлов или истечении срока действия надгробий.

Эти несоответствия устраняются в процессе восстановления. Восстановление синхронизирует данные между узлами, сравнивая их соответствующие наборы данных для их общих диапазонов токенов и потоковую передачу различий для любых несинхронизированных разделов между узлами. Сравнивает данные с деревьями Меркла, которые представляют собой иерархию хешей.

Инкрементальное и полное восстановление

Есть 2 вида восстановления: полное и инкрементальное. Полное восстановление выполняется для всех данных в восстанавливаемом диапазоне токенов. Инкрементальное восстановление восстанавливает только те данные, которые были записаны после предыдущего инкрементального восстановления.

Инкрементальное восстановление является типом восстановления по умолчанию, и при регулярном выполнении может значительно сократить время и затраты на выполнение восстановления. Однако важно понимать, что после того, как инкрементальное восстановление пометит данные как восстановленные, он не будет пытаться восстановить их снова. Это нормально для синхронизации пропущенных записей, но не защищает от таких вещей, как повреждение диска, потеря данных из-за ошибки оператора или ошибки в RT.KeyValue. По этой причине следует время от времени проводить полное восстановление.

Использование и лучшие практики

Так как восстановление может привести к увеличению нагрузки на диск и сеть, RT.KeyValue не запускает его автоматически. Он запускается оператором через nodetool.

По умолчанию используется инкрементное восстановление, которое запускается с помощью следующей команды:

|  |
| --- |
| nodetool repair |

Полное восстановление можно запустить с помощью следующей команды:

|  |
| --- |
| nodetool repair --full |

Кроме того, восстановление можно запустить на одном пространстве ключей:

|  |
| --- |
| nodetool repair [options] <keyspace\_name> |

Или даже на конкретных таблицах:

|  |
| --- |
| nodetool repair [options] <keyspace\_name> <table1> <table2> |

Команда восстановления восстанавливает только диапазоны токенов на восстанавливаемом узле, она не восстанавливает весь кластер. По умолчанию восстановление будет выполняться для всех диапазонов токенов, реплицированных узлом, на котором выполняется восстановление, что приведет к дублированию работы, если вы запустите его на каждом узле. Флаг -pr восстанавливает только «первичные» диапазоны на узле, поэтому вы можете восстановить весь кластер, запустив nodetool repair -pr на каждом узле в одном центре обработки данных.

Конкретная частота восстановления, подходящая для вашего кластера, зависит от нескольких факторов. Однако, если вы только начинаете и ищете, с чего начать, возможно, будет разумным проводить инкрементальное восстановление каждые 1-3 дня и полное восстановление каждые 1-3 недели. Если вы не хотите проводить инкрементальное восстановление, лучше всего начать с полного восстановления каждые 5 дней.

Как минимум, восстановление следует запускать достаточно часто, чтобы период отсрочки gc никогда не истек для неисправленных данных. В противном случае удаленные данные могут появиться снова. При льготном периоде gc по умолчанию, равном 10 дням, восстановление каждого узла в вашем кластере не реже одного раза в 7 дней предотвратит это, обеспечивая при этом достаточный резерв, чтобы учесть задержки.

Другие параметры

* -pr, --partitioner-range

Ограничивает восстановление "первичным" диапазоном токенов востанавливаемого узла. Первичный диапазон - это просто диапазон токенов, для которого узел является первой репликой в кольце.

* -prv, --preview

Оценивает объем потоковой передачи, который может произойти для данной команды восстановления. Это строит деревья Меркла и выводит ожидаемую потоковую активность, но фактически не выполняет потоковую передачу. По умолчанию оцениваются инкрементальные восстановления, добавьте флаг --full, чтобы оценить полное восстановление.

* -vd, --validate

Проверяет, что восстановленные данные одинаковы на всех узлах. Подобно --preview, он строит и сравнивает деревья Меркла восстановленных данных, но не выполняет потоковую передачу. Это полезно для устранения неполадок. Если это показывает, что восстановленные данные не синхронизированы, следует запустить полное восстановление.

Пример полного восстановления

Полное восстановление обычно требуется для перераспределения данных после увеличения коэффициента репликации пространства ключей или после добавления узла в кластер. Полное восстановление предполагает потоковую передачу SSTables. Чтобы продемонстрировать полное восстановление, начните с трехузлового кластера.

|  |
| --- |
| [ec2-user@ip-10-0-2-238 ~]$ nodetool status  Datacenter: us-east-1  =====================  Status=Up/Down  |/ State=Normal/Leaving/Joining/Moving  -- Address Load Tokens Owns Host ID Rack  UN 10.0.1.115 547 KiB 256 ? b64cb32a-b32a-46b4-9eeb-e123fa8fc287 us-east-1b  UN 10.0.3.206 617.91 KiB 256 ? 74863177-684b-45f4-99f7-d1006625dc9e us-east-1d  UN 10.0.2.238 670.26 KiB 256 ? 4dcdadd2-41f9-4f34-9892-1f20868b27c7 us-east-1c |

Создайте пространство ключей с коэффициентом репликации 3:

|  |
| --- |
| cqlsh> DROP KEYSPACE cqlkeyspace;  cqlsh> CREATE KEYSPACE CQLKeyspace  ... WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3}; |

Добавьте таблицу в пространство ключей:

|  |
| --- |
| cqlsh> use cqlkeyspace;  cqlsh:cqlkeyspace> CREATE TABLE t (  ... id int,  ... k int,  ... v text,  ... PRIMARY KEY (id)  ... ); |

Добавьте данные таблицы:

|  |
| --- |
| cqlsh:cqlkeyspace> INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');  cqlsh:cqlkeyspace> INSERT INTO t (id, k, v) VALUES (1, 1, 'val1');  cqlsh:cqlkeyspace> INSERT INTO t (id, k, v) VALUES (2, 2, 'val2'); |

В запросе перечислены добавленные данные:

|  |
| --- |
| cqlsh:cqlkeyspace> SELECT \* FROM t;  id | k | v  ----+---+------  1 | 1 | val1  0 | 0 | val0  2 | 2 | val2  (3 rows) |

Внесите следующие изменения в трехузловой кластер:

* Увеличьте коэффициент репликации с 3 до 4.
* Добавьте 4-й узел в кластер.

При увеличении коэффициента репликации выводится следующее сообщение, указывающее, что требуется полное восстановление согласно:

|  |
| --- |
| cqlsh:cqlkeyspace> ALTER KEYSPACE CQLKeyspace  ... WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 4};  Warnings :  When increasing replication factor you need to run a full (-full) repair to distribute the  data. |

Выполните полное восстановление таблицы cqlkeyspace пространства ключей t с помощью следующей команды:

|  |
| --- |
| nodetool repair -full cqlkeyspace t |

Полное восстановление завершается примерно через секунду, как указано в выводе:

|  |
| --- |
| [ec2-user@ip-10-0-2-238 ~]$ nodetool repair -full cqlkeyspace t  [2019-08-17 03:06:21,445] Starting repair command #1 (fd576da0-c09b-11e9-b00c-1520e8c38f00), repairing keyspace cqlkeyspace with repair options (parallelism: parallel, primary range: false, incremental: false, job threads: 1, ColumnFamilies: [t], dataCenters: [], hosts: [], previewKind: NONE, # of ranges: 1024, pull repair: false, force repair: false, optimise streams: false)  [2019-08-17 03:06:23,059] Repair session fd8e5c20-c09b-11e9-b00c-1520e8c38f00 for range [(-8792657144775336505,-8786320730900698730], (-5454146041421260303,-5439402053041523135], (4288357893651763201,4324309707046452322], ... , (4350676211955643098,4351706629422088296]] finished (progress: 0%)  [2019-08-17 03:06:23,077] Repair completed successfully  [2019-08-17 03:06:23,077] Repair command #1 finished in 1 second  [ec2-user@ip-10-0-2-238 ~]$ |

Команда nodetool tpstats должна отображать завершенный ремонт как Repair-Task> Completed, значение столбца 1:

|  |
| --- |
| [ec2-user@ip-10-0-2-238 ~]$ nodetool tpstats  Pool Name Active Pending Completed Blocked All time blocked  ReadStage 0 0 99 0 0  …  Repair-Task 0 0 1 0 0  RequestResponseStage 0 0 2078 0 0 |

Восстановление при чтении (Read repair)

Восстановление при чтении - это процесс восстановления реплик данных во время запроса на чтение. Если все реплики, участвующие в запросе чтения на заданном уровне согласованности чтения, согласованы, данные возвращаются клиенту, и восстановление чтения не требуется. Но если реплики, участвующие в запросе на чтение на данном уровне согласованности, не согласованы, выполняется восстановление при чтении, чтобы сделать реплики, участвующие в запросе чтения, согласованными. Клиенту возвращаются самые свежие данные. Восстановление при чтении выполняется на переднем плане и блокируется тем, что ответ не возвращается клиенту до тех пор, пока восстановление при чтении не будет завершено и не будут созданы актуальные данные.

Ожидание монотонного чтения кворума

RT.KeyValue использует блокирующее восстановление чтения, чтобы гарантировать ожидание «монотонных чтений кворума», т.е. что при 2 последовательных чтениях кворума гарантируется, что 2 не получит что-то более старое, чем 1, и это даже в том случае, если неудачная запись кворума сделала запись самого актуального значения только для меньшинства реплик. "Кворум" означает большинство узлов среди реплик.

Конфигурация монотонного чтения на уровне таблицы

В RT.KeyValue представлена поддержка конфигурации монотонных чтений на уровне таблицы. В схему таблицы добавлена опция read\_repair table с блокировкой (blocking) опций (по умолчанию) и без нее (none).

Параметр read\_repair настраивает поведение восстановления при чтении, позволяя настраивать различные параметры производительности и согласованности. Поведение восстановления при чтении влияет на два свойства согласованности.

* Монотонное чтение кворума: обеспечивается BLOCKING. Монотонное чтение кворума не позволяет при некоторых обстоятельствах возвращаться назад во времени. Когда монотонное чтение кворума не обеспечивается и запись не достигает кворума реплик, она может быть видна при одном чтении, а затем исчезнуть при последующем чтении.
* Атомарность записи: предоставлено NONE. Атомарность записи не позволяет при чтении возвращать частично примененные записи. RT.KeyValue пытается обеспечить атомарность записи на уровне разделов, но поскольку при восстановлении чтения восстанавливаются только данные, охватываемые оператором SELECT, восстановление при чтении может нарушить атомарность записи, когда данные читаются на более детальном уровне, чем они записываются. Например, восстановление при чтении может нарушить атомарность записи, если вы записываете несколько строк в кластерный раздел в пакете, но затем выбираете одну строку, указав столбец кластеризации в операторе SELECT.

Blocking

Настройка по умолчанию. Когда read\_repair установлен в BLOCKING и начато восстановление чтения, чтение будет блокироваться при записи, отправляемой в другие реплики, до тех пор, пока записи не достигнут CL. Обеспечивает монотонное чтение кворума, но не атомарность записи на уровне раздела.

None

Когда read\_repair установлен в NONE, координатор будет согласовывать любые различия между репликами, но не будет пытаться их исправить. Обеспечивает атомарность записи на уровне раздела, но не монотонное чтение кворума.

Пример использования параметра NONE для параметра read\_repair следующий:

|  |
| --- |
| CREATE TABLE ks.tbl (k INT, c INT, v INT, PRIMARY KEY (k,c)) with read\_repair='NONE'"); |

Пример восстановления при чтении

Чтобы проиллюстрировать восстановление чтения на примере, представьте, что клиент отправляет запрос чтения с уровнем согласованности чтения ДВА в 5-узловой кластер, как показано на Рисунке 2. Уровень согласованности чтения определяет, сколько узлов реплик должны вернуть ответ, прежде чем запрос чтения будет считается успешным.

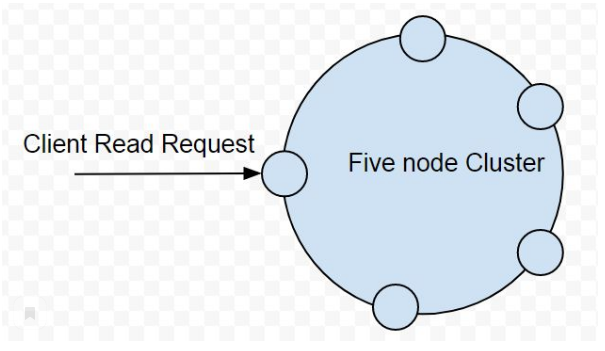


Рисунок 2. Клиент отправляет запрос на чтение в кластер из 5 узлов

Три узла размещают реплики для запрошенных данных, как показано на Рисунке 3. При уровне согласованности чтения TWO два узла реплики должны возвращать ответ, чтобы запрос чтения считался успешным. Если узел, на который клиент отправляет запрос хостам, реплика запрошенных данных должна быть отправлена только одному другому узлу-реплике. Но если принимающий узел не размещает реплику для запрошенных данных, узел становится узлом-координатором и пересылает запрос чтения на узел, на котором размещена реплика. Запрос прямого чтения направляется самому быстрому узлу (как определено динамическим снитчем), как показано на Рисунке 3. Запрос прямого чтения - это полное чтение, возвращающее запрошенные данные.

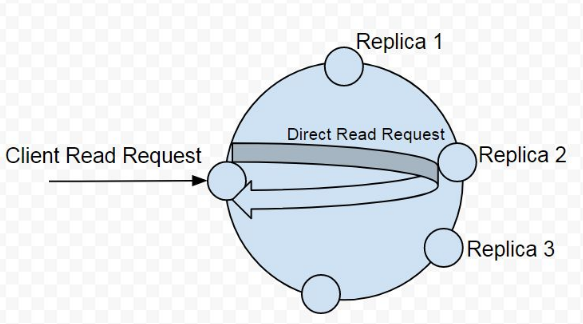


Рисунок 3. Запрос прямого чтения, отправленный на узел самой быстрой реплики

Затем узел-координатор отправляет необходимое количество дополнительных запросов для удовлетворения уровня согласованности, который равен TWO. Узел-координатор должен отправить еще один запрос на чтение, всего два. Все запросы на чтение, дополнительные к первому прямому запросу на чтение, являются дайджест-запросами на чтение. Запрос на чтение дайджеста не является полным чтением и возвращает только хеш-значение данных. Возвращается только хеш-значение, чтобы уменьшить сетевой трафик данных. В обсуждаемом примере узел-координатор отправляет один запрос на чтение дайджеста узлу, на котором размещена реплика, как показано на Рисунке 4.

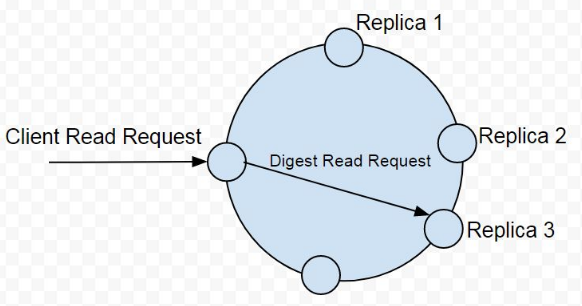


Рисунок 4. Координатор отправляет запрос на чтение дайджеста

Узел-координатор получил полную копию данных от одного узла и хеш-значение для данных от другого узла. Для сравнения возвращаемых данных для полной копии данных рассчитывается хеш-значение. Два значения хеш-функции сравниваются. Если хеш-значения совпадают, восстановление чтения не требуется, и клиенту возвращается полная копия запрошенных данных. Узел-координатор выполнил всего два запроса на чтение реплики, потому что в этом примере уровень согласованности чтения равен TWO. Если бы уровень согласованности был выше, например THREE, три узла реплики должны были бы ответить на запрос чтения, и только если бы все значения дайджеста или хеш-функции совпадали с хеш-значением полной копии данных, запрос на чтение считался бы успешным и данные возвращаются клиенту.

Но если значение хеш-функции из запроса на чтение дайджеста не совпадает с хеш-значением данных из запроса полного чтения первого узла реплики, это означает, что существует несогласованность в репликах. Чтобы исправить несоответствие, выполняется восстановление чтения.

Например, предположим, что этот дайджест-запрос возвращает хеш-значение, которое не совпадает с хеш-значением данных из прямого запроса полного чтения. Нам нужно будет сделать реплики согласованными, для которых узел-координатор отправляет прямой (полный) запрос на чтение узлу реплики, которому он ранее отправил дайджест-запрос на чтение, как показано на Рисунке 5.

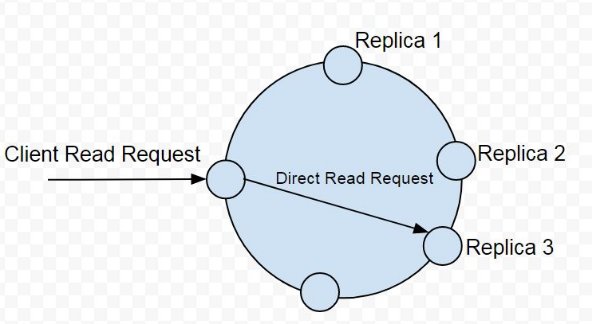


Рисунок 5. Координатор отправляет запрос прямого чтения на узел-реплику, которому он отправил дайджест-запрос чтения

После получения данных от второго узла-реплики координатор получает данные от двух узлов-реплик. Требуются только две реплики, так как в этом примере уровень согласованности чтения равен TWO. Данные из двух реплик сравниваются и на основе временных меток выбирается самая последняя реплика. Может потребоваться объединение данных для создания актуальной копии данных, если одна реплика содержит данные только для некоторых столбцов. В этом примере, если обнаруживается, что данные из первого запроса прямого чтения устарели, а данные из второго запроса полного чтения являются последними, необходимо выполнить восстановление реплики 2. Если новая обновленная версия данные создаются путем слияния двух реплик, для которых потребуется восстановление при чтении на обеих задействованных репликах. Например, восстановление при чтении выполняется для реплики 2, как показано на Рисунке 6.

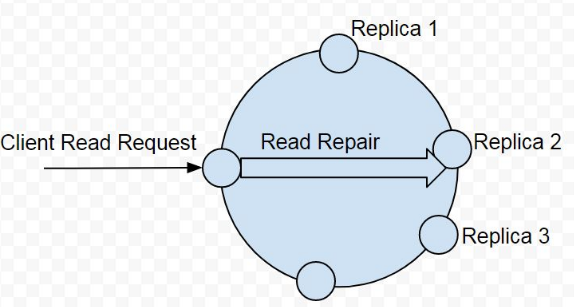


Рисунок 6. Координатор выполняет восстановление при чтении.

Самые свежие данные возвращаются клиенту, как показано на Рисунке 7. Из трех реплик Реплика 1 даже не считывается и, следовательно, не восстанавливается. Реплика 2 восстановлена. Реплика 3 является самой последней и возвращена клиенту.

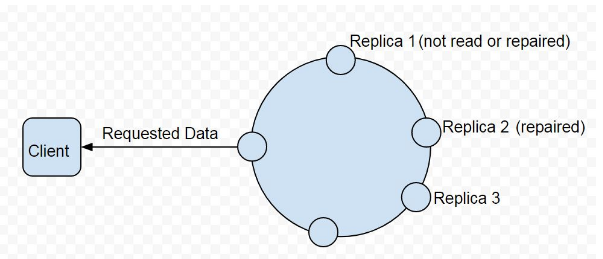


Рисунок 7. Самые свежие данные, возвращаемые клиенту

Уровень согласованности чтения и Восстановление при чтении

Последовательность чтения наиболее важна при определении необходимости исправления чтения. Как показано в Таблице 3. Восстановление при чтении не требуется для всех уровней согласованности.

**Таблица 3. Восстановление при чтении на основе уровня согласованности чтения**

|  |  |
| --- | --- |
| **Уровень согласованности чтения** | **Описание** |
| ONE | Восстановление при чтении не выполняется, поскольку данные из первого запроса прямого чтения удовлетворяют уровню согласованности ONE. Для поиска несоответствий в данных запросы на чтение дайджеста не используются. |
| TWO | Восстановление при чтении выполняется, если обнаруживаются несоответствия в данных, определяемые запросами прямого и дайджест-чтения. |
| THREE | Восстановление при чтении выполняется, если обнаруживаются несоответствия в данных, определяемые запросами прямого и дайджест-чтения. |
| LOCAL\_ONE | Восстановление при чтении не выполняется, поскольку данные из запроса прямого чтения от ближайшей реплики удовлетворяют уровню согласованности LOCAL\_ONE. Для поиска несоответствий в данных не используются дайджест-запросы чтения. |
| LOCAL\_QUORUM | Восстановление при чтении выполняется, если обнаруживаются несоответствия в данных, определяемые запросами прямого и дайджест-чтения. |
| QUORUM | Восстановление при чтении выполняется, если обнаруживаются несоответствия в данных, определяемые запросами прямого и дайджест-чтения. |

Если выполняется восстановление при чтении, оно выполняется только на устаревших репликах, участвующих в запросе на чтение. Количество реплик, задействованных в запросе на чтение, будет зависеть от уровня согласованности чтения (в примере это два).

Диагностические события для восстановления при чтении.

RT.KeyValue добавляет диагностические события для восстановления чтении, которые можно использовать для раскрытия такой информации, как:

* Связанные конечные точки.
* Дайджест ответов по конечным точкам.
* Затронутые ключи разделов.
* Предполагаемые чтения / записи.
* Обновление негабаритных.

Background восстановления при чтении

Восстановление при чтении не является альтернативой другим видам восстановления, таким как полное восстановление или замена узла, который продолжает выходить из строя. Данные, возвращаемые даже после выполнения восстановления при чтении, могут быть не самыми актуальными данными, если уровень согласованности отличается от уровня, требующего ответа от всех реплик.

Резервные копии (Backups)

RT.KeyValue хранит данные в неизменяемых файлах SSTable. Резервные копии в базе данных RT.KeyValue - это резервные копии данных базы данных, которые хранятся в виде файлов SSTable. Резервные копии используются для нескольких целей, включая следующие:

* Хранить копию данных для долговечности.
* Чтобы иметь возможность восстановить таблицу, если данные таблицы потеряны из-за сбоя узла / раздела / сети.
* Чтобы иметь возможность переносить файлы SSTable на другой компьютер; для портативности.

Типы резервных копий

RT.KeyValue поддерживает два типа стратегий резервного копирования:

* Снапшот (snapshots).
* Инкрементальные резервные копии (incremental backups).

Снапшот - это копия файлов таблицы SSTable в определенный момент времени, созданная с помощью жестких ссылок. DDL для создания таблицы также сохраняется. Снапшоты могут быть созданы пользователем или созданы автоматически. Параметр snapshot\_before\_compaction в файле cassandra.yaml определяет, создаются ли снапшоты перед каждым уплотнением. По умолчанию для snapshot\_before\_compaction установлено значение false. Снапшоты могут быть созданы автоматически перед усечением пространства ключей или удалением таблицы, установив для auto\_snapshot значение true (по умолчанию) в cassandra.yaml. Усечения могут быть отложены из-за автоматических снапшотов, а другой параметр в cassandra.yaml определяет, как долго координатор должен ждать завершения усечения. По умолчанию RT.KeyValue ждет 60 секунд для завершения автоматического создания снапшотов.

Инкрементальное резервное копирование - это копия файлов SSTable таблицы, созданная жесткой связью, когда memtables сбрасывается на диск как SSTables. Обычно инкрементальные резервные копии объединяются со снапшотами, чтобы сократить время резервного копирования, а также дисковое пространство. Добавочные резервные копии не включены по умолчанию и должны быть включены явно в cassandra.yaml (с настройкой incremental\_backups) или с помощью nodetool. После включения RT.KeyValue создает жесткую ссылку на каждую SSTable, сбрасываемую или передаваемую локально, в подкаталоге резервных копий / данных пространства ключей. Также создаются инкрементальные резервные копии системных таблиц.

Структура каталога данных

Структура каталогов данных RT.KeyValue состоит из разных каталогов для пространств ключей и таблиц с файлами данных в каталогах таблиц. Резервные копии каталогов и моментальные снапшоты для хранения резервных копий и моментальных снапшотов, соответственно, для конкретной таблицы также хранятся в каталоге таблиц. Структура каталогов для RT.KeyValue проиллюстрирована на Рисунке 8.

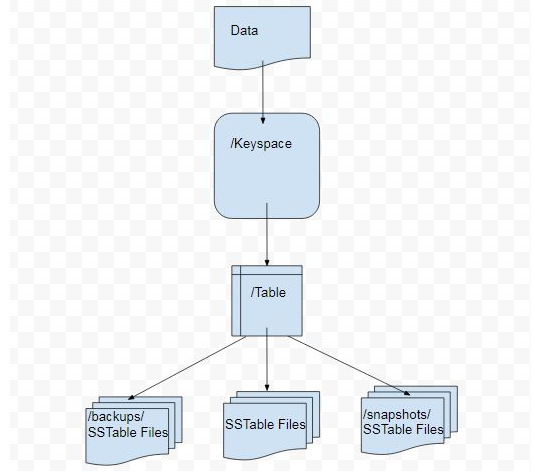


Рисунок 8. Структура каталогов для данных RT.KeyValue

Настройка примеров таблиц для резервных копий и снапшотов

В этом разделе мы создадим несколько примеров данных, которые можно использовать для демонстрации инкрементных резервных копий и моментальных снапшотов. Мы использовали трехузловой кластер RT.KeyValue. Сначала создаются пространства ключей. Затем в пространстве ключей создаются таблицы и добавляются данные таблицы. Мы использовали два пространства ключей cqlkeyspace и catalogkeyspace с двумя таблицами в каждом.

Создайте пространство ключей cqlkeyspace:

|  |
| --- |
| CREATE KEYSPACE cqlkeyspace  WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3}; |

Создайте две таблицы t и t2 в пространстве ключей cqlkeyspace.

|  |
| --- |
| USE cqlkeyspace;  CREATE TABLE t (  id int,  k int,  v text,  PRIMARY KEY (id)  );  CREATE TABLE t2 (  id int,  k int,  v text,  PRIMARY KEY (id)  ); |

Добавьте данные в таблицы:

|  |
| --- |
| INSERT INTO t (id, k, v) VALUES (0, 0, 'val0');  INSERT INTO t (id, k, v) VALUES (1, 1, 'val1');  INSERT INTO t2 (id, k, v) VALUES (0, 0, 'val0');  INSERT INTO t2 (id, k, v) VALUES (1, 1, 'val1');  INSERT INTO t2 (id, k, v) VALUES (2, 2, 'val2'); |

Запросите таблицу, чтобы получить список данных:

|  |
| --- |
| SELECT \* FROM t;  SELECT \* FROM t2; |

приводит к:

|  |
| --- |
| id | k | v  ----+---+------  1 | 1 | val1  0 | 0 | val0  (2 rows)  id | k | v  ----+---+------  1 | 1 | val1  0 | 0 | val0  2 | 2 | val2  (3 rows) |

Создайте второй catalogkeyspace:

|  |
| --- |
| CREATE KEYSPACE catalogkeyspace  WITH replication = {'class': 'SimpleStrategy', 'replication\_factor' : 3}; |

Создайте журнал (journal) с двумя таблицами и журнал (magazine) в catalogkeyspace:

|  |
| --- |
| USE catalogkeyspace;  CREATE TABLE journal (  id int,  name text,  publisher text,  PRIMARY KEY (id)  );  CREATE TABLE magazine (  id int,  name text,  publisher text,  PRIMARY KEY (id)  ); |

Добавьте данные в таблицы:

|  |
| --- |
| INSERT INTO journal (id, name, publisher) VALUES (0, 'Apache Cassandra Magazine', 'Apache Cassandra');  INSERT INTO journal (id, name, publisher) VALUES (1, 'Couchbase Magazine', 'Couchbase');  INSERT INTO magazine (id, name, publisher) VALUES (0, 'Apache Cassandra Magazine', 'Apache Cassandra');  INSERT INTO magazine (id, name, publisher) VALUES (1, 'Couchbase Magazine', 'Couchbase'); |

Запросите таблицы, чтобы получить список данных:

|  |
| --- |
| SELECT \* FROM catalogkeyspace.journal;  SELECT \* FROM catalogkeyspace.magazine; |

приводит к:

|  |
| --- |
| id | name | publisher  ----+---------------------------+------------------  1 | Couchbase Magazine | Couchbase  0 | Apache Cassandra Magazine | Apache Cassandra  (2 rows)  id | name | publisher  ----+---------------------------+------------------  1 | Couchbase Magazine | Couchbase  0 | Apache Cassandra Magazine | Apache Cassandra  (2 rows) |

Снапшоты

В этом разделе мы демонстрируем создание снапшотов. Команда, используемая для создания снапшота, представляет собой nodetool snapshot с использованием:

|  |
| --- |
| $ nodetool help snapshot |

приводит к:

|  |
| --- |
| NAME  nodetool snapshot - Take a snapshot of specified keyspaces or a snapshot  of the specified table  SYNOPSIS  nodetool [(-h <host> | --host <host>)] [(-p <port> | --port <port>)]  [(-pp | --print-port)] [(-pw <password> | --password <password>)]  [(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]  [(-u <username> | --username <username>)] snapshot  [(-cf <table> | --column-family <table> | --table <table>)]  [(-kt <ktlist> | --kt-list <ktlist> | -kc <ktlist> | --kc.list <ktlist>)]  [(-sf | --skip-flush)] [(-t <tag> | --tag <tag>)] [--] [<keyspaces...>]  OPTIONS  -cf <table>, --column-family <table>, --table <table>  The table name (you must specify one and only one keyspace for using  this option)  -h <host>, --host <host>  Node hostname or ip address  -kt <ktlist>, --kt-list <ktlist>, -kc <ktlist>, --kc.list <ktlist>  The list of Keyspace.table to take snapshot.(you must not specify  only keyspace)  -p <port>, --port <port>  Remote jmx agent port number  -pp, --print-port  Operate in 4.0 mode with hosts disambiguated by port number  -pw <password>, --password <password>  Remote jmx agent password  -pwf <passwordFilePath>, --password-file <passwordFilePath>  Path to the JMX password file  -sf, --skip-flush  Do not flush memtables before snapshotting (snapshot will not  contain unflushed data)  -t <tag>, --tag <tag>  The name of the snapshot  -u <username>, --username <username>  Remote jmx agent username  --  This option can be used to separate command-line options from the  list of argument, (useful when arguments might be mistaken for  command-line options  [<keyspaces...>]  List of keyspaces. By default, all keyspaces |

Настройка снапшотов

Чтобы продемонстрировать создание снапшотов с помощью Nodetool в командной строке, мы установили для параметра auto\_snapshots значение false в файле cassandra.yaml:

|  |
| --- |
| auto\_snapshot: false |

Также установите для snapshot\_before\_compaction значение false, чтобы отключить автоматическое создание снапшотов перед сжатием:

|  |
| --- |
| snapshot\_before\_compaction: false |

Создание снапшотов

Прежде чем создавать какие-либо снапшоты, выполните поиск снапшотов:

|  |
| --- |
| $ find -name snapshots |

Мы будем использовать примеры пространств ключей и таблиц для создания снапшотов.

Создание снапшотов всех таблиц в пространстве ключей

Используя приведенный выше синтаксис, создайте моментальный снапшот с именем catalog-ks для всех таблиц в пространстве ключей catalogkeyspace:

|  |
| --- |
| $ nodetool snapshot --tag catalog-ks catalogkeyspace |

приводит к:

|  |
| --- |
| Requested creating snapshot(s) for [catalogkeyspace] with snapshot name [catalog-ks] and  options {skipFlush=false}  Snapshot directory: catalog-ks |

Используя приведенную выше команду find, можно найти каталоги моментальных снапшотов и snapshots с перечисленными файлами, подобными следующим:

|  |
| --- |
| ./cassandra/data/data/catalogkeyspace/journal-296a2d30c22a11e9b1350d927649052c/snapshots  ./cassandra/data/data/catalogkeyspace/magazine-446eae30c22a11e9b1350d927649052c/snapshots |

Аналогичным образом можно создавать снапшоты всех таблиц в нескольких пространствах ключей:

|  |
| --- |
| $ nodetool snapshot --tag catalog-cql-ks catalogkeyspace, cqlkeyspace |

Создание снапшотов одной таблицы в пространстве ключей

Чтобы сделать снапшот одной таблицы, синтаксис команды nodetool snapshot выглядит следующим образом:

|  |
| --- |
| $ nodetool snapshot --tag <tag> --table <table> --<keyspace> |

Используя приведенный выше синтаксис, создайте снапшот журнала таблиц (magazine) в пространстве ключей catalogkeyspace:

|  |
| --- |
| $ nodetool snapshot --tag magazine --table magazine catalogkeyspace |

приводит к:

|  |
| --- |
| Requested creating snapshot(s) for [catalogkeyspace] with snapshot name [magazine] and  options {skipFlush=false}  Snapshot directory: magazine |

Создание снапшотов нескольких таблиц из одного и того же пространства ключей

Чтобы сделать снапшоты нескольких таблиц в пространстве ключей, список *Keyspace.table* должен быть указан с параметром --kt-list. Например, создайте снапшоты для таблиц t и t2 в пространстве ключей cqlkeyspace:

|  |
| --- |
| $ nodetool snapshot --kt-list cqlkeyspace.t,cqlkeyspace.t2 --tag multi-table |

приводит к:

|  |
| --- |
| Requested creating snapshot(s) for ["CQLKeyspace".t,"CQLKeyspace".t2] with snapshot name [multi-  table] and options {skipFlush=false}  Snapshot directory: multi-table |

Можно создать несколько снапшотов одного и того же набора таблиц и пометить их разными именами. В качестве примера создайте еще один снапшот для того же набора таблиц t и t2 в пространстве ключей cqlkeyspace и пометьте снапшоты по-разному:

|  |
| --- |
| $ nodetool snapshot --kt-list cqlkeyspace.t, cqlkeyspace.t2 --tag multi-table-2 |

приводит к:

|  |
| --- |
| Requested creating snapshot(s) for ["CQLKeyspace".t,"CQLKeyspace".t2] with snapshot name [multi-  table-2] and options {skipFlush=false}  Snapshot directory: multi-table-2 |

Создание снапшота нескольких таблиц из разных пространств ключей

Чтобы сделать снапшоты нескольких таблиц, находящихся в разных пространствах ключей, синтаксис команды такой же, как когда несколько таблиц находятся в одном пространстве ключей. Каждый <keyspace>. <table> необходимо указывать отдельно в параметре --kt-list.

Например, создайте снапшот для таблицы t в пространстве cqlkeyspace и журнал таблиц (journal) в пространстве ключей каталога и пометьте снапшот multi-ks.

|  |
| --- |
| $ nodetool snapshot --kt-list catalogkeyspace.journal,cqlkeyspace.t --tag multi-ks |

приводит к:

|  |
| --- |
| Requested creating snapshot(s) for [catalogkeyspace.journal,cqlkeyspace.t] with snapshot  name [multi-ks] and options {skipFlush=false}  Snapshot directory: multi-ks |

Список Моментальных снапшотов

Чтобы вывести список снапшотов, используйте команду nodetool listsnapshots. Перечисляются все снапшотов, которые мы создали в предыдущих примерах:

|  |
| --- |
| $ nodetool listsnapshots |

приводит к:

|  |
| --- |
| Snapshot Details:  Snapshot name Keyspace name Column family name True size Size on disk  multi-table cqlkeyspace t2 4.86 KiB 5.67 KiB  multi-table cqlkeyspace t 4.89 KiB 5.7 KiB  multi-ks cqlkeyspace t 4.89 KiB 5.7 KiB  multi-ks catalogkeyspace journal 4.9 KiB 5.73 KiB  magazine catalogkeyspace magazine 4.9 KiB 5.73 KiB  multi-table-2 cqlkeyspace t2 4.86 KiB 5.67 KiB  multi-table-2 cqlkeyspace t 4.89 KiB 5.7 KiB  catalog-ks catalogkeyspace journal 4.9 KiB 5.73 KiB  catalog-ks catalogkeyspace magazine 4.9 KiB 5.73 KiB  Total TrueDiskSpaceUsed: 44.02 KiB |

Поиск каталогов снапшотов

Каталоги snapshots могут быть перечислены с помощью команды find –name snapshots:

|  |
| --- |
| $ find -name snapshots |

приводит к:

|  |
| --- |
| ./cassandra/data/data/cqlkeyspace/t-d132e240c21711e9bbee19821dcea330/snapshots  ./cassandra/data/data/cqlkeyspace/t2-d993a390c22911e9b1350d927649052c/snapshots  ./cassandra/data/data/catalogkeyspace/journal-296a2d30c22a11e9b1350d927649052c/snapshots  ./cassandra/data/data/catalogkeyspace/magazine-446eae30c22a11e9b1350d927649052c/snapshots |

Чтобы вывести список снапшотов для конкретной таблицы, сначала перейдите в каталог снапшотов для этой таблицы. Например, перечислите снапшоты для таблицы catalogkeyspace/journal:

|  |
| --- |
| $ cd ./cassandra/data/data/catalogkeyspace/journal-296a2d30c22a11e9b1350d927649052c/snapshots && ls -l |

приводит к:

|  |
| --- |
| total 0  drwxrwxr-x. 2 ec2-user ec2-user 265 Aug 19 02:44 catalog-ks  drwxrwxr-x. 2 ec2-user ec2-user 265 Aug 19 02:52 multi-ks |

В каталоге моментальных снапшотов (snapshots) перечислены файлы SSTable в моментальном снапшоте. Файл schema.cql также создается в каждом снапшоте, который определяет схему, которая может воссоздать таблицу с помощью CQL при восстановлении из снапшота:

|  |
| --- |
| $ cd catalog-ks && ls -l |

приводит к:

|  |
| --- |
| total 44  -rw-rw-r--. 1 ec2-user ec2-user 31 Aug 19 02:44 manifest.jsonZ  -rw-rw-r--. 4 ec2-user ec2-user 47 Aug 19 02:38 na-1-big-CompressionInfo.db  -rw-rw-r--. 4 ec2-user ec2-user 97 Aug 19 02:38 na-1-big-Data.db  -rw-rw-r--. 4 ec2-user ec2-user 10 Aug 19 02:38 na-1-big-Digest.crc32  -rw-rw-r--. 4 ec2-user ec2-user 16 Aug 19 02:38 na-1-big-Filter.db  -rw-rw-r--. 4 ec2-user ec2-user 16 Aug 19 02:38 na-1-big-Index.db  -rw-rw-r--. 4 ec2-user ec2-user 4687 Aug 19 02:38 na-1-big-Statistics.db  -rw-rw-r--. 4 ec2-user ec2-user 56 Aug 19 02:38 na-1-big-Summary.db  -rw-rw-r--. 4 ec2-user ec2-user 92 Aug 19 02:38 na-1-big-TOC.txt  -rw-rw-r--. 1 ec2-user ec2-user 814 Aug 19 02:44 schema.cql |

Очистка моментальных снапшотов

Снапшоты можно очистить или удалить с помощью команды nodetool clearsnapshot. Необходимо указать либо конкретное имя снапшота, либо параметр –all.

Например, удалите снапшот с именем magazine из пространства ключей cqlkeyspace:

|  |
| --- |
| $ nodetool clearsnapshot -t magazine cqlkeyspace |

или удалите все снапшоты из cqlkeyspace с параметром –all:

|  |
| --- |
| $ nodetool clearsnapshot -all cqlkeyspace |

Инкрементальные резервные копии

В следующих разделах мы обсудим настройку и создание инкрементальных резервных копий (incremental backups).

Настройка инкрементального резервного копирования

Для создания инкрементальных резервных копий установите для incremental\_backups значение true в cassandra.yaml.

|  |
| --- |
| incremental\_backups: true |

Это единственная настройка, необходимая для создания инкрементальных резервных копий. По умолчанию для параметра incremental\_backups установлено значение false, поскольку для каждого сброса данных создается новый набор файлов SSTable, и если необходимо запустить несколько операторов CQL, каталог backups может быстро заполниться и израсходовать хранилище, необходимое для хранения данных таблицы. Добавочное резервное копирование также можно включить в командной строке с помощью команды nodetool enablebackup. Инкрементальное резервное копирование можно отключить с помощью команды nodetool disablebackup. Статус инкрементальных резервных копий, включены ли они, можно проверить с помощью nodetool statusbackup.

Создание инкрементальных резервных копий

После создания каждой таблицы удалите данные таблицы с помощью команды nodetool flush. Создаются инкрементные резервные копии.

|  |
| --- |
| $ nodetool flush cqlkeyspace t  $ nodetool flush cqlkeyspace t2  $ nodetool flush catalogkeyspace journal magazine |

Поиск инкрементальных резервных Копий

Инкрементальные резервные копии создаются в каталоге данных data RT.KeyValue в каталоге таблиц. Резервные копии можно найти с помощью следующей команды:

|  |
| --- |
| $ find -name backups |

приводит к:

|  |
| --- |
| ./cassandra/data/data/cqlkeyspace/t-d132e240c21711e9bbee19821dcea330/backups  ./cassandra/data/data/cqlkeyspace/t2-d993a390c22911e9b1350d927649052c/backups  ./cassandra/data/data/catalogkeyspace/journal-296a2d30c22a11e9b1350d927649052c/backups  ./cassandra/data/data/catalogkeyspace/magazine-446eae30c22a11e9b1350d927649052c/backups |

Создание инкрементальной резервной копии

В этом разделе более подробно обсуждается создание инкрементальных резервных копий с использованием ранее созданного пространства ключей и таблицы.

Очистите пространство ключей и таблицу:

|  |
| --- |
| $ nodetool flush cqlkeyspace t |

Поиск резервных копий и каталог backups отобразит каталог резервных копий, даже если мы еще не добавили никаких табличных данных.

|  |
| --- |
| $ find -name backups |

приводит к:

|  |
| --- |
| ./cassandra/data/data/cqlkeyspace/t-d132e240c21711e9bbee19821dcea330/backups |

Проверка каталога backups покажет, что файлов резервных копий также нет:

|  |
| --- |
| $ cd ./cassandra/data/data/cqlkeyspace/t-d132e240c21711e9bbee19821dcea330/backups && ls -l |

приводит к:

|  |
| --- |
| total 0 |

Если к данным добавляется строка данных, запуск команды nodetool flush очистит данные таблицы и будет создана инкрементальная резервная копия:

|  |
| --- |
| $ nodetool flush cqlkeyspace t  $ cd ./cassandra/data/data/cqlkeyspace/t-d132e240c21711e9bbee19821dcea330/backups && ls -l |

приводит к:

|  |
| --- |
| total 36  -rw-rw-r--. 2 ec2-user ec2-user 47 Aug 19 00:32 na-1-big-CompressionInfo.db  -rw-rw-r--. 2 ec2-user ec2-user 43 Aug 19 00:32 na-1-big-Data.db  -rw-rw-r--. 2 ec2-user ec2-user 10 Aug 19 00:32 na-1-big-Digest.crc32  -rw-rw-r--. 2 ec2-user ec2-user 16 Aug 19 00:32 na-1-big-Filter.db  -rw-rw-r--. 2 ec2-user ec2-user 8 Aug 19 00:32 na-1-big-Index.db  -rw-rw-r--. 2 ec2-user ec2-user 4673 Aug 19 00:32 na-1-big-Statistics.db  -rw-rw-r--. 2 ec2-user ec2-user 56 Aug 19 00:32 na-1-big-Summary.db  -rw-rw-r--. 2 ec2-user ec2-user 92 Aug 19 00:32 na-1-big-TOC.txt |

|  |
| --- |
| **Примечание.** |
| Каталог backups для любой таблицы, такой как cqlkeyspace/t, создается в каталоге data для этой таблицы. |

Добавление еще одной строки данных и очистка приведет к созданию еще одного набора файлов инкрементальных резервных копий. Файлы SSTable имеют метку времени, что отличает первую инкрементальную резервную копию от второй:

|  |
| --- |
| total 72  -rw-rw-r--. 2 ec2-user ec2-user 47 Aug 19 00:32 na-1-big-CompressionInfo.db  -rw-rw-r--. 2 ec2-user ec2-user 43 Aug 19 00:32 na-1-big-Data.db  -rw-rw-r--. 2 ec2-user ec2-user 10 Aug 19 00:32 na-1-big-Digest.crc32  -rw-rw-r--. 2 ec2-user ec2-user 16 Aug 19 00:32 na-1-big-Filter.db  -rw-rw-r--. 2 ec2-user ec2-user 8 Aug 19 00:32 na-1-big-Index.db  -rw-rw-r--. 2 ec2-user ec2-user 4673 Aug 19 00:32 na-1-big-Statistics.db  -rw-rw-r--. 2 ec2-user ec2-user 56 Aug 19 00:32 na-1-big-Summary.db  -rw-rw-r--. 2 ec2-user ec2-user 92 Aug 19 00:32 na-1-big-TOC.txt  -rw-rw-r--. 2 ec2-user ec2-user 47 Aug 19 00:35 na-2-big-CompressionInfo.db  -rw-rw-r--. 2 ec2-user ec2-user 41 Aug 19 00:35 na-2-big-Data.db  -rw-rw-r--. 2 ec2-user ec2-user 10 Aug 19 00:35 na-2-big-Digest.crc32  -rw-rw-r--. 2 ec2-user ec2-user 16 Aug 19 00:35 na-2-big-Filter.db  -rw-rw-r--. 2 ec2-user ec2-user 8 Aug 19 00:35 na-2-big-Index.db  -rw-rw-r--. 2 ec2-user ec2-user 4673 Aug 19 00:35 na-2-big-Statistics.db  -rw-rw-r--. 2 ec2-user ec2-user 56 Aug 19 00:35 na-2-big-Summary.db  -rw-rw-r--. 2 ec2-user ec2-user 92 Aug 19 00:35 na-2-big-TOC.txt |

Восстановление из инкрементальных резервных копий и моментальных снапшотов

Двумя основными инструментами / командами для восстановления таблицы после ее удаления являются:

* sstableloader
* nodetool import

Моментальный снапшот содержит по существу тот же набор файлов SSTable, что и инкрементальная резервная копия с несколькими дополнительными файлами. Снапшот включает файл schema.cql для DDL схемы для создания таблицы на CQL. Резервная копия таблицы не включает DDL, который должен быть получен из моментального снапшота при восстановлении из инкрементальной резервной копии.

Сжатие (compression)

RT.KeyValue предлагает операторам возможность настраивать сжатие для каждой таблицы. Сжатие уменьшает размер данных на диске за счет сжатия SSTable в настраиваемом пользователем сжатии chunk\_length\_in\_kb. Поскольку RT.KeyValue SSTables неизменяемы, затраты ЦП на сжатие необходимы только при записи SSTable - последующие обновления данных будут попадать в разные SSTables, поэтому RT.KeyValue не нужно будет распаковывать, перезаписывать и повторно сжимать данные при выполнении команд UPDATE. При чтении RT.KeyValue найдет соответствующие сжатые фрагменты на диске, распакует весь фрагмент, а затем продолжит выполнение оставшейся части пути чтения (слияние данных с дисков и таблиц memtables, восстановление чтения и т.д.).

Алгоритмы сжатия обычно сводятся к следующему:

* **Скорость сжатия**: насколько быстро алгоритм сжатия сжимает данные. Это очень важно для путей сброса и сжатия, поскольку данные должны быть сжаты перед записью на диск.
* **Скорость декомпрессии**: насколько быстро алгоритм сжатия распаковывает данные. Это критически важно для путей чтения и сжатия, поскольку данные должны быть прочитаны с диска целиком и распакованы, прежде чем их можно будет вернуть.
* **Коэффициент**: на какой коэффициент уменьшаются несжатые данные. RT.KeyValue обычно измеряет это как размер данных на диске относительно несжатого размера. Например, коэффициент 0.5 означает, что размер данных на диске составляет 50% от размера несжатых данных. RT.KeyValue предоставляет это соотношение для каждой таблицы как поле SSTable Compression Ratio в таблице nodetool tablestats.

RT.KeyValue по умолчанию предлагает пять алгоритмов сжатия, которые обеспечивают разные компромиссы в этих областях. Хотя сравнительный анализ алгоритмов сжатия зависит от многих факторов (параметров алгоритма, таких как уровень сжатия, сжимаемость входных данных, базовый класс процессора и т.д.), Следующая Таблица 4. должна помочь вам выбрать отправную точку на основе требований вашего приложения с чрезвычайно грубой оценкой различных вариантов по их производительности в этих областях (A относительно хорош, F относительно плох):

**Таблица 4. Алгоритмы сжатия**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритм сжатия** | **Класс** | **Сжатие** | **Декомпрессия** | **Соотношение** |
| LZ4 | LZ4Compressor | A+ | A+ | С+ |
| LZ4HC | LZ4Compressor | C+ | A+ | В+ |
| Zstd | ZstdCompressor | A- | A- | А+ |
| Snappy | SnappyCompressor | A- | A | С |
| Deflate (zlib) | DeflateCompressor | C | C | А |

Вообще говоря, для приложения, критичного к производительности (задержка или пропускная способность), LZ4 является правильным выбором, поскольку он обеспечивает отличное соотношение на затраченный цикл ЦП. Вот почему это выбор по умолчанию в RT.KeyValue.

Однако для приложений, критически важных для хранения (занимающих место на диске), Zstd может быть лучшим выбором, поскольку он может значительно улучшить LZ4.

Snappy сохранен для обратной совместимости, и LZ4, как правило, будет предпочтительнее.

Deflate сохраняется для обратной совместимости, и обычно предпочтительнее Zstd.

Уплотнение (Compaction)

Стратегии

Выбор правильной стратегии сжатия для вашей рабочей нагрузки обеспечит наилучшую производительность как для запросов, так и для самого сжатия.

Стратегия многоуровневого уплотнения по размеру (Size Tiered Compaction Stratagy, STCS)

Стратегия уплотнения по умолчанию. Полезно как запасной вариант, когда другие стратегии не подходят для рабочей нагрузки. Наиболее полезно для рабочих нагрузок, не являющихся чистыми временными рядами, с вращающимися дисками, или когда количество операций ввода-вывода из LCS слишком велико.

Стратегия выравнивания уплотнения (Leveled Compaction Strategy, LCS)

Стратегия выравнивания уплотнения (LCS) оптимизирована для тяжелых рабочих нагрузок чтения или рабочих нагрузок с большим количеством обновлений и удалений. Это не лучший выбор для неизменяемых данных временных рядов.

Стратегия сжатия временного окна (Time Window Compaction Strategy, TCS)

Стратегия сжатия временного окна разработана для данных с TTL, в основном неизменяемых временных рядов.

Виды уплотнения

Концепция уплотнения используется для различных типов операций в RT.KeyValue, общая черта этих операций заключается в том, что они принимают одну или несколько таблиц SSTables и выводят новые таблицы SSTables. Типы уплотнений:

* **Незначительное уплотнение**

срабатывает автоматически в RT.KeyValue.

* **Сильное уплотнение**

пользователь выполняет сжатие по всем SSTables на узле.

* **Определяемое пользователем уплотнение**

пользователь запускает уплотнение для данного набора SSTables.

* **Скраб**

попробуйте исправить любые сломанные SSTables. Это может фактически удалить действительные данные, если эти данные повреждены, в этом случае вам нужно будет выполнить полное восстановление на узле.

* **UpgradeSSTables**

обновите SSTables до последней версии. Запустите это после обновления до новой основной версии.

* **Очистка**

удалить любые диапазоны, которыми этот узел больше не владеет, обычно запускается на соседних узлах после того, как узел был загружен, поскольку этот узел будет владеть некоторыми диапазонами этих узлов.

* **Восстановление вторичного индекса**

перестроить вторичные индексы на узле.

* **Антиуплотнение**

после восстановления диапазоны, которые были фактически восстановлены, отделяются от SSTables, которые существовали на момент начала ремонта.

* **Уплотнение поддиапазона**

Можно сжать только данный поддиапазон - это может быть полезно, если вы знаете, что токен ведет себя некорректно - либо собирает много обновлений, либо много удаляет. (nodetool compact -st x -et y) выберет все SSTables, содержащие диапазон между x и y, и выполнит сжатие для этих SSTables. Для STCS это, скорее всего, будет включать все SSTables, но с LCS он может выполнить сжатие для подмножества SSTables. С LCS результирующий sstable окажется в L0.

Команды Compaction nodetool

Утилита nodetool <nodetool> предоставляет ряд команд, связанных с уплотнением:

enableautocompaction

Включить уплотнение.

disableautocompaction

Отключить уплотнение.

setcompactionthroughput

Максимальная скорость сжатия - по умолчанию 16 МБ / с, но имейте в виду, что достичь этой пропускной способности, скорее всего, невозможно.

compactionstats

Статистика о текущих и ожидаемых уплотнениях.

compactionhistory

Cсведения о последних уплотнениях.

setcompactionthreshold

Установите минимальный / максимальный стабильный счетчик, когда запускать уплотнение, по умолчанию 4/32.

Мониторинг

Метрики в RT.KeyValue управляются с помощью библиотеки метрик Dropwizard. Эти метрики можно запрашивать через JMX или передавать во внешние системы мониторинга с помощью ряда встроенных и сторонних плагинов репортера.

Метрики собираются для одного узла. Оператор может использовать внешнюю систему мониторинга для их агрегирования.

Типы метрик

Все показатели, сообщаемые RT.KeyValue, относятся к одному из следующих типов:

Gauge

Мгновенное измерение значения.

Counter

Датчик для экземпляра AtomicLong. Обычно это используется для отслеживания изменений с момента последнего вызова, чтобы увидеть, есть ли большое увеличение по сравнению с нормой.

Histogram

Измеряет статистическое распределение значений в потоке данных. + Помимо минимума, максимума, среднего и т.д., Он также измеряет медианный, 75-й, 90-й, 95-й, 98-й, 99-й и 99,9-й процентили.

Timer

Измеряет как частоту вызова определенного фрагмента кода, так и гистограмму его продолжительности.

Latency

Специальный тип, который отслеживает задержку (в микросекундах) с помощью таймера и счетчика, который отслеживает общую задержку, накопленную с момента запуска. Первый полезен, если вы отслеживаете изменение общей задержки с момента последней проверки. К каждому имени метрики этого типа будут добавлены «Задержка» и «Общая задержка».

Meter

Метрика счетчика, которая измеряет среднюю пропускную способность и пропускную способность скользящего среднего за одну, пять и пятнадцать минут, экспоненциально взвешенную.

Табличные метрики

Каждая таблица в RT.KeyValue имеет метрики, отвечающие за отслеживание ее состояния и производительности.

К именам показателей добавляются конкретные имена пространства ключей (Keyspace) и таблицы (Table).

Сообщаемый формат имени:

* Название показателя

org.apache.cassandra.metrics.Table.<MetricName>.<Keyspace>.<Table>

* JMX MBean

org.apache.cassandra.metrics:type=Table keyspace=<Keyspace> scope=<Table> name=<MetricName>

Метрик репортер

Метрики RT.KeyValue могут быть экспортированы в ряд систем мониторинга с помощью ряда встроенных и сторонних плагинов репортера.

Конфигурация этих плагинов управляется проектом конфигурации metrics Reporter. Пример файла конфигурации находится по conf/metrics-reporter-config-sample.yaml.

После настройки вы просто запускаете RT.KeyValue с флагом -Dcassandra.metricsReporterConfigFile=metrics-reporter-config.yaml. Указанный файл .yaml и любые сторонние jar-файлы репортеров должны находиться в пути к классам RT.KeyValue.

Массовая загрузка (Bulk loading)

Массовая загрузка данных RT.KeyValue поддерживается различными инструментами. Данные для массовой загрузки должны быть в форме SSTables. RT.KeyValue не поддерживает прямую загрузку данных в любом другом формате, таком как CSV, JSON и XML. Хотя команда cqlsh COPY может загружать данные CSV, это не лучший вариант для объемов данных. Массовая загрузка используется для:

* Восстановление инкрементальных резервных копий и снапшотов. Резервные копии и снапшоты уже представлены в виде файлов SSTables.
* Загрузите существующие SSTables в другой кластер. Данные могут иметь разное количество узлов или стратегию репликации.
* Загрузите внешние данные в кластер.

Инструменты для массовой загрузки

RT.KeyValue предоставляет две команды или инструменты для массовой загрузки данных:

* RT.KeyValue Bulk loader, также называемый sstableloader
* Команда nodetool import

nodetool import и sstableloader доступен, если каталог bin установки RT.KeyValue находится в переменной среде PATH. Или к ним можно получить доступ прямо из каталога bin. В примерах используются пространства ключей и таблицы, созданные в резервных копиях (backups).

Использование sstableloader

sstableloader - это основной инструмент для массовой загрузки данных. sstableloader передает файлы данных SSTable в работающий кластер в соответствии со стратегией репликации и фактором репликации. Таблица для загрузки данных не обязательно должна быть пустой.

Единственные требования для запуска sstableloader:

* Один или несколько начальных хостов, разделенных запятыми, для подключения и получения кольцевой информации.
* Путь к каталогу для загрузки SSTables.

|  |
| --- |
| sstableloader [options] <dir\_path> |

Sstableloader массово загружает SSTables, найденные в каталоге <dir\_path>, в настроенный кластер. <dir\_path> используется как имя целевого пространства ключей / таблицы. Например, чтобы загрузить файл SSTable с именем Standard1-g-1-Data.db в Keyspace1/Standard1, вам потребуются файлы Standard1-g-1-Data.db и Standard1-g-1-Index.db в папке /path/to/Keyspace1/Standard1/.

Использование nodetool import

Рекомендуется импортировать SSTables в таблицу с помощью команды nodetool import вместо устаревшей команды nodetool refresh. Команда nodetool import имеет возможность загружать новые таблицы SSTables из отдельного каталога.

Использование команды:

|  |
| --- |
| nodetool [(-h <host> | --host <host>)] [(-p <port> | --port <port>)]  [(-pp | --print-port)] [(-pw <password> | --password <password>)]  [(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]  [(-u <username> | --username <username>)] import  [(-c | --no-invalidate-caches)] [(-e | --extended-verify)]  [(-l | --keep-level)] [(-q | --quick)] [(-r | --keep-repaired)]  [(-t | --no-tokens)] [(-v | --no-verify)] [--] <keyspace> <table>  <directory> ... |

Необходимо указать пространство ключей аргументов, имя таблицы и каталог.

Массовая загрузка внешних данных

Массовая загрузка внешних данных напрямую не поддерживается ни одним из инструментов, которые мы обсуждали, включая sstableloader и nodetool import. Для nodetool import и sstableloader данные должны быть в форме SSTables. RT.KeyValue поддерживает Java API для создания SSTables из входных данных с использованием Java-класса org.apache.cassandra.io.sstable.CQLSSTableWriter. Впоследствии для массовой загрузки SSTables используется либо sstableloader, либо nodetool import.

Change Data Capture

Обзор

Система отслеживания измененных данных (CDC) предоставляет механизм для пометки определенных таблиц для архивирования, а также для отказа от записи в эти таблицы после достижения настраиваемого размера на диске для журнала CDC. Оператор может включить CDC для таблицы, установив свойство таблицы cdc=true (либо при создании таблицы <create-table-statement>, либо при изменении ее <alter-table-statement>). После создания CommitLogSegment жесткая ссылка на сегмент создается в каталоге, указанном в cassandra.yaml. В сегменте fsync на диске, если данные CDC присутствуют где-либо в сегменте, также создается файл <segment\_name> \_cdc.idx с целочисленным смещением того, сколько данных в исходном сегменте сохраняется на диске. После очистки последнего сегмента к файлу \_cdc.idx будет добавлена ​​вторая строка с понятным для человека словом «COMPLETED», указывающим, что RT.KeyValue завершила всю обработку файла.

Мы используем индексный файл, а не просто побуждаем клиентов анализировать журнал в реальном времени из дескриптора, отображаемого в память, поскольку данные могут быть отражены в буфере ядра, который еще не сохранен на диске. Анализ только до указанного смещения в файле \_cdc.idx гарантирует, что вы проанализируете данные CDC только на предмет надежных данных.

Порог разрешенного общего дискового пространства указан в yaml, когда вновь выделенные сегменты CommitLogSegments не будут разрешать данные CDC до тех пор, пока потребитель не проанализирует и не удалит файлы из указанного каталога cdc\_raw.

Конфигурация

Включение или отключение CDC

CDC можно включить или отключить через свойство таблицы cdc, например:

|  |
| --- |
| CREATE TABLE foo (a int, b text, PRIMARY KEY(a)) WITH cdc=true;  ALTER TABLE foo WITH cdc=true;  ALTER TABLE foo WITH cdc=false; |

параметры cassandra.yaml

Для CDC доступны следующие параметры cassandra.yaml:

cdc\_enabled (**по умолчанию:** **false**)

Включение или отключение операций CDC на уровне узла.

cdc\_raw\_directory (**по умолчанию**: $CASSANDRA\_HOME/data/cdc\_raw)

Место назначения для CommitLogSegments, подлежащего перемещению после сброса всех соответствующих memtables.

cdc\_free\_space\_in\_mb: (**по умолчанию**: **минимум 4096 и 1/8 объема тома**)

Вычисляется как сумма всех активных сегментов CommitLogSegments, разрешающих CDC. Все очищенные сегменты CDC в cdc\_raw\_directory.

cdc\_free\_space\_check\_interval\_ms (**по умолчанию:** **250**)

При наличии емкости мы ограничиваем частоту, с которой мы пересчитываем пространство, занимаемое cdc\_raw\_directory, чтобы предотвратить ненужное сжигание циклов ЦП. По умолчанию выполняется проверка 4 раза в секунду.

Чтение сегментов CommitLog

Используйте CommitLogReader.java. Использование довольно простое, для использования доступны различные сигнатуры. Чтобы обрабатывать мутации, считанные с диска, реализуйте CommitLogReadHandler.

Предупреждение

Не включайте CDC без какого-либо процесса потребления.

Если CDC включен на узле, а затем таблица cdc\_free\_space\_in\_mb заполнится, а в последствии записи в таблицы с поддержкой CDC будут отклонены, если не будет установлен какой-либо процесс потребления.

Фильтры Блума (Bloom filters)

В пути чтения RT.KeyValue объединяет данные на диске (в SSTables) с данными в RAM (в memtables). Чтобы избежать проверки каждого файла данных SSTable для запрашиваемого раздела, RT.KeyValue использует структуру данных, известную как фильтр Блума.

Фильтры Блума - это вероятностная структура данных, которая позволяет RT.KeyValue определять одно из двух возможных состояний:

* данные определенно не существуют в данном файле;
* данные, вероятно, существуют в данном файле.

Хотя фильтры Блума не могут гарантировать, что данные существуют в данной SSTable, фильтры Блума можно сделать более точными, позволив им потреблять больше оперативной памяти. У операторов есть возможность настроить это поведение для каждой таблицы, установив для bloom\_filter\_fp\_chance значение с плавающей запятой между 0 и 1.

Значение по умолчанию для bloom\_filter\_fp\_chance составляет 0,1 для таблиц, использующих LeveledCompactionStrategy, и 0,01 для всех остальных случаев.

Фильтры Блума хранятся в ОЗУ, но хранятся вне кучи, поэтому операторы не должны учитывать фильтры Блума при выборе максимального размера кучи. По мере повышения точности (по мере приближения bloom\_filter\_fp\_chance к 0) использование памяти увеличивается нелинейно - фильтр bloom для bloom\_filter\_fp\_chance = 0.01 потребует примерно в три раза больше памяти, чем та же таблица с bloom\_filter\_fp\_chance = 0.1.

Типичные значения для bloom\_filter\_fp\_chance обычно составляют от 0,01 (1%) до 0,1 (10%) вероятности ложного срабатывания, когда RT.KeyValue может сканировать SSTable на наличие строки, только чтобы обнаружить, что она не существует на диске. Параметр следует настраивать в зависимости от варианта использования:

* Пользователи с большим объемом оперативной памяти и более медленными дисками могут извлечь выгоду из установки для bloom\_filter\_fp\_chance численно меньшее значение (например, 0,01), чтобы избежать лишних операций ввода-вывода.
* Пользователи с меньшим объемом оперативной памяти, более плотными узлами или очень быстрыми дисками могут допускать более высокий bloom\_filter\_fp\_chance, чтобы сэкономить оперативную память за счет избыточных операций ввода-вывода.
* В рабочих нагрузках, которые редко читают или которые выполняют только чтение путем сканирования всего набора данных (например, рабочие нагрузки аналитики), установка bloom\_filter\_fp\_chance на гораздо большее число является приемлемой.

Изменение

Вероятность ложного срабатывания фильтра Блума отображается в выходных данных DESCRIBE TABLE как поле bloom\_filter\_fp\_chance. Операторы могут изменить значение с помощью оператора ALTER TABLE:

|  |
| --- |
| ALTER TABLE keyspace.table WITH bloom\_filter\_fp\_chance=0.01 |

Однако операторы должны знать, что это изменение происходит не сразу: фильтр Блума вычисляется при записи файла и сохраняется на диске в качестве компонента фильтра таблицы SSTable. После выполнения оператора ALTER TABLE новые файлы на диск будут записаны с новым bloom\_filter\_fp\_chance, но существующие sstables не будут изменены до тех пор, пока они не будут уплотнены, если оператору требуется изменение bloom\_filter\_fp\_chance для вступления в силу, они могут запустить перезапись SSTable, используя nodetool scrub или nodetool upgradesstables -a, оба из которых будут перестраивать sstables на диске, регенерируя фильтры Блума в процессе.

1. Полное описание файла local\_system\_data\_file\_directory



1. содержимое logback.xml файла по умолчанию

