



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Προχωρημένα Θέματα Βάσεων Δεδομένων

Αναφορά για την Εξαμηνιαία Εργασία:

Χρήση του Apache Spark στις Βάσεις Δεδομένων

Ον/μο : Μπερέτσος Θεόδωρος
Α.Μ.: : 03111612

Ον/μο : Ντόκος Χρήστος
Α.Μ.: : 03117xxx

Ον/μο : Στάβαρης Δημοσθένης
Α.Μ.: : 03117xxx

Ημερομηνία παράδοσης: 18/03/2022

Μέρος 1^ο: Υπολογισμός Αναλυτικών Ερωτημάτων με τα APIs του Apache Spark

Ζητούμενο 1

Έγινε λήψη του dataset `movie_data.tar.gz`. Αποσυμπιέστηκε. Δημιουργήθηκαν τα directories **files** και **outputs** στο Hadoop file system. Τέλος, φορτώθηκαν τα 3 CSV αρχεία που μας δόθηκαν στο `hdfs` στο φάκελο **files** εκτελώντας τις παρακάτω εντολές στον **master** (βλ. Εικόνα 1).

```
user@master:~$ wget 'http://www.cslab.ntua.gr/courses/atds/movie_data.tar.gz'
--2022-03-11 19:41:19-- http://www.cslab.ntua.gr/courses/atds/movie_data.tar.gz
Resolving www.cslab.ntua.gr (www.cslab.ntua.gr)... 147.102.3.238
Connecting to www.cslab.ntua.gr (www.cslab.ntua.gr)|147.102.3.238|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 184259305 (176M) [application/x-gzip]
Saving to: 'movie_data.tar.gz'

movie_data.tar.gz      100%[=====] 175.72M  109MB/s  in 1.6s

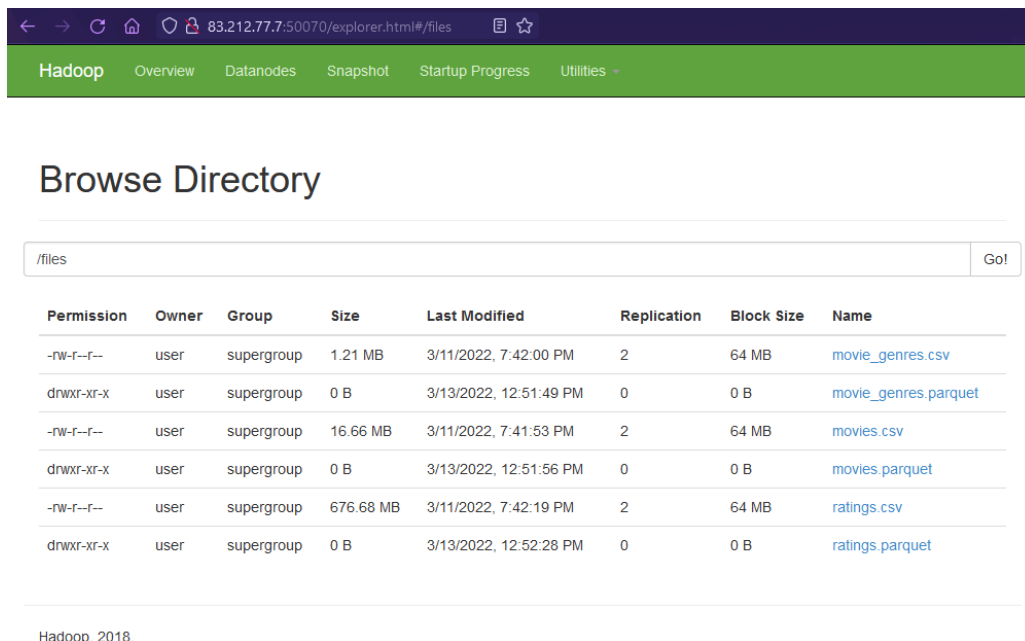
2022-03-11 19:41:21 (109 MB/s) - 'movie_data.tar.gz' saved [184259305/184259305]

user@master:~$ tar -xzf movie_data.tar.gz
user@master:~$ hadoop fs -mkdir hdfs://master:9000/files
user@master:~$ hadoop fs -put movies.csv hdfs://master:9000/files/.
user@master:~$ hadoop fs -put movie_genres.csv hdfs://master:9000/files/.
user@master:~$ hadoop fs -put ratings.csv hdfs://master:9000/files/.
user@master:~$ hadoop fs -mkdir hdfs://master:9000/outputs
user@master:~$ hadoop fs -ls hdfs://master:9000/
Found 2 items
drwxr-xr-x - user supergroup          0 2022-03-11 19:42 hdfs://master:9000/files
drwxr-xr-x - user supergroup          0 2022-03-11 19:42 hdfs://master:9000/outputs
user@master:~$
```

Εικόνα 1: Εντολές φόρτωσης .csv αρχείων στο `hdfs`

Ζητούμενο 2

Χρησιμοποιήθηκε το script με όνομα `csv2parquet.py` για την μετατροπή των αρχείων CSV σε Parquet. Εποπτικά η εικόνα των 6 αρχείων (3 CSV και 3 Parquet) στο directory `files` μέσα από το Web UI του Hadoop είναι η εξής (βλ. Εικόνα 2):



← → ↺ 83.212.77.7:50070/explorer.html#/files

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/files Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	user	supergroup	1.21 MB	3/11/2022, 7:42:00 PM	2	64 MB	movie_genres.csv
drwxr-xr-x	user	supergroup	0 B	3/13/2022, 12:51:49 PM	0	0 B	movie_genres.parquet
-rw-r--r--	user	supergroup	16.66 MB	3/11/2022, 7:41:53 PM	2	64 MB	movies.csv
drwxr-xr-x	user	supergroup	0 B	3/13/2022, 12:51:56 PM	0	0 B	movies.parquet
-rw-r--r--	user	supergroup	676.68 MB	3/11/2022, 7:42:19 PM	2	64 MB	ratings.csv
drwxr-xr-x	user	supergroup	0 B	3/13/2022, 12:52:28 PM	0	0 B	ratings.parquet

Hadoop, 2018.

Εικόνα 2: Web UI Hadoop

Ζητούμενο 3

Υλοποιήθηκαν διαφορετικές λύσεις για κάθε ένα από τα ερωτήματα Q1-Q5, χρησιμοποιώντας το RDD API και τη Spark SQL. Συγκεκριμένα στην υλοποίηση της τελευταίας η είσοδος έγινε με τους εξής δύο τρόπους: αρχεία CSV και αρχεία Parquet. Παρακάτω παρουσιάζονται οι ψευδοκώδικες σε Map Reduce που εφαρμόστηκαν στις υλοποιήσεις με το RDD API.

Χρησιμοποιήθηκαν οι μέθοδοι `map()`, `reduceByKey()`, `filter()`, `count()`, `join()` και `sortByKey()` από το RDD API.

Ερώτημα Q1

```
# movies = movies.csv

map(movies, value):
    for movie in movies:
        line = movie.split(',')
        # movie = (movie_id, title, description, publish_date,
        #         duration, cost, income, favoured)
        title = line[1]
        year = line[3].split('-')[0]
        publish_date = line[3]
        cost = line[5]
        income = line[6]
        if ((publish_date != '') and (cost != 0) and
            (income != 0) and (year > 2000)):
            profit = ((income - cost) / cost) * 100
            emit(year, (title, profit))

reduce(year, (title, profit)):
    max_profit = 0
    max_title = ''
    for title, profit in (title, profit):
        if (max_profit > profit):
            max_profit = profit
            max_title = title
    emit(year, (max_title, max_profit))
```

Ερώτημα Q2

```
# ratings = ratings.csv

map(ratings, value):
    for movie in movies:
        line = movie.split(',')
        # ratings = (user_id, movie_id, rating, timestamp))
```

```

        user_id = line[0]
        rating = line[2]
        emit(user_id, (rating, 1))

reduce(user_id, (rating, 1)):
    sum_of_ratings = 0
    cnt_of_movies = 0
    for rating, cnt in (rating, 1):
        sum_of_ratings += rating
        cnt_of_movies++
    emit(user_id, (sum_of_ratings, cnt_of_movies))

map(user_id, (sum_of_ratings, cnt_of_movies)):
    avg_rating = sum_of_ratings / cnt_of_movies
    emit(user_id, avg_rating)

all_users = (user_id, avg_rating).count()
users_above_3 = (user_id, avg_rating).filter(avg_rating >= 3.0)

percentage = users_above_3 / all_users * 100

```

Ερώτημα Q3

```

# ratings = ratings.csv
# genres = genres.csv

map(ratings, value):
    for movie in movies:
        line = movie.split(',')
        # ratings = (user_id, movie_id, rating, timestamp))
        movie_id = line[1]
        rating = line[2]
        emit(movie_id, (rating, 1))

reduce(movie_id, (rating, 1)):
    sum_of_ratings = 0
    cnt_of_ratings = 0
    for rating, cnt in (rating, 1):
        sum_of_ratings += rating
        cnt_of_ratings++
    emit(movie_id, (sum_of_ratings, cnt_of_ratings))

map(movie_id, (sum_of_ratings, cnt_of_ratings)):
    avg_rating = sum_of_ratings / cnt_of_ratings

```

```

    emit(movie_id, (avg_rating, cnt_of_ratings))

map(genres, value):
    for genre in genres:
        line = genre.split(',')
        # genres = (movie_id, genre)
        movie_id = line[0]
        genre = line[1]
        emit(movie_id, genre)

join((movie_id, genre), (movie_id, (avg_rating, cnt_of_ratings))):
    emit(movie_id, (genre, (avg_rating, cnt_of_ratings)))

map(movie_id, (genre, (avg_rating, cnt_of_ratings))):
    for line in lines:
        emit(genre, (avg_rating, 1))

reduce(genre, (avg_rating, 1)):
    sum_of_avg_ratings = 0
    cnt_of_avg_ratings = 0
    for avg_rating, cnt in (avg_rating, 1):
        sum_of_avg_ratings += avg_rating
        cnt_of_avg_ratings++
    emit(movie_id, (sum_of_avg_ratings, cnt_of_avg_ratings))

map(movie_id, (sum_of_avg_ratings, cnt_of_avg_ratings)):
    avg_rating_per_genre = sum_of_avg_ratings / cnt_of_avg_ratings
    emit(genre, (avg_rating_per_genre, cnt_of_avg_rating_per_genre))

```

Ερώτημα Q4

```

# movies = movies.csv
# genres = genres.csv

map(movies, value):
    for movie in movies:
        line = movie.split(',')
        # movie = (movie_id, title, description, publish_date,
        #         duration, cost, income, favoured)
        movie_id = line[0]
        title = line[1]
        description = line[2]
        publish_date = line[3]

```

```

        year = line[3].split('-')[0]
        if ((title != '') and (publish_date != '') and
            (year != '') and (year >= 2000) and (year <= 2019)):
            word_count_in_description = 0
            for word in description.split():
                word_count_in_description++
            emit(movie_id, (year, word_count_in_description))

map(genres, value):
    for genre in genres:
        line = genre.split(',')
        # genres = (movie_id, genre)
        movie_id = line[0]
        genre = line[1]
        if (genre == 'Drama'):
            emit(movie_id, 'Drama')

join((movie_id, (year, word_count_in_description)), (movie_id, 'Drama')):
    emit(movie_id, ((year, word_count_in_description), 'Drama'))

map(movie_id, ((year, word_count_in_description), 'Drama')):
    if (year >=2000) and (year <=2004):
        period = '2000-2004'
    elif (year >=2005) and (year <=2009):
        period = '2005-2009'
    elif (year >=2010) and (year <=2014):
        period = '2010-2014'
    else:
        period = '2015-2019'
    emit(period, (word_count_in_description, 1))

reduce(period, (word_count_in_description, 1)):
    sum_of_words = 0
    cnt_of_movies = 0
    for words, cnt in (word_count_in_description, 1):
        sum_of_words += words
        cnt_of_movies++
    emit(period, (sum_of_words, cnt_of_movies))

map(period, (sum_of_words, cnt_of_movies)):
    avg_cnt_of_words = sum_of_words / cnt_of_movies
    emit(period, avg_cnt_of_words)

```

Ερώτημα Q5

Lorem Ipsum

Ζητούμενο 4

Lorem Ipsum

Μέρος 2º: Υλοποίηση και μελέτη συνένωσης σε ερωτήματα και Μελέτη του βελτιστοποιητή του Spark

Ζητούμενο 1

Lorem Ipsum

Ζητούμενο 2

Lorem Ipsum

Ζητούμενο 3

Lorem Ipsum

Ζητούμενο 4

Lorem Ipsum