Προχωρημένα Θέματα Βάσεων Δεδομένων

Εξαμηνιαία Εργασία

Ομάδα: 23

Ημερομηνία παράδοσης: 15/01/2023

Μέλη:

Στύλος Αλέξανδρος – Παναγιώτης (el18410),

Χελάς Στέφανος (el18241)

Link

Το link στο GitHub που περιέχει όλο τον κώδικα που υλοποιήσαμε είναι:

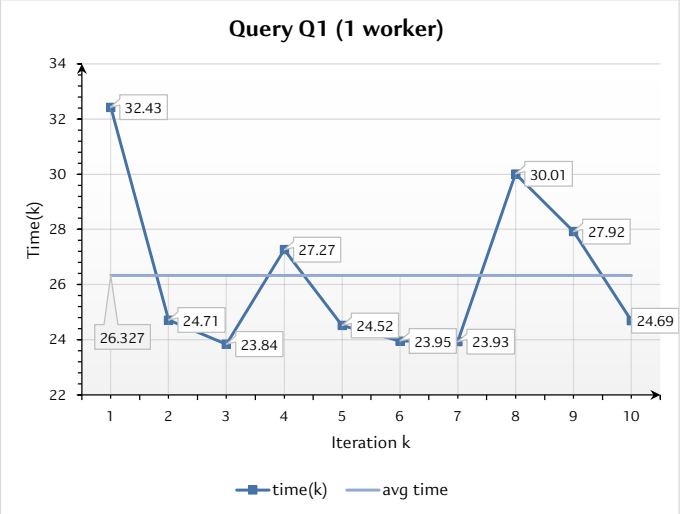
Ερώτημα 2

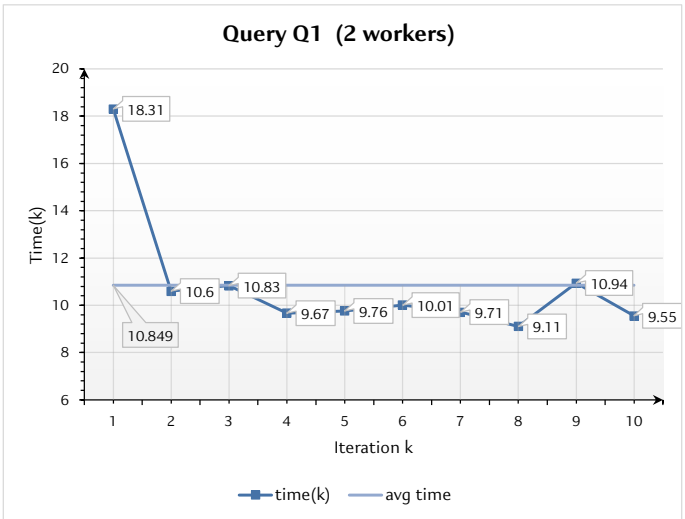
- Query Q1

Το αποτέλεσμα που προκύπτει είναι η εξής εγγραφή:

Vendor ID	tpep_pickup_ datetime	tpep_dropoff_ datetime	passenger _count	trip_ distance	Ratecode ID	store_and_ fwd_flag	PULocation ID	DOLocation ID	payment_ type	fare_ amount	extra	mta_tax	tip_ amount	tolls_ amount	improvement _surcharge	total_ amount	congestion _surcharge	airport_ fee	Location ID	Borough	Zone	service_ zone
2	2022-03-17 12:27:47	2022-03-17 12:27:58	1	0	1	N	12	12	1	2.5	0	0.5	40	0	0.3	45.8	2.5	0	12	Manhattan	Battery Park	Yellow Zone

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:



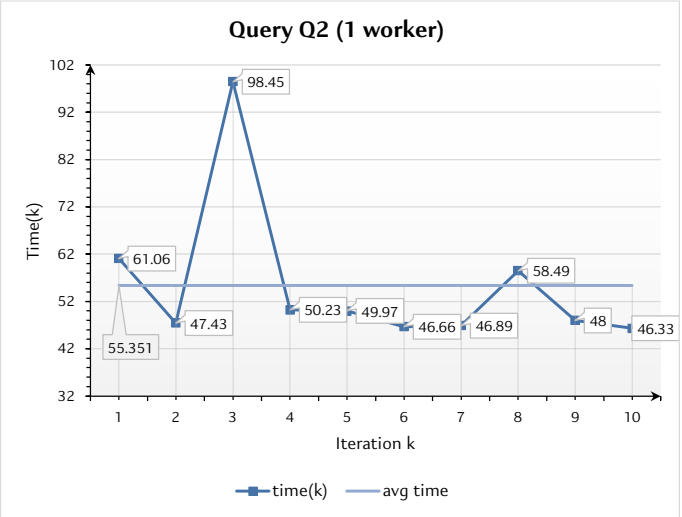


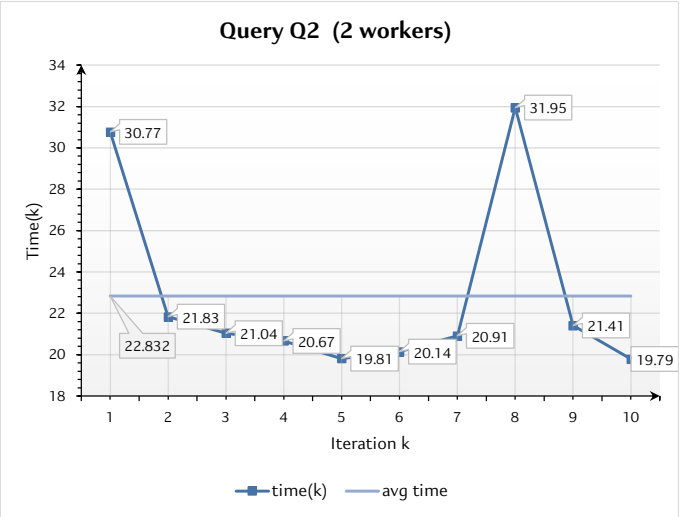
- Query Q2

Προκύπτουν οι παρακάτω εγγραφές ως αποτέλεσμα:

month	max_tolls_ amount	VendorID	tpep_pickup_ datetime	tpep_dropoff_ datetime	passenger _count	trip_ distance	Ratecode ID	store_and_ fwd_flag	PULocation ID	DOLocation ID	payment_ type	fare_ amount	extra	mta_tax	tip_amount	improvement _surcharge	total_ amount	congestion_ surcharge	airport_ fee
January	193.3	1	2022-01-22 11:39:07	2022-01-22 12:31:09	1	33.4	1	Y	70	265	4	88	0	0.5	0	0.3	282.1	0	0
February	95	1	2022-02-18 02:33:30	2022-02-18 02:35:28	1	1.3	1	N	265	265	1	3	0.5	0.5	19.85	0.3	119.15	0	0
March	235.7	1	2022-03-11 20:08:32	2022-03-11 20:09:45	1	0	1	N	265	265	1	2.5	1	0.5	48	0.3	288	0	0
April	911.87	1	2022-04-29 04:31:21	2022-04-29 04:32:30	2	0	1	N	249	249	3	3	3	0.5	0	0.3	918.67	2.5	0
May	813.75	1	2022-05-21 16:47:48	2022-05-21 17:05:47	1	2.4	3	N	239	246	3	31.5	0	0	0	0.3	845.55	0	0
June	800.09	1	2022-06-12 16:51:46	2022-06-12 17:56:48	9	22	1	N	142	132	2	67.5	2.5	0.5	0	0.3	870.89	2.5	0

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:





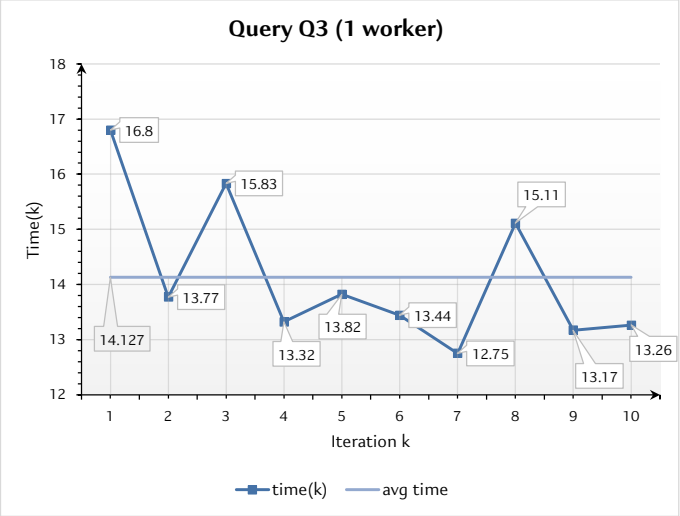
Ερώτημα 3

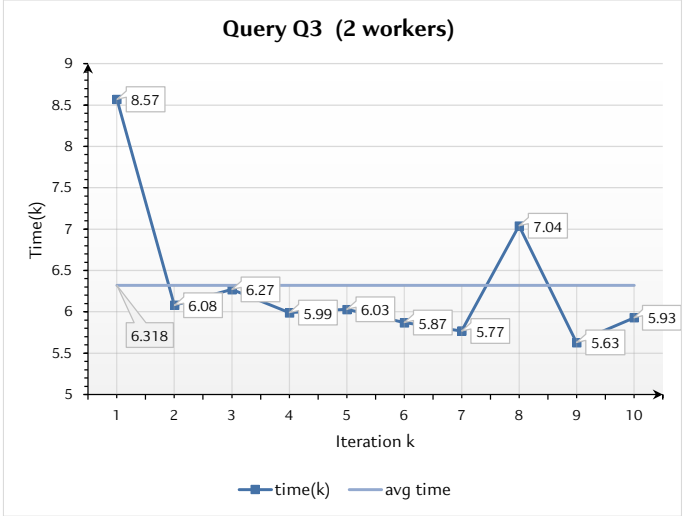
- Query Q3

Το αποτέλεσμα που προκύπτει είναι το εξής:

fortnight	avg distance	avg charge
[2022-01-01, 2022-01-15]	5.576410378	19.90370264
[2022-01-16, 2022-01-31]	5.097880367	19.14882164
[2022-02-01, 2022-02-15]	6.248888338	19.49197907
[2022-02-16, 2022-02-28]	5.849460516	20.1876918
[2022-03-01, 2022-03-15]	6.480485434	20.65227817
[2022-03-16, 2022-03-31]	5.556944936	21.12092055
[2022-04-01, 2022-04-15]	5.679323078	21.51555909
[2022-04-16, 2022-04-30]	5.800344708	21.42808838
[2022-05-01, 2022-05-15]	6.249697852	21.92157035
[2022-05-16, 2022-05-31]	7.906694182	22.77194878
[2022-06-01, 2022-06-15]	6.315157337	22.46630531
[2022-06-16, 2022-06-30]	6.174138575	22.33138064

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:



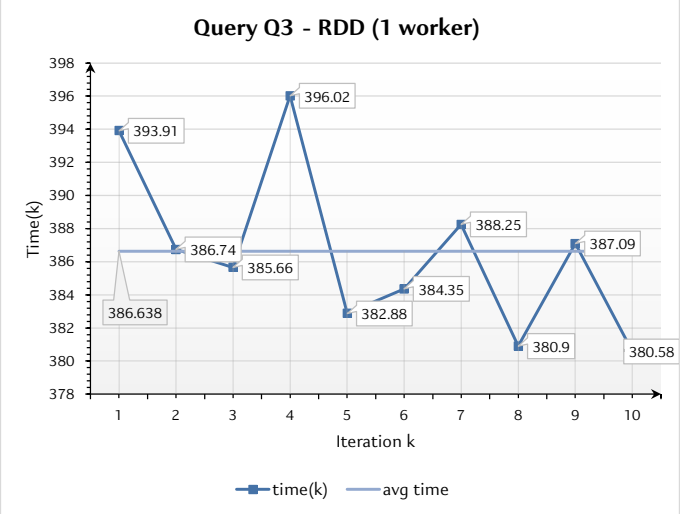


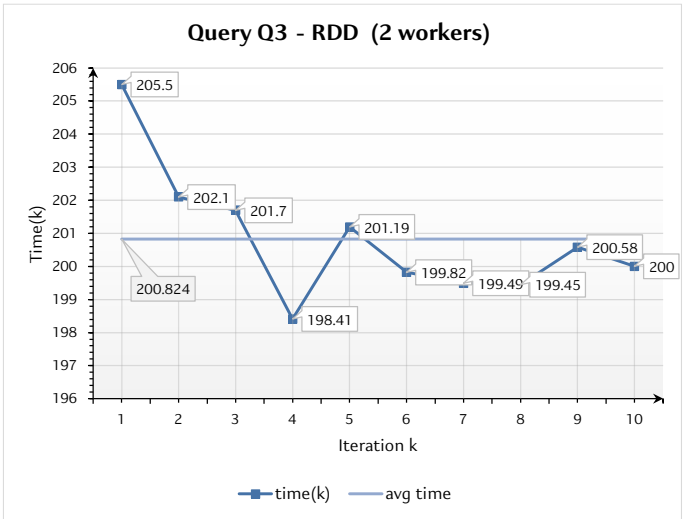
- Query Q3 – RDD

Προκύπτει το ίδιο αποτέλεσμα με το προηγούμενη περίπτωση, δηλαδή έχουμε τις εγγραφές:

fortnight	avg distance	avg charge
[2022-01-01, 2022-01-15]	5.576410378	19.90370264
[2022-01-16, 2022-01-31]	5.097880367	19.14882164
[2022-02-01, 2022-02-15]	6.248888338	19.49197907
[2022-02-16, 2022-02-28]	5.849460516	20.1876918
[2022-03-01, 2022-03-15]	6.480485434	20.65227817
[2022-03-16, 2022-03-31]	5.556944936	21.12092055
[2022-04-01, 2022-04-15]	5.679323078	21.51555909
[2022-04-16, 2022-04-30]	5.800344708	21.42808838
[2022-05-01, 2022-05-15]	6.249697852	21.92157035
[2022-05-16, 2022-05-31]	7.906694182	22.77194878
[2022-06-01, 2022-06-15]	6.315157337	22.46630531
[2022-06-16, 2022-06-30]	6.174138575	22.33138064

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:





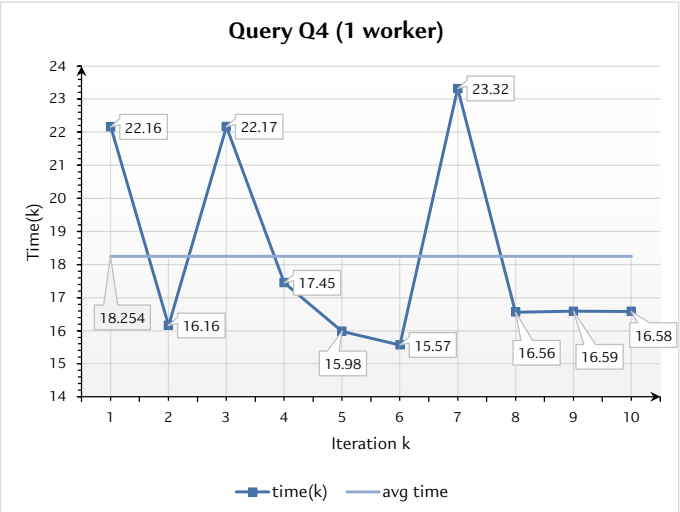
Ερώτημα 4

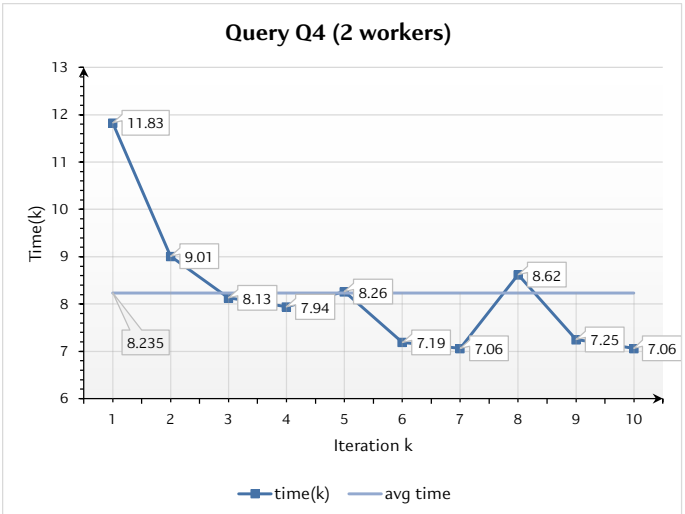
- Query Q4

Το αποτέλεσμα που προκύπτει είναι:

day	hour_interval	number_passengers/hour/day
Monday	[0, 1)	1.529945651
Monday	[1, 2)	1.527838567
Monday	[2, 3)	1.508072619
Tuesday	[0, 1)	1.467988771
Tuesday	[1, 2)	1.444286792
Tuesday	[2, 3)	1.423199399
Wednesday	[0, 1)	1.420031388
Wednesday	[1, 2)	1.417512474
Wednesday	[2, 3)	1.410452081
Thursday	[1, 2)	1.408848021
Thursday	[0, 1)	1.401229186
Thursday	[2, 3)	1.401148965
Friday	[23, 0)	1.405382315
Friday	[1, 2)	1.402590729
Friday	[0, 1)	1.401038253
Saturday	[23, 0)	1.475576918
Saturday	[22, 23)	1.444813976
Saturday	[2, 3)	1.423058114
Sunday	[23, 0)	1.522606766
Sunday	[22, 23)	1.506817619
Sunday	[0, 1)	1.499315428

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:



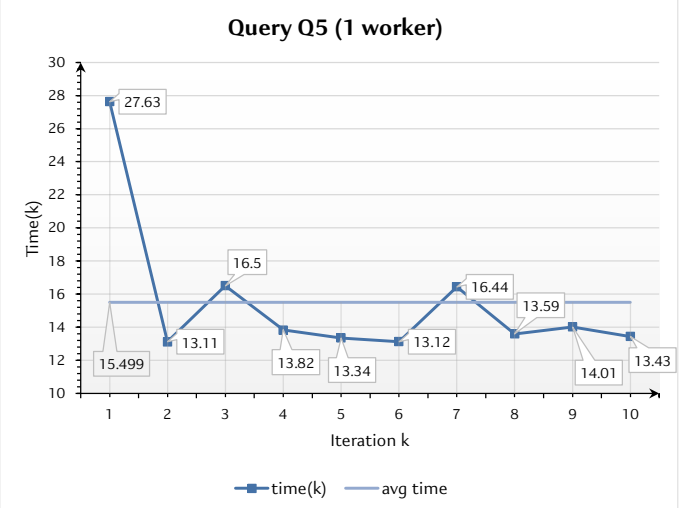


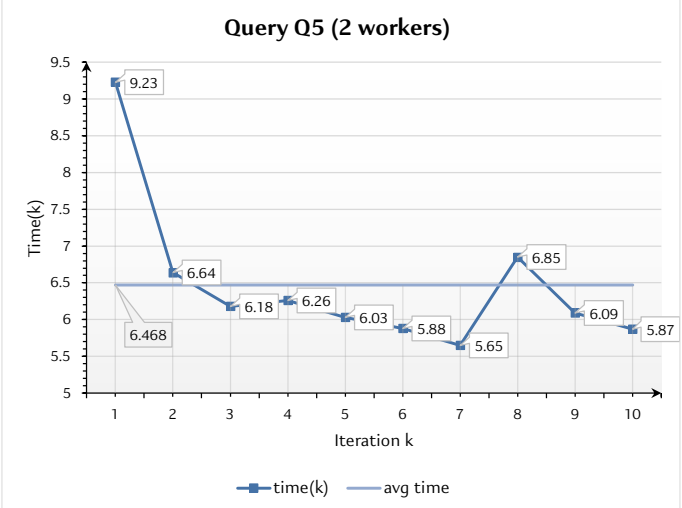
- Query Q5

Το αποτέλεσμα που προκύπτει περιλαμβάνει τις παρακάτω εγγραφές:

month	date	max % tip/fare
January	2022-01-09	1688800
January	2022-01-31	1100000
January	2022-01-01	500000
January	2022-01-03	250000
January	2022-01-16	120000
February	2022-02-21	450000
February	2022-02-13	296900
February	2022-02-09	250000
February	2022-02-27	250000
February	2022-02-24	150000
March	2022-03-18	1000000
March	2022-03-21	700000
March	2022-03-26	200000
March	2022-03-05	200000
March	2022-03-19	100000
April	2022-04-12	2250000
April	2022-04-02	1250000
April	2022-04-21	1200000
April	2022-04-03	400000
April	2022-04-30	200000
May	2022-05-12	750000
May	2022-05-20	650000
May	2022-05-16	200000
May	2022-05-15	200000
May	2022-05-06	120000
June	2022-06-25	1550000
June	2022-06-13	1500000
June	2022-06-10	650000
June	2022-06-16	570000
June	2022-06-18	400000

Για κάθε περίπτωση αριθμού εργατών πραγματοποιούμε 10 επαναλήψεις εκτέλεσης του Query, υπολογίζουμε το χρόνο εκτέλεσης και βρίσκουμε το μέσο όρο των τιμών αυτών. Έτσι, προκύπτει:

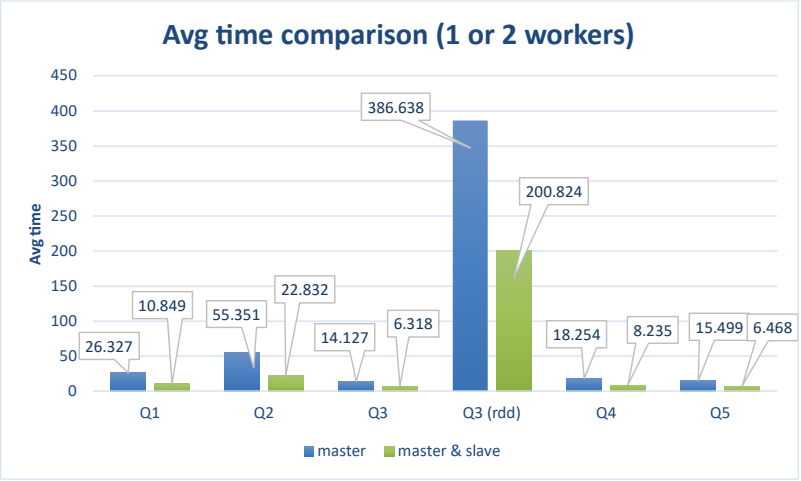


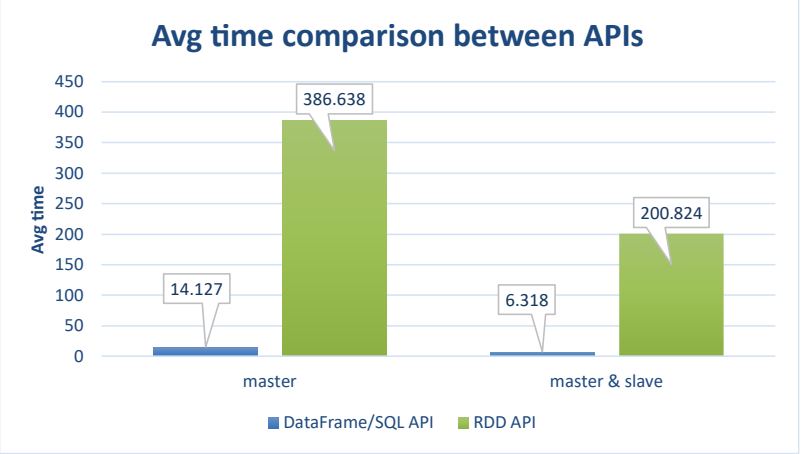


Σχόλια

Μπορούμε να συγκεντρώσουμε τις παραπάνω τιμές που βρήκαμε για τον μέσο χρόνο που χρειάζεται ένα Query σ' ένα συγκεντρωτικό πινακάκι:

1 worker - master		2 worker - master & slave		Transition from 1 worker to 2 workers
Query	Average time [sec]	Query	Average time [sec]	Average time reduction %
Q1	26.327	Q1	10.849	-58.79
Q2	55.351	Q2	22.832	-58.75
Q3	14.127	Q3	6.318	-55.28
Q3 (RDD)	386.638	Q3 (RDD)	200.824	-48.06
Q4	18.254	Q4	8.235	-54.89
Q5	15.499	Q5	6.468	-58.27





Βλέπουμε ότι η χρήση δύο workers, δηλαδή ενός master και ενός slave που συνεργάζονται σ' ένα cluster για την εκτέλεση ενός Query, μείωσε σημαντικά το χρόνο που απαιτείται σε σχέση με την περίπτωση εκείνη, στην οποία έχουμε έναν μόνο master που εργάζεται ως worker. Ενδεικτικά, η μείωση αυτή του χρόνου εκτέλεσης ξεκινάει από ~50% και μπορεί να φτάσει μέχρι και το ~60%.

Τέλος, αξίζει να αναφερθούμε και σε μια χρονική σύγκριση μεταξύ των διαφορετικών APIs που χρησιμοποιήσουμε για το Query Q3 στο Ερώτημα 3. Παρατηρούμε ότι η χρήση του DataFrame/SQL API οδηγεί στο ίδιο αποτέλεσμα συντριπτικά γρηγορότερα σε σχέση με τη χρήση του RDD API. Τα συμπεράσματα αυτά φαίνεται και στα διπλανά διαγράμματα.