Arenadata™ Hadoop

Версия - v2.1.2

Руководство пользователя по работе с HDFS

2020 Arenadata™

Оглавление

1 Общий обзор 3

2 Команды Shell 5

3 Secondary NameNode 9

4 Checkpoint Node 10

5 Backup Node 12

6 Import Checkpoint 13

7 Balancer 14

8 Rack Awareness 16

9 Безопасный режим 17

10 Команда fsck 18

11 Команда fetchdt 20

12 Recovery Mode 21

13 Upgrade и Rollback 22

14 DataNode Hot Swap Drive 24

15 Права доступа к файлам и безопасность 25

16 Масштабируемость 26

Руководство является отправной точкой для пользователей, работающих с распределенной файловой системой **HDFS** (Hadoop Distributed File System) либо в составе кластера **Hadoop**, либо в качестве автономной распределенной файловой системы общего назначения. Несмотря на то, что **HDFS** предназначена для работы в разных средах без особой настройки, практические знания о файловой системе могут значительно улучшить эксплуатационные свойства и диагностику конкретного кластера.

Руководство может быть полезно администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих внедрение и эксплуатацию кластера.

Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Общий обзор

HDFS – это основное распределенное хранилище, используемое приложениями **Hadoop**. Кластер **HDFS** в основном состоит из NameNode, который управляет метаданными файловой системы, и DataNode, в которых хранятся фактические данные. Диаграмма архитектуры **HDFS** отображает основные взаимодействия между NameNode, DataNodes и клиентами: клиенты связываются с NameNode для изменения метаданных файла и выполняют фактический ввод-вывод файла непосредственно с помощью DataNodes.

Далее приведены некоторые из основных функций, которые могут представлять интерес для пользователей:

- Hadoop, включая HDFS, хорошо подходит для распределенного хранения и распределенной обработки: он отказоустойчив, масштабируем и чрезвычайно прост в расширении. MapReduce, хорошо известный своей простотой и применимостью для распределенных приложений, является неотъемлемой частью Hadoop.
- HDFS легко настраивается с конфигурацией по умолчанию, которая подходит для многих установок. Но при этом в большинстве случаев в зависимости от специфики решаемых задач конфигурацию необходимо настраивать.
- Hadoop написан на Java и поддерживается на всех основных платформах.
- Hadoop поддерживает shell-подобные команды для непосредственного взаимодействия с HDFS.
- NameNode и Datanodes имеют встроенные веб-интерфейсы, которые позволяют легко проверять текущее состояние кластера.
- Новые функции и улучшения регулярно внедряются в HDFS. Ниже приведен перечень полезных функций HDFS:
  - Права доступа к файлам и аутентификация;
  - Rack awareness: учет физического местоположения узла при планировании задач и выделении хранилища;
  - Safemode: административный режим для сопровождения;
  - fsck: утилита для диагностики работоспособности файловой системы, поиска отсутствующих файлов или блоков;
  - fetchdt: утилита для извлечения DelegationToken и сохранения его в файле в локальной системе;
  - Balancer: инструмент для балансировки кластера, когда данные неравномерно распределены между DataNodes;
  - Upgrade и rollback: после обновления программного обеспечения можно выполнить откат до состояния HDFS перед обновлением в случае непредвиденных проблем;

– Secondary NameNode: выполняет периодические checkpoints пространства имен и помогает поддерживать размер файла, содержащего журнал изменений HDFS, в определенных пределах в NameNode;

– Checkpoint node: выполняет периодические checkpoints пространства имен и помогает минимизировать размер журнала с изменениями HDFS, хранящийся в NameNode. NameNode позволяет использовать несколько узлов Checkpoint одновременно, если в системе не зарегистрированы резервные узлы;

– Backup node: расширение узла Checkpoint. В дополнение к checkpointing он также получает поток изменений от NameNode и поддерживает свою собственную копию пространства имен в памяти, которая всегда синхронизируется с активным состоянием пространства имен NameNode. Только один резервный узел может быть зарегистрирован с помощью NameNode одновременно.

NameNode и DataNode запускают внутренний веб-сервер для отображения базовой информации о текущем состоянии кластера. В конфигурации по умолчанию главная страница NameNode находится по адресу http://namenode-name:9870/. На ней перечислены узлы данных в кластере и основные статистические данные кластера. Веб-интерфейс также можно использовать для просмотра файловой системы, используя ссылку “Browse the file system” на главной странице NameNode.

Права доступа к файлам в системе схожи с правами доступа на других известных платформах, таких как Linux. В настоящее время безопасность ограничивается простыми разрешениями на доступ к файлам. Пользователь, запускающий NameNode, рассматривается как суперпользователь для **HDFS**, однако, будущие версии **HDFS** будут поддерживать сетевые протоколы аутентификации, такие как **Kerberos**, для аутентификации пользователей и шифрования передачи данных.

HDFS имеет один NameNode для каждого кластера. В настоящее время общий объем памяти, доступный на NameNode, является основным ограничением масштабируемости. В очень больших кластерах увеличение среднего размера файлов, хранящихся в **HDFS**, помогает увеличить размер кластера без увеличения требований к памяти для NameNode.

Глава 2

Команды Shell

Hadoop включает в себя различные shell-подобные команды, напрямую взаимодействующие с **HDFS** и другими файловыми системами, которые поддерживает **Hadoop**. Команда `bin/hdfs dfs -help` выводит список команд, поддерживаемых оболочкой **Hadoop**, а `bin/hdfs dfs -help command-name` отображает более подробную справку. Команды поддерживают большинство обычных операций файловой системы, таких как копирование файлов, изменение прав доступа к файлам и т.д. Но также поддерживаются некоторые специфические операции **HDFS**, например, изменение репликации файлов.

Команда `bin/hdfs dfsadmin` поддерживает операции, связанные с администрированием **HDFS**, а `bin/hdfs dfsadmin -help` выводит список всех поддерживаемых в данный момент команд, например:

- `-report` – сообщает основную статистику HDFS. Некоторые из этих сведений также доступны на главной странице NameNode;
- `-safemode` – хотя обычно это не требуется, администратор может вручную войти или покинуть режим Safemode;
- `-finalizeUpgrade` – удаляет предыдущую резервную копию кластера, созданную во время последнего обновления;
- `-refreshNodes` – обновляет интерфейс NameNode с набором узлов DataNode, разрешенным подключаться к NameNode. По умолчанию NameNodes повторно считывает имена узлов DataNode в файле, определенном dfs.hosts и dfs.hosts.exclude. Хосты в dfs.hosts – это узлы данных, которые являются частью кластера. Если в dfs.hosts есть записи, то только хосты в нем могут регистрироваться с помощью NameNode. Записи в dfs.hosts.exclude – это узлы данных, которые должны быть выведены из эксплуатации. Альтернативно, если для *dfs.namenode.hosts.provider.classname* задано значение *org.apache.hadoop.hdfs.server.blockmanagement.CombinedHostFileManager*, все хосты include и exclude указываются в файле JSON, заданном dfs.hosts. Вывод из эксплуатации завершает DataNodes, когда все их записи среплицированы в другие DataNodes, при этом списанные узлы не выключаются автоматически и не выбираются для записи новых реплик;
- `-printTopology` – вывод топологии кластера. Отображение дерева стоек и узлов данных, прикрепленных к трекам, как показано в NameNode.

Пример использования:

```
hdfs dfsadmin [-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]]
hdfs dfsadmin [-safemode enter | leave | get | wait | forceExit]
hdfs dfsadmin [-saveNamespace [-beforeShutdown]]
hdfs dfsadmin [-rollEdits]
hdfs dfsadmin [-restoreFailedStorage true |false |check]
hdfs dfsadmin [-refreshNodes]
hdfs dfsadmin [-setQuota <quota> <dirname>...<dirname>]
```

```
hdfs dfsadmin [-clrQuota <dirname>...<dirname>]
hdfs dfsadmin [-setSpaceQuota <quota> [-storageType <storagetype>] <dirname>...<dirname>]
hdfs dfsadmin [-clrSpaceQuota [-storageType <storagetype>] <dirname>...<dirname>]
hdfs dfsadmin [-finalizeUpgrade]
hdfs dfsadmin [-rollingUpgrade [<query> |<prepare> |<finalize>]]
hdfs dfsadmin [-upgrade [query | finalize]
hdfs dfsadmin [-refreshServiceAcl]
hdfs dfsadmin [-refreshUserToGroupsMappings]
hdfs dfsadmin [-refreshSuperUserGroupsConfiguration]
hdfs dfsadmin [-refreshCallQueue]
hdfs dfsadmin [-refresh <host:ipc_port> <key> [arg1..argn]]
hdfs dfsadmin [-reconfig <namenode|datanode> <host:ipc_port> <start |status |properties>]
hdfs dfsadmin [-printTopology]
hdfs dfsadmin [-refreshNamenodes datanodehost:port]
hdfs dfsadmin [-getVolumeReport datanodehost:port]
hdfs dfsadmin [-deleteBlockPool datanode-host:port blockpoolId [force]]
hdfs dfsadmin [-setBalancerBandwidth <bandwidth in bytes per second>]
hdfs dfsadmin [-getBalancerBandwidth <datanode_host:ipc_port>]
hdfs dfsadmin [-fetchImage <local directory>]
hdfs dfsadmin [-allowSnapshot <snapshotDir>]
hdfs dfsadmin [-disallowSnapshot <snapshotDir>]
hdfs dfsadmin [-shutdownDatanode <datanode_host:ipc_port> [upgrade]]
hdfs dfsadmin [-evictWriters <datanode_host:ipc_port>]
hdfs dfsadmin [-getDatanodeInfo <datanode_host:ipc_port>]
hdfs dfsadmin [-metasave filename]
hdfs dfsadmin [-triggerBlockReport [-incremental] <datanode_host:ipc_port>]
hdfs dfsadmin [-listOpenFiles [-blockingDecommission] [-path <path>]]
hdfs dfsadmin [-help [cmd]]
```

- `-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]` – сообщает основную информацию и статистику файловой системы. Использование *dfs* может отличаться от *du*, поскольку оно измеряет необработанное пространство, используемое репликацией, контрольными суммами, снапшотами и т.д. на всех DN. Дополнительные флаги можно применять для фильтрации списка отображаемых DataNodes;
- `-safemode enter|leave|get|wait|forceExit` – команда обслуживания безопасного режима Safe mode – состояние Namenode, в котором он:
  - Не принимает изменения в пространстве имен (только для чтения);
  - Не копирует и не удаляет блоки.

  Безопасный режим вводится автоматически при запуске Namenode и автоматически выключается, когда настроенный минимальный процент блоков удовлетворяет условию минимальной репликации. Если Namenode обнаруживает какую-либо аномалию, он остается в безопасном режиме, пока проблема не будет решена. Если эта аномалия является следствием преднамеренного действия, администратор может использовать команду `-safemode forceExit` для принудительного выхода из безопасного режима. Случаи, когда это может потребоваться:
  - Метаданные Namenode не согласованы. Если Namenode обнаруживает, что метаданные были изменены вне диапазона и могут привести к потере данных, то Namenode переходит в состояние *forceExit*. В этот момент пользователь может либо перезапустить Namenode с правильными файлами метаданных, либо использовать *forceExit* (если потеря данных допустима);
  - Откат приводит к тому, что метаданные заменяются, и в редких случаях он может вызвать режим принудительного выхода из безопасного режима в Namenode.

  Безопасный режим можно ввести вручную, но тогда отключить его можно будет тоже только вручную;

- `-saveNamespace [-beforeShutdown]` – сохранение текущего пространства имен в каталогах хранилища и сброс журнала изменений. Требуется безопасный режим. Если задана опция *beforeShutdown*, NameNode делает контрольную точку тогда и только тогда, когда в течение временного окна контрольная точка не была установлена (настраиваемое количество периодов контрольных точек). Обычно функция используется перед закрытием NameNode для предотвращения возможного повреждения fsimage или журнала изменений;
- `-rollEdits` – откат журнала редактирования на активном NameNode;
- `-restoreFailedStorage true|false|check` – включение/выключение автоматической попытки восстановления неудачных реплик хранилища. Если сбойное хранилище снова станет доступным, система попытается восстановить журнал изменений и/или fsimage во время контрольной точки. Опция *check* возвращает текущую настройку;
- `-refreshNodes` – повторное чтение хостов и исключение файлов для обновления набора Datanodes, которым разрешено подключаться к Namenode, и тех, которые должны быть выведены из эксплуатации или повторно введены;
- `-finalizeUpgrade` – завершение обновления HDFS. Datanodes удаляют свои рабочие каталоги предыдущей версии, после чего Namenode делает то же самое. На этом процесс обновления завершается;
- `-upgrade query|finalize` – query: запрос текущего состояния обновления; finalize: завершить обновление HDFS (эквивалентно *finalizeUpgrade*);
- `-refreshServiceAcl` – перезагрузка файла политики авторизации на уровне сервиса;
- `-refreshUserToGroupsMappings` – обновить сопоставления пользователей и групп;
- `-refreshSuperUserGroupsConfiguration` – обновить сопоставления proxy-групп суперпользователя;
- `-refreshCallQueue` – перезагрузить очередь вызовов из конфига;
- `-refresh <host:ipc_port> <key> [arg1..argn]` – запуск обновления во время выполнения ресурса, указанного <key> на <host: ipc_port>. Все остальные аргументы после отправляются на хост;
- `-reconfig <datanode |namenode> <host:ipc_port> <start|status|properties>` – запуск реконфигурации, либо получение статуса текущей реконфигурации, либо получение списка реконфигурируемых свойств. Второй параметр указывает тип узла;
- `-printTopology` – отобразить дерево стоек и их узлов, как передается в Namenode;
- `-refreshNamenodes datanodehost:port` – перезагрузка файлов конфигурации для указанного datanode, прекращение обслуживания удаленных пулов блоков и старт обслуживания новых пулов блоков;
- `-getVolumeReport datanodehost:port` – получить отчет об объеме для указанного datanode;
- `-deleteBlockPool datanode-host:port blockpoolId [force]` – при принудительном вводе каталог пула блоков для указанного идентификатора блока данных на указанном datanode удаляется вместе с его содержимым, в противном случае каталог удаляется, только если он пуст. Команда не будет выполнена, если datanode все еще обслуживает пул блоков. Для выключения сервиса пула блоков на datanode использовать *refreshNamenodes*;
- `-setBalancerBandwidth <bandwidth in bytes per second>` – изменение пропускной способности сети, используемой каждым datanode во время балансировки блоков HDFS. <bandwidth> – максимальное число байтов в секунду, которое будет использоваться каждой datanode. Это значение переопределяет параметр *dfs.datanode.balance.bandwidthPerSec*. При этом новое значение не является постоянным в узле DataNode;
- `-getBalancerBandwidth <datanode_host:ipc_port>` – получение пропускной способности сети (в байтах в секунду) для указанного datanode. Это максимальная пропускная способность сети, используемая datanode при балансировке блоков HDFS;
- `-fetchImage <local directory>` – загрузка последнего fsimage из NameNode и сохранение его в указанном локальном каталоге;

- `-allowSnapshot <snapshotDir>` – разрешение на создание снапшотов каталога. Если операция завершается успешно, каталог становится моментальным снимком;
- `-disallowSnapshot <snapshotDir>` – запрет на создание снапшотов каталога, который будет создан. Все снимки каталога должны быть удалены перед включением функции;
- `-shutdownDatanode <datanode_host:ipc_port> [upgrade]` – отправить запрос на отключение для указанного datanode;
- `-evictWriters <datanode_host:ipc_port>` – заставляет datanode выселить всех клиентов, которые пишут блок. Функция полезна, когда эксплуатация приостановлена из-за медленных писателей;
- `-getDatanodeInfo <datanode_host:ipc_port>` – получение информации о указанном datanode;
- `-metasave filename` – сохранение основных структур данных Namenode в *filename* в каталоге, указанном свойством *hadoop.log.dir*. Файл *filename* перезаписывается, если уже существует. При этом *filename* содержит одну строку для каждого из следующих:
  - Сообщения heartbeat узлов Datanodes с Namenode;
  - Ожидающие репликации блоки;
  - Копируемые в настоящее время блоки;
  - Ожидающие удаления блоки.
- `-triggerBlockReport [-incremental] <datanode_host:ipc_port>` – запуск отчета о блокировке для указанного datanode. Если указано значение *incremental*, то отчет о полном блоке;
- `-listOpenFiles [-blockingDecommission] [-path <path>]` – список всех открытых файлов, которыми в данный момент управляет NameNode, а также имя клиента и клиентский компьютер, к которому они обращаются. Список открытых файлов фильтруется по заданному типу и пути;
- `-help [cmd]` – справка для указанной команды или для всех команд, если ни одна не указана.

Глава 3

Secondary NameNode

NameNode хранит изменения в файловой системе в виде журнала изменений, добавляемого к родному файлу. Когда NameNode запускается, сервис считывает состояние **HDFS** из файла образа, fsimage, а затем применяет изменения из файла журнала. После чего он записывает новое состояние **HDFS** в fsimage и начинает обычную работу с пустым файлом правок. Поскольку NameNode объединяет fsimage и редактирует файлы только во время запуска, файл журнала изменений может со временем стать очень большим. Другим побочным эффектом большего файла журнала изменений является то, что следующий перезапуск NameNode занимает больше времени.

В свою очередь Secondary NameNode время от времени объединяет файлы журнала изменений с fsimage и сохраняет размер журнала в пределах лимита. Поскольку требования к памяти у вторичного и основного NameNode одинаковы, Secondary NameNode обычно запускается на другом узле, чем NameNode.

Запуск процесса контрольной точки на Secondary NameNode управляется двумя параметрами конфигурации:

- `dfs.namenode.checkpoint.period` – по умолчанию установлено значение на 1 час; определяет максимальную задержку между двумя последовательными контрольными точками;
- `dfs.namenode.checkpoint.txns` – по умолчанию установлено значение на 1 миллион; определяет количество незарегистрированных транзакций на NameNode, которые будут вызывать следующую контрольную точку, даже если период ее не был достигнут.

Secondary NameNode хранит последнюю контрольную точку в каталоге, который структурирован таким же образом, как и каталог основного NameNode. Получается, образ с контрольной точки всегда готов к считыванию активным NameNode при необходимости.

Пример использования:

```
hdfs secondarynamenode [-checkpoint [force]] | [-format] | [-geteditsize]
```

- `-checkpoint [force]` – контрольные точки SecondaryNameNode, если *EditLog size*> = *fs.checkpoint.size*. Если используется принудительно, контрольная точка не зависит от размера EditLog;
- `-format` – форматирование локального хранилища во время запуска;
- `-geteditsize` – вывод количества незарегистрированных транзакций на NameNode.

Глава 4

Checkpoint Node

NameNode сохраняет свое пространство имен с помощью двух файлов: fsimage, который является последней контрольной точкой пространства имен, и журнал изменений (log) в пространстве имен с момента контрольной точки. Когда NameNode запускается, он объединяет fsimage и журнал изменений, чтобы обеспечить актуальное представление метаданных файловой системы. Затем NameNode перезаписывает fsimage с новым состоянием **HDFS** и начинает новый журнал изменений.

Checkpoint Node периодически создает контрольные точки пространства имен. Он загружает fsimage и изменения из активного NameNode, объединяет их локально и загружает новое изображение обратно в активный NameNode. Checkpoint Node обычно работает на компьютере, отличном от NameNode, поскольку их требования к памяти одиинаковы. Запуск Checkpoint Node осуществляется командой `bin/hdfs namenode -checkpoint` на указанном в конфигурационном файле узле.

Расположение Checkpoint Node (или Backup) и сопровождающего его веб-интерфейса настраивается с помощью переменных конфигурации *dfs.namenode.backup.address* и *dfs.namenode.backup.http-address*.

Запуск процесса контрольной точки на Checkpoint Node контролируется двумя параметрами конфигурации:

- `dfs.namenode.checkpoint.period` – по умолчанию установлено значение на 1 час; определяет максимальную задержку между двумя последовательными контрольными точками;
- `dfs.namenode.checkpoint.txns` – по умолчанию установлено значение на 1 миллион; определяет количество незарегистрированных транзакций на NameNode, которые будут вызывать срочную контрольную точку, даже если период ее не был достигнут.

Checkpoint Node хранит последнюю контрольную точку в каталоге, который структурирован таким же образом, как и каталог NameNode. Таким образом контрольное изображение всегда доступно для чтения из NameNode при необходимости.

В файле конфигурации кластера можно указать несколько Checkpoint Node

Пример использования:

```
hdfs namenode [-backup] |
      [-checkpoint] |
      [-format [-clusterid cid ] [-force] [-nonInteractive] ] |
      [-upgrade [-clusterid cid] [-renameReserved<k-v pairs>] ] |
      [-upgradeOnly [-clusterid cid] [-renameReserved<k-v pairs>] ] |
      [-rollback] |
      [-rollingUpgrade <rollback |started> ] |
      [-importCheckpoint] |
      [-initializeSharedEdits] |
```

```
[-bootstrapStandby [-force] [-nonInteractive] [-skipSharedEditsCheck] ] |
[-recover [-force] ] |
[-metadataVersion ]
```

- `-backup` – запуск Backup Node;
- `-checkpoint` – запуск Checkpoint Node;
- `-format [-clusterid cid]` – форматирование указанного NameNode. Запускает NameNode, форматирует его и затем выключает. Будет генерировать NameNodeFormatException, если имя dir уже существует и если переформатирование для кластера отключено;
- `-upgrade [-clusterid cid] [-renameReserved <k-v pairs>]` – Namenode должен быть запущен с опцией обновления до новой версии Hadoop;
- `-upgradeOnly [-clusterid cid] [-renameReserved <k-v pairs>]` – обновляет и закрывает указанный NameNode;
- `-rollback` – откат NameNode до предыдущей версии. Следует использовать после остановки кластера и разворачивания старой версии Hadoop;
- `-rollingUpgrade <rollback|started>` – *query*: запрос текущего состояния обновления; *prepare*: подготовка нового обновления; *finalize*: завершение текущего обновления;
- `-importCheckpoint` – загрузка изображения из каталога контрольных точек и сохранение его в текущем. Контрольная точка dir читается из свойства *dfs.namenode.checkpoint.dir*;
- `-initializeSharedEdits` – форматирование нового общего каталога изменений и копирование его в достаточное количество сегментов журнала для запуска резервного NameNode;
- `-bootstrapStandby [-force] [-nonInteractive] [-skipSharedEditsCheck]` – позволяет загружать резервные каталоги хранилища NameNode путем копирования последнего снимка пространства имен из активного NameNode. Используется при первой настройке кластера высокой доступности. Опция *-force* или *-nonInteractive* имеет то же значение, что и *-format*. Опция *-skipSharedEditsCheck* пропускает проверку правок, которая гарантирует достаточное количество правок, уже находящихся в общем каталоге, для запуска с последней контрольной точки;
- `-recover [-force]` – восстановление потерянных метаданных в поврежденной файловой системе;
- `-metadataVersion` – печать версии метаданных программного обеспечения и образа (при условии наличия настроенных каталогов).

Глава 5

Backup Node

Backup Node обеспечивает ту же функциональность контрольных точек, что и Checkpoint Node, а также поддерживает в памяти обновленную копию пространства имен файловой системы, которая всегда синхронизируется с активным состоянием NameNode. Наряду с принятием потока изменений файловой системы журнала из NameNode и сохранением его на диске, Backup Node также применяет эти изменения к своей собственной копии пространства имен в памяти, создавая таким образом резервную копию пространства имен.

Backup Node не нужно загружать fsimage и редактировать файлы с активного NameNode для создания контрольной точки, как это требуется для Checkpoint Node или Secondary NameNode, поскольку Backup Node уже имеет актуальный статус состояния пространства имен в памяти. Процесс создания контрольной точки Backup узла является более эффективным, поскольку ему нужно только сохранить пространство имен в локальном файле fsimage и сбросить изменения.

Поскольку Backup Node поддерживает копию пространства имен в памяти, его требования к RAM такие же, как и у узла NameNode.

NameNode поддерживает один Backup Node. Узлы Checkpoint не могут быть зарегистрированы, если используется Backup

Backup Node настраивается так же, как узел Checkpoint, начиная с `bin/hdfs namenode -backup`.

Расположение Backup Node (или Checkpoint) и его веб-интерфейса настраивается с помощью переменных конфигурации *dfs.namenode.backup.address* и *dfs.namenode.backup.http-address*.

Использование Backup Node обеспечивает возможность запуска NameNode без постоянного хранилища, делегируя всю ответственность за сохранение состояния пространства имен Backup Node. Для этого необходимо запустить NameNode с параметром `-importCheckpoint`, а также не указывать постоянные каталоги хранения типа edits *dfs.namenode.edits.dir* в конфигурации NameNode.

Пример использования приведен в главе *Checkpoint Node*.

Глава 6

Import Checkpoint

В случае потери всех копий образов и файлов транзакций в NameNode может быть импортирована последняя контрольная точка. Для этого необходимо:

- Создать пустой каталог, указанный в переменной конфигурации *dfs.namenode.name.dir*;
- Указать местоположение каталога контрольных точек в переменной конфигурации *dfs.namenode.checkpoint.dir*;
- Запустить NameNode с параметром `-importCheckpoint`.

При этом NameNode загружает контрольную точку из каталога *dfs.namenode.checkpoint.dir*, а затем сохраняет ее в каталог NameNode, заданный в *dfs.namenode.name.dir*. Если в *dfs.namenode.name.dir* содержится допустимый образ, NameNode выдает ошибку. NameNode проверяет консистентность образа в *dfs.namenode.checkpoint.dir*, но не изменяет его каким-либо образом.

Пример использования приведен в главе *Checkpoint Node*.

Глава 7

Balancer

Данные **HDFS** не всегда могут быть размещены равномерно по всем DataNode. Одной из распространенных причин является добавление новых DataNodes в существующий кластер. При размещении новых блоков (данные для файла хранятся в виде серии блоков) NameNode учитывает различные параметры, прежде чем выбирать узлы DataNodes для получения этих блоков. Некоторые из них:

- Политика для хранения одной из реплик блока на том же узле, что и узел, который записывает блок;
- Необходимо распределить различные реплики блока по стойкам, чтобы кластер мог пережить потерю всей стойки;
- Одна из реплик обычно размещается в той же стойке, что и узел, выполняющий запись в файл, что снижает количество операций ввода-вывода между стойками;
- Данные HDFS распределяются равномерно по узлам DataNodes в кластере.

Из-за множества конкурирующих соображений данные могут быть неравномерно размещены между узлами DataNodes. **HDFS** предоставляет инструмент для администраторов, который анализирует размещение блоков и осуществляет перебалансировку данных через DataNode.

Пример использования:

```
hdfs balancer
      [-policy <policy>]
      [-threshold <threshold>]
      [-exclude [-f <hosts-file> | <comma-separated list of hosts>]]
      [-include [-f <hosts-file> | <comma-separated list of hosts>]]
      [-source [-f <hosts-file> | <comma-separated list of hosts>]]
      [-blockpools <comma-separated list of blockpool ids>]
      [-idleiterations <idleiterations>]
      [-runDuringUpgrade]
```

- `-policy <policy>` – *datanode* (по умолчанию): кластер сбалансирован, если каждый datanode сбалансирован; *blockpool*: кластер сбалансирован, если каждый пул блоков в каждой datanode сбалансирован;
- `-threshold <threshold>` – процент емкости диска; перезапись порога по умолчанию;
- `-exclude -f <hosts-file> | <comma-separated list of hosts>` – исключение указанных datanode из балансировки;
- `-include -f <hosts-file> | <comma-separated list of hosts>` – включение только указанных datanode для балансировки;

- `-source -f <hosts-file> | <comma-separated list of hosts>` – выбор только указанных datanode в качестве исходных узлов;
- `-blockpools <comma-separated list of blockpool ids>` – работа балансировщика только на blockpools, включенных в этот список;
- `-idleiterations <iterations>` – максимальное количество холостых итераций перед выходом; перезапись количества итераций по умолчанию (*5*);
- `-runDuringUpgrade` – следует ли запускать балансировщик во время текущего обновления HDFS. Действие не влияет на используемое пространство на перегруженных машинах, поэтому нежелательно;
- `-h|--help` – показывает используемый инструмент и справочную информацию.

Для остановки процесса перебалансировки администратор может просто нажать комбинацию клавиш *Ctrl-C*.

Политика blockpool является более строгой, чем политика datanode

Помимо указанных параметров введена функция закрепления (pinning feature) для предотвращения перемещения определенных реплик балансировщиком или mover. По умолчанию функция отключена и может быть включена с помощью свойства конфигурации *dfs.datanode.block-pinning.enabled*. В включенном состоянии она влияет только на блоки, которые записываются в указанные в вызове `create()` узлы. Функция полезна при необходимости локального сохранения данных для таких приложений, как **HBase regionserver**.

Глава 8

Rack Awareness

Кластер **HDFS** может распознавать топологию стоек, в которых размещены все узлы. Важно настроить эту топологию, чтобы оптимизировать емкость и использование данных.

Детальная информация приведена по ссылке rack awareness.

Глава 9

Безопасный режим

Во время запуска NameNode загружает состояние файловой системы из fsimage и файла журнала изменений. Затем чтобы не начать преждевременную репликацию блоков, он ожидает, когда DataNodes сообщат о своих блоках, не смотря на то, что в кластере уже существует достаточное количество реплик. В течение этого времени NameNode остается в безопасном режиме (Safemode). Безопасный режим для NameNode по существу является режимом только для чтения для кластера **HDFS**, где он не допускает каких-либо изменений в файловой системе или блоках. Обычно NameNode выходит из Safemode автоматически после того, как узлы DataNodes сообщают, что большинство блоков файловой системы доступно.

При необходимости система **HDFS** может быть размещена в безопасном режиме явно с помощью команды `bin/hdfs dfsadmin -safemode`. Включен ли режим Safemode отображается на первой странице NameNode.

Глава 10

Команда fsck

HDFS поддерживает команду **fsck** для проверки различных несоответствий. Команда предназначена для сообщения о проблемах с различными файлами, например, об отсутствующих блоках для файла или недостаточно реплицированных блоках. Но в отличие от традиционной утилиты **fsck** для собственных файловых систем, эта команда не исправляет обнаруженные ошибки.

Обычно NameNode автоматически исправляет большинство сбоев. По умолчанию **fsck** игнорирует открытые файлы, но предоставляет возможность выбора всех файлов во время создания отчетов. Команда **fsck** системы **HDFS** не является командой оболочки **Hadoop**, она может быть запущена путем `bin/hdfs fsck`.

Команда **fsck** может быть запущена по всей файловой системе или только на подмножестве файлов.

Пример использования:

```
hdfs fsck <path>
       [-list-corruptfileblocks |
       [-move | -delete | -openforwrite]
       [-files [-blocks [-locations | -racks | -replicaDetails | -upgradedomains]]]
       [-includeSnapshots] [-showprogress]
       [-storagepolicies] [-maintenance]
       [-blockId <blk_Id>]
```

- `path` – начало проверки с указанного пути;
- `-delete` – удаление поврежденных файлов;
- `-files` – отобразить проверяемые файлы;
- `-files -blocks` – отобразить отчет о блокировке;
- `-files -blocks -locations` – отобразить расположение каждого блока;
- `-files -blocks -racks` – отобразить топологию сети для расположения узлов данных;
- `-files -blocks -replicaDetails` – отобразить каждую копию реплики;
- `-files -blocks -upgradedomains` – отобразить обновления доменов для каждого блока;
- `-includeSnapshots` – включить данные снапшотов, если указанный путь указывает на каталог снапшотов или под ним есть каталоги снапшотов;
- `-list-corruptfileblocks` – отобразить список отсутствующих блоков и файлов, к которым они принадлежат;
- `-move` – переместить поврежденные файлы в */lost+found*;

- `-openforwrite` – отобразить файлы, открытые для записи;
- `-showprogress` – отобразить точки для прогресса в выводе; по умолчанию выключено (без прогресса);
- `-storagepolicies` – отобразить сводную политику хранения для блоков;
- `-maintenance` – отобразить детали узла;
- `-blockId` – отобразить информацию о блоке.

Глава 11

Команда fetchdt

Для получения Delegation Token и сохранения его в файле в локальной системе **HDFS** поддерживает команду **fetchdt**. Этот токен впоследствии можно использовать для доступа к защищенному серверу (например, NameNode) с незащищенного клиента. Утилита использует RPC или HTTPS (через **Kerberos**) для получения токена и, следовательно, требует наличия тикетов **Kerberos** перед запуском (необходимо запустить **kinit** для получения тикетов). В **HDFS** команда **fetchdt** не является командой оболочки **Hadoop** и может быть запущена как `bin/hdfs fetchdt DTfile`. После получения токена можно запустить команду **HDFS** без тикетов **Kerberos**, указав переменную среды `HADOOP_TOKEN_FILE_LOCATION` в файл токена делегирования.

Пример использования:

```
hdfs fetchdt <opts> <token_file_path>
```

- `--webservice NN_Url` – URL для связи с NN (начинается с *http* или *https*);
- `--renewer name` – обновленное название токена делегирования;
- `--cancel` – отменить токен делегирования;
- `--renew` – обновить токен делегирования. Токен делегирования должен быть получен с использованием параметра *-renewer name*;
- `--print` – отобразить токен делегирования;
- `token_file_path` – путь к файлу для хранения токена.

Глава 12

Recovery Mode

Как правило, обычно настраивается несколько мест хранения метаданных. И в случае, если хранилище повреждается, метаданные можно прочитать из других хранилищ. Однако, что делать, если единственное доступное хранилище повреждено? Для таких ситуаций существует специальный режим запуска NameNode, называемый режимом восстановления – Recovery Mode, который позволяет восстановить большую часть данных.

Запуск NameNode в режиме восстановления осуществляется следующим образом: `namenode -recover`.

В режиме восстановления NameNode в командной строке интерактивно запрашивает о возможных действиях, которые необходимо предпринять для восстановления данных. Для того, чтобы не получать эти запросы, можно указать опцию `-force`, которая заставляет режим Recovery Mode всегда выбирать первый вариант действий, обычно являющийся самым разумным.

Поскольку режим Recovery Mode может привести к потере данных, всегда необходимо перед его использованием делать резервную копию журнала изменений и fsimage

Глава 13

Upgrade и Rollback

При обновлении **Hadoop** на существующем кластере, как и при любом обновлении программного обеспечения, возможны новые ошибки или несовместимые изменения, которые влияют на существующие приложения и не были обнаружены ранее. Система позволяет администраторам вернуться к более ранней версии **Hadoop** и откатить кластер до состояния, в котором он находился до обновления. Но **HDFS** может иметь одну такую резервную копию. Перед обновлением администраторам необходимо удалить существующую резервную копию с помощью команды `bin/hadoop dfsadmin -finalizeUpgrade`. Далее кратко описывается типичная процедура обновления:

- Перед обновлением программного обеспечения Hadoop завершить работу, если имеется существующая резервная копия.
- Остановить кластер и распространить новую версию Hadoop.
- Запустить новую версию с параметром *-upgrade* (`sbin/start-dfs.sh -upgrade`).
- Завершить обновление, как только новая HDFS считается работающей (возможно, после нескольких дней эксплуатации). Важно обратить внимание, что пока кластер не будет завершен, удаление файлов, существовавших до обновления, не освобождает реальное дисковое пространство на узлах данных DataNodes.
- При необходимости вернуться к старой версии:
  - Остановить кластер и распространить более раннюю версию Hadoop;
  - Выполнить команду отката в namenode (`bin/hdfs namenode -rollback`);
  - Запустить кластер с возможностью отката (`sbin/start-dfs.sh -rollback`).

При обновлении **HDFS** до новой версии необходимо переименовать или удалить все пути, зарезервированные в новой версии **HDFS**, в ином случае, когда NameNode встречает зарезервированный путь во время обновления, выдает ошибку:

```
/.reserved is a reserved path and .snapshot is a reserved path component in this version of HDFS. Please␣
↪rollback and delete or rename this path, or upgrade with the -renameReserved [key-value pairs] option to␣
↪automatically rename these paths during upgrade.
```

Указание `-upgrade -renameReserved [optional key-value pairs]` заставляет NameNode автоматически переименовывать любые зарезервированные пути, найденные во время запуска. Например, чтобы переименовать все пути с именами *.snapshot* на *.my-snapshot* и *.reserved* на *.my-reserved*, необходимо указать `-upgrade -renameReserved .snapshot=.my-snapshot,.reserved=.my-reserved`.

Если с ключом *-renameReserved* не указаны пары ключ-значение, NameNode добавляет суффиксы зарезервированным путям в виде *.<LAYOUT-VERSION>.UPGRADE_RENAMED*, например,

`.snapshot.-51.UPGRADE_RENAMED`.

Есть некоторые оговорки к этому процессу переименования и по возможности рекомендуется перед обновлением сначала выполнять `hdfs dfsadmin -saveNamespace`. Это связано с тем, что может возникнуть несогласованность данных, если операция журнала изменений ссылается на место назначения автоматически переименованного файла.

Глава 14

DataNode Hot Swap Drive

Datanode поддерживает возможность горячей замены дисков, поэтому пользователь может добавлять или заменять узлы DataNode в **HDFS**, не выключая их. Далее кратко описана типичная процедура замены Hot Swap Drive:

- При наличии новых каталогов хранения пользователь должен отформатировать их и подключить соответствующим образом;
- Пользователь обновляет конфигурацию DataNode *dfs.datanode.data.dir*, чтобы отразить каталоги томов данных, которые будут использоваться;
- Пользователь запускает `dfsadmin -reconfig datanode HOST:PORT start` для инициализации процесса реконфигурации. Пользователь может использовать `dfsadmin -reconfig datanode HOST:PORT status` для запроса текущего состояния задачи реконфигурации;
- После завершения задачи реконфигурации пользователь может безопасно размонтировать удаленные каталоги томов данных и физически удалить диски.

Глава 15

Права доступа к файлам и безопасность

Права доступа к файлам схожи с правами доступа к файлам на других известных платформах, таких как **Linux**. В настоящее время безопасность ограничена простыми правами доступа к файлам. Пользователь, запускающий NameNode, для **HDFS** рассматривается как суперпользователь. Будущие версии **HDFS** будут поддерживать сетевые протоколы аутентификации, такие как Kerberos, для аутентификации пользователей и шифрования передачи данных.

Глава 16

Масштабируемость

В настоящее время **Hadoop** может работать на кластерах с тысячами узлов, где **HDFS** имеет по одному NameNode для каждого кластера. Таким образом общий объем памяти, доступный на NameNode, является основным ограничением масштабируемости. В очень больших кластерах увеличение среднего размера файлов, хранящихся в **HDFS**, способствует увеличению размера кластера без повышения требований к памяти для NameNode.

Конфигурация по умолчанию может не подходить для очень больших кластеров