УДК 004.021

АНАЛИЗ ЭФФЕКТИВНОСТИ АЛГОРИТМА MAPREDUCE ПРИ РЕШЕНИИ ЗАДАЧИ ВЫЧИСЛЕНИЯ СУММАРНОЙ ВЫРУЧКИ ПРЕДПРИЯТИЙ РЕГИОНА С ПОМОЩЬЮ APACHE HADOOP

Диденко А.А., магистрант гр. ПИм-171, I курс
Научный руководитель: Пимонов А.Г., д.т.н., профессор
Кузбасский государственный технический университет имени Т.Ф. Горбачева
г. Кемерово

Введение. В современном обществе все чаще возникают проблемы анализа большого объема данных, таких как логи покупок, посещений интернет-сайтов, геоданных и много другого. Для решения данной проблемы компания Google разработала и выложила в открытый доступ алгоритм распределенных вычислений MapReduce [1], который послужил основой для свободно распространяемого проекта Apache Software Foundation, состоящего из набора утилит, библиотек и фреймворка для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов [2]. В данной статье представлены анализ и демонстрация работы данного алгоритма на примере решения задачи по вычислению суммарной выручки предприятий в регионе с помощью Apache Hadoop на кластере, поднятом в виртуальной машине.

Постановка задачи. Есть лог из 4 млн. транзакций, представленный в формате: `Date \t Time \t Region \t Product \t Amount \t Method \n`. Пример записей из лога представлен на рис. 1. Необходимо вычислить суммарную выручку предприятий в регионе.

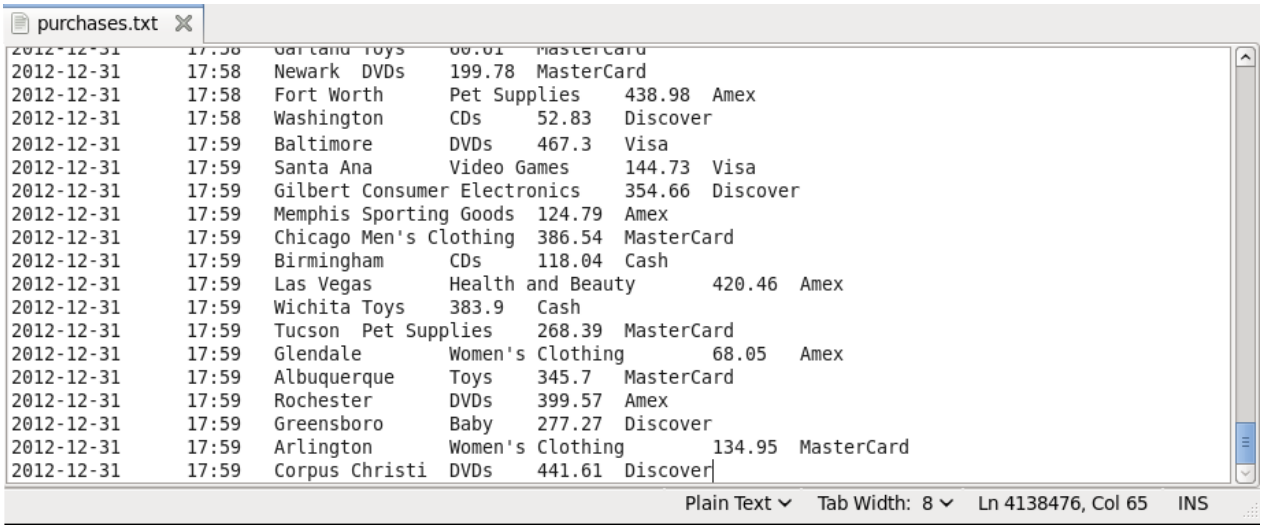

Рис. 1. Лог транзакций

Решение данной задачи стандартными методами программирования, например, с помощью программы на Java, которая парсит этот лог построчно и суммирует суммы покупок по регионам, занимает довольно большой

промежуток времени, а конкретно около 40 минут для моей системы. Это неприемлемый результат для такого относительно небольшого набора данных. Теперь решим эту же задачу, но уже с помощью алгоритма MapReduce. Данный алгоритм состоит из нескольких шагов [3]. Первый шаг – это применение функции `Map` к каждому элементу исходного лога. Функция возвращает либо ноль, либо создает объект, состоящий из пары значений `Key/Value`, в нашем случае `Key – Region/Value – Amount`. Примеры таких объектов: [`Washington/52.83`], [`Baltimore/467.3`], [`Washington/37.23`] и так далее. Следующим шагом алгоритм отсортирует все объекты методом `shuffle and sort`, а также создает новые экземпляры объектов, где все значения (`Value`) будут сгруппированы по ключу. Примеры таких объектов: [`Washington: 52.83, 37.23, …`], [`Baltimore: 467.3, …`] и так далее. Последним шагом будет выполнение функции `Reduce` для каждого сгенерированного объекта. С помощью этой функции будут просуммированы все элементы `Value` объекта, а результаты ее работы возвращены в коллекцию в формате `Key: Result`. Все три шага изображены на рис. 2. Примеры результирующих объектов: [`Washington: 90.06`], [`Baltimore: 467.3`] и так далее.

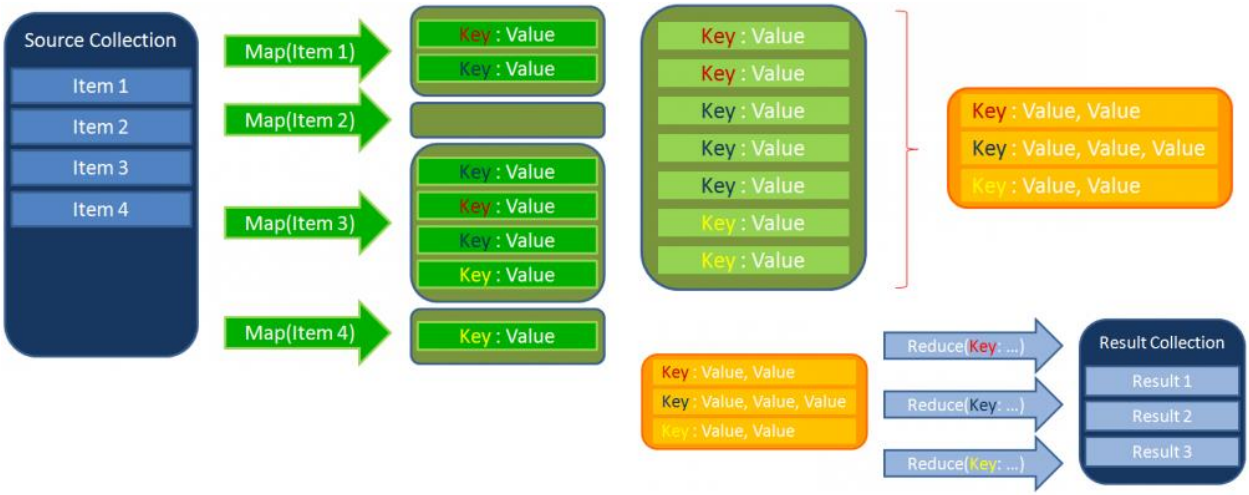

Рис. 2. Алгоритм MapReduce

Для реализации данного алгоритма был выбран язык программирования Python, а также был создан виртуальный кластер в программе Oracle VM VirtualBox. Код Map и Reduce для данной задачи представлен на рис. 3, 4.

Процесс и результаты решения задачи представлены на рис. 5. Кроме того во время решения данной задачи на виртуальном кластере собиралась статистика, представленная на рис. 6., из которой видно, что для решения было создано четыре параллельные задачи `map` и одна – `reduce`. Распределение времени, потраченного на копирование, сортировку и сборку в процентах представлено на последнем графике (рис. 6).

*mapper.py

```
import sys

for line in sys.stdin:
    data = line.strip().split("\t")
    if len(data) == 6:
        date, time, store, item, cost, payment = data
        print "{0}\t{1}".format(store, cost)
```

Рис. 3. Mapper.py

*reducer.py

```
#!/usr/bin/python

import sys

salesTotal = 0
oldKey = None

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisKey, thisSale = data_mapped

    if oldKey and oldKey != thisKey:
        print oldKey, "\t", salesTotal
        oldKey = thisKey;
        salesTotal = 0

    oldKey = thisKey
    salesTotal += float(thisSale)

if oldKey != None:
    print oldKey, "\t", salesTotal
```

Рис. 4. Reducer.py

Заключение. В данной статье был рассмотрен алгоритм MapReduce на примере решения задачи по вычислению суммарной выручки предприятий в регионе с помощью Apache Hadoop на виртуальном кластере. В результате было наглядно продемонстрировано, что для данного лога и конкретного компьютера алгоритм MapReduce в 30-35 раз эффективнее по времени, чем линейное решение, написанное на Java.

```
File  Edit  View  Search  Terminal  Help
[training@localhost code]$ hs mapper.py reducer.py myinput output2
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-training/hadoop-unjar62176556
484266683/] [] /tmp/streamjob7816135680932716611.jar tmpDir=null
18/03/13 15:16:30 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
18/03/13 15:16:30 WARN snappy.LoadSnappy: Snappy native library is available
18/03/13 15:16:30 INFO snappy.LoadSnappy: Snappy native library loaded
18/03/13 15:16:30 INFO mapred.FileInputFormat: Total input paths to process : 1
18/03/13 15:16:31 INFO streaming.StreamJob: getLocalDirs(): [/var/lib/hadoop-hdf
s/cache/training/mapred/local]
18/03/13 15:16:31 INFO streaming.StreamJob: Running job: job_201803131444_0004
18/03/13 15:16:31 INFO streaming.StreamJob: To kill this job, run:
18/03/13 15:16:31 INFO streaming.StreamJob: UNDEF/bin/hadoop job  -Dmapred.job.t
racker=0.0.0.0:8021 -kill job_201803131444_0004
18/03/13 15:16:31 INFO streaming.StreamJob: Tracking URL: http://0.0.0.0:50030/j
obdetails.jsp?jobid=job_201803131444_0004
18/03/13 15:16:32 INFO streaming.StreamJob:  map 0%  reduce 0%
18/03/13 15:16:45 INFO streaming.StreamJob:  map 6%  reduce 0%
18/03/13 15:16:46 INFO streaming.StreamJob:  map 13%  reduce 0%
18/03/13 15:16:48 INFO streaming.StreamJob:  map 18%  reduce 0%
18/03/13 15:16:49 INFO streaming.StreamJob:  map 23%  reduce 0%
18/03/13 15:16:51 INFO streaming.StreamJob:  map 28%  reduce 0%
18/03/13 15:16:52 INFO streaming.StreamJob:  map 33%  reduce 0%
18/03/13 15:16:54 INFO streaming.StreamJob:  map 39%  reduce 0%
18/03/13 15:16:55 INFO streaming.StreamJob:  map 45%  reduce 0%
18/03/13 15:16:57 INFO streaming.StreamJob:  map 50%  reduce 0%
18/03/13 15:17:04 INFO streaming.StreamJob:  map 75%  reduce 0%
18/03/13 15:17:07 INFO streaming.StreamJob:  map 90%  reduce 25%
18/03/13 15:17:10 INFO streaming.StreamJob:  map 100%  reduce 25%
18/03/13 15:17:16 INFO streaming.StreamJob:  map 100%  reduce 77%
18/03/13 15:17:19 INFO streaming.StreamJob:  map 100%  reduce 89%
18/03/13 15:17:23 INFO streaming.StreamJob:  map 100%  reduce 100%
18/03/13 15:17:24 INFO streaming.StreamJob: Job complete: job_201803131444_0004
18/03/13 15:17:24 INFO streaming.StreamJob: Output: output2
[training@localhost code]$ hadoop fs -ls output2
Found 3 items
-rw-r--r--   1 training supergroup          0 2018-03-13 15:17 output2/_SUCCESS
drwxr-xr-x   - training supergroup          0 2018-03-13 15:16 output2/_logs
-rw-r--r--   1 training supergroup       2296 2018-03-13 15:17 output2/part-0000
```

```
File  Edit  View  Search  Terminal  Help
Albuquerque     10052311.42
Anaheim         10076416.36
Anchorage       9933500.4
Arlington       10072207.97
Atlanta         9997146.7
Aurora  9992970.92
Austin  10057158.9
Bakersfield     10031208.92
Baltimore       10096521.45
Baton Rouge     10131273.23
Birmingham      10076606.52
Boise   10039166.74
Boston  10039473.28
Buffalo         10001941.19
Chandler        9919559.86
Charlotte       10112531.34
Chesapeake      10038504.92
Chicago         10062522.07
Chula Vista     9974951.34
Cincinnati      10139505.74
Cleveland       10067835.84
Colorado Springs        10061105.87
Columbus        10035241.03
Corpus Christi  9976522.77
Dallas  10066548.45
Denver  10031534.87
Detroit         9979260.76
Durham  10153890.21
El Paso         10016409.97
Fort Wayne      10132594.02
Fort Worth      10120830.65
Fremont         10053242.36
Fresno  9976260.26
Garland         10071043.92
Gilbert         10062115.19
Glendale        10044493.97
Greensboro      10033781.39
Henderson       10053416.05
:
```

Рис. 5. Процесс и результаты решения задачи алгоритмом MapReduce

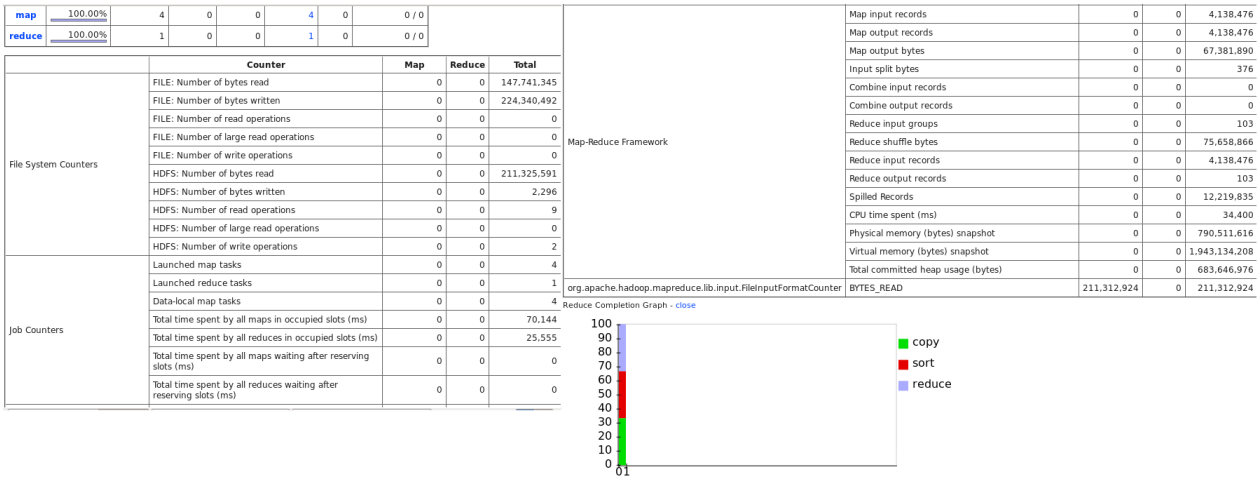

Рис. 6. Статистика решения задачи на виртуальном кластере

Список литературы:

1. Википедия MapReduce [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/MapReduce, свободный (дата обращения 22.03.2018).
2. Википедия Hadoop [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Hadoop, свободный (дата обращения: 22.03.2018).

3. Что такое MapReduce?] // Статья блога о разработке, программировании на C#/.NET, и не только [Электронный ресурс]. – Режим доступа: http://regfordev.blogspot.ru/2015/09/mapreduce.html#.WrQJDOjFKUm, свободный (дата обращения 22.03.2018).