Προχωρημένα Θέματα Βάσεων Δεδομένων

Ροή Λ - 9ο Εξάμηνο

Αναφορά Εξαμηνιαίας Εργασίας

Ονοματεπώνυμο	**Α.Μ.**
Πελαγία Δρακοπούλου	03118027
Μαρία Νεφέλη Κονδύλη	03118147

Φεβρουάριος 2023

1.

Η εγκατάσταση περιλαμβάνει τα εξής συστήματα:

- HDFS: Κατανεμημένο filesystem, από το οποίο διαβάζουν και γράφουν τα Spark jobs και παρέχεται από το Hadoop.
- Spark: Open source framework για επεξεργασία δεδομένων, που περιλαμβάνει υλοποίηση του προγραμματιστικού μοντέλου MapReduce.

Από το site του okeanos knossos δημιουργούμε 2 εικονικές μηχανές (virtual machines) τις οποίες ονομάζουμε spark master και spark slave αντίστοιχα.
Στα VMs συνδεόμαστε μέσω ssh, εγκαθιστούμε σε αυτά τα παραπάνω συστήματα και τα συνδέουμε μεταξύ τους, έτσι ώστε το Spark να μπορεί να διαβάζει τα αρχεία που έχουμε αποθηκεύσει στο HDFS.
Πιο συγκεκριμένα, πραγματοποιούμε τα απαραίτητα configurations στα αρχεία των συστημάτων αυτών ώστε να λειτουργεί με τον επιθυμητό τρόπο η εφαρμογή μας.

Σημείωση: Τα configurations αυτά αναγράφονται λεπτομερώς στο readme file του github repo του project. (https://github.com/peldrak/ATDS_pyspark_project.git)

Ο γενικός σχεδιασμός βασίζεται στα παρακάτω χαρακτηριστικά:

- Ο master θέλουμε να έχει τους παρακάτω ρόλους:
    - HDFS NameNode: Κεντρικός κόμβος του HDFS, ο οποίος περιέχει την πληροφορία με την αντιστοίχιση των blocks των αρχείων με τους αντίστοιχους DataNodes
    - HDFS DataNode: Περιέχει blocks από τα αρχεία του HDFS.
    - Spark Master node: Κεντρικός κόμβος του Spark ο οποίος μέσω του ενσωματωμένου cluster manager ελέγχει τους διαθέσιμους πόρους και με βάση αυτούς δρομολογεί και διαχειρίζεται τις κατανεμημένες εφαρμογές που τρέχουν τον cluster.
    - Spark Worker node: Διεργασία του Spark, η οποία τρέχει σε κάθε κόμβο του cluster και διαχειρίζεται τις προς εκτέλεση διεργασίες των κατανεμημένων εφαρμογών στον κόμβο αυτό.

- Ενώ ο slave τους εξής:
    - HDFS DataNode
    - Spark Worker node

Δημιουργούμε το αρχείο code.py το οποίο περιέχει όλο τον κώδικα για την εκτέλεση των ερωτημάτων.
Δημιουργούμε το spark session στο οποίο θα τρέξει ο κώδικάς μας στο url: spark://192.168.0.1:7077

Στη συνέχεια ορίζουμε μία λίστα από τα paths των parquet αρχείων στο HDFS και μέσω της εντολής spark.read.parquet αποθηκεύουμε το περιεχόμενο των αρχείων σε ένα ενιαίο dataframe. Δημιουργούμε επίσης και το αντίστοιχο RDD μέσω της εντολής df.rdd. Αντίστοιχη διαδικασία ακολουθούμε και για το csv αρχείο.

2.

Εκτελούμε τα ερωτήματα Q1 και Q2 χρησιμοποιώντας το DataFrame API και παρακάτω παρουσιάζουμε τα αποτελέσματα και τους χρόνους εκτέλεσης τους με χρήση 1 και 2 workers.

- Για το Q1 αναζητούμε τη διαδρομή με το μεγαλύτερο φιλοδώρημα (tip) τον Μάρτιο και σημείο άφιξης το "Battery Park".

Output:

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
2	2022-03-17 12:27:47	2022-03-17 12:27:58	1.0	0.0

RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount
1.0	N	12	12	1	2.5

extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount
0.0	0.5	40.0	0.0	0.3	45.8

congestion_surcharge	airport_fee	LocationID	Borough	Zone	service_zone
2.5	0.0	12	Manhattan	Battery Park	Yellow Zone

Χρόνοι εκτέλεσης:

	1 worker	2 workers
Execution Time (sec)	22.19	17.58

- Για το Q2 αναζητούμε για κάθε μήνα, η διαδρομή με το υψηλότερο ποσό στα διόδια, αγνοώντας μηδενικά ποσά.

Output:

VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
1	2022-01-22 11:39:07	2022-01-22 12:31:09	1.0	33.4

1	2022-02-18 02:33:30	2022-02-18 02:35:28	1.0	1.3
1	2022-03-11 20:08:32	2022-03-11 20:09:45	1.0	0.0
1	2022-04-29 04:31:21	2022-04-29 04:32:30	2.0	0.0
1	2022-05-21 16:47:48	2022-05-21 17:05:47	1.0	2.4
1	2022-06-12 16:51:46	2022-06-12 17:56:48	9.0	22.0

RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount
1.0	Y	70	265	4	88.0
1.0	N	265	265	1	3.0
1.0	N	265	265	1	2.5
1.0	N	249	249	3	3.0
3.0	N	239	246	3	31.5
1.0	N	142	132	2	67.5

extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount
0.0	0.5	0.0	193.3	0.3	282.1
0.5	0.5	19.85	95.0	0.3	119.15
1.0	0.5	48.0	235.7	0.3	288.0
3.0	0.5	0.0	911.87	0.3	918.67
0.0	0.0	0.0	813.75	0.3	845.55
2.5	0.5	0.0	800.09	0.3	870.89

congestion_surcharge	airport_fee	month	max(Tolls_amount)
0.0	0.0	1	193.3
0.0	0.0	2	95.0
0.0	0.0	3	235.7
2.5	0.0	4	911.87
0	0.0	5	813.75
2.5	0.0	6	800.09

Χρόνοι εκτέλεσης:

	1 worker	2 workers
Execution Time (sec)	29.94	20.75

3.

Εκτελούμε το ερώτημα Q3 χρησιμοποιώντας DataFrame API και RDD API και παρουσιάζουμε τα αποτελέσματα και τους χρόνους εκτέλεσης με χρήση 1 και 2 workers.

- Στο Q3 αναζητούμε τον μέσο όρο της απόστασης και του κόστους, ανά 15 ημέρες, για όλες τις διαδρομές με σημείο αναχώρησης διαφορετικό από το σημείο άφιξης.

DataFrame

Output:

start	end	Avg(Total_amount)	Avg(Trip_distance)
2022-01-01 00:00:00	2022-01-16 00:00:00	19.903702637879007	5.576410377852007
2022-01-16 00:00:00	2022-01-31 00:00:00	19.03660791389491	4.804840472309411
2022-01-31 00:00:00	2022-02-15 00:00:00	19.553891327960553	5.950485844928121
2022-02-15 00:00:00	2022-03-02 00:00:00	20.17207809365826	6.1857672125677
2022-03-02 00:00:00	2022-03-17 00:00:00	20.692357713183547	6.60698631990843
2022-03-17 00:00:00	2022-04-01 01:00:00	21.118287307889744	5.524788048396609
2022-04-01 01:00:00	2022-04-16 01:00:00	21.513246092852775	5.679221475787186
2022-04-16 01:00:00	2022-05-01 01:00:00	21.43101017447181	5.800096624033294
2022-05-01 01:00:00	2022-05-16 01:00:00	21.929327001976056	6.255316989977559
2022-05-16 01:00:00	2022-05-31 01:00:00	22.80847294458172	8.000620246151973
2022-05-31 01:00:00	2022-06-15 01:00:00	22.444346976981922	6.372734051706068
2022-06-15 01:00:00	2022-06-30 01:00:00	22.352414801280606	6.154210472858656
2022-06-30 01:00:00	2022-07-15 01:00:00	22.091536970956383	5.93465896170917

Χρόνοι εκτέλεσης:

	1 worker	2 workers
Execution Time (sec)	16.62	10.3

RDD

Για το RDD αντιστοιχίζουμε τα 15νθήμερα των μηνών με αριθμούς από το 1 έως το 12.

Output:

Fortnight	Avg(Total_amount)	Avg(Trip_distance)
1	19.903702637879007	5.576410377852007
2	19.14882164234129	5.097880367275346
3	19.491979067238603	6.24888833846387
4	20.18769180439039	5.849460516243601
5	20.65227817417907	6.480485434052824
6	21.120920554171548	5.5569449358506535
7	21.515559094583587	5.679323077938295
8	21.428088376232783	5.800344707645977
9	21.9215703489091	6.249697852127242
10	22.771948777963715	7.906694182348757

11	22.466305309343248	6.315157336730177
12	22.322209933302126	6.173665478162396

Χρόνοι εκτέλεσης:

	1 worker	2 workers
Execution Time (sec)	299.12	173.92

4.

Εκτελούμε τα ερωτήματα Q4 και Q5 χρησιμοποιώντας το DataFrame API και παρακάτω παρουσιάζουμε τα αποτελέσματα και τους χρόνους εκτέλεσης τους με χρήση 1 και 2 workers.

- Για το Q4 αναζητούμε τις top 3 ώρες αιχμής ανά ημέρα της εβδομάδος, εννοώντας τις ώρες της ημέρας με τον μεγαλύτερο αριθμό επιβατών σε μια κούρσα ταξί.

Output:

weekday	hour	passenger_count	rank
1	19	15006.54	1
1	18	14840.15	2
1	15	14694.81	3
2	19	16070.81	1
2	21	15782.35	2
2	18	14976.04	3
3	21	18441.85	1
3	20	18368.54	2
3	19	16635.23	3
4	21	17480.19	1
4	19	16091.35	2
4	20	15828.46	3
5	21	20550.23	1
5	20	18185.69	2
5	22	17727.46	3
6	21	19590.84	1
6	22	18789.36	2
6	19	17468.48	3
7	20	18140.38	1

7	21	16959.58	2
7	18	16753.38	3

Χρόνοι εκτέλεσης:

	1 worker	**2 workers**
Execution Time (sec)	33.68	20.94

- Για το Q5 αναζητούμε τις κορυφαίες top 5 ημέρες ανά μήνα στις οποίες οι κούρσες είχαν το μεγαλύτερο ποσοστό σε tip.

Output:

DayOfMonth	**Month**	**Tip_percentage_per_day(%)**
9	1	45.66283036959261
31	1	43.91081781426741
1	1	29.232385664667696
29	1	24.005644390312366
16	1	23.36952183947488
21	2	25.971838067219053
13	2	24.535298013813403
9	2	23.90590714658549
10	2	23.345227307186523
27	2	23.289467018935383
18	3	29.69141014657725
21	3	27.567380399656017
26	3	22.70981817419811
5	3	22.559160264555523
12	3	22.096141323139058
12	4	48.49086831936195
2	4	31.182343581733473
21	4	30.443809755531518
3	4	24.334691397434888
30	4	22.02484774478965
12	5	32.42044332475968
20	5	26.035244715051892
16	5	23.639931330245297
15	5	22.05169698578769
6	5	21.83148230862558
13	6	38.45733182725732

25	6	32.88334436951147
10	6	27.414238108844213
16	6	25.53350445284533
20	6	24.216057680366593

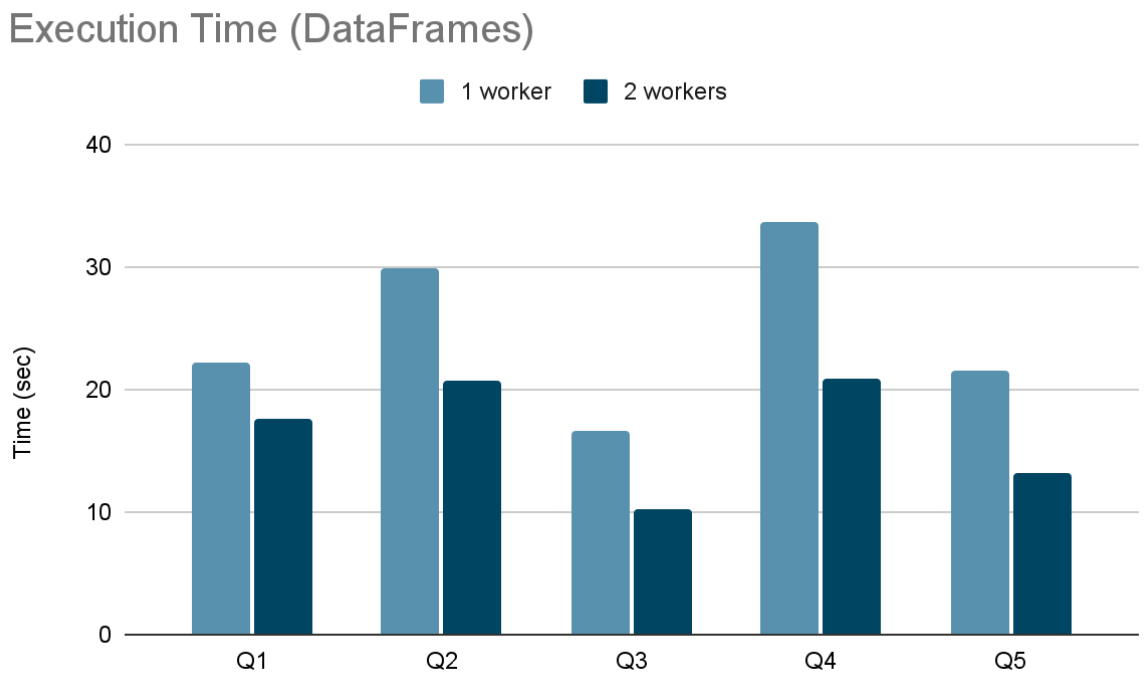
Χρόνοι εκτέλεσης:

	1 worker	**2 workers**
Execution Time (sec)	21.59	13.11

Συνολικός πίνακας σύγκρισης των χρόνων εκτέλεσης των ερωτημάτων με 1 και 2 workers:

Συνολικά Δεδομένα:

	Execution Time (sec)	
	1 worker	**2 workers**
Q1	22.19	17.58
Q2	29.94	20.75
Q3 DF	16.62	10.3
Q3 RDD	299.12	173.92
Q4	33.68	20.94
Q5	21.59	13.11



Παρατηρούμε μεγάλη χρονική διαφορά μεταξύ των εκτελέσεων με RDD και με Dataframe. Αυτό συμβαίνει γιατί τα RDDs είναι αρκετά πιο αργά στη εκτέλεση απλών πράξεων όπως την ομαδοποίηση δεδομένων σε σχέση με τα Dataframes. Επίσης, τα Dataframes έχουν built-in optimizer σε αντίθεση με τα RDDs.
Συνολικά, σε όλες τις περιπτώσεις παρατηρούμε ότι ο χρόνος εκτέλεσης με έναν worker είναι μεγαλύτερος του χρόνου εκτέλεσης με 2 workers.