**Лабораторная 3 Прудников**

# **Задание 1**

Скрипт hbase-prudnikov.sh\* загружает файлы из обычной файловой системы (не hdfs) в таблицу hbase. \*\*\*

Предполагается что скрипт лежит в той же папке, где и файлы.

Скрипт принимает 1 аргумент – имя загружаемого файла. Далее генерируется файл tmp\_pk\_filename, где все строчки нумеруются, то есть создается колонка primary key, которая в hbase будет использоваться как ключ. Далее создается tmp папка в hadoop fs и этот временный файл перемещается туда (для загрузки в hbase файлы должны находиться в hadoop fs). Далее запускается mapreduce job для создания структуры hbaseHfiles storefile из tmp файла, где указывается структура итоговой таблицы. Далее запускается job который загружает получившуюся структуру в hbase, после этого все временные директории и файлы можно удалить.

Последняя команда запускает job RowCounter для подсчета количества строк в загруженной таблице.

В hbase получается такая структура таблицы: HBASE\_ROW\_KEY, edge: source\_node\_id, edge: target\_node\_id, edge: weight, где edge - это column family, логическое объединение схожих по смыслу колонок и после: сами названия таблиц, так как одна строчка файла логически представляет из себя ребро графа, то я объединил колонки по этому признаку в hbase.

# **Задание 2**

Cкрипт Task3.scala\*\* (результат работы такой же, как и Task2) принимает 2 аргумента args(0) - имя таблицы, args(1) - путь к выходному файлу.

Скрипт подключается к hbase, далее указываются какие колонки из какой таблицы нужно будет сканировать. Далее получаем RDD из нашей таблицы, нас интересует часть с result, поэтому с помощью map забираем только её, она хранит в себе rowkey и отсканированные колонки, которые мы задали ранее.

Далее получаем обычное key-value rdd путем забора значений нужных колонок из объекта result, далее применяем reduceByKey и сохраняем результат в файл как п Task2.

Результаты работы Task2 и Task3 совпадают.

Время работы:

* для graph1:
  + Task2 - < 1min
  + Task3 (hbase) - < 1 min
* для graph2:
  + Task2 - ~2 min
  + Task3 (hbase) - ~5 min

Видно, что чтение графа2 из таблицы hbase работает дольше, чем из обычного файла, возможно потому что получившаяся таблица hbase весит 7.4Gb, а сам исходный файл всего 900 Mb

\* Скрипты работают на виртуальной машине cloudera-quickstart 5.8.0

\*\*Note: для компилирования Task3 я добавлял в pom.xml для maven зависимость:

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase</artifactId>

<version>0.90.2</version>

</dependency>

\*\*Note: в архиве решения находятся все файлы для запуска Task3 (папка src, pom, и модифицированные скрипты для запуска run1 run2)

\*\*\*Note: при загрузке graph2 скрипт hbase-prudnikov.sh «съедает» ~21+ Gb на жестком диске, а сама получившаяся таблица весит 7.4Gb