Основы работы с Hadoop

Максим Губин

Томск

econophysica

HDFS: Командная строка

- Пользовательские Команды
  - **hdfs dfs** - запускает команды файловой системы на HDFS
  - **hdfs fsck** - запускает команду проверки файловой системы HDFS
- Команды администрирования
  - **hdfs dfsadmin** - запускает команды администрирования HDFS

econophysica

HDFS: Команды

- Показать содержимое директории

```
hdfs dfs -ls
hdfs dfs -ls /
hdfs dfs -ls -R /var
```

- Показать использование места на диске

```
hdfs dfs -du -h /
hdfs dfs -du /hbase/data/hbase/namespace/
hdfs dfs -du -h /hbase/data/hbase/namespace/
hdfs dfs -du -s /hbase/data/hbase/namespace/
```


econophysica

HDFS: Команды

Перенести данные в HDFS

```
hdfs dfs -mkdir mydata
hdfs dfs -ls
hdfs dfs -copyFromLocal data/somefile.avro mydata
hdfs dfs -ls -R
```

- Перенести данные обратно в локальную файловую систему

```
cd data/
hdfs dfs -copyToLocal mydata/somefile.avro data.avro
md5sum somefile.avro data.avro
```


econophysica

HDFS: Команды

- Посмотреть acl для файла

```
hdfs dfs -getfacl mydata/somefile.avro
```

- Посмотреть статистику для файла (%r – фактор репликации)

```
hdfs dfs -stat "%r" mydata/somefile.avro
```

- Писать в HDFS из stdin

```
echo “foo bar" | hdfs dfs -put - testdata/myfile.txt
hdfs dfs -ls –R
hdfs dfs -cat testdata/myfile.txt
```


econophysica

HDFS: Команды (fsck)

- Удалить файл

```
hdfs dfs -rm mydata/somefile.avro
hdfs dfs -ls -R
```

- Посмотреть расположение блоков файла

```
hdfs fsck mydata/somefile.avro -files -blocks
-locations
```

- Посмотреть список отсутствующих блоков и файлов, которым они принадлежат

```
hdfs fsck / -list-corruptfileblocks
```


econophysica

HDFS: Команды администрирования

- Запросить отчет об состоянии кластера

```
hdfs dfsadmin -report
```

- Дерево стоек и узлов в них

```
hdfs dfsadmin -printTopology
```

- Получить статус определенного узла данных

```
hdfs dfsadmin -getDatanodeInfo
localhost:50020
```


econophysica

HDFS: Команды

- Получить список namenode кластера

```
hdfs getconf -namenodes
```

- Сбросить образ файловой системы HDFS в XML-файл

```
cd /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current
hdfs oiv -i fsimage_0000000000000003388 -o
/tmp/fsimage.xml -p XML
```


econophysica

Пример обработки данных на Pig

```
Lines=LOAD ‘input/hadoop.log’ AS (line: chararray);
Words = FOREACH Lines GENERATE  FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE  group, COUNT(Words);
Results = ORDER Words BY Counts DESC;
Top5 = LIMIT Results 5;
STORE Top5 INTO /output/top5words;
```


econophysica

Доступ к Pig

Варианты:

- Пакетный режим: отправить скрипт напрямую
- Интерактивный режим: Grunt, командная строка
- Java-класс PigServer, JDBC-подобный интерфейс

Режимы исполнения:

- Локальный режим: pig –x local
- Режим Mapreduce: pig –x mapreduce

econophysica

Типы данных Pig

- Скалярные типы:
  - Int, long, float, double, boolean, null, chararray, bytearry;
- Сложные типы: field, tuple, bag, relation;
- field - это поле данных
- tuple - это упорядоченный набор полей
- bag - это коллекция кортежей
- Отношение это bag

Примеры:
Tuple - строка в базе данных
(0002576169, Том, 20, 4,0)
Сумка это Таблица или представление в базе данных
{(0002576169, Том, 20, 4,0),
 (0002576170, Mike, 20, 3.6),
 (0002576171 Люси, 19, 4,0),…. }

econophysica

Запуск сценариев Pig

- Локальный режим
  - Используется локальный хост и локальная файловая система
  - Ни Hadoop, ни HDFS не требуются
  - Полезно для прототипирования и отладки
- Режим MapReduce
  - Запуск на кластере Hadoop и HDFS
- Пакетный режим - запустить скрипт напрямую
  - pig -x local my_pig_script.pig
  - pig -x mapreduce my_pig_script.pig
- Интерактивный режим использует оболочку Pig для запуска скрипта
  - Grunt> Lines = LOAD '/input/input.txt' AS (строка: chararray);
  - Grunt> Unique = DISTINCT Lines;
  - Grunt> DUMP Unique;

econophysica

Операции Pig

- Загрузка данных
  - LOAD loads input data
  - Lines=LOAD ‘input/access.log’ AS (line: chararray);
- Проекция
  - FOREACH … GENERTE … (similar to SELECT)
  - Применяет набор преобразований к каждой записи.
- Группировка
  - GROUP группирует записи с одинаковым ключом
- Dump/Store
  - DUMP выводит результат на экран, STORE сохраняет на диск
- Аггрегации
  - AVG, COUNT, MAX, MIN, SUM

econophysica

Загрузка данных в Pig

```
students = load 'student.txt' using PigStorage('\t')
                        as (studentid: int, name:chararray, age:int, gpa:double);
```

- PigStorage: загружает / сохраняет отношения, используя текстовый формат с разделителями полей
- TextLoader: загружает отношения из простого текстового формата
- BinStorage: загружает / сохраняет отношения из или в двоичные файлы
- PigDump: хранит отношения, записывая представление toString () кортежей, по одному на строку

econophysica

Pig: FOREACH

```
studentid = FOREACH  students  GENERATE  studentid, name;
```

- The Foreach … перебирает элементы bag и преобразует их.
- В результате Foreach тоже получается bag
- Элементы называются так же, как во входном bag

econophysica

Позиционная нотация в Pig

```
students = LOAD 'student.txt' USING PigStorage() AS (name:chararray, age:int, gpa:float);
DUMP A;
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
studentname = Foreach students Generate $1 as studentname;
```

- Поля можно адресовать по их позиции.

econophysica

Pig: Group

```
B = GROUP A BY age;
C = COGROUP A BY name, B BY name;
```

- Группирует данные в одно или несколько отношений
- Операторы GROUP и COGROUP идентичны.
- Оба оператора работают с одним или несколькими отношениями.
- Для удобства чтения GROUP используется в выражениях, включающих одно отношение
- COGROUP используется в утверждениях, включающих два или более отношений.

econophysica

Pig: Dump&Store

```
A = LOAD ‘input/pig/multiquery/A’;
B = FILTER A by $1 == “apple”;
C = FILTER A by $1 == “apple”;
SOTRE B INTO “output/b”
STORE C INTO “output/c”
```

- Pig способен распознать, что B и C оба основаны на A, и сгенерировать A только один раз;
- Dump: выводит на экран;
- Store: сохраняет на диск.

econophysica

Pig: Count

```
X = FOREACH B GENERATE COUNT(A);
```

- Вычислить количество элементов в сумке
- Используйте функцию COUNT для вычисления количества элементов в сумке.
- Для COUNT требуется предыдущий оператор GROUP ALL для глобальных счетчиков и оператор GROUP BY для счетчиков групп.

econophysica

Pig: Order

```
student = ORDER students BY gpa DESC;
```

- Сортирует отношение на основе одного или нескольких полей
- В Pig отношения неупорядочены.
- Если вы сортируете отношение A для получения отношения X, отношения A и X по-прежнему содержат одинаковые элементы.

econophysica

Spark Context

- Главная точка входа в функциональность Spark
- Создан для вас в Spark shell как переменная sc
- В автономных программах вы создаете свои собственные контексты

```
from pyspark import SparkContext

sc = SparkContext(“masterUrl”, “name”, “sparkHome”, [“library.py”]))
```


econophysica

Spark: создаем RDD


```
# Превратить локальную коллекцию в RDD
sc.parallelize ([1, 2, 3])

# Загрузить текстовый файл из локальной FS, HDFS или S3
sc.textFile («file.txt»)
sc.textFile («Каталог / *. TXT»)
sc.textFile («HDFS: // NameNode: 9000 / путь / файл»)

# Использовать любой существующий Hadoop InputFormat
sc.hadoopFile (keyClass, valClass, inputFmt, conf)
```


econophysica

Spark: трансформации RDD

```
nums = sc.parallelize ([1, 2, 3])
# Пропустить каждый элемент через функцию
squares = nums.map (lambda x: x * x) # => {1, 4, 9}

# Отфильтровать по предикату
even = squares.filter (lambda x: x% 2 == 0) # => {4}

# Сопоставить каждый элементу нескольким другим
nums.flatMap(lambda x: range(0, x))  # => {0, 0, 1, 0, 1, 2}
```


econophysica

Spark: действия с RDD


```
nums = sc.parallelize([1, 2, 3])

# Взять значение RDD в локальную переменную
nums.collect() # => [1, 2, 3]

# Взять первые K элементов
nums.take(2)   # => [1, 2]

# Получить количество элементов
nums.count()   # => 3

# Соединить элементы ассоциативной функцией
nums.reduce(lambda x, y: x + y)  # => 6

# Сохранить результаты в текстовый файл

nums.saveAsTextFile(“hdfs://file.txt”)
```


econophysica

Задание: HDFS

- Подключиться к удаленной машине;
- Найти набор данных на удаленной машине командами HDFS;
- Оценить размер набора данных командой HDFS;
- Исследовать права доступа к набору данных командой HDFS;

econophysica

Задание: Pig

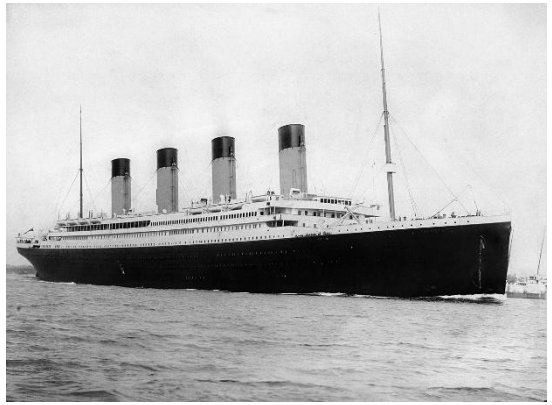
- Открыть оболочку командной строки Pig;
- Загрузить набор данных в Pig;
- Выполнить describe чтобы убедиться, что набор загружен;
- Отфильтровать пассажиров по значению пола: женский;
- Подсчитать количество выживших женщин.

Задание: Spark

- Открыть оболочку командной строки Spark;
- Загрузить набор данных в Spark;
- Отфильтровать пассажиров по значению пола: женский;
- Подсчитать количество выживших женщин.

econophysica

Описание набора данных



- **Pclass** Класс пассажира (1 = 1st; 2 = 2nd; 3 = 3rd)
- **survival** Выживание (0 = No; 1 = Yes)
- **Name** Имя
- **sex** Пол
- **Age** Возраст
- **Sibsp** Количество братьев/сестёр/супругов на борту
- **Parch** Количество детей/родителей на борту
- **Ticket** Номер билета
- **Fare** Стоимость проезда, в британских фунтах
- **Cabin** Каюта
- **Embarked** Порт посадки (C = Cherbourg; Q = Queenstown; S = Southampton)

econophysica

Спасибо за внимание!

mgubin@tpu.ru

econophysica

Подсказки: загрузить набор данных в Spark


```
raw = LOAD '/home/developer/datasets/titanic-passengers.csv'

USING PigStorage(',')

AS (PassengerId:int, Survived:boolean, Pclass:int, Name:chararray,

Sex:chararray, Age:int, SibSp:int, Parch:int, Ticket:chararray,

Fare:float, Cabin:chararray, Embarked:chararray);



passengers = FILTER raw by $3 != 'Name';
```


econophysica

Подсказки: загрузить набор данных в Spark

```
raw_data =
spark.read.csv('/home/developer/datasets/titanic-passengers.csv',
header = 'True', inferSchema='True')
```