

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

**ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ
ΕΞΑΜΗΝΙΑ ΕΡΓΑΣΙΑ 2022-2023**

ΟΜΑΔΑ 40

Ονοματεπώνυμο
Νικόλαος Μπλέτσας
Γεώργιος Τζουμανέκας

Αριθμός Μητρώου
03118899
03118095

Github link

<https://github.com/GeoTzoum/atds-team40>

Εισαγωγή files στο hdfs

Αρχικά εγκαταστήσαμε με τα κατάλληλα configurations το Spark και το HDFS. Για να ξεκινήσουμε την λειτουργία του HDFS εκτελούμε την εντολή *start-dfs.sh* και για την να ξεκινήσουμε την λειτουργία του Spark εκτελούμε την εντολή *start-all.sh*. Δημιουργούμε το directory με την εντολή *hadoop fs -mkdir hdfs://master:9000/files/* στο οποίο φορτώνουμε το ένα αρχείο με την εντολή *hadoop fs -put hdfs://master:9000/files/taxi+_zone_lookup.csv*. Έπειτα φτιάχνουμε ένα νέο directory *files/taxi* στο *hadoop* που θα εισάγουμε όλα τα *parquet* αρχεία με τον ίδιο τρόπο.

Αποτελέσματα από την εκτέλεση των queries

Q1 sql:

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee
2	2022-03-17 12:27:47	2022-03-17 12:27:58	1.0	0.0	1.0	N	12	12	1	2.5	0.0	0.5	48.0	0.0	0.3	45.8	2.5	0.0

Q2 sql:

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee
1	2022-01-22 11:39:07	2022-01-22 12:31:09	1.0	33.4	1.0	N	70	265	4	88.0	0.0	0.5	0.0	193.3	0.3	282.1	0.0	0.0
1	2022-02-18 02:33:30	2022-02-18 02:35:28	1.0	1.3	1.0	N	265	265	1	3.0	0.5	0.5	19.85	95.0	0.3	119.15	0.0	0.0
1	2022-03-11 20:00:32	2022-03-11 20:09:46	1.0	0.0	1.0	N	265	265	1	2.5	1.0	0.5	48.0	235.9	0.3	288.9	0.0	0.0
1	2022-04-29 04:31:21	2022-04-29 04:32:38	2.0	0.0	1.0	N	240	240	3	3.0	3.0	0.5	0.0	911.87	0.3	918.67	2.5	0.0
1	2022-05-21 16:47:48	2022-05-21 17:05:47	1.0	2.4	3.0	N	239	246	3	31.5	0.0	0.0	0.0	813.75	0.3	845.55	0.0	0.0
1	2022-05-12 16:51:46	2022-05-12 17:56:48	9.0	22.0	1.0	N	142	132	2	67.5	2.5	0.5	0.0	680.99	0.3	678.69	2.5	0.0

Q3 sql:

YR	MN	Month_Part	avg(Trip_distance)	avg(total_amount)
2022	1	Second Half	5.097880367275346	19.14882164234129
2022	1	First Half	5.576410377852007	19.903702637879007
2022	2	First Half	6.248888338463885	19.491979067237448
2022	2	Second Half	5.849460516243601	20.18769180439039
2022	3	First Half	6.480485434052824	20.652278174179074
2022	3	Second Half	5.5569449358506535	21.120920554171548
2022	4	Second Half	5.800344707645977	21.428088376232783
2022	4	First Half	5.679323077938295	21.515559094583587
2022	5	First Half	6.249697852127242	21.921570348909114
2022	5	Second Half	7.906694182348757	22.771948777963715
2022	6	First Half	6.315157336730177	22.466305309343248
2022	6	Second Half	6.199832889000861	22.34284989271931

Q3 rdd:

```

time, avg(total amount)
((1, 'First Half'), 19.903702637879007)
((1, 'Second Half'), 19.14882164234129)
((2, 'First Half'), 19.491979067237448)
((2, 'Second Half'), 20.18769180439039)
((3, 'First Half'), 20.652278174179074)
((3, 'Second Half'), 21.120920554171548)
((4, 'First Half'), 21.515559094583587)
((4, 'Second Half'), 21.428088376232783)
((5, 'First Half'), 21.921570348909114)
((5, 'Second Half'), 22.771948777963715)
((6, 'First Half'), 22.466305309343248)
((6, 'Second Half'), 22.331380641103525)

time, avg(trip distance)
((1, 'First Half'), 5.576410377852007)
((1, 'Second Half'), 5.097880367275346)
((2, 'First Half'), 6.248888338463885)
((2, 'Second Half'), 5.849460516243601)
((3, 'First Half'), 6.480485434052824)
((3, 'Second Half'), 5.5569449358506535)
((4, 'First Half'), 5.679323077938295)
((4, 'Second Half'), 5.800344707645977)
((5, 'First Half'), 6.249697852127242)
((5, 'Second Half'), 7.906694182348757)
((6, 'First Half'), 6.315157336730177)
((6, 'Second Half'), 6.174138574511356)

```

Q4 sql:

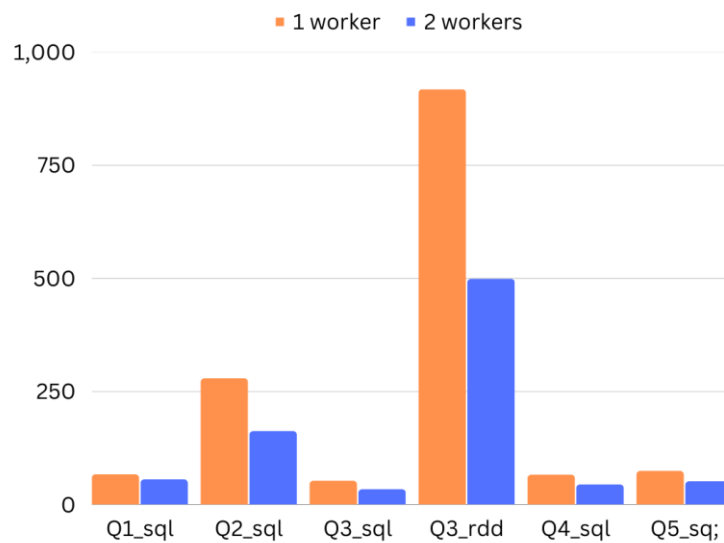
hour	day	passengers
0	Sunday	1.5299456507188562
1	Sunday	1.527838567375201
2	Sunday	1.5080726185191242
0	Monday	1.4679887711672552
1	Monday	1.4442867916810471
2	Monday	1.4231993989051486
0	Tuesday	1.4200313882151518
1	Tuesday	1.4175124740006593
2	Tuesday	1.4104520814693964
1	Wednesday	1.4088480212656305
0	Wednesday	1.4012291857176276
2	Wednesday	1.4011489645958584
23	Thursday	1.4052969946783969
1	Thursday	1.4019816054943737
0	Thursday	1.4012800328564583
23	Friday	1.475576918073731
22	Friday	1.444813976205668
2	Friday	1.4230581143524386
23	Saturday	1.522606766277207
22	Saturday	1.5068176194011382
0	Saturday	1.4993154284898547

Q5 sql:

day	month	percentage
29	January	13.281695466343933
30	January	12.647559063953313
22	January	12.50531584153524
15	January	12.488568453584604
23	January	12.45176062008483
5	February	12.587826720466092
6	February	12.531148066105837
13	February	12.483880549707122
4	February	12.443687124886978
10	February	12.402989406799005
9	March	12.525862493714753
10	March	12.454514040286558
30	March	12.436711363326735
12	March	12.430243546465958
31	March	12.400258696885896
7	April	12.431798590349711
6	April	12.39052216804565
27	April	12.385832070322706
28	April	12.357687843637914
1	April	12.315532876414338
12	May	12.37559961476351
4	May	12.344841041276652
18	May	12.335820238740737
25	May	12.329909569029104
19	May	12.31987827649278
9	June	12.386117303827776
16	June	12.37938530386768
23	June	12.325498543404183
8	June	12.315642048296459
15	June	12.307720439633318

Χρόνοι εκτέλεσης ερωτημάτων

Στο παρακάτω διάγραμμα φαίνονται οι χρόνοι για την εκτέλεση των ερωτημάτων με τους δύο workers και αφού κλείσαμε manually τον έναν worker.



Παρατηρούμε πως σε όλες τις περιπτώσεις οι 2 workers ήταν πιο γρήγοροι από τον 1 worker στην εκτέλεση των ερωτημάτων. Αυτό είναι αναμενόμενο καθώς με τους 2 workers χρησιμοποιούμε διπλάσιο αριθμό cores, σε σχέση με τον 1. Με 2 workers το κάθε query χωρίζεται σε subtasks για τον κάθε worker και εκτελείται παράλληλα, άρα τελικά και πιο γρήγορα. Με τους 2 workers επίσης έχουμε μεγαλύτερο resilience, δηλαδή αν αποτύχει ο ένας worker μπορεί ο άλλος να συνεχίσει και δεν αποτυγχάνει ολόκληρο το query.

Συγκρίνοντας τον χρόνο στο Q3 ανάμεσα στο Spark SQL και στο RDD Api βλέπουμε ότι το RDD κάνει πολύ περισσότερο χρόνο (και στις δύο περιπτώσεις για workers). Αυτό είναι λογικό καθώς το RDD Api δεν διαθέτει κάποιο inbuilt μηχανισμό βελτιστοποίησης, όπως το Spark SQL που χρησιμοποιεί τον catalyst optimizer.

Συγκεκριμένα οι χρόνοι εκτέλεσης σε πίνακα:

Query	1 worker	2 workers
Q1_sql	66.113	54.938
Q2_sql	278.263	162.119
Q3_sql	52.078	33.520
Q3_rdd	917.271	498.667
Q4_sql	65.386	44.125
Q5_sql	73.646	50.873