Дата последней сборки документа:
3 апреля 2021 г. в 04:58

Заметки по большим данным и облачным технологиям

Подвойский А.О.

Здесь приводятся заметки по некоторым вопросам, касающимся больших данных, облачных технологий, машинного обучения, анализа данных, программирования на языках `Python`, `R` и прочим сопряженным вопросам так или иначе, затрагивающим работу с данными.

Содержание

1. Основные термины и определения

Витрина данных (Data Mart) – срез хранилища данных, представляющий собой массив тематической, узконаправленной информации, ориентированный, например, на пользователей одной рабочей группы или департамента.

2. Установка Hadoop на MacOS X

Установить `hadoop` на MacOS X можно с помощью менеджера пакетов `brew`

```
brew install hadoop
```

После установки Hadoop остается внести несколько изменений в конфигурационные файлы и настроить переменные окружения.

В файле, расположенном

по пути `/usr/local/Cellar/hadoop/3.3.0/libexec/etc/hadoop/hadoop-env.sh` изменить путь до JAVA_HOME. Узнать домашнюю директорию java можно так

```
/usr/libexec/java_home # /usr/local/Cellar/openjdk/15.0.1
```

/usr/local/Cellar/hadoop/3.3.0/libexec/etc/hadoop/hadoop-env.sh

```
# The java implementation to use...
# variable ...
export JAVA_HOME=/usr/local/Cellar/opejdk/15.0.1
```

Далее в файле `core-site.xml` нужно внести следующие изменения

core-site.xml

```
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/Cellar/hadoop/hdfs/tmp</value>
    <description>A base for other temporary directories</description>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:8020</value>
  </property>
</configuration>
```

Теперь внесем изменения в файл `mapred-site.xml`

mapred-site.xml

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:8021</value>
  </property>
</configuration>
```

И, наконец, внесем изменения в файл

hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

Проверим включен ли ssh

```
ssh localhost
```

Если эта команда вызывает ошибку, то следует настроить ssh следующим образом (нужно сообщить системе о ключах, которые мы собираемся использовать)

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

Последним шагом требуется отформатировать HDFS

```
cd /usr/local/opt/hadoop
hdfs namenode -format
```

Вывод последней команды должен содержать следующую строку

```
...
2021-04-02 14:58:30,137 INFO common.Storage: Storage directory /usr/local/Cellar/hadoop/hdfs/tmp
    /dfs/name has been successfully formatted.
```

Теперь можно задать псевдоними для команд запуска и остановки hadoop сервисов

~/.zshrc

```
alias hstart="/usr/local/Cellar/hadoop/3.3.0/sbin/start-all.sh"
alias hstop="/usr/local/Cellar/hadoop/3.3.0/sbin/stop-all.sh"

source ~/.zshrc
```

Теперь можно запустить Hadoop

```
hstart
```

и проверить запущенные сервисы

```
jps
# ----
47728 ResourceManager
47538 SecondaryNameNode
47826 NodeManager
47908 Jps
47302 NameNode
47401 DataNode
```

Получить доступ к Hadoop можно через web-интерфейс

- http://localhost:9870: менеджер ресурсов,
- http://localhost:8088: трекер заданий,
- http://localhost:8042: информация по узлам.

3. Компоненты экосистемы Hadoop

Avro – система сериализации для выполнения эффективных межъязыковых вызовов RPC и долгосрочного хранения данных.

MapReduce – модель распределенной обработки данных и исполнительная среда, работающая на больших кластерах типовых машин.

HDFS – распределенная файловая система, работающая на больших кластерах стандартных машин.

Hive – распределенное хранилище данных. В принципе Hive можно называть платформой пакетного обработки или СУБД. Hive управляет данными, хранимыми в HDFS, и предоставляет язык запросов на базе SQL (которые преобразуются ядром времени выполнения в задания MapReduce) для работы с этими данными.

HBase – распределенная нереляционная столбцово-ориентированная *база данных*, построенная на основе HDFS. HBase использует HDFS для организации хранения данных и поддерживает как пакетные вычисления с использованием MapReduce, так и точечные запросы (произвольное чтение данных).

Sqoop – инструмент эффективной массовой пересылки данных между структурированными хранилищами (такими, как реляционные базы данных) и HDFS.

Oozie – сервис запуска и планирования заданий Hadoop (включая задания MapReduce, Pig, Hive и Sqoop jobs).

4. Apache Drill

5. Apache HBase

HBase – распределенная нереляционная (столбцово-ориентирования) база данных формата «ключ-значение».

5.1. Установка и запуск

Подробности, связанные с установкой различных режимах (автономном, распределенном и т.д.) можно узнать на странице https://hbase.apache.org/book.html.

Скачать tar-архив можно здесь https://www.apache.org/dyn/closer.lua/hbase/2.4.0/hbase-2.4.0-bin.tar.gz

```
curl -O https://apache-mirror.rbc.ru/pub/apache/hbase/2.4.0/hbase-2.4.0-bin.tar.gz
```

Теперь следует распоковать архив

```
tar -xvzf hbase-2.4.0...
```

перейти в директорию `hbase-2.4.0` и задать путь до java в файле `hbase-env.sh`, раскоментировав нужную строку

conf/hbase-env.sh

```
export JAVA_HOME=/usr/local/Cellar/openjdk/15.0.1
```

В конфигурационном файле команданой оболочки удобно задать переменные окружения для Java и HBase

~/.zshrc

```
# for HBase
export JAVA_HOME="/usr/local/Cellar/openjdk/15.0.1"
export PATH="${PATH}:/Users/leor.finkelberg/hbase/hbase-2.4.0/bin"
```

Диреткорию размещения java на MacOS X следует искать с помощью менеджера пакетов `brew`

```
brew list java # /usr/local/Cellar/openjdk/15.0.1/bin/java
```

ВАЖНО: обновить java, можно скачав соответствующую версию с ресурса https://www.oracle.com/java/technologies/javase-jdk15-downloads.html.

Запустить HBase можно с помощью сценария командной оболочки из `bin/`

```
start-hbase.sh
```

Подключиться к запущенному экземпляру можно так

```
hbase shell
```

Для того чтобы убедиться, что процесс HMaster запущен можно воспользоваться утилитой `jps`.

Бывает удобно следить за работой приложения с помощью Web-интерфейса, доступного на http://localhost:16010.

Закончить сессию можно с помощью команды `quit`. Затем нужно остановить HBase

```
stop-hbase.sh
```

6. Apache Cassandra

7. Apache Kylin

8. Apache Impala

Apache Impala – это массово-параллельный механизм интерактивного выполнения SQL-запросов к данным, хранящимся в HDFS, HBase или Amazon Simple Storage (S3). Также Impala называют MPP-движком или *распределенной СУБД*.

Impala использует тот же синтаксис SQL (HiveQL), драйвер ODBC и пользовательский интерфейс как и Apache Hive.

Чтобы избежать задержки, Impala обходит MapReduce для прямого доступа к данным с помощью специализированного механизма распределенных запросов. В результате производительность на порядок выше чем у Hive (платформа Hive построена на базе MapReduce).

9. Приемы работы с `hadoop fs`

ВАЖНО: при работе с локальной файловой системой, HDFS, WebHDFS, S3 FS и т.д. следует пользоваться `hadoop fs`, а при работе с HDFS – `hdfs dfs`[1].

Вывести наполнение директории

```
hdfs dfs -ls / # наполнение корня директории
hdfs dfs -ls -d /hadoop
hdfs dfs -ls -h /data
hdfs dfs -ls -R /hadoop # вывести рекурсивно список всех файлов в поддиректориях hadoop
hdfs dfs -ls /hadoop/dat* # вывести список всех файлов по шаблону
```

Вывести информацию об используемом дисковом пространстве

```
hdfs dfs -df hdfs:/
```

Вывести информацию о здоровье файловой системы Hadoop

[1] `hdfs dfs` используется вместо `hadoop dfs`, считающейся устаревшей

```
hdfs fsck /
```

Создать директорию на HDFS

```
hdfs dfs -mkdir /hadoop
```

Скопировать файл, расположенный в локальной файловой системе, на HDFS

```
hdfs dfs -copyFromLocal ~/python_scripts /hadoop
```

Скопировать файл, расположенный на HDFS, в локальную файловую систему

```
hdfs dfs -copyToLocal /hadoop/work/DBSCAN_test.py ~/GARBAGE
```

Задать значение фактора реплицирования (по умолчанию равен 3)

```
hdfs dfs -setrep -w 2 /usr/sample
```

Скопировать директорию с одного узла кластера на другой

```
hdfs dfs -distcp hdfs://namenodeA/apache_hadoop hdfs://namenodeB/hadoop
```

Вывести информацию по статистике файлов/директорий; здесь `%b` – блоки

```
hdfs dfs -stat "%F %u:%g %b %y %n" /hadoop/work/DBSCAN_test.py
# regular file leor.finkelberg:supergroup 597 2021-04-02 22:01:50 DBSCAN_test.py
```

10. Apache Hive

Apache Hive представляет собой *систему управления базами данных* (СУБД), построенную поверх Hadoop. В официальной документации платформа описана как *хранилище данных* (DWH). Apache Hive позволяет работать с данными, хранящимися в HDFS или HBase, с помощью языка HiveQL, очень близкому к стандартному SQL.

ВАЖНО: Apache Hive не является реляционной базой данных, не поддерживает онлайн-транзакции (OLTP), в том числе обновления отдельных строк таблицы.

Apache Hive более удобна в использовании по сравнению с Hadoop и Pig, поскольку не требует изучения специального доменно-специфичного языка (Pig Lating) или Java. А это значит, что платформа Hive является универсальной, то есть может быть задействована как

- система *пакетного* анализа,
- *нереляционное хранилище данных*,
- часть процесса ETL.

Hive и другие фреймворки, построенные *на базе MapReduce*, лучше всего подходят для *длительных пакетных заданий*, например, для пакетной обработки заданий типа ETL.

При работе с Hive можно выделить следующие объекты

- Базы данных,
- Таблицы,
- Партиции,
- Бакеты.

База данных в Hive представляет собой то же, что и база данных в реляционных СУБД. База данных в Hive – это пространство имен, содержащее таблицы. Команда создания базы данных выглядит следующим образом

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Здесь DATABASE и SCHEMA одно и то же.

Пример создания базы данных

```
CREATE DATABASE userdb;
```

Для переключения на соответствующую базу данных используется команда USE

```
USE userdb;
```

В целом таблицы в Hive представляют собой то же, что и таблицы в классических реляционных базах данных, но есть и отличия. Основное отличие таблиц Hive состоит в том, что они хранятся в виде *обычных файлов* на HDFS. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и т.д.

Пример создания таблицы

```
CREATE TABLE IF NOT EXISTS employee (
  eid INT,
  name STRING,
  salary STRING,
  destination STRING
)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Здесь создается таблица, данные которой будут храниться в виде обычных csv-файлов. Столбцы разделены символом табуляции. После этого можно загрузить данные в таблицу.

Так как Hive по сути представляет собой движок для трансляции SQL-запросов в MapReduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблице. Для того чтобы избежать полного сканирования данных по некоторым столбцам таблицы можно провести партиционирование таблицы. Это означает, что данные, относящиеся к разным значениям будут физически храниться в разных папках на HDFS.

Для создания партиционированной таблицы необходимо указать по каким столбцам будет выполняться партиционирование

```
CREATE TABLE IF NOT EXISTS employee_partitioned (
  eid INT,
  name STRING,
  salary STRING,
  destination STRING
)
COMMENT 'Employee details'
PARTITIONED BY (birth_year INT, birth_month STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

При заливке данных в такую таблицу необходимо явно указать, в какой партиции требуется разместить данные

```
LOAD DATA PATH '/user/root/sample.txt' OVERWRITE
INTO TABLE employee_partitioned
PARTITION (birth_year=1998, birth_month='May');
```

10.1. Форматы хранения данных

Форматы хранения данных:

- AVRO: формат на основе JSON, включающий поддержку RPC и сериализацию,
- Parquet: колоночный формат хранения,
- ORC: быстрый колоночный формат хранения,
- RCFile: формат размещения данных для реляционных таблиц,
- SequenceFile: бинарный формат данных с записью определенных типов данных.

Текстовый файл Пример создания таблицы в текстовом формате. Формат текстового файла используется по умолчанию для Hive.

```
CREATE TABLE country (
  name STRING,
  states ARRAY<STRING>,
  cities_and_size MAP<STRING, INT>,
  parties STRUCT<name STRING, votes FLOAT, members INT>
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\001'
COLLECTION ITEMS TERMINATED BY '\002'
MAP KEYS TERMINATED BY '\003'
LINES TERMINATED BY '\n'
STORED AS TextFile;
```

Файл последовательностей (SequenceFile) Формат SequenceFile является значением по умолчанию в Hadoop и хранит данные в *парах ключ-значение*. Большинство инструментов в экосистеме Hadoop могут читать формат SequenceFile

```
CREATE TABLE country (
  name STRING,
  population INT
)
STORED AS SequenceFile;
```

Файл столбцов строк (RCFile) Грубо говоря, RCFile это стратегия хранения данных в формате «строки столбцов». RCFile сочетает в себе достоинства как строко-ориентированных, так столбцово-ориентированных стратегий. Чтобы сериализовать таблицу, RCFile разбивает ее сначала по горизонтали, а затем по вертикали, вместо того чтобы разбить ее только по горизонтали, как это делается в обычных реляционных СУБД. Горизонтальное разбиение разбивает таблицу, как следует из названия стратегии, по горизонтали на несколько групп строк на основе заданного пользователем значения, определяющим размер группы строк.

```
CREATE TABLE country (
  name STRING,
  population INT
)
```

```
STORED AS RCFile;
```

Пример. Пусть есть таблица вида

```
c1 | c2 | c3 | c4
---+----+----+---
11 | 12 | 13 | 14
21 | 22 | 23 | 24
31 | 32 | 33 | 34
41 | 42 | 43 | 44
51 | 52 | 53 | 54
```

Как обсуждалось выше на первом этапе таблица разбивается по горизонтали на группы строк

```
c1 | c2 | c3 | c4
---+----+----+---
11 | 12 | 13 | 14
----------------
21 | 22 | 23 | 24
----------------
31 | 32 | 33 | 34
================ -- граница групп строк
41 | 42 | 43 | 44
----------------
51 | 52 | 53 | 54
```

Затем в каждой группе строк RCFile разбивает данные по вертикали

```
1-ая группа         2-ая группа
c1 | 11 | 21 | 31 || 41 | 51
c2 | 12 | 22 | 32 || 42 | 52
c3 | 13 | 23 | 33 || 43 | 53
c4 | 14 | 24 | 34 || 44 | 54
```

То есть таблица будет сериализована как

```
11, 21, 31; 41, 51;
12, 22, 32; 42, 52;
13, 23, 33; 43, 53;
14, 24, 34; 44, 54;
```

ORC файл ORC файл следует рассматривать как альтернативу RCFile

11. Логическая витрина для доступа к большим данным

Пример. Рассмотрим некоторый промышленный комплекс, обладающий огромным количеством оборудования, обвешанного различными датчиками, регулярно сообщающими сведения о состоянии этого оборудования. Для простоты рассмострим только два аггрегата (котел и резервуар), и три датчика (температуры котла и резервуара, а также давления в котле).

Эти датчики контролируются АСУ разных производителей и выдают информацию в разные хранилища: сведения о температуре и давлении в котле поступают в HBase, а данные о температуре в резервуаре пишутся в лог-файлы, расположенные в HDFS.

Данные о датчиках могут храниться, например, в `PostgreSQL`, а показания этих датчиков – в HDFS, HBase и т.п. Теперь пусть мы хотим предоставить аналитику возможность делать запросы. Заранее построить и запрограммировать сложные запросы не получится. Выполнение любого

сложного, тяжелого запроса требует связывания данных из разных источников, в том числе из находящихся за пределами нашего модельного примера. Извне могут поступать, например, справочные сведения о рабочих диапазонах температуры и давления для разных видов оборудования, фасетные классификаторы, позволяющие определить, какое оборудование является маслонаполненным и др. Все подобные запросы аналитик формулирует в терминах концептуальной модели предметной области, то есть ровно в тех выражениях, в которых он думает о работе своего предприятия.

Витрина данных – предметно-ориентированная и, как правило, содержащая данные по одному из направлений деятельности компании база данных. Она отвечает тем же требованиям, что и хранилище данных, но в отличие от него, нейтрально к приложениям. В витрине информация храниться оптимизированно с точки зрения решения конкретных задач.

Витрины данных имеют следующие достоинства:

- пользователи ведят и работают только с теми данными, которые им действительно нужны,
- для витрин данных не требуется использовать мощные вычислительные средства.

К недостаткам витрин данных можно отнести сложность контроля целостности и противоречивости данных.

Список иллюстраций

Список литературы

1. *Сенько А.* Работа с BigData в облаках. Обработка и хранение данных с примерами из Microsoft Azure. – СПб.: Питер, 2019. – 448 с.
2. *Уайт Т.* Hadoop: Подробное руководство. – СПб.: Питер, 2013. – 672 с.