CPSC 4770/6770

Distributed and Cluster Computing

Lecture 15: Optimization

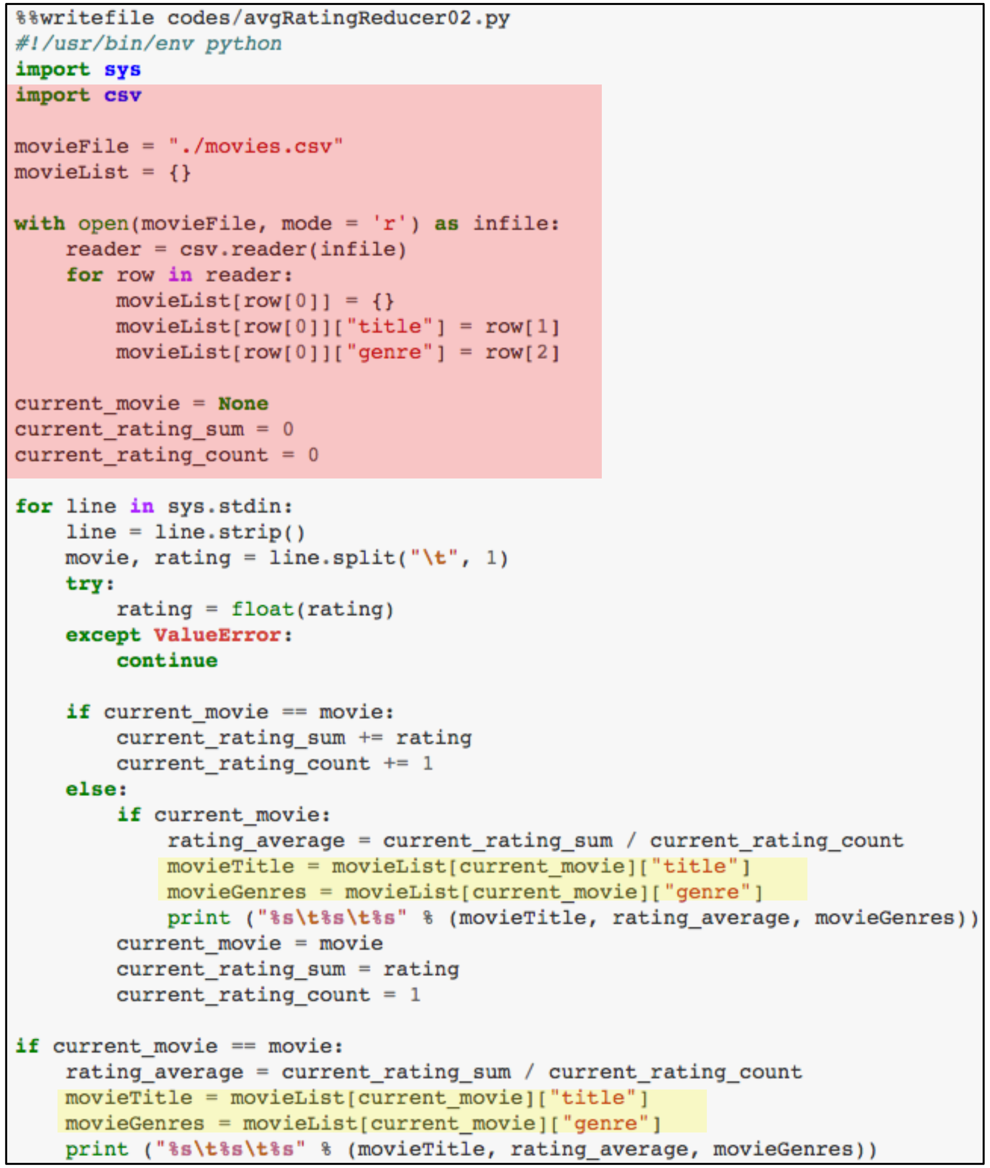
First Principle

- First principle of optimizing Hadoop workflow: **Reduce data movement in the shuffle phase**

```
hdfs dfs -rm -r intro-to-hadoop/output-movielens-02
mapred streaming \
    -input intro-to-hadoop/movielens/ratings.csv \
    -output intro-to-hadoop/output-movielens-02 \
    -file ./codes/avgRatingMapper04.py \
    -mapper avgRatingMapper04.py \
    -file ./codes/avgRatingReducer01.py \
    -reducer avgRatingReducer01.py \
    -file ./movielens/movies.csv
```

  - What is being passed from Map to Reduce?
  - Can reducer do the same thing as mapper, that is, to load in external data?
  - If we load external data on the reduce side, do we need to do so on the map side?

avgRatingReducer02.py

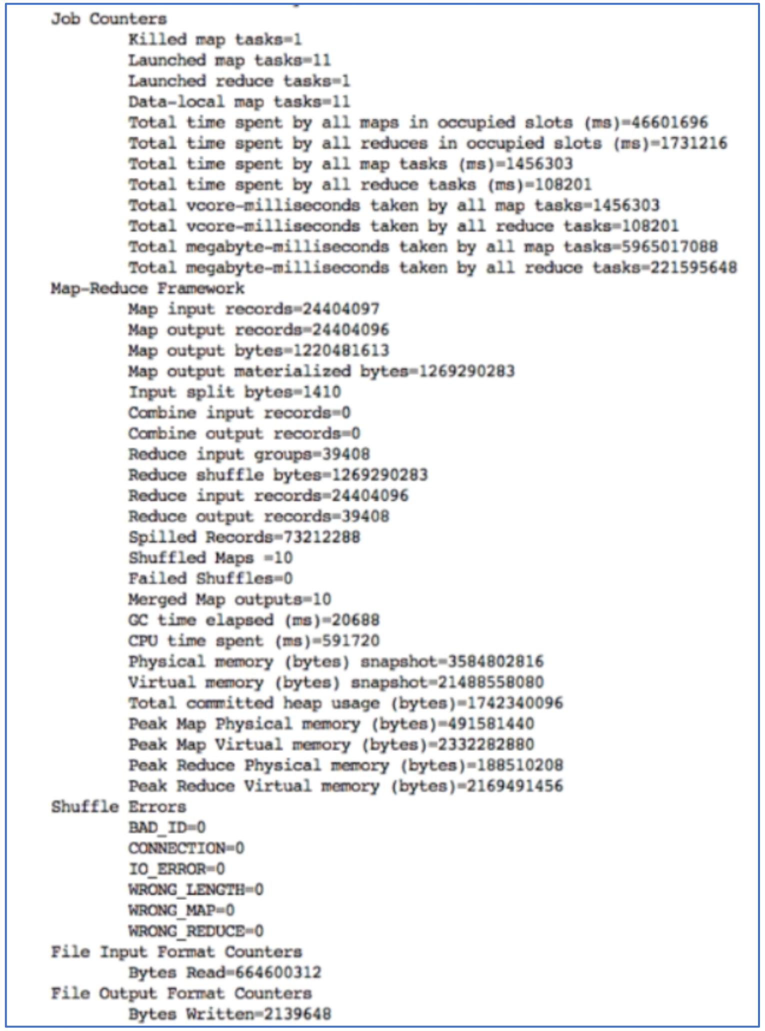


```
%%writefile codes/avgRatingReducer02.py
#!/usr/bin/env python
import sys
import csv

movieFile = "./movies.csv"
movieList = {}

with open(movieFile, mode = 'r') as infile:
    reader = csv.reader(infile)
    for row in reader:
        movieList[row[0]] = {}
        movieList[row[0]]["title"] = row[1]
        movieList[row[0]]["genre"] = row[2]

current_movie = None
current_rating_sum = 0
current_rating_count = 0

for line in sys.stdin:
    line = line.strip()
    movie, rating = line.split("\t", 1)
    try:
        rating = float(rating)
    except ValueError:
        continue

    if current_movie == movie:
        current_rating_sum += rating
        current_rating_count += 1
    else:
        if current_movie:
            rating_average = current_rating_sum / current_rating_count
            movieTitle = movieList[current_movie]["title"]
            movieGenres = movieList[current_movie]["genre"]
            print ("%s\t%s\t%s" % (movieTitle, rating_average, movieGenres))
        current_movie = movie
        current_rating_sum = rating
        current_rating_count = 1

if current_movie == movie:
    rating_average = current_rating_sum / current_rating_count
    movieTitle = movieList[current_movie]["title"]
    movieGenres = movieList[current_movie]["genre"]
    print ("%s\t%s\t%s" % (movieTitle, rating_average, movieGenres))
```

```
hdfs dfs -rm -r intro-to-hadoop/output-movielens-03
mapred streaming \
    -input /repository/movielens/ratings.csv \
    -output intro-to-hadoop/output-movielens-03 \
    -file avgRatingMapper02.py \
    -mapper avgRatingMapper02.py \
    -file avgRatingReducer02.py \
    -reducer avgRatingReducer02.py \
    -file ./movielens/movies.csv

hdfs dfs -ls intro-to-hadoop/output-movielens-02
hdfs dfs -ls intro-to-hadoop/output-movielens-03

hdfs dfs -cat intro-to-hadoop/output-movielens-03/part-00000 \
  2>/dev/null | head -n 10
```

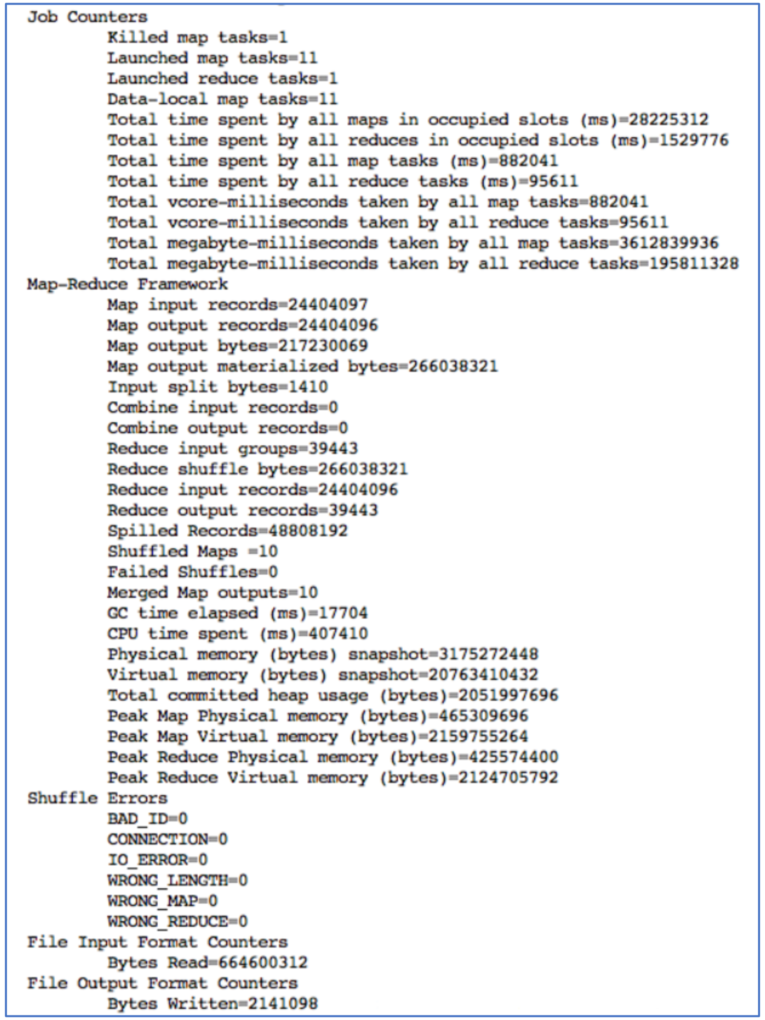
How does the number shuffle bytes in this example compare to the previous example?

Performance Comparison



```
    Job Counters
            Killed map tasks=1
            Launched map tasks=11
            Launched reduce tasks=1
            Data-local map tasks=11
            Total time spent by all maps in occupied slots (ms)=46601696
            Total time spent by all reduces in occupied slots (ms)=1731216
            Total time spent by all map tasks (ms)=1456303
            Total time spent by all reduce tasks (ms)=108201
            Total vcore-milliseconds taken by all map tasks=1456303
            Total vcore-milliseconds taken by all reduce tasks=108201
            Total megabyte-milliseconds taken by all map tasks=5965017088
            Total megabyte-milliseconds taken by all reduce tasks=221595648
    Map-Reduce Framework
            Map input records=24404097
            Map output records=24404096
            Map output bytes=1220481613
            Map output materialized bytes=1269290283
            Input split bytes=1410
            Combine input records=0
            Combine output records=0
            Reduce input groups=39408
            Reduce shuffle bytes=1269290283
            Reduce input records=24404096
            Reduce output records=39408
            Spilled Records=73212288
            Shuffled Maps =10
            Failed Shuffles=0
            Merged Map outputs=10
            GC time elapsed (ms)=20688
            CPU time spent (ms)=591720
            Physical memory (bytes) snapshot=3584802816
            Virtual memory (bytes) snapshot=21488558080
            Total committed heap usage (bytes)=1742340096
            Peak Map Physical memory (bytes)=491581440
            Peak Map Virtual memory (bytes)=2332282880
            Peak Reduce Physical memory (bytes)=188510208
            Peak Reduce Virtual memory (bytes)=2169491456
    Shuffle Errors
            BAD_ID=0
            CONNECTION=0
            IO_ERROR=0
            WRONG_LENGTH=0
            WRONG_MAP=0
            WRONG_REDUCE=0
    File Input Format Counters
            Bytes Read=664600312
    File Output Format Counters
            Bytes Written=2139648
```

Baseline run



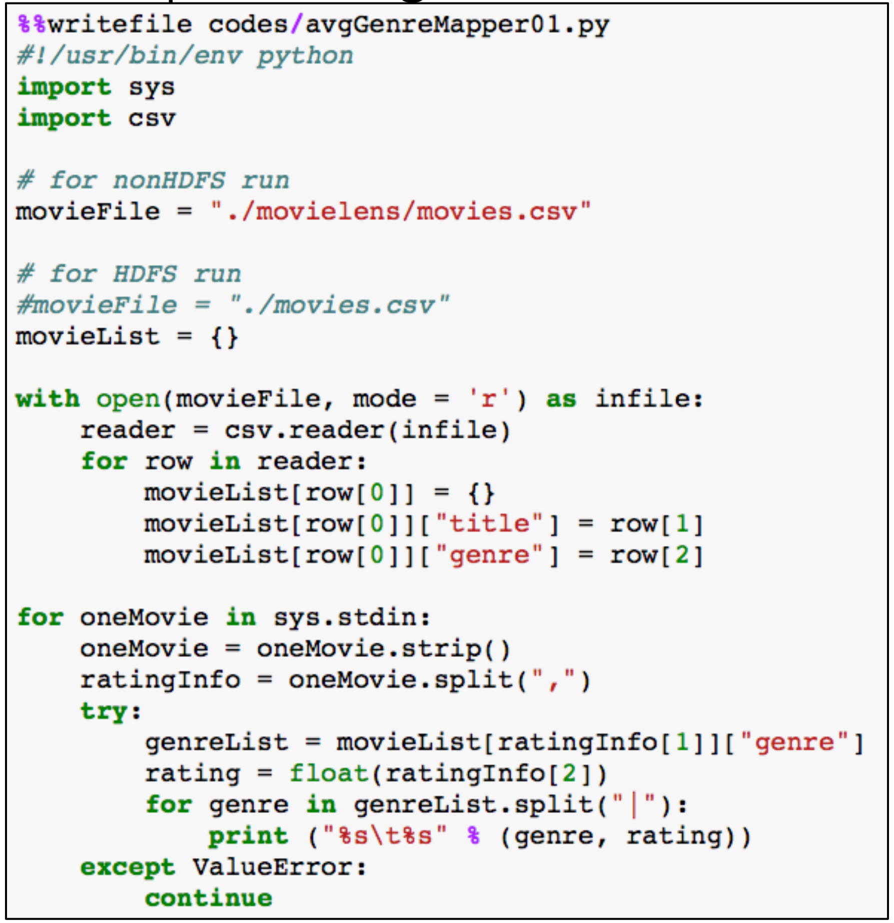
```
Job Counters
        Killed map tasks=1
        Launched map tasks=11
        Launched reduce tasks=1
        Data-local map tasks=11
        Total time spent by all maps in occupied slots (ms)=28225312
        Total time spent by all reduces in occupied slots (ms)=1529776
        Total time spent by all map tasks (ms)=882041
        Total time spent by all reduce tasks (ms)=95611
        Total vcore-milliseconds taken by all map tasks=882041
        Total vcore-milliseconds taken by all reduce tasks=95611
        Total megabyte-milliseconds taken by all map tasks=3612839936
        Total megabyte-milliseconds taken by all reduce tasks=195811328
Map-Reduce Framework
        Map input records=24404097
        Map output records=24404096
        Map output bytes=217230069
        Map output materialized bytes=266038321
        Input split bytes=1410
        Combine input records=0
        Combine output records=0
        Reduce input groups=39443
        Reduce shuffle bytes=266038321
        Reduce input records=24404096
        Reduce output records=39443
        Spilled Records=48808192
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=17704
        CPU time spent (ms)=407410
        Physical memory (bytes) snapshot=3175272448
        Virtual memory (bytes) snapshot=20763410432
        Total committed heap usage (bytes)=2051997696
        Peak Map Physical memory (bytes)=465309696
        Peak Map Virtual memory (bytes)=2159755264
        Peak Reduce Physical memory (bytes)=425574400
        Peak Reduce Virtual memory (bytes)=2124705792
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=664600312
File Output Format Counters
        Bytes Written=2141098
```

Optimized run

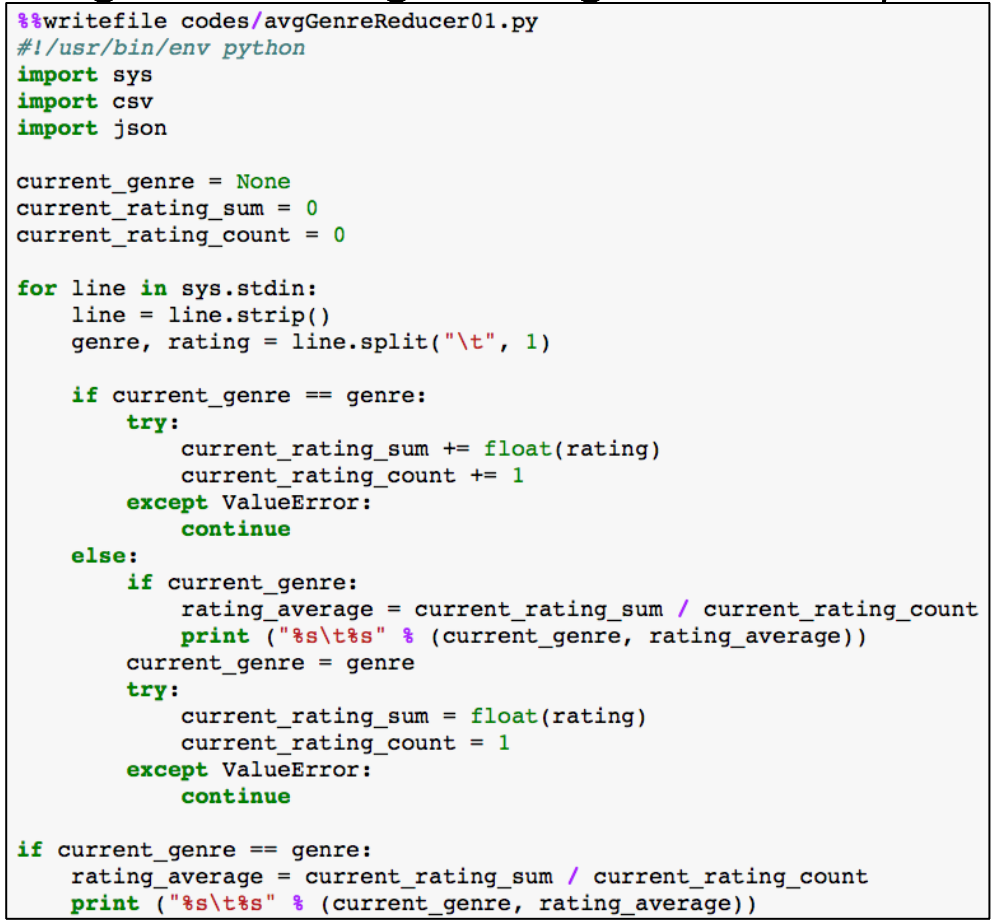
Common Optimization Approaches

- In-mapper reduction of key/value pairs
- Additional combiner function

Example: Find genres which have the highest average ratings over the years



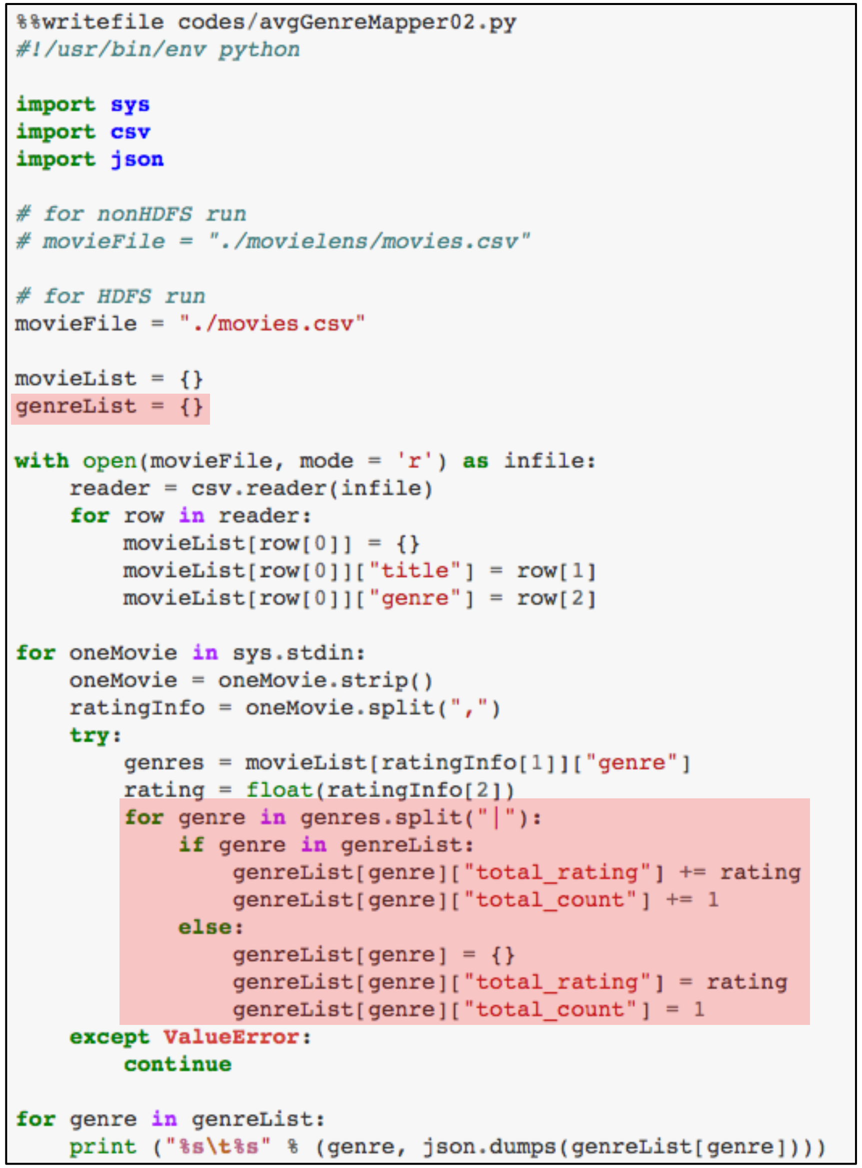
```
%%writefile codes/avgGenreMapper01.py
#!/usr/bin/env python
import sys
import csv

# for nonHDFS run
movieFile = "./movielens/movies.csv"

# for HDFS run
#movieFile = "./movies.csv"
movieList = {}

with open(movieFile, mode = 'r') as infile:
    reader = csv.reader(infile)
    for row in reader:
        movieList[row[0]] = {}
        movieList[row[0]]["title"] = row[1]
        movieList[row[0]]["genre"] = row[2]

for oneMovie in sys.stdin:
    oneMovie = oneMovie.strip()
    ratingInfo = oneMovie.split(",")
    try:
        genreList = movieList[ratingInfo[1]]["genre"]
        rating = float(ratingInfo[2])
        for genre in genreList.split("|"):
            print ("%s\t%s" % (genre, rating))
    except ValueError:
        continue
```



```
%%writefile codes/avgGenreReducer01.py
#!/usr/bin/env python
import sys
import csv
import json

current_genre = None
current_rating_sum = 0
current_rating_count = 0

for line in sys.stdin:
    line = line.strip()
    genre, rating = line.split("\t", 1)

    if current_genre == genre:
        try:
            current_rating_sum += float(rating)
            current_rating_count += 1
        except ValueError:
            continue
    else:
        if current_genre:
            rating_average = current_rating_sum / current_rating_count
            print ("%s\t%s" % (current_genre, rating_average))
        current_genre = genre
        try:
            current_rating_sum = float(rating)
            current_rating_count = 1
        except ValueError:
            continue

if current_genre == genre:
    rating_average = current_rating_sum / current_rating_count
    print ("%s\t%s" % (current_genre, rating_average))
```

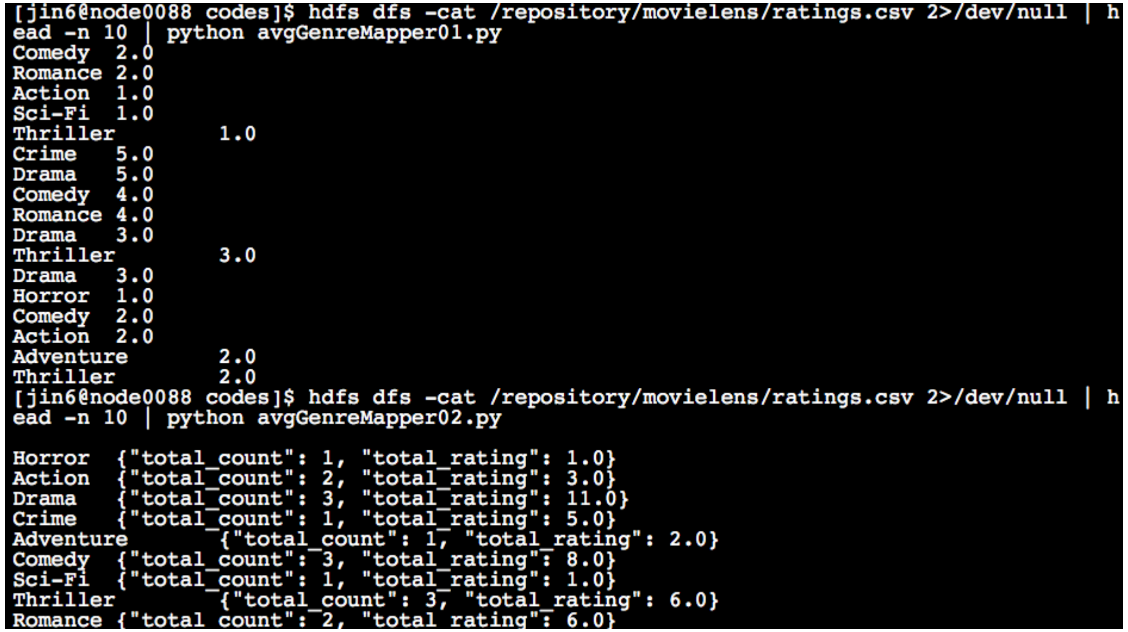
- cat –n codes/avgGenreMapper01.py
- cat –n codes/avgGenreReducer01.py
- hdfs dfs -rm -r intro-to-hadoop/output-movielens-04
- mapred streaming **-input** intro-to-hadoop/movielens/ratings.csv **-output** intro-to-hadoop/output-movielens-04 **-file** ./codes/avgGenreMapper01.py **-mapper** avgGenreMapper01.py **-file** ./avgGenreReducer01.py **-reducer** avgGenreReducer01.py **-file** ./movielens/movies.csv

Optimization through in-mapper reduction of key/value pairs



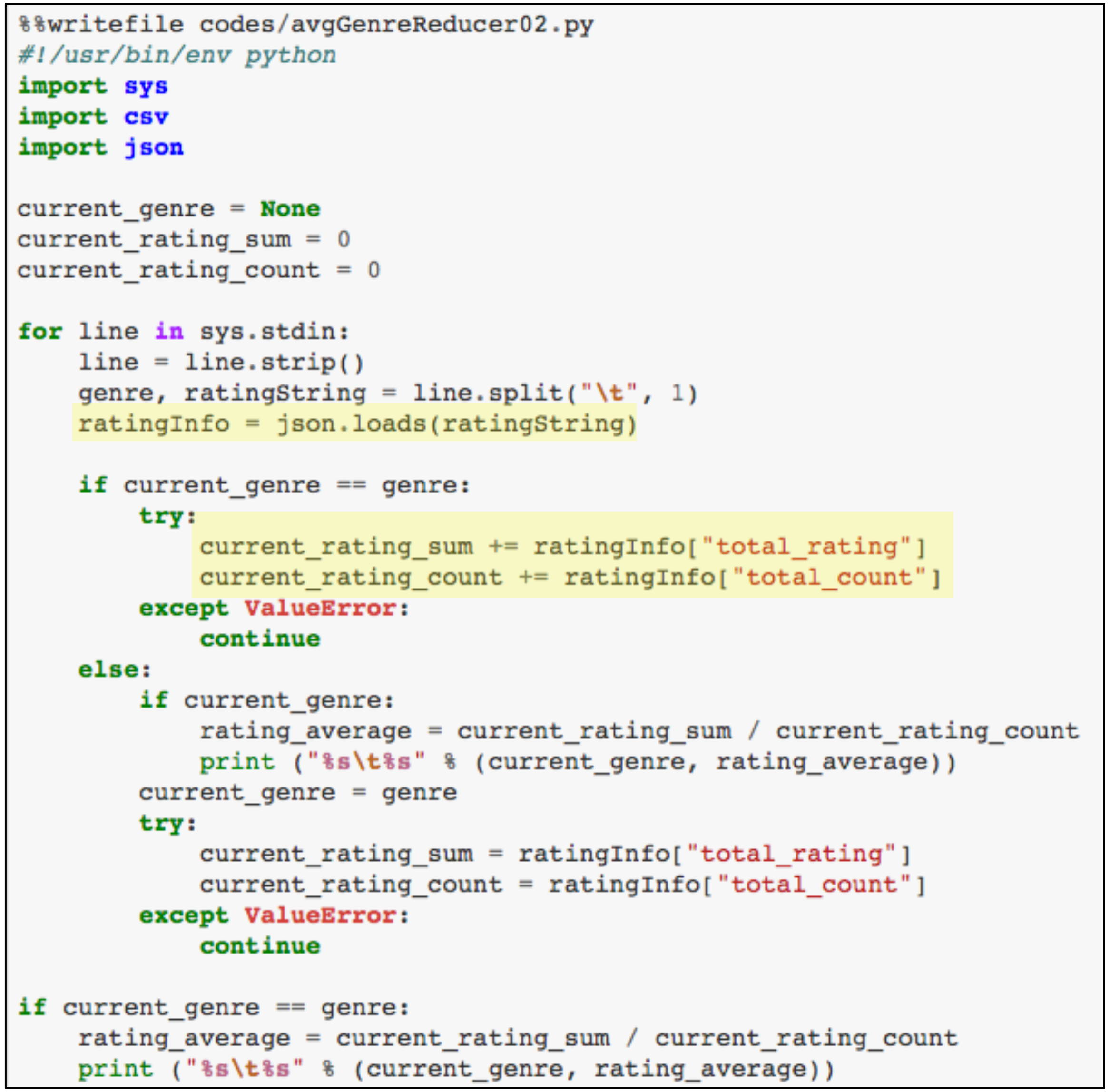
```
%%writefile codes/avgGenreMapper02.py
#!/usr/bin/env python

import sys
import csv
import json

# for nonHDFS run
# movieFile = "./movielens/movies.csv"

# for HDFS run
movieFile = "./movies.csv"

movieList = {}
genreList = {}

with open(movieFile, mode = 'r') as infile:
    reader = csv.reader(infile)
    for row in reader:
        movieList[row[0]] = {}
        movieList[row[0]]["title"] = row[1]
        movieList[row[0]]["genre"] = row[2]

for oneMovie in sys.stdin:
    oneMovie = oneMovie.strip()
    ratingInfo = oneMovie.split(",")
    try:
        genres = movieList[ratingInfo[1]]["genre"]
        rating = float(ratingInfo[2])
        for genre in genres.split("|"):
            if genre in genreList:
                genreList[genre]["total_rating"] += rating
                genreList[genre]["total_count"] += 1
            else:
                genreList[genre] = {}
                genreList[genre]["total_rating"] = rating
                genreList[genre]["total_count"] = 1
    except ValueError:
        continue

for genre in genreList:
    print ("%s\t%s" % (genre, json.dumps(genreList[genre])))
```

- cat -n codes/avgGenreMapper01.py
- Make sure to use nonHDFS run in the code avgGenreMapper01.py for the following command:
  - hdfs dfs -cat intro-to-hadoop/movielens/ratings.csv 2>/dev/null \
    | head -n 10 | python avgGenreMapper01.py
- hdfs dfs -cat /repository/movielens/ratings.csv 2>/dev/null | head -n 10 | python avgGenreMapper02.py



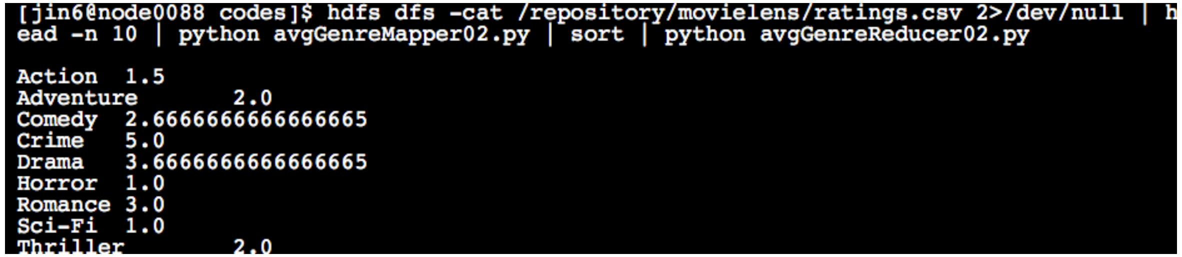
```
[jin6@node0088 codes]$ hdfs dfs -cat /repository/movielens/ratings.csv 2>/dev/null | h
ead -n 10 | python avgGenreMapper01.py
Comedy  2.0
Romance 2.0
Action  1.0
Sci-Fi  1.0
Thriller        1.0
Crime   5.0
Drama   5.0
Comedy  4.0
Romance 4.0
Drama   3.0
Thriller        3.0
Drama   3.0
Horror  1.0
Comedy  2.0
Action  2.0
Adventure       2.0
Thriller        2.0
[jin6@node0088 codes]$ hdfs dfs -cat /repository/movielens/ratings.csv 2>/dev/null | h
ead -n 10 | python avgGenreMapper02.py

Horror  {"total_count": 1, "total_rating": 1.0}
Action  {"total_count": 2, "total_rating": 3.0}
Drama   {"total_count": 3, "total_rating": 11.0}
Crime   {"total_count": 1, "total_rating": 5.0}
Adventure       {"total_count": 1, "total_rating": 2.0}
Comedy  {"total_count": 3, "total_rating": 8.0}
Sci-Fi  {"total_count": 1, "total_rating": 1.0}
Thriller        {"total_count": 3, "total_rating": 6.0}
Romance {"total_count": 2, "total_rating": 6.0}
```

Optimization through in-mapper reduction of key/value pairs (cont.)



```
%%writefile codes/avgGenreReducer02.py
#!/usr/bin/env python
import sys
import csv
import json

current_genre = None
current_rating_sum = 0
current_rating_count = 0

for line in sys.stdin:
    line = line.strip()
    genre, ratingString = line.split("\t", 1)
    ratingInfo = json.loads(ratingString)

    if current_genre == genre:
        try:
            current_rating_sum += ratingInfo["total_rating"]
            current_rating_count += ratingInfo["total_count"]
        except ValueError:
            continue
    else:
        if current_genre:
            rating_average = current_rating_sum / current_rating_count
            print ("%s\t%s" % (current_genre, rating_average))
        current_genre = genre
        try:
            current_rating_sum = ratingInfo["total_rating"]
            current_rating_count = ratingInfo["total_count"]
        except ValueError:
            continue

if current_genre == genre:
    rating_average = current_rating_sum / current_rating_count
    print ("%s\t%s" % (current_genre, rating_average))
```

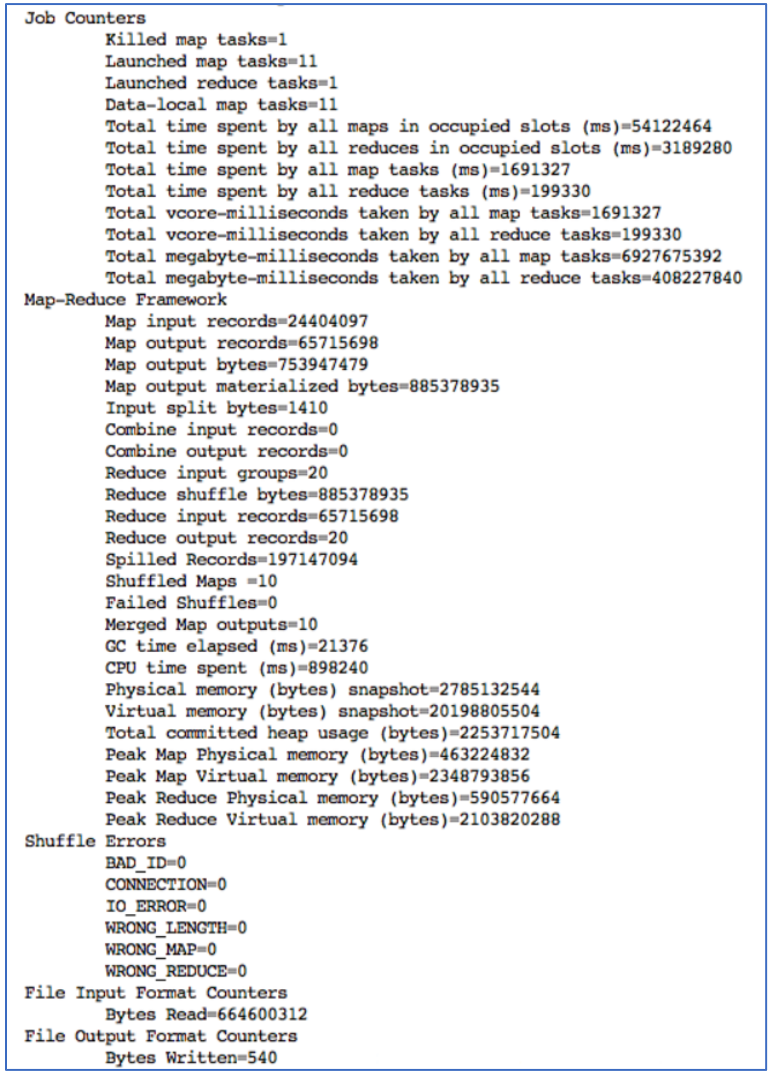
- cat -n codes/avgGenreReducer02.py
- hdfs dfs -cat intro-to-hadoop/movielens/ratings.csv 2>/dev/null | head -n 10 | python avgGenreMapper02.py | sort | python avgGenreReducer02.py



```
[jin6@node0088 codes]$ hdfs dfs -cat /repository/movielens/ratings.csv 2>/dev/null | h
ead -n 10 | python avgGenreMapper02.py | sort | python avgGenreReducer02.py

Action  1.5
Adventure       2.0
Comedy  2.6666666666666665
Crime   5.0
Drama   3.6666666666666665
Horror  1.0
Romance 3.0
Sci-Fi  1.0
Thriller        2.0
```

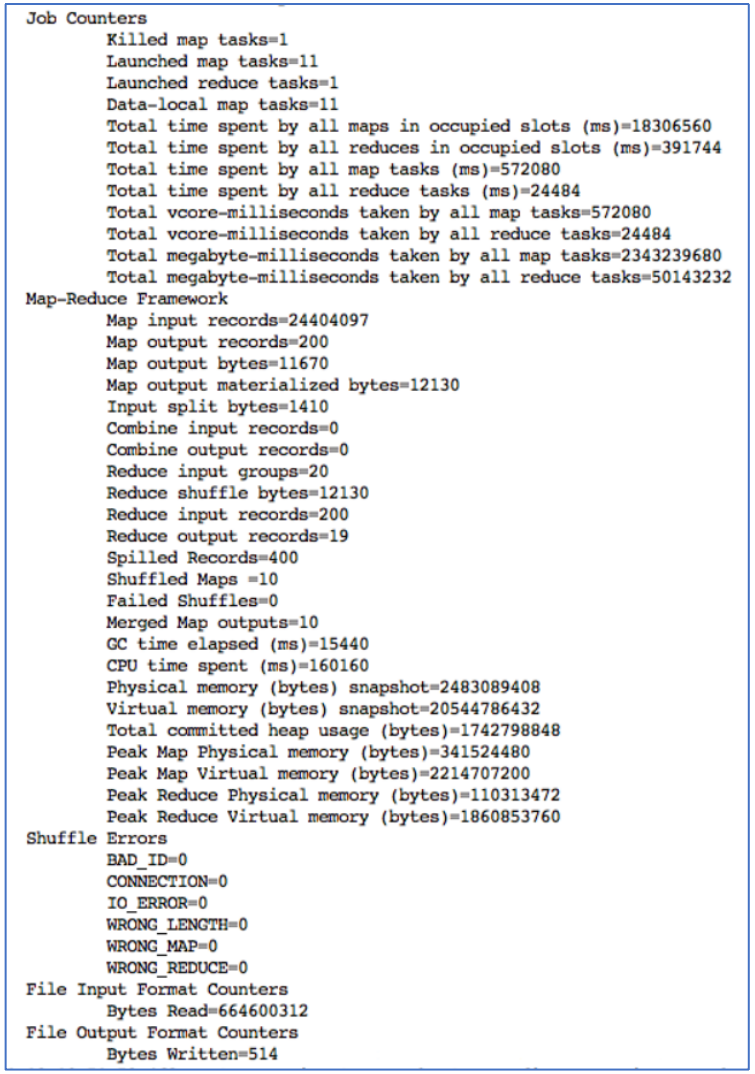
- hdfs dfs -rm -R intro-to-hadoop/output-movielens-05
- mapred streaming **-input** intro-to-hadoop/movielens/ratings.csv **-output** intro-to-hadoop/output-movielens-05 **-file** ./codes/avgGenreMapper02.py **-mapper** avgGenreMapper02.py **-file** ./codes/avgGenreReducer02.py -reducer avgGenreReducer02.py **-file** ./movielens/movies.csv
- hdfs dfs -cat intro-to-hadoop/output-movielens-05/part-00000
- hdfs dfs -cat intro-to-hadoop/output-movielens-04/part-00000
- How different are the number of shuffle bytes between the two jobs?

Performance Comparison



```
Job Counters
        Killed map tasks=1
        Launched map tasks=11
        Launched reduce tasks=1
        Data-local map tasks=11
        Total time spent by all maps in occupied slots (ms)=54122464
        Total time spent by all reduces in occupied slots (ms)=3189280
        Total time spent by all map tasks (ms)=1691327
        Total time spent by all reduce tasks (ms)=199330
        Total vcore-milliseconds taken by all map tasks=1691327
        Total vcore-milliseconds taken by all reduce tasks=199330
        Total megabyte-milliseconds taken by all map tasks=6927675392
        Total megabyte-milliseconds taken by all reduce tasks=408227840
Map-Reduce Framework
        Map input records=24404097
        Map output records=65715698
        Map output bytes=753947479
        Map output materialized bytes=885378935
        Input split bytes=1410
        Combine input records=0
        Combine output records=0
        Reduce input groups=20
        Reduce shuffle bytes=885378935
        Reduce input records=65715698
        Reduce output records=20
        Spilled Records=197147094
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=21376
        CPU time spent (ms)=898240
        Physical memory (bytes) snapshot=2785132544
        Virtual memory (bytes) snapshot=20198805504
        Total committed heap usage (bytes)=2253717504
        Peak Map Physical memory (bytes)=463224832
        Peak Map Virtual memory (bytes)=2348793856
        Peak Reduce Physical memory (bytes)=590577664
        Peak Reduce Virtual memory (bytes)=2103820288
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=664600312
File Output Format Counters
        Bytes Written=540
```

Baseline run



```
Job Counters
        Killed map tasks=1
        Launched map tasks=11
        Launched reduce tasks=1
        Data-local map tasks=11
        Total time spent by all maps in occupied slots (ms)=18306560
        Total time spent by all reduces in occupied slots (ms)=391744
        Total time spent by all map tasks (ms)=572080
        Total time spent by all reduce tasks (ms)=24484
        Total vcore-milliseconds taken by all map tasks=572080
        Total vcore-milliseconds taken by all reduce tasks=24484
        Total megabyte-milliseconds taken by all map tasks=2343239680
        Total megabyte-milliseconds taken by all reduce tasks=50143232
Map-Reduce Framework
        Map input records=24404097
        Map output records=200
        Map output bytes=11670
        Map output materialized bytes=12130
        Input split bytes=1410
        Combine input records=0
        Combine output records=0
        Reduce input groups=20
        Reduce shuffle bytes=12130
        Reduce input records=200
        Reduce output records=19
        Spilled Records=400
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=15440
        CPU time spent (ms)=160160
        Physical memory (bytes) snapshot=2483089408
        Virtual memory (bytes) snapshot=20544786432
        Total committed heap usage (bytes)=1742798848
        Peak Map Physical memory (bytes)=341524480
        Peak Map Virtual memory (bytes)=2214707200
        Peak Reduce Physical memory (bytes)=110313472
        Peak Reduce Virtual memory (bytes)=1860853760
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=664600312
File Output Format Counters
        Bytes Written=514
```

Optimized run

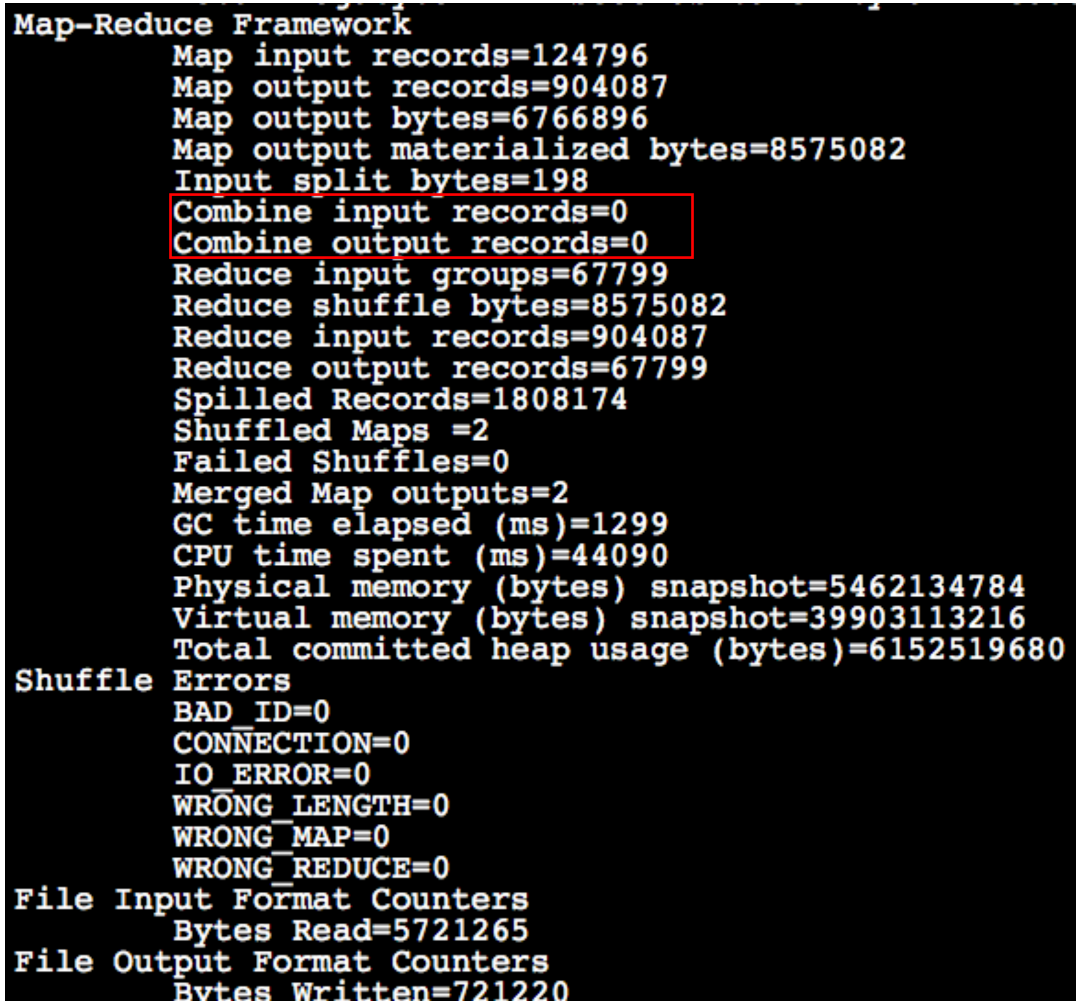
Optimization through combiner function

- The combiner in MapReducer is also known as ‘Mini-reducer’. The primary job of Combiner is to process the output data from the mapper, before passing it to Reducer. It runs after the mapper and before the Reducer and its use is optional.

-
  ```
  mapred streaming -input intro-to-hadoop/complete-shakespeare.txt -output intro-to-hadoop/output-wordcount-01 -file ./codes/wcMapper.py -mapper wcMapper.py -file ./codes/wcReducer.py -reducer wcReducer.py
  ```

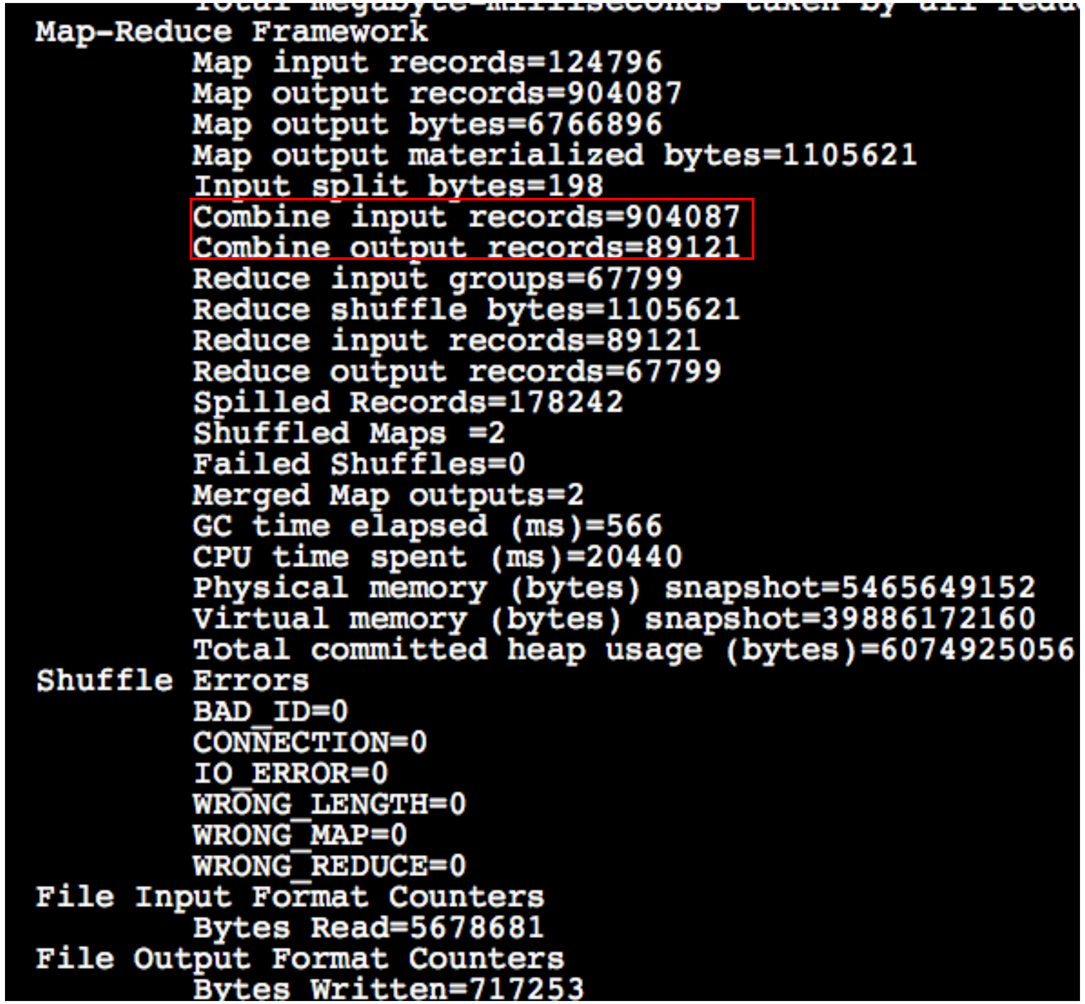
-
  ```
  mapred streaming -input intro-to-hadoop/complete-shakespeare.txt -output intro-to-hadoop/output-wordcount-02 -file ./codes/wcMapper.py -mapper wcMapper.py -file ./codes/wcReducer.py -reducer wcReducer.py -combiner wcReducer.py
  ```

Optimization through combiner function (Cont.)



```
Map-Reduce Framework
        Map input records=124796
        Map output records=904087
        Map output bytes=6766896
        Map output materialized bytes=8575082
        Input split bytes=198
        Combine input records=0
        Combine output records=0
        Reduce input groups=67799
        Reduce shuffle bytes=8575082
        Reduce input records=904087
        Reduce output records=67799
        Spilled Records=1808174
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=1299
        CPU time spent (ms)=44090
        Physical memory (bytes) snapshot=5462134784
        Virtual memory (bytes) snapshot=39903113216
        Total committed heap usage (bytes)=6152519680
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=5721265
File Output Format Counters
        Bytes Written=721220
```

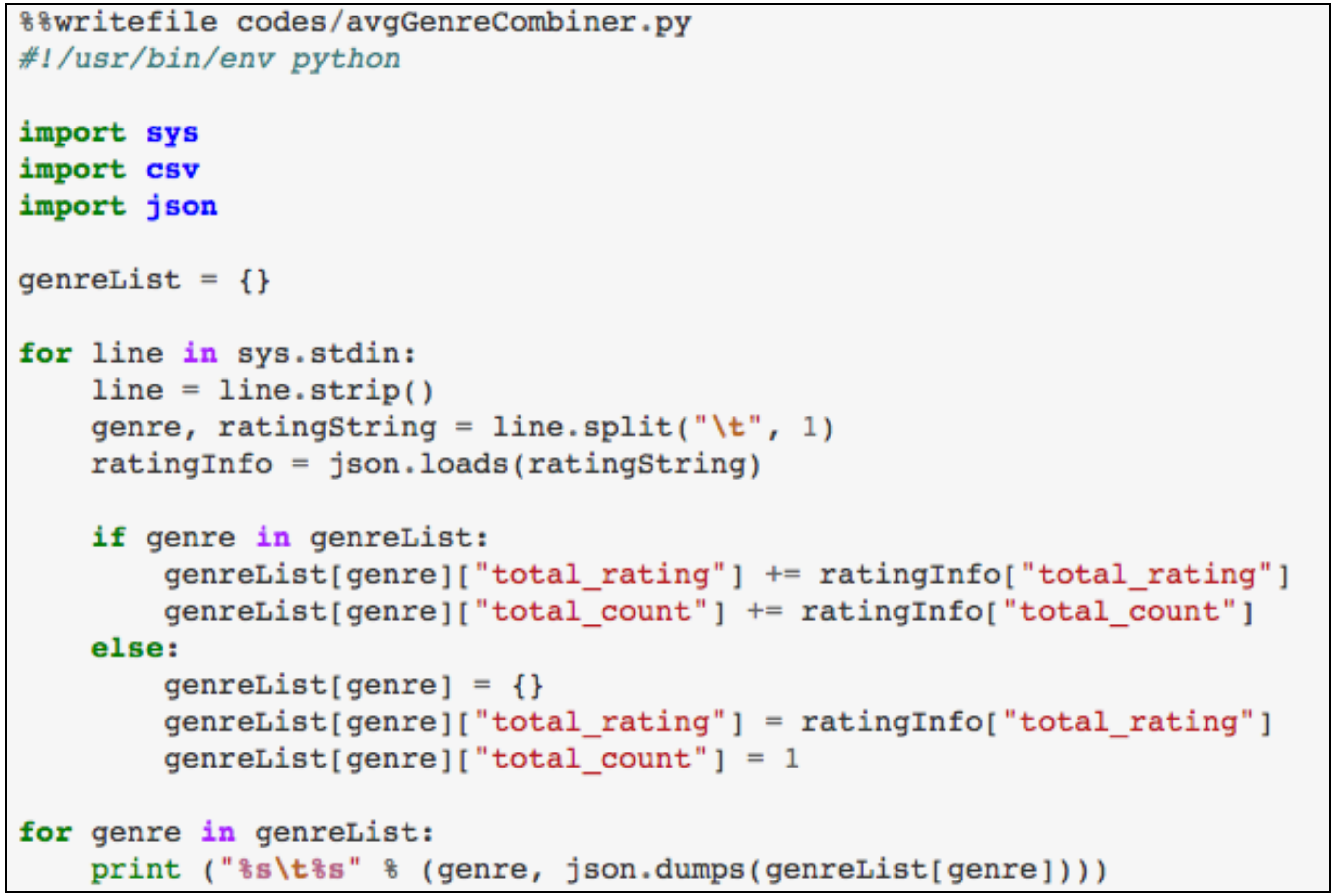
Without combiner



```
        Total megabyte-milliseconds taken by all redu
Map-Reduce Framework
        Map input records=124796
        Map output records=904087
        Map output bytes=6766896
        Map output materialized bytes=1105621
        Input split bytes=198
        Combine input records=904087
        Combine output records=89121
        Reduce input groups=67799
        Reduce shuffle bytes=1105621
        Reduce input records=89121
        Reduce output records=67799
        Spilled Records=178242
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=566
        CPU time spent (ms)=20440
        Physical memory (bytes) snapshot=5465649152
        Virtual memory (bytes) snapshot=39886172160
        Total committed heap usage (bytes)=6074925056
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=5678681
File Output Format Counters
        Bytes Written=717253
```

With combiner

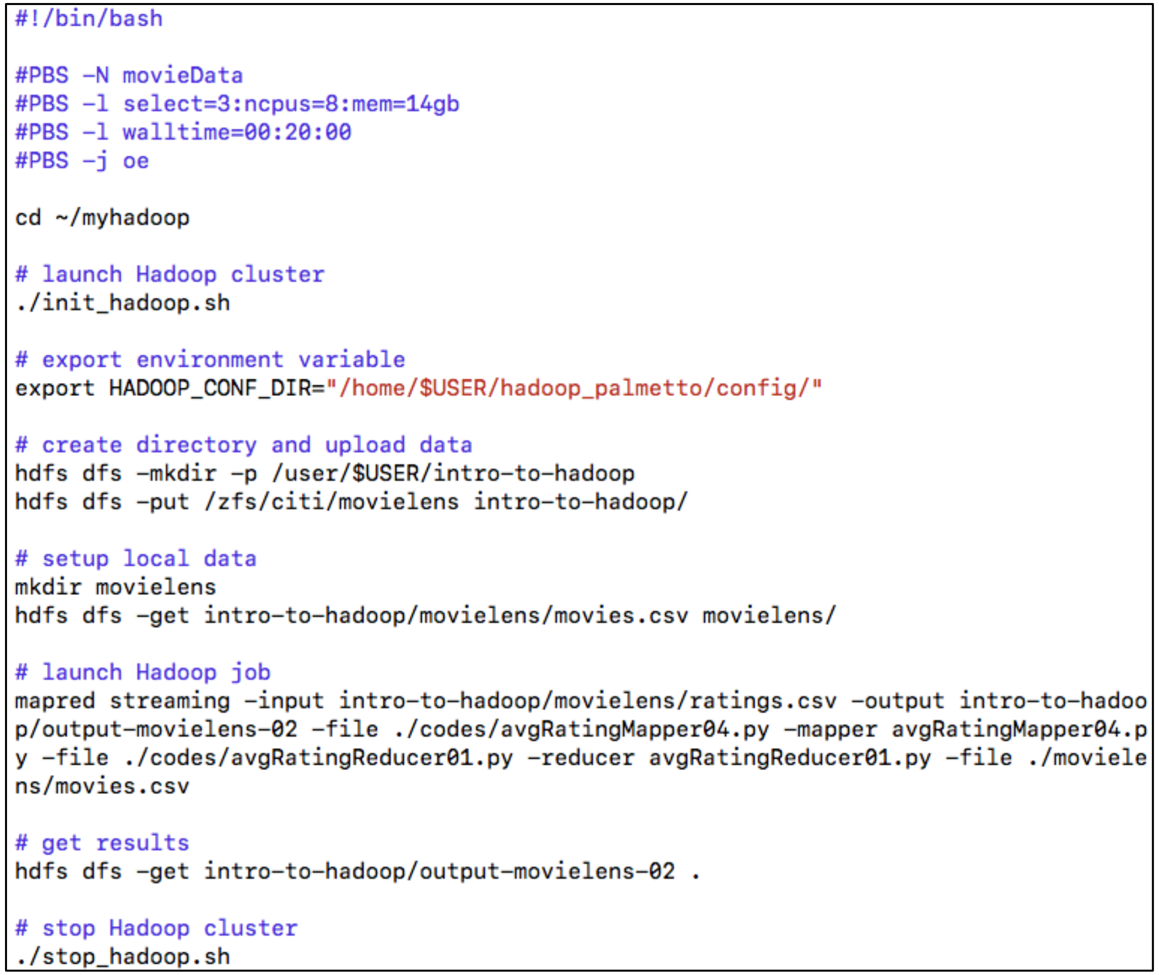
Add combiner to the movie example



```
%%writefile codes/avgGenreCombiner.py
#!/usr/bin/env python

import sys
import csv
import json

genreList = {}

for line in sys.stdin:
    line = line.strip()
    genre, ratingString = line.split("\t", 1)
    ratingInfo = json.loads(ratingString)

    if genre in genreList:
        genreList[genre]["total_rating"] += ratingInfo["total_rating"]
        genreList[genre]["total_count"] += ratingInfo["total_count"]
    else:
        genreList[genre] = {}
        genreList[genre]["total_rating"] = ratingInfo["total_rating"]
        genreList[genre]["total_count"] = 1

for genre in genreList:
    print ("%s\t%s" % (genre, json.dumps(genreList[genre])))
```

- hdfs dfs -rm -r intro-to-hadoop/output-movielens-06
- mapred streaming **-input** intro-to-hadoop/movielens/ratings.csv **-output** intro-to-hadoop/output-movielens-06 **-file** ./codes/avgGenreMapper02.py **-mapper** avgGenreMapper02.py **-file** ./codes/avgGenreReducer02.py **-reducer** avgGenreReducer02.py **-file** ./codes/avgGenreCombiner.py **-combiner** avgGenreCombiner.py **-file** ./movielens/movies.csv
- How different are the number of shuffle bytes between the two jobs?

Integrating Hadoop Job into Palmetto Workflow



```
#!/bin/bash

#PBS -N movieData
#PBS -l select=3:ncpus=8:mem=14gb
#PBS -l walltime=00:20:00
#PBS -j oe

cd ~/myhadoop

# launch Hadoop cluster
./init_hadoop.sh

# export environment variable
export HADOOP_CONF_DIR="/home/$USER/hadoop_palmetto/config/"

# create directory and upload data
hdfs dfs -mkdir -p /user/$USER/intro-to-hadoop
hdfs dfs -put /zfs/citi/movielens intro-to-hadoop/

# setup local data
mkdir movielens
hdfs dfs -get intro-to-hadoop/movielens/movies.csv movielens/

# launch Hadoop job
mapred streaming -input intro-to-hadoop/movielens/ratings.csv -output intro-to-hadoo
p/output-movielens-02 -file ./codes/avgRatingMapper04.py -mapper avgRatingMapper04.p
y -file ./codes/avgRatingReducer01.py -reducer avgRatingReducer01.py -file ./moviele
ns/movies.csv

# get results
hdfs dfs -get intro-to-hadoop/output-movielens-02 .

# stop Hadoop cluster
./stop_hadoop.sh
```

- With hdp module, Hadoop job can be invoked within a Palmetto PBS script, allowing the integration of large-data processing components into a standard HPC workflow.
  - Open a terminal, ssh to login001 (DUO required), and submit this script
    - ssh login001
    - cd
    - cat –n ~/myhadoop/codes/movieAnalyzer.pbs
    - qsub ~/myhadooop/codes/movieAnalyzer.pbs
  - View the final output when the job is finished
    - qstat –anu $USER
    - cat part-00000 2>/dev/null | head -n 20