ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

<<Big Data Εργασία 1 - Αναφορά Map Reduce>>

Τσορμπάρη Παρασκευή (ics22050)

Φουλίδης Δημήτριος (iis22026)

Θεσσαλονίκη, Νοέμβριος 2024

Πίνακας περιεχομένων

1. Ο Αλγόριθμος του προβλήματος

Ο αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του προβλήματος βασίστηκε στο μοντέλο MapReduce το οποίο μετασχηματίζει τα δεδομένα εισόδου σε ζεύγη κλειδιού - τιμής (key-value pairs) με την χρήση των συναρτήσεων map και reduce. Συγκεκριμένα, στο πρόβλημα των συγγραφικών ομάδων, η υλοποίηση βασίστηκε σε δύο φάσεις MapReduce. Κατά τη πρώτη φάση, γίνεται επεξεργασία των δεδομένων έτσι ώστε να βρεθεί το μέσο μέγεθος όλων των συγγραφικών ομάδων, ενώ στη δεύτερη γίνεται περαιτέρω επεξεργασία με στόχο την εύρεση του αριθμού των ομάδων με μέγεθος μικρότερο από τον μέσο όρο.

1.1 Πρώτος κύκλος

Στην πρώτη φάση, ο Mapper αξιοποίησε το δεύτερο πεδίο κάθε εγγραφής με τα ονόματα των συγγραφέων από τα αρχεία εισόδου για να εξάγει κατάλληλα ζεύγη κλειδιού-τιμής. Το κάθε κλειδί αντιπροσωπεύει τους συγγραφείς κάθε ομάδας, ενώ η τιμή αντιστοιχεί στο μέγεθός της, δηλαδή στο πλήθος των συγγραφέων που την απαρτίζουν.

Επειδή ο Mapper δημιουργεί ζεύγη για κάθε εγγραφή στα δεδομένα, κάποιες συγγραφικές ομάδες και το μέγεθός τους εμφανίζονται παραπάνω από μια φορά στην έξοδο. Ο Reducer, λοιπόν, λαμβάνει ως είσοδο τα αποτελέσματα από τον Mapper και για κάθε μοναδικό κλειδί καταγράφει το μέγεθος της ομάδας που αναπαριστά μόνο μία φορά, σε ένα μοναδικό ζεύγος στην έξοδο. Επιπλέον, καταγράφεται ο συνολικός αριθμός ομάδων και συγγραφέων προκειμένου να υπολογιστεί το μέσο μέγεθος ομαδων. Με αυτό τον τρόπο, η έξοδος της πρώτης φάσης περιέχει κάθε ομάδα καταγεγραμμένη μόνο μία φορά και παρέχει τα κατάλληλα δεδομένα για τον υπολογισμό που ζητείται.

1ος Mapper

Είσοδος:

id;authors;;;;...
1442405;Anna Bernasconi|Carsten Damm|Igor Shparlinski;;;;; …
1442405;Anna Bernasconi|Carsten Damm|Igor Shparlinski;;;;; …
2456312;John Smith|Maria Gonzalez|Ahmed Khan|Oliver Brown|Su Wei;;;;; …
3478924;Emily Davis|Hiroshi Tanaka;;;;; …
…

Έξοδος:

“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“John Smith|Maria Gonzalez|Ahmed Khan|Oliver Brown|Su Wei”, 4
“Emily Davis|Hiroshi Tanaka”, 2
…

Ψευδοκώδικας:

```
map (input_key, input_value)
        for each record in input_value //i.e. in the document text
                record_fields: = split(record)
                authors: = split(record_fields[1]) //save author names
                emit_intermediate(record_fields[1] + “,” , authors.length)
```

1ος Reducer

Είσοδος:

“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“John Smith|Maria Gonzalez|Ahmed Khan|Oliver Brown|Su Wei”, 4
“Emily Davis|Hiroshi Tanaka”, 2
…

Έξοδος:

“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“John Smith|Maria Gonzalez|Ahmed Khan|Oliver Brown|Su Wei”, 4
“Emily Davis|Hiroshi Tanaka”, 2
…

Ψευδοκώδικας:

```
//team = record_fields[1] + “,”
//values = [authors.length, authors.length…]
reduce (team, values)
        firstValue:= values[0]
        size+= values[0].get()
        team_counter++
        emit_intermediate (team, firstValue)
```

1.2 Δεύτερος κύκλος

Ο Mapper του δεύτερου κύκλου δέχεται ως είσοδο την έξοδο του πρώτου Reducer και εξάγει ζεύγη κλειδιού-τιμής για τις ομάδες που έχουν μέγεθος μικρότερο από τον μέσο όρο. Το κλειδί είναι το “BelowAverage” και παραμένει ίδιο για όλες τις εγγραφές, ενώ η τιμή για κάθε κλειδί ορίζεται ως 1. Ο δεύτερος Reducer συγκεντρώνει το άθροισμα των τιμών του μοναδικού κλειδιού που προκύπτει από τον Mapper, το οποίο ουσιαστικά αντιπροσωπεύει τον αριθμό των ομάδων που έχουν μέγεθος μικρότερο από τον μέσο όρο. Λόγω της ύπαρξης ενός μόνο κλειδιού εισόδου, θα παραχθεί ένα μόνο ζεύγος κλειδιού-τιμής.

2ος Mapper

Είσοδος:

“Anna Bernasconi|Carsten Damm|Igor Shparlinski”, 3
“John Smith|Maria Gonzalez|Ahmed Khan|Oliver Brown|Su Wei”, 4
“Emily Davis|Hiroshi Tanaka”, 2
…

Έξοδος:

BelowAverage, 1
BelowAverage, 1
BelowAverage, 1
…

Ψευδοκώδικας:

```
map (input_key, input_value)
        for each line in input_value
                team_and_size: = split(line)
                //average size calculated from run configuration data
                if(team_and_size[1] < averageSize)
                        emit_intermediate(“BelowAverage”, 1)
```

2ος Reducer

Είσοδος:

BelowAverage, 1
BelowAverage, 1
BelowAverage, 1
…

Έξοδος:

”Number of teams below average size”, 1133676

Ψευδοκώδικας:

```
//values = [1,1,1…]
reduce (“BelowAverage”, Iterator values)
        for each val in values
                teamsBelowAverage += val.get()
        emit(“Number of teams below average size, teamsBelowAverage)
```

2. Πακέτα και παράμετροι συστήματος

Για την λύση του προβλήματος χρησιμοποιήθηκαν οι βιβλιοθήκες του Hadoop, οι οποίες προστέθηκαν ως maven dependencies. Δεν χρησιμοποιήθηκε κάποιο άλλο εξωτερικό πακέτο. Σχετικά με τις παραμέτρους του συστήματος, χρειάστηκε επέκταση του αποθηκευτικού χώρου της εικονικής μηχανής (επιλέχθηκαν τα 30GB) για την επιτυχή εκτέλεση του προγράμματος και δηλώθηκαν ορίσματα για τον φάκελο εισόδου και τους φακέλους εξόδου των κύκλων στο eclipse. Ενδεικτικά, για 1 Reduce Task:

1. Project path: /home/bigdata/BigDataMapReduce
2. Path εισόδου των δεδομένων: /home/bigdata/input/MapReduceInput (args0)
3. Path εξόδου αποτελεσμάτων 1ου κύκλου: /home/bigdata/Desktop/BigData_Task1/Results/1_Reduce_Task/teams_and_size.txt (args1)
4. Path εξόδου αποτελεσμάτων 2ου κύκλου: /home/bigdata/Desktop/BigData_Task1/Results/1_Reduce_Task/finalAverage.txt (args2)

Η δημιουργία του Jar αρχείου έγινε κάνοντας export το project ως JAR File και επιλέγοντας την κύρια κλάση App.java ως κλάση εκτέλεσης. Η εκτέλεση του προγράμματος στο τερματικό προϋποθέτει την έναρξη των σχετικών υπηρεσιών κατανεμημένης αποθήκευσης και διαχείρισης πόρων HDFS και YARN του Hadoop. Αρχικά, το Hadoop βασίζεται στο πρωτόκολλο SSH (Secure Shell) για την επικοινωνία μεταξύ των κόμβων μέσα στο κατανεμημένο σύστημα. Για την αυθεντικοποίηση των εν λόγω συνδέσεων χωρίς password, χρησιμοποιείται η εντολή **ssh-keygen** για τη δημιουργία ενός RSA key και η **ssh-copy-id** για την αντιγραφή του στο αρχείο εγκεκριμένων κλειδιών authorized_keys.

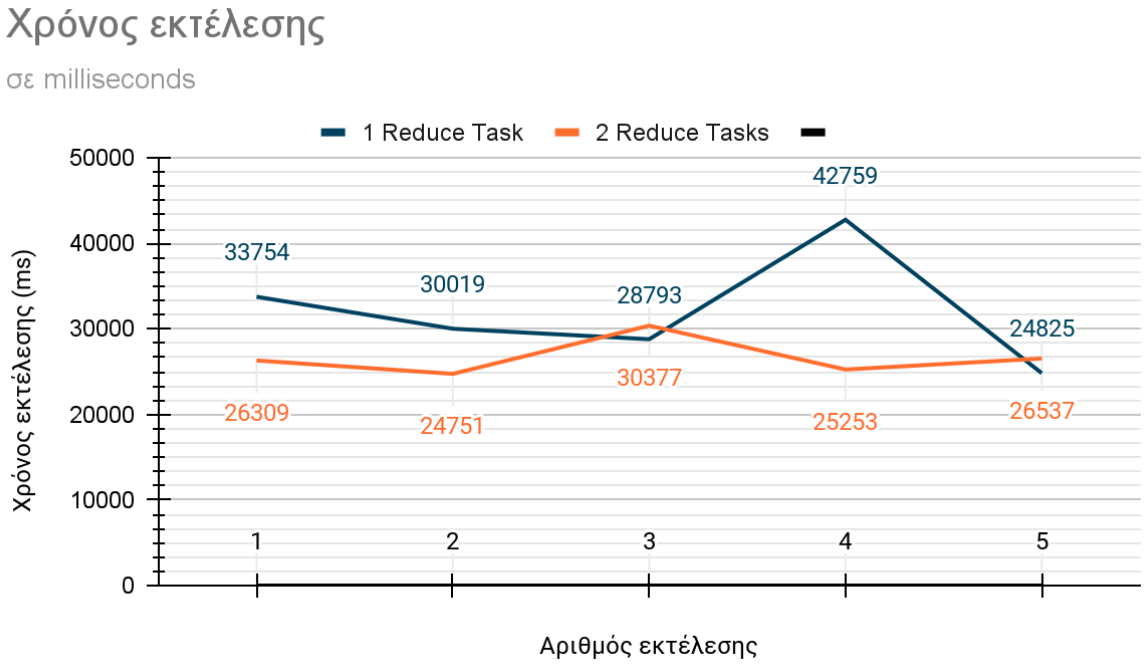
Στη συνέχεια, πραγματοποιείται SSH σύνδεση με την εντολή **ssh localhost** και εκκινούνται οι παραπάνω υπηρεσίες με τις εντολές **start-dfs.sh** και **start-yarn.sh**. Είναι σημαντικό να σημειωθεί πως χρειάζεται αντιγραφή των αρχείων εισόδου από την τοπική αποθήκευση στο σύστημα HDFS για την εκτέλεση του αρχείου και την κατανεμημένη του επεξεργασία εκτελώντας την εντολή **hdfs dfs -put local_path target_path,** όπου **local_path** είναι το μονοπάτι στο οποίο βρίσκονται τοπικά τα αρχεία εισόδου και **target_path** το επιθυμητό μονοπάτι αποθήκευσης μέσα στο HDFS. Τέλος, για την εκτέλεση του προγράμματος χρησιμοποιείται η εντολή hadoop jar fileName.jar **target_path job1OutputPath finalOutputPath**, όπου **target_path** το μονοπάτι του αρχείου εισόδου στο HDFS, **job1OutputPath** το μονοπάτι αποθήκευσης του αποτελέσματος του πρώτου Reducer στο HDFS και **finalOutputPath** το μονοπάτι αποθήκευσης του τελικού αποτελέσματος.

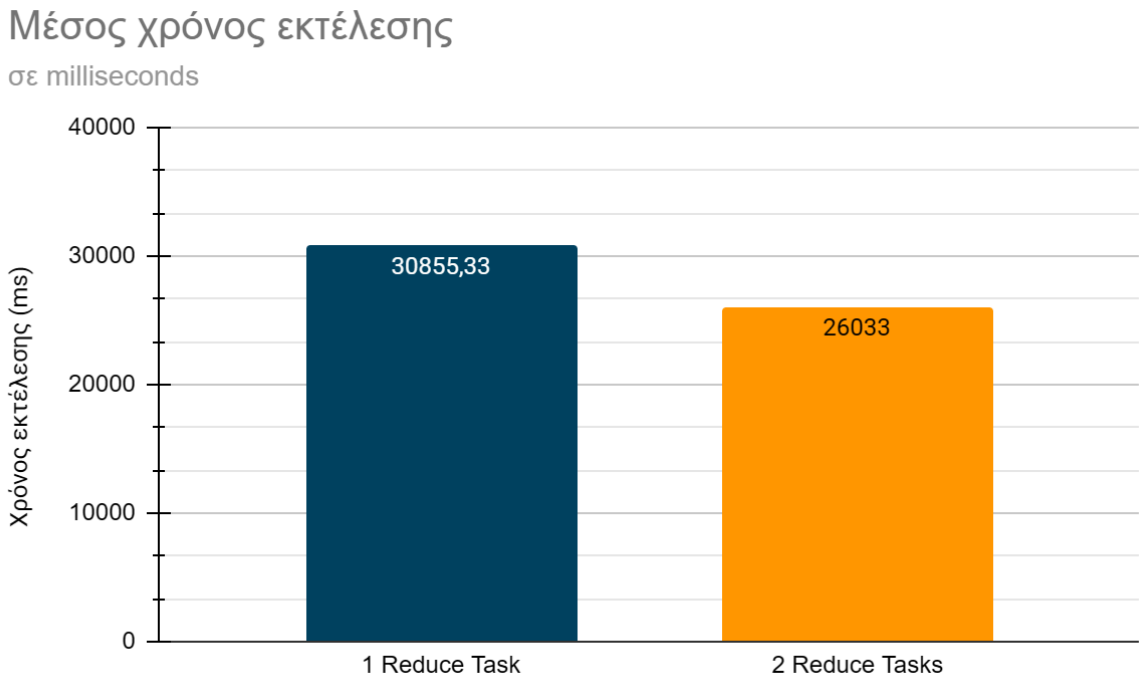
3. Χρόνοι εκτέλεσης, γραφικές παραστάσεις και σχολιασμός

Χρόνοι εκτέλεσης με 1 και 2 Reduce Tasks σε milliseconds (Ο μέσος χρόνος υπολογίστηκε με τον μέσο όρο των χρόνων εκτέλεσης χωρίς τις 2 ακραίες τιμές):

	1 Reduce Task	**2 Reduce Tasks**
1.	33754 ms	26309 ms
2.	30019 ms	24751 ms
3.	28793 ms	30377 ms
4.	42759 ms	25253 ms
5.	24825 ms	26537 ms
Μέσος χρόνος	**30855,33 ms**	**26033 ms**

- Διαφορά Μέσου χρόνου εκτέλεσης: 30855,33 - 26033 = **4822,33 ms**
- Διακύμανση χρόνου εκτέλεσης με 1 Reduce Task: 42759 - 24825 = **17934 ms**
- Διακύμανση χρόνου εκτέλεσης με 2 Reduce Tasks: 30377 - 25253 = **5626 ms**





Συγκριτικά σχόλια

1. Γενικά, η εκτέλεση με **2 Reduce Tasks** φαίνεται να είναι ταχύτερη από την εκτέλεση με **1 Reduce Task**. Αυτό είναι αναμενόμενο, καθώς η κατανομή του φόρτου εργασίας σε περισσότερα Reduce Tasks οδηγεί σε καλύτερη παράλληλη εκτέλεση και ταχύτερη ολοκλήρωση.

2. Η διαφορά στους μέσους χρόνους εκτέλεσης είναι σημαντική (**4822,33 ms ή 15,63%**) πράγμα που επαληθεύει πως η εκτέλεση με 2 Reduce Tasks είναι πιο αποτελεσματική.

3. Η αποτελεσματικότητα των 2 Reduce Tasks είναι εμφανής και από τη μεγαλύτερη διακύμανση στους χρόνους εκτέλεσης που σημειώνεται (**17934 ms** με 1 Task έναντι **5626** ms με 2). Αυτό υποδεικνύει ότι το σύστημα δεν καταφέρνει να εξισορροπήσει τον φόρτο το ίδιο αποδοτικά με μόνο 1 Reduce Task. Οι χρόνοι με 2 Reduce Tasks είναι πιο σταθεροί και με μικρότερες διακυμάνσεις, γεγονός που δείχνει ότι η παράλληλη εκτέλεση με δύο Reduce Tasks βοηθά στην παραπάνω εξομάλυνση.