MAPREDUCE 실습

학습 목표

- Python 으로 mapper와 reducer를 구현해 확인해 본다.
- cloudera 이용하여 mapreduce를 수행해 본다.

MapReduce 잡 실행

환경 :
Centos 6.3
Hadoop 2.x
Cloudera
vmware

01 Hadoop 버전 확인
$ hadoop version

```
[training@localhost ~]$ hadoop version
Hadoop 2.0.0-cdh4.1.1
Subversion file:///data/1/jenkins/workspace/generic-package-centos32-6/topdir/BUILD/hadoop-2.0.0-cdh4.1.1/src/hadoop-common-project/hadoop-common -r
581959ba23e4af85afd8db98b7687662fe9c5f20
Compiled by jenkins on Tue Oct 16 11:07:59 PDT 2012
From source with checksum 95f5c7f30b4030f1f327758e7b2bd61f
```

02 myinput 디렉터리 생성

```
hadoop  fs  -mkdir  myinput
hadoop  fs  -ls /user/training [user 명이 training일 경우]
```

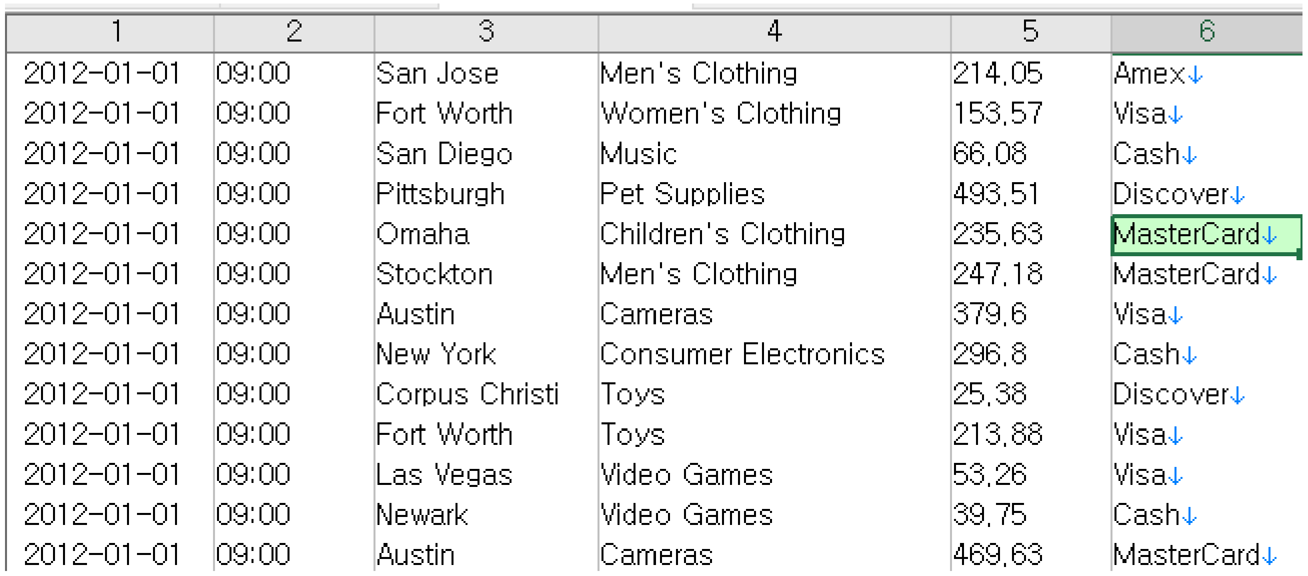
02 myinput에 purchases.txt를 넣기 (430만줄, 201MB)

# 디렉터리 이동 후, 파일을 myinput 으로 넣기

```
cd  /home/training/udacity_training/data
hadoop  fs   -put  purchases.txt  myinput
```

```
[training@localhost data]$ hadoop fs -ls myinput
Found 1 items
```

```
-rw-r--r--   1 training supergroup  211312924 2019-02-19 09:52 myinput/purchases.txt
```



1	2	3	4	5	6
2012-01-01	09:00	San Jose	Men's Clothing	214.05	Amex↓
2012-01-01	09:00	Fort Worth	Women's Clothing	153.57	Visa↓
2012-01-01	09:00	San Diego	Music	66.08	Cash↓
2012-01-01	09:00	Pittsburgh	Pet Supplies	493.51	Discover↓
2012-01-01	09:00	Omaha	Children's Clothing	235.63	MasterCard↓
2012-01-01	09:00	Stockton	Men's Clothing	247.18	MasterCard↓
2012-01-01	09:00	Austin	Cameras	379.6	Visa↓
2012-01-01	09:00	New York	Consumer Electronics	296.8	Cash↓
2012-01-01	09:00	Corpus Christi	Toys	25.38	Discover↓
2012-01-01	09:00	Fort Worth	Toys	213.88	Visa↓
2012-01-01	09:00	Las Vegas	Video Games	53.26	Visa↓
2012-01-01	09:00	Newark	Video Games	39.75	Cash↓
2012-01-01	09:00	Austin	Cameras	469.63	MasterCard↓

03 mapper and reducer 실행

(아래 명령줄은 한줄이다. 길기에 두줄로 분리시켜둠. 실제 명령넣을 때는 한줄로 넣어야 함)
[사용법]

hadoop jar, <a path to a jar> -mapper   <mapper>   -reducer   <reducer>   -file   <맵퍼 코드 파일>   -file   <리듀서 코드 파일>
-input   <HDFS의 입력디렉터리> -output <출력 데이터를 쓸 리듀서의 출력 디렉터리>

해당 소스 디렉터리로 이동 후, 명령 실행

# 디렉터리 이동

```
cd /home/training/udacity_training/code
```

# mapper와 reducer 실행

```
hadoop jar  /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.1.1.jar -mapper mapper.py  -reducer  reducer.py  -file  mapper.py  -file  reducer.py  -input  myinput  -output  joboutput
```

(* mapper.py, reducer.py 코드가 있는 곳에서 실행)

04 결과 확인

# 결과 확인

hadoop fs   -ls

# joboutput 폴더의 리스트 확인

hadoop fs -ls   joboutput

확인 파일
...   joboutput/_SUCCESS   => 성공적인 수행
....   joboutput/_logs   => job이 실행되는 동안 일어났던 정보 로그를 포함.
....   joboutput/part-00000 => 우리가 수행했던 Job(잡) 하나의 리듀서로부터의 결과 파일

[실행 결과]

```
[training@localhost code]$ hadoop fs  -ls
Found 2 items
drwxr-xr-x   - training supergroup          0 2018-05-07 06:31 joboutput
drwxr-xr-x   - training supergroup          0 2018-05-07 06:22 myinput
[training@localhost code]$ hadoop  fs  -ls   joboutput
Found 3 items
-rw-r--r--   1 training supergroup          0 2018-05-07 06:31 joboutput/_SUCCESS
drwxr-xr-x   - training supergroup          0 2018-05-07 06:27 joboutput/_logs
-rw-r--r--   1 training supergroup       2296 2018-05-07 06:31 joboutput/part-00000
```

# 생성된 파일의 내용 보기 (less : Enter 키 입력 한줄씩 출력, Space bar 한 화면씩 출력)
hadoop fs -cat joboutput/part-00000 | less

# HDFS의 생성된 결과를 내 pc의 txt파일로 가져오기
hadoop fs -get joboutput/part-00000   mylocfile.txt

```
[training@localhost code]$ head -n 20 mylocfile.txt
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
```

05 Hadoop job 명령 실행

# shell에 의한 mapper와 reducer의 실행

hs mapper.py reducer.py myinput joboutput

[실행 결과] joboutput 가 있어, 디렉터리가 존재한다고 함.

hs mapper.py reducer.py myinput newoutputdir

[결과]

```
[training@localhost code]$ hs mapper.py reducer.py myinput joboutput
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-training/hadoop-unjar9181741705364652401/] [] /tmp/streamjob8795202652725235274.jar tmpDir=null
19/02/19 10:06:54 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
19/02/19 10:06:55 INFO mapred.JobClient: Cleaning up the staging area hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_201902190936_0002
19/02/19 10:06:55 ERROR security.UserGroupInformation: PriviledgedActionException as:training (auth:SIMPLE) cause:org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://0.0.0.0:8020/user/training/joboutput already exists
19/02/19 10:06:55 ERROR streaming.StreamJob: Error launching job , Output path already exists : Output directory hdfs://0.0.0.0:8020/user/training/joboutput already exists
Streaming Command Failed!
```

# 참고 자료 hs

hs 스크립트 소스 코드 확인(

```
run_mapreduce ()
{
hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.1.1.jar -mapper $1 -reducer $2 -file $1 -file $2 -input $3 -output $4
}
```

[실행결과]

```
[training@localhost data]$ hadoop fs -mkdir myinput
[training@localhost data]$ hadoop fs -put purchases.txt myinput
[training@localhost data]$ hadoop fs -ls myinput
Found 1 items
-rw-r--r--   1 training supergroup  211312924 2018-05-07 06:22 myinput/purchases.txt
```

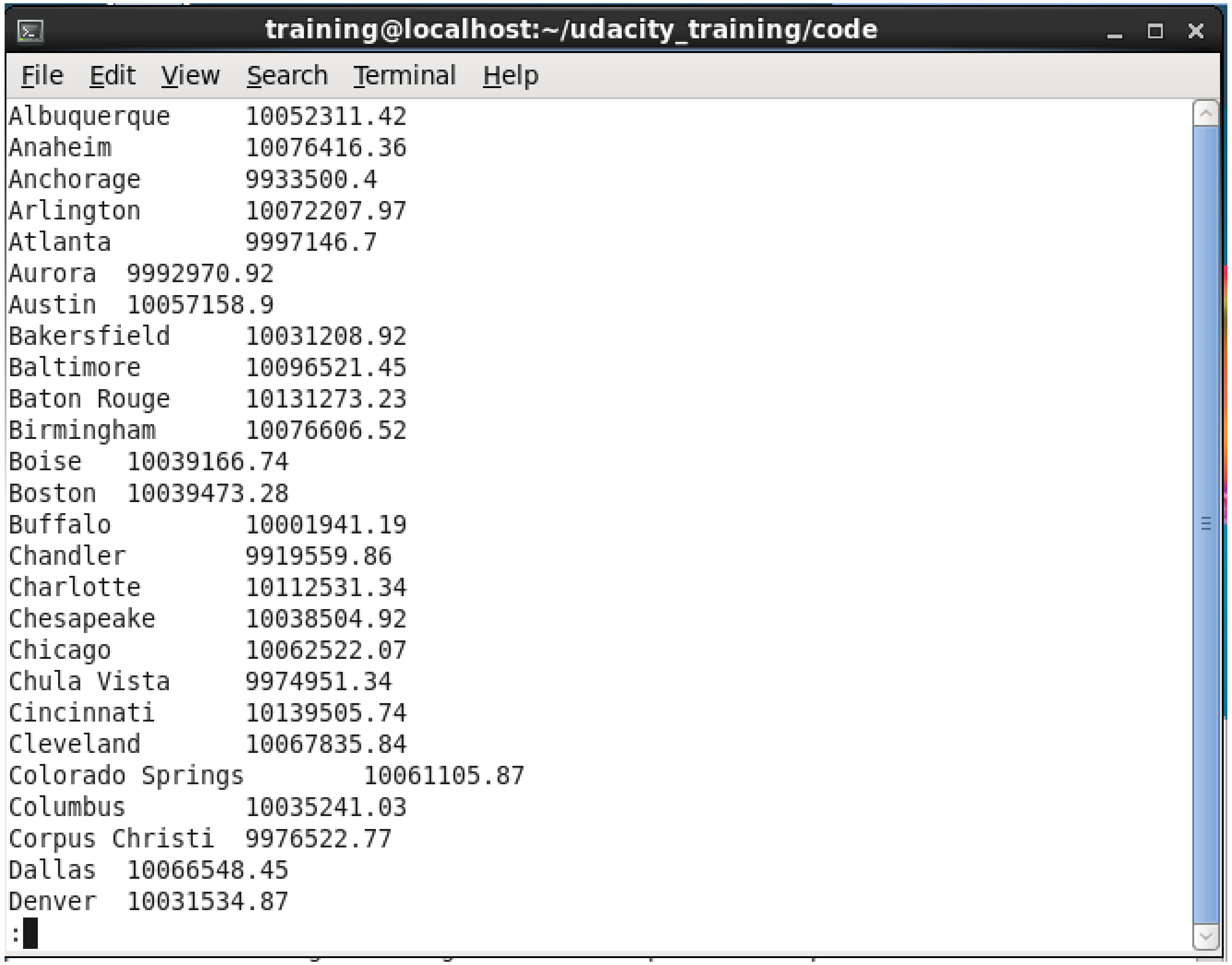
```
[training@localhost code]$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.1.1.jar -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py -input myinput -output joboutput
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-training/hadoop-unjar4354191662551748265/] [] /tmp/streamjob389135337284658968.jar tmpDir=null
18/05/07 06:27:23 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
18/05/07 06:27:24 WARN snappy.LoadSnappy: Snappy native library is available
18/05/07 06:27:24 INFO snappy.LoadSnappy: Snappy native library loaded
18/05/07 06:27:24 INFO mapred.FileInputFormat: Total input paths to process : 1
18/05/07 06:27:25 INFO streaming.StreamJob: getLocalDirs(): [/var/lib/hadoop-hdfs/cache/training/mapred/local]
18/05/07 06:27:25 INFO streaming.StreamJob: Running job: job_201805070616_0001
18/05/07 06:27:25 INFO streaming.StreamJob: To kill this job, run:
18/05/07 06:27:25 INFO streaming.StreamJob: UNDEF/bin/hadoop job  -Dmapred.job.tracker=0.0.0.0:8021 -kill job_201805070616_0001
18/05/07 06:27:25 INFO streaming.StreamJob: Tracking URL: http://0.0.0.0:50030/jobdetails.jsp?jobid=job_201805070616_0001
18/05/07 06:27:26 INFO streaming.StreamJob:  map 0%  reduce 0%
18/05/07 06:27:49 INFO streaming.StreamJob:  map 3%  reduce 0%
18/05/07 06:27:52 INFO streaming.StreamJob:  map 5%  reduce 0%
18/05/07 06:27:56 INFO streaming.StreamJob:  map 9%  reduce 0%
18/05/07 06:27:59 INFO streaming.StreamJob:  map 11%  reduce 0%
18/05/07 06:28:02 INFO streaming.StreamJob:  map 12%  reduce 0%
18/05/07 06:28:03 INFO streaming.StreamJob:  map 13%  reduce 0%
18/05/07 06:28:06 INFO streaming.StreamJob:  map 15%  reduce 0%
18/05/07 06:28:09 INFO streaming.StreamJob:  map 18%  reduce 0%
...
18/05/07 06:28:51 INFO streaming.StreamJob:  map 49%  reduce 0%
18/05/07 06:28:54 INFO streaming.StreamJob:  map 50%  reduce 0%
18/05/07 06:29:22 INFO streaming.StreamJob:  map 53%  reduce 8%
18/05/07 06:29:25 INFO streaming.StreamJob:  map 58%  reduce 8%
18/05/07 06:29:26 INFO streaming.StreamJob:  map 59%  reduce 17%
18/05/07 06:29:29 INFO streaming.StreamJob:  map 68%  reduce 17%
18/05/07 06:29:32 INFO streaming.StreamJob:  map 78%  reduce 17%
18/05/07 06:29:35 INFO streaming.StreamJob:  map 79%  reduce 17%
18/05/07 06:29:36 INFO streaming.StreamJob:  map 80%  reduce 17%
18/05/07 06:29:39 INFO streaming.StreamJob:  map 81%  reduce 25%
18/05/07 06:29:42 INFO streaming.StreamJob:  map 83%  reduce 25%
18/05/07 06:29:45 INFO streaming.StreamJob:  map 86%  reduce 25%
...
18/05/07 06:30:01 INFO streaming.StreamJob:  map 96%  reduce 25%
18/05/07 06:30:04 INFO streaming.StreamJob:  map 99%  reduce 25%
18/05/07 06:30:07 INFO streaming.StreamJob:  map 100%  reduce 25%
18/05/07 06:30:13 INFO streaming.StreamJob:  map 100%  reduce 33%
18/05/07 06:30:22 INFO streaming.StreamJob:  map 100%  reduce 67%
18/05/07 06:30:25 INFO streaming.StreamJob:  map 100%  reduce 69%
```

```
18/05/07 06:30:28 INFO streaming.StreamJob:  map 100%  reduce 71%
...
18/05/07 06:31:08 INFO streaming.StreamJob:  map 100%  reduce 93%
18/05/07 06:31:11 INFO streaming.StreamJob:  map 100%  reduce 95%
18/05/07 06:31:15 INFO streaming.StreamJob:  map 100%  reduce 97%
18/05/07 06:31:18 INFO streaming.StreamJob:  map 100%  reduce 98%
18/05/07 06:31:21 INFO streaming.StreamJob:  map 100%  reduce 100%
18/05/07 06:31:27 INFO streaming.StreamJob: Job complete: job_201805070616_0001
18/05/07 06:31:27 INFO streaming.StreamJob: Output: joboutput


[training@localhost code]$ hadoop fs  -ls
Found 2 items
drwxr-xr-x   - training supergroup          0 2018-05-07 06:31 joboutput
drwxr-xr-x   - training supergroup          0 2018-05-07 06:22 myinput
[training@localhost code]$ hadoop  fs  -ls   joboutput
Found 3 items
-rw-r--r--   1 training supergroup          0 2018-05-07 06:31 joboutput/_SUCCESS
drwxr-xr-x   - training supergroup          0 2018-05-07 06:27 joboutput/_logs
-rw-r--r--   1 training supergroup       2296 2018-05-07 06:31 joboutput/part-00000
```

hadoop fs -cat joboutput/part-00000 | less

[결과 파일]


training@localhost:~/udacity_training/code
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
:
```

```
[training@localhost code]$ hadoop  fs  -get  joboutput/part-00000   mylocfile.txt
[training@localhost code]$ ls -ltr
total 12
-rwxrwxr-x 1 training training  743 Sep  9  2013 reducer.py
-rwxrwxr-x 1 training training  424 Sep 10  2013 mapper.py
-rwxr-xr-x 1 training training 2296 May  7 06:38 mylocfile.txt
```

참고 python 소스 코드

# mapper.py

```
#!/usr/bin/python

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

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)
```

# reducer.py

```
#!/usr/bin/python

import sys

salesTotal = 0
oldKey = None

# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
#
# All the sales for a particular store will be presented,
# then the key will change and we'll be dealing with the next store

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        # Something has gone wrong. Skip this line.
        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
```

# 실습과제 1

위의 Hadoop의 MapReduce는 스토어별 총 합계를 구하였다.
(1) 이와 비슷하게 Mapper와 Reducer를 이용하여 item 별 총합계를 구해서 계산해 보자.
확인한 결과 화면을 캡쳐해 보자.

# 도전과제 1

(2) Item별 총 데이터 횟수를 구해보자.
(3) store별 총 데이터 횟수를 구해보자.

확인한 결과 화면을 캡쳐해 보자.