Πανεπιστήμιο Μακεδονίας

Τμήμα Εφαρμοσμένης Πληροφορικής

Μάθημα: BIG DATA - Ανάλυση Δεδομένων Μεγάλου Όγκου

Ομαδική Εργασία 1

Θέμα: MapReduce, Hadoop

Παρασκευή Ξανθοπούλου (it1490)
Κωνσταντίνος Τσεμπέρης (ics20071)

Καθηγητής: Αλέξανδρος Καρακασίδης
Ακαδημαϊκό Έτος: 2023-2024

Θεσσαλονίκη, Νοέμβριος 2023

Περιεχόμενα

1. Περιγραφή Αλγορίθμου **3**
1.1. Πακέτα/Παράμετροι **5**
2. Χρονική Αξιολόγηση Διαμορφώσεων **6**
2.1 Πίνακες 6
2.2 Γραφικές Παραστάσεις 8
2.1 Elapsed Time 8
2.2 Average Map Time 9
2.3 Average Shuffle Time 10
2.4 Average Merge Time 11
2.5 Average Reduce Time 12
2.5 Σύγκριση 13

1. Περιγραφή Αλγορίθμου

Ο αλγόριθμος που χρησιμοποιήθηκε αποτελείται από δύο κύκλους εκτέλεσης MapReduce, έναν πρώτο κύκλο για την αντιστοίχιση των IP με τη συχνότητά τους και έναν δεύτερο κύκλο για την ταξινόμηση των IP με βάση τη συχνότητα.

Πρώτος κύκλος

MAP()

```
columns = Split(line, “,”)
for every line of Block:
     Ip = columns[0]
     emit(<Ip, 1>)
end for
```

Σχόλια

- Για κάθε γραμμή εισόδου, χωρίζουμε με βάση το “,‘ στις στήλες ενός πίνακα columns, παίρνουμε το Ip από τη στήλη columns[0], και εκπέμπουμε το ζεύγος key-value <Ip, 1>.

REDUCE()

```
for every Ip of Keys:
     RETRIEVE(Ip, values = [1,1, … ,1])
     emit(<Ip, sum(values)>)
end for
```

Σχόλια

- Το MapReduce framework συγκεντρώνει όλα τα key-value ζεύγη από το Map και τα ομαδοποιεί σύμφωνα με το κλειδί (Ip).
- Στη φάση Reduce, για κάθε κλειδί (Ip), προσθέτουμε όλες τις τιμές (values = [1,1, … ,1]) για τη συγκεκριμένη διεύθυνση Ip, υπολογίζοντας τη συχνότητα της και εκπέμπουμε το ζεύγος <Ip, συχνότητα>.
- Τα αποτελέσματα του πρώτου κύκλου είναι ήδη ομαδοποιημένα σύμφωνα με τη διεύθυνση IP και αποτελούν την είσοδο του δεύτερου κύκλου.

Δεύτερος κύκλος

MAP()

```
parts = Split(line, “\\s+”)
for every line:
     Ip = parts[0]
     frequency = parts[1]
     compositeKey.Set(frequency, Ip)
     emit(<compositeKey, Ip>)
end for
```

Σχόλια

- Χωρίζουμε κάθε γραμμή εισόδου (με βάση το κενό) σε δύο μέρη, όπου το πρώτο μέρος είναι η Ip και το δεύτερο μέρος η συχνότητα (frequency). Δημιουργούμε ένα σύνθετο κλειδί (compositeKey) το οποίο αποτελείται από το ζεύγος <συχνότητα, Ip>. Εκπέμπουμε το ζεύγος <compositeKey, Ip>.
- Κατά τη φάση Ταξινόμησης και Μείξης, τα ζεύγη ταξινομούνται ως προς το compositeKey, του οποίου πρώτο στοιχείο είναι η συχνότητα. Με τον τρόπο αυτό πετυχαίνουμε την ταξινόμηση ως προς τη συχνότητα.
- Για την ταξινόμηση χρησιμοποιείται η κλάση CompositeKeyComparator όπου περιλαμβάνεται η μέθοδος compare για τη σύγκριση και ταξινόμηση ως προς τη συχνότητα με φθίνουσα σειρά (σε περίπτωση ίσων συχνοτήτων ταξινομούμε με αύξουσα Ip).

REDUCE()

```
for every Ip of compositeKeys:
     emit(<Ip, compositeKey.getFrequency()>)
end for
```

Σχόλια

- Η φάση Reduce δέχεται ως είσοδο τα ζευγη <CompositeKey, Ip> ταξινομημένα ως προς τη συχνότητα με φθίνουσα σειρά.
- Εκπέμπουμε το ζεύγος <Ip, συχνότητα> δημιουργώντας έτσι μια λίστα με τις Ip ταξινομημένες με φθίνουσα σειρά συχνότητας.

1.1. Πακέτα/Παράμετροι

Στο πλαίσιο της υλοποίησης του προγράμματος MapReduce, χρησιμοποιήθηκαν διάφορα πακέτα τα οποία βοήθησαν στην επιτυχή εκτέλεση και στην παραγωγή των απαιτούμενων αποτελεσμάτων. Αναλύουμε τις εξαρτήσεις και τις παραμέτρους συστήματος που τροποποιήθηκαν:

Πακέτα:

- **java.io.DataInput** και **java.io.DataOutput**: Τα πακέτα αυτά χρησιμοποιήθηκαν για την ανάγνωση και την εγγραφή δεδομένων στις συναρτήσεις write και readFields της κλάσης CompositeKey.
- **java.io.IOException**: Το πακέτο αφορά τον χειρισμό εξαιρέσεων που σχετίζονται με το I/O.
- **org.apache.hadoop.conf.Configuration**, **org.apache.hadoop.fs.Path**, **org.apache.hadoop.io.***, **org.apache.hadoop.mapreduce.***, κλπ: βιβλιοθήκες που παρέχονται από το Hadoop.

Παράμετροι Συστήματος:

- **mapreduce.job.reduces**: Αυτή η παράμετρος συστήματος ρυθμίστηκε για τον προσδιορισμό του πλήθους των διεργασιών Reduce που εκτελούνται σε κάθε κύκλο MapReduce. Αντί για την εντολή -D mapreduce.job.reduces=(#tasks), το πλήθος διεργασιών καθορίστηκε προγραμματιστικά στον κώδικα, χρησιμοποιώντας τη μέθοδο job.setNumReduceTasks(int num).

2. Χρονική Αξιολόγηση Διαμορφώσεων

Πραγματοποιήθηκαν μετρήσεις χρόνων εκτέλεσης για τις εξής 6 διαμορφώσεις:

- ένας κόμβος: 1,2 και 4 διεργασίες Reduce
- δύο κόμβοι: 1,2 και 4 διεργασίες Reduce

Για την εκτέλεση σε έναν κόμβο, εκτελέστηκε από τον κόμβο slave η εντολή **~/hadoop/sbin/yarn-daemon.sh stop nodemanager**, ώστε να τερματιστεί η υπηρεσία NodeManager στον συγκεκριμένο κόμβο. Ως αποτέλεσμα, οι επόμενες εργασίες του Hadoop έτρεξαν μόνο στον κόμβο master.

2.1 Πίνακες

Για έναν κόμβο:

Reduce Tasks	Iter	Elapsed Time	Av. Map Time	Av. Shuffle Time	Av. Merge Time	Av. Reduce Time
1	1	~~4:22~~	~~0:12~~	~~2:23~~	~~0:06~~	~~0:57~~
	2	4:26	0:12	2:24	0:07	1:01
	3	4:27	0:12	2:24	0:06	1:03
	4	4:27	0:12	2:23	0:07	1:04
	5	~~4:31~~	~~0:12~~	~~2:24~~	~~0:07~~	~~0:57~~
	Av.	**4:26**	**0:12**	**2:23**	**0:06**	**1:02**
2	1	~~4:09~~	~~0:11~~	~~2:24~~	~~0:03~~	~~0:29~~
	2	4:10	0:10	2:23	0:03	0:29
	3	4:10	0:10	2:22	0:03	0:29
	4	4:13	0:11	2:25	0:04	0:29
	5	~~4:12~~	~~0:10~~	~~2:26~~	~~0:03~~	~~0:30~~
	Av.	**4:11**	**0:10**	**2:23**	**0:03**	**0:29**
4	1	~~4:27~~	~~0:12~~	~~1:54~~	~~0:03~~	~~0:14~~
	2	4:27	0:12	1:55	0:03	0:14
	3	4:28	0:12	1:56	0:03	0:14
	4	4:29	0:12	1:54	0:04	0:14
	5	~~4:37~~	~~0:13~~	~~1:56~~	~~0:03~~	~~0:14~~
	Av.	**4:28**	**0:12**	**1:55**	**0:03**	**0:14**

Για δύο κόμβους:

Reduce Tasks	Iter	Elapsed Time	Av. Map Time	Av. Shuffle Time	Av. Merge Time	Av. Reduce Time
1	1	~~3:08~~	~~0:14~~	~~1:35~~	~~0:06~~	~~0:59~~
	2	3:09	0:14	1:33	0:07	1:00
	3	3:19	0:12	1:47	0:06	1:00
	4	3:19	0:14	1:42	0:07	1:00
	5	~~3:37~~	~~0:16~~	~~1:57~~	~~0:06~~	~~0:59~~
	Av.	**3:15**	**0:13**	**1:40**	**0:06**	**1:00**
2	1	~~2:38~~	~~0:12~~	~~1:29~~	~~0:05~~	~~0:30~~
	2	2:41	0:11	1:33	0:07	0:29
	3	2:43	0:13	1:31	0:07	0:30
	4	2:44	0:14	1:33	0:07	0:30
	5	~~2:45~~	~~0:11~~	~~1:37~~	~~0:06~~	~~0:30~~
	Av.	**2:42**	**0:12**	**1:32**	**0:07**	**0:29**
4	1	~~2:27~~	~~0:11~~	~~1:27~~	~~0:03~~	~~0:15~~
	2	2:28	0:13	1:28	0:03	0:14
	3	2:31	0:12	1:29	0:04	0:14
	4	2:31	0:12	1:31	0:03	0:13
	5	~~2:43~~	~~0:15~~	~~1:39~~	~~0:04~~	~~0:14~~
	Av.	**2:30**	**0:12**	**1:29**	**0:03**	**0:13**

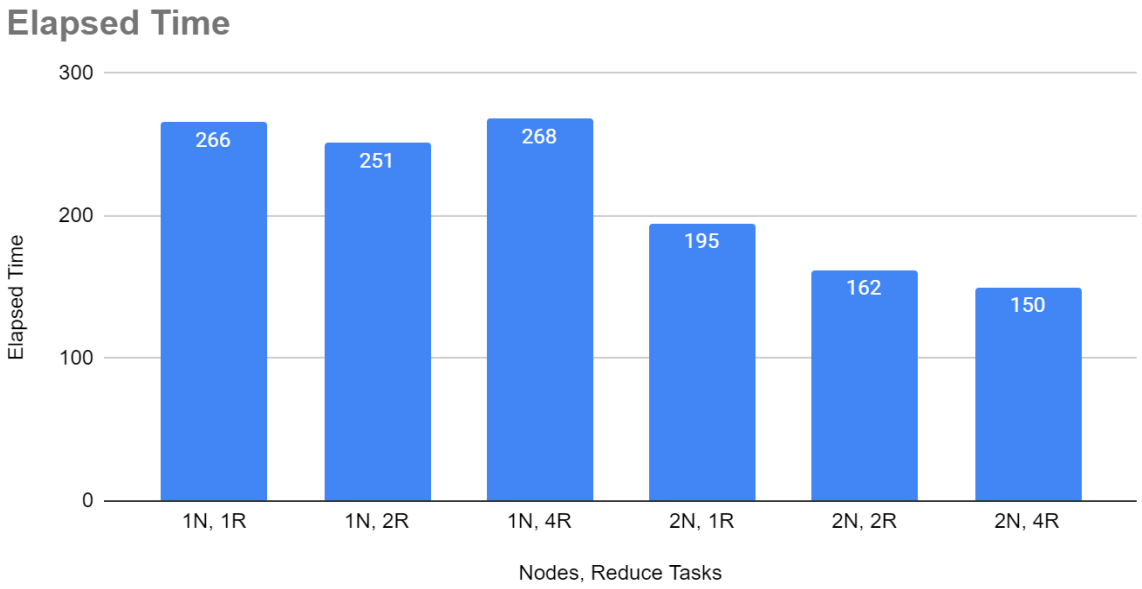
Οι μέσοι χρόνοι για κάθε διαμόρφωση, μετά από μετατροπή σε δευτερόλεπτα

Nodes, Reduce Tasks	**Elapsed Time**	**Av. Map Time**	**Av. Shuffle Time**	**Av. Merge Time**	**Av. Reduce Time**
1N, 1R	266	12	143	6	62
1N, 2R	251	10	143	3	29
1N, 4R	268	12	115	3	14
2N, 1R	195	13	100	6	60
2N, 2R	162	12	92	7	29
2N, 4R	150	12	89	3	13

Όπως μπορούμε να δούμε, η απόδοση ποικίλλει ανάλογα με τον αριθμό των κόμβων και των διεργασιών Reduce. Διαφορετικές διαμορφώσεις μπορούν να οδηγήσουν σε διαφορές στο χρόνο εκτέλεσης και στους χρόνους που αφορούν συγκεκριμένες εργασίες (π.χ. map, shuffle, merge, reduce). Ακολουθούν γραφικές παραστάσεις βάσει των μετρήσεων του παραπάνω πίνακα και σχολιασμός των ευρημάτων.

2.2 Γραφικές Παραστάσεις

2.1 Elapsed Time



1 κόμβος (1N) vs 2 κόμβων (2N):

- Η χρήση δύο κόμβων οδηγεί σε ταχύτερους συνολικούς χρόνους εκτέλεσης σε σύγκριση με τη χρήση μόνο ενός κόμβου.
- Αυτό αναδεικνύει το πλεονέκτημα της κατανομής του φόρτου εργασίας σε πολλούς κόμβους για παράλληλη επεξεργασία, οδηγώντας σε πιο αποτελεσματική εκτέλεση εργασιών.

Αριθμός διεργασιών Reduce (1R, 2R, 4R):

- Παρατηρείται αξιοσημείωτη μείωση του χρόνου εκτέλεσης καθώς αυξάνεται ο αριθμός των διεργασιών Reduce.
- Οι περισσότερες διεργασίες Reduce οδηγούν σε μικρότερους χρόνους, υποδεικνύοντας ότι η παραλληλοποίηση της φάσης Reduce μειώνει σημαντικά τους χρόνους ολοκλήρωσης της εργασίας.

Διαμόρφωση 1N, 4R:

- Για τη διαμόρφωση ενός κόμβου και τεσσάρων διεργασιών Reduce, παρατηρήθηκε ο μέγιστος χρόνος εκτέλεσης. Αυτό συμβαίνει είτε λόγω ανισορροπίας των δεδομένων (τα δεδομένα δεν διανέμονται ομοιόμορφα μεταξύ των Reducers) είτε λόγω του Overhead (η ύπαρξη περισσότερων Reducers αυξάνει τον χρόνο εκτέλεσης λόγω αρχικοποίησης των διεργασιών, μεταφοράς των δεδομένων και συγχώνευσης των ενδιάμεσων δεδομένων).

Διαμόρφωση 1N, 2R:

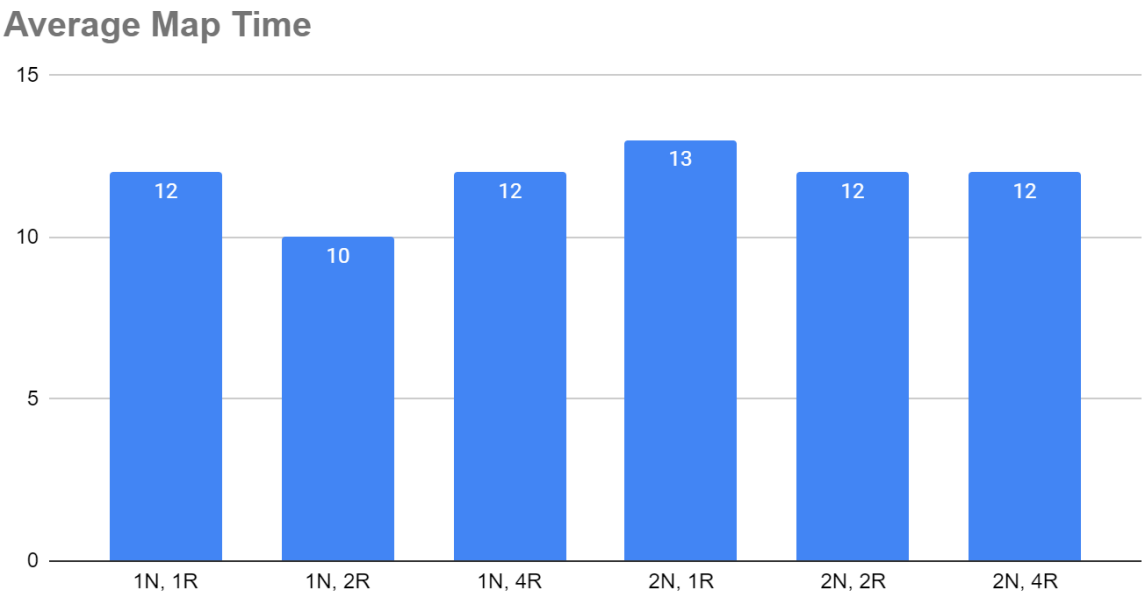
- Μεταξύ των διαμορφώσεων ενός κόμβου, η χρήση 2 διεργασιών Reduce (1N, 2R) οδηγεί στον ταχύτερο χρόνο ολοκλήρωσης (251 δευτερόλεπτα).
- Αυτό υποδηλώνει ότι η ύπαρξη ενός μέτριου αριθμού διεργασιών Reduce μπορεί να είναι πιο αποδοτική από μία ή τέσσερις διεργασίες Reduce όταν χρησιμοποιείται μόνο ένας κόμβος.

Διαμόρφωση 2N, 4R:

- Μεταξύ των διαμορφώσεων 2 κόμβων, η χρήση 4 διεργασιών Reduce (2N, 4R) οδηγεί στον ταχύτερο χρόνο ολοκλήρωσης (150 δευτερόλεπτα).
- Αυτό ενισχύει την ιδέα ότι η κατανομή του φόρτου εργασίας και η αύξηση του παραλληλισμού οδηγούν σε καλύτερες επιδόσεις.

Συνοπτικά, οι μετρήσεις δείχνουν ότι η αύξηση του αριθμού των κόμβων και του αριθμού των διεργασιών Reduce οδηγεί γενικά σε βελτιωμένη απόδοση. Η χρήση περισσότερων κόμβων επιτρέπει την παράλληλη επεξεργασία και η αύξηση του αριθμού των διεργασιών Reduce κατανέμει αποτελεσματικά το φόρτο εργασίας Reduce, με αποτέλεσμα την ταχύτερη εκτέλεση εργασιών.

2.2 Average Map Time



1 κόμβος (1N) vs 2 κόμβοι (2N):

- Κατά μέσο όρο, φαίνεται ότι η χρήση δύο κόμβων (2N) απαιτεί ελαφρώς περισσότερο χρόνο για τις εργασίες Map σε σύγκριση με τη χρήση ενός κόμβου (1N). Αυτό

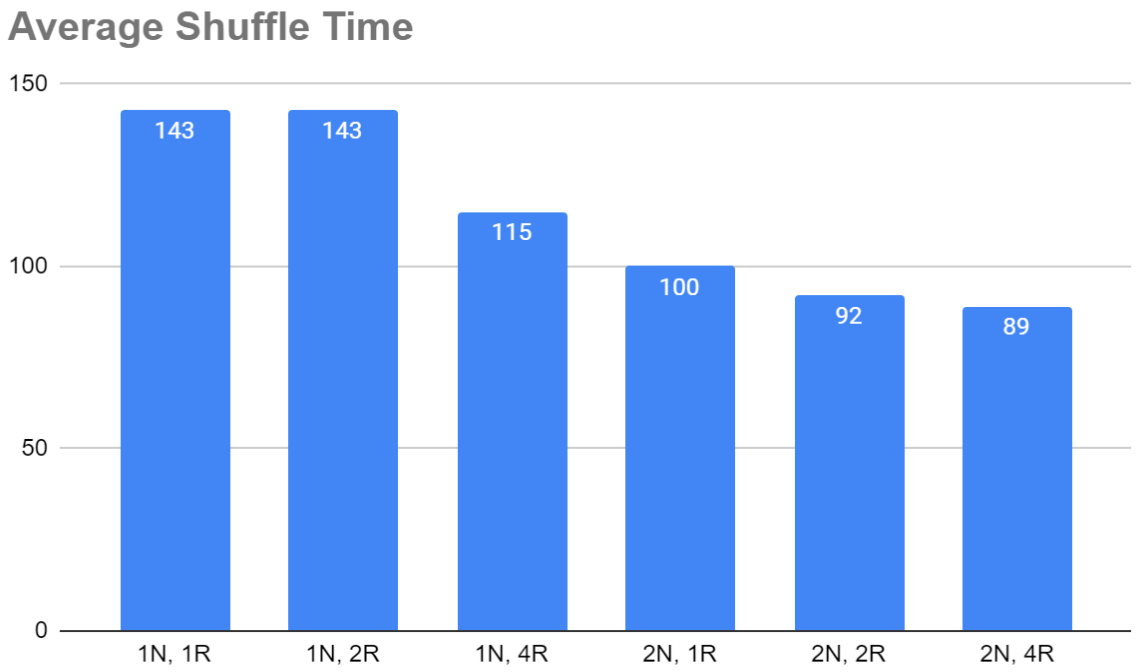
υποδεικνύει ότι η επιβάρυνση από τη διαχείριση πολλαπλών κόμβων μπορεί να οδηγήσει σε μικρή αύξηση του χρόνου εκτέλεσης εργασιών Map.

Αριθμός διεργασιών Reduce (1R, 2R, 4R):

- Ο αριθμός των διεργασιών Reduce δεν φαίνεται να έχει αντίκτυπο στον μέσο χρόνο Map.

Συνοπτικά, οι μετρικές δείχνουν ότι ο μέσος χρόνος Map μπορεί να επηρεαστεί πολύ λίγο από τον αριθμό των κόμβων και των διεργασιών Reduce.

2.3 Average Shuffle Time



1 κόμβος (1N) vs 2 κόμβοι (2N):

- Γενικά, η χρήση 2 κόμβων (2N) οδηγεί σε μικρότερους μέσους χρόνους Μείξης σε σύγκριση με τη χρήση ενός κόμβου (1N). Αυτό υποδηλώνει ότι τα οφέλη της παράλληλης επεξεργασίας και της κατανομής των δεδομένων είναι εμφανή στη φάση αυτή.

Αριθμός διεργασιών Reduce(1R, 2R, 4R):

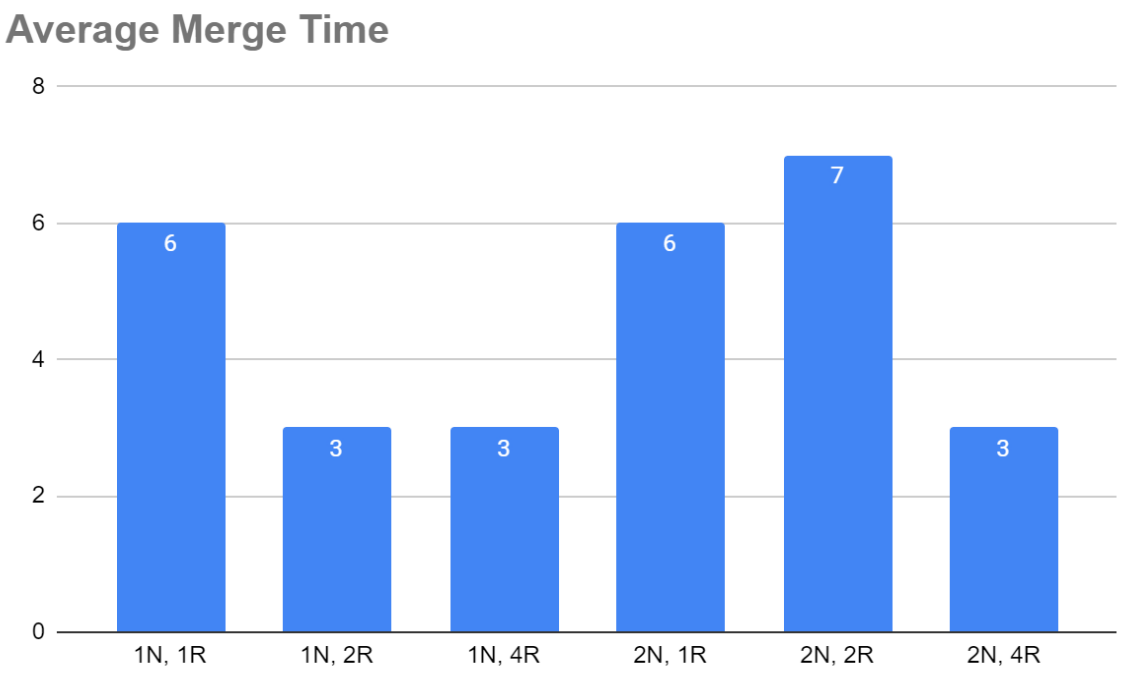
- Καθώς αυξάνεται ο αριθμός των διεργασιών Reduce, ο μέσος χρόνος Μείξης τείνει να μειώνεται, υποδεικνύοντας ότι η παραλληλοποίηση της φάσης Reduce συμβάλλει στην αποτελεσματικότερη Μείξη των δεδομένων.

Διαμόρφωση 2N, 4R:

- Η διαμόρφωση δύο κόμβων και τεσσάρων διεργασιών Reduce αποδίδει τον μικρότερο μέσο χρόνο Μείξης (89 δευτερόλεπτα). Αυτό υποδεικνύει ότι ο συνδυασμός πολλών κόμβων με περισσότερες διεργασίες Reduce οδηγεί σε πιο αποδοτική φάση Μείξης.

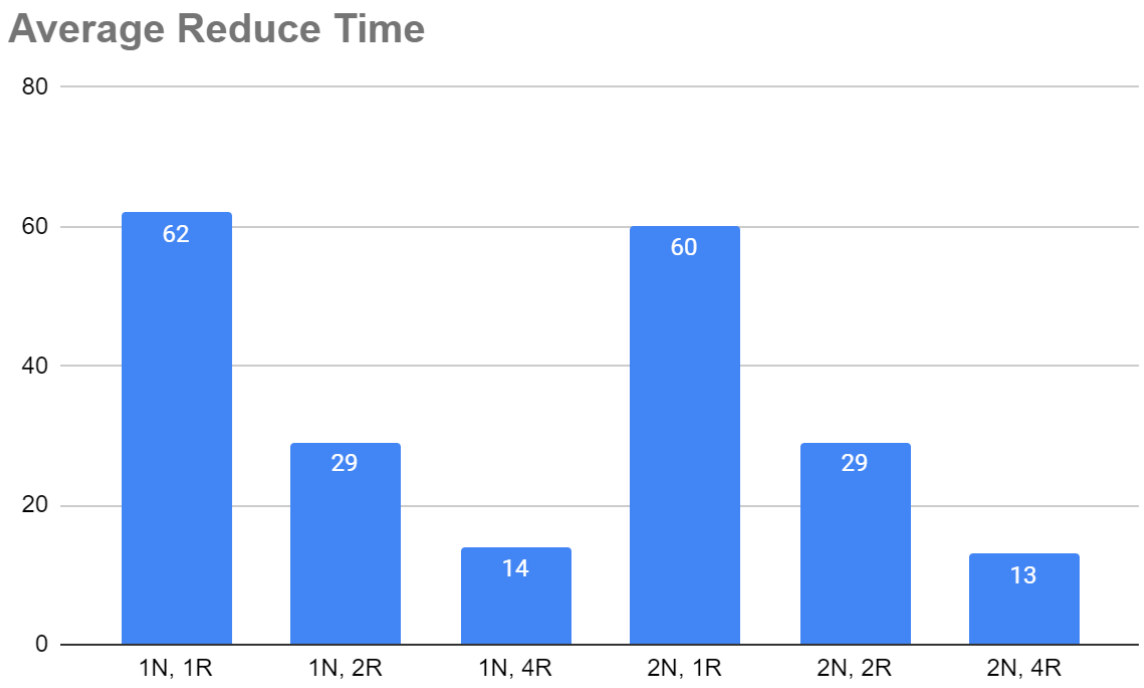
Συνοπτικά, η παράλληλη επεξεργασία και η αύξηση του αριθμού των διεργασιών Reduce οδηγούν γενικά σε βελτιωμένη απόδοση, όσον αφορά τους μέσους χρόνους Μείξης.

2.4 Average Merge Time



Ο μέσος χρόνος συγχώνευσης είναι σχετικά χαμηλός και δεν διαφέρει σημαντικά ανάλογα με τον αριθμό των κόμβων ή των διεργασιών Reduce.

2.5 Average Reduce Time



1 κόμβος (1N) vs 2 κόμβοι (2N):

- Ο αριθμός των κόμβων (1N ή 2N) δεν επηρεάζει σημαντικά τον μέσο χρόνο Reduce.

Αριθμός διεργασιών Reduce (1R, 2R, 4R):

- Η αύξηση του αριθμού των διεργασιών Reduce οδηγεί γενικά σε συντομότερους μέσους χρόνους Reduce, υποδεικνύοντας ότι η παραλληλοποίηση της φάσης Reduce οδηγεί σε πιο αποτελεσματική επεξεργασία.

Διαμόρφωση 1N, 4R:

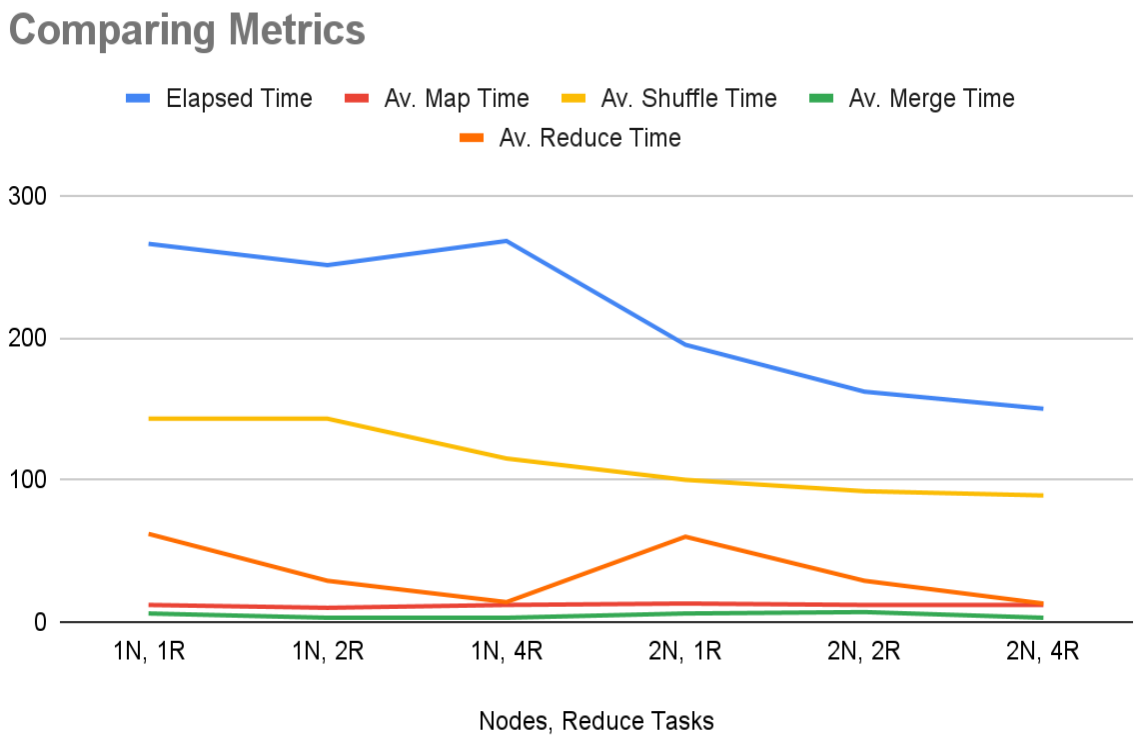
- Μεταξύ των διαμορφώσεων 1 κόμβου, η χρήση 4 διεργασιών Reduce (1N, 4R) οδηγεί στον συντομότερο μέσο χρόνο Reduce (14 δευτερόλεπτα).

Διαμόρφωση 2N, 4R:

- Μεταξύ των διαμορφώσεων 2 κόμβων, η χρήση 4 διεργασιών Reduce (2N, 4R) οδηγεί στον συντομότερο μέσο χρόνο Reduce (13 δευτερόλεπτα).

Συνοπτικά, ο αριθμός των διεργασιών Reduce επηρεάζει σημαντικά τον μέσο χρόνο της φάσης Reduce. Οι διαμορφώσεις με μεγαλύτερο αριθμό διεργασιών Reduce τείνουν να έχουν μικρότερους μέσους χρόνους Reduce, υποδεικνύοντας τα οφέλη της παραλληλοποίησης της φάσης Reduce. Τόσο οι διαμορφώσεις 1 κόμβου όσο και οι διαμορφώσεις 2 κόμβων παρουσιάζουν παρόμοια μοτίβα όσον αφορά την επίδραση του αριθμού των διεργασιών Reduce στην αποδοτικότητα της επεξεργασίας.

2.5 Σύγκριση



Συνολικά, οι μετρικές δείχνουν ότι οι διαμορφώσεις με μεγαλύτερο αριθμό διεργασιών Reduce (2R και 4R) τείνουν να είναι πιο αποδοτικές, οδηγώντας σε συντομότερους χρόνους και πιο ισορροπημένη κατανομή εργασιών. Επιπλέον, οι διαμορφώσεις 2 κόμβων (2N) γενικά υπερτερούν των διαμορφώσεων 1 κόμβου (1N) όσον αφορά τους χρόνους εκτέλεσης και τους χρόνους Reduce, υποδεικνύοντας τα οφέλη της παράλληλης επεξεργασίας.