ΟΜΑΔΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ BIG DATA / ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕΓΑΛΟΥ ΟΓΚΟΥ

Μέλη ομάδας:

Ζέρβας Μιχάλης (ics20015)

Δρίκος Χρήστος(ics20046)

Περιεχόμενα

Για την υλοποίηση χρησιμοποιήθηκε ένας κύκλος MapReduce.

1. Αλγόριθμος που χρησιμοποιήσαμε για να επιλύσουμε το πρόβλημα

1. Ο mapper λαμβάνει ως είσοδο μιά παραγγελία
2. Δημιουργεί όλους τους δυνατούς μοναδικούς συνδυασμούς προϊόντων από τη συγκεκριμένη παραγγελία. Η σειρά των προϊόντων δεν παίζει κάποιο ρόλο. Παράδειγμα ο συνδυασμός F1 F3 θεωρείται ίδιος με τον F3 F1.
3. Ο Mapper περνάει ως κλειδί κάθε δυνατό συνδυασμό προϊόντων που έχει παράχθηκε και ως value για τον καθένα από αυτούς την τιμή 1. Παράδειγμα < [F1 F3] , 1 >
4. Οι Combiners λαμβάνουν ως input το output των Mappers και παράγουν ως έξοδο: για key, κάθε συνδυασμό προϊόντων (προφανώς έναν τη φορά) και για value το άθροισμα των values που αντιστοιχούν σε κάθε συνδυασμό. Π.χ. < [F1 F3] , 300 >
5. Ο Reducer, παίρνοντας ως είσοδο τα αποτελέσματα του Combiner, αθροίζει συνολικά για κάθε συνδυασμό προϊόντων όλα τα values που έχει πάρει από τους combiners και αν το άθροισμα είναι μεγαλύτερο ή ίσο από το threshold, γράφεται ως έξοδος το αποτέλεσμα με key τον συγκεκριμένο συνδυασμό και value το πόσες φορές εμφανίστηκε. Αν είναι μικρότερο από το threshold το αγνοεί. Π.χ. < [F1 F3] , 6000 >

2. Περιορισμοί και θεωρήσεις της λύσης

<u>Περιορισμοί:</u> Ο κώδικας του combiner έπρεπε να είναι λίγο διαφορετικός από του reducer λόγω ελέγχου threshold. Επιλέξαμε να κάνουμε συμπίεση των αποτελεσμάτων που παράγει ο mapper, διότι έτσι μειώνεται ο όγκος των δεδομένων που παράγει.Συνεπώς μεταφέρονται γρηγορότερα στο δίκτυο και μειώνονται τα I/O στον δίσκο. Επιλέξαμε την συμπίεση Snappy για να έχουμε την γρηγορότερη δυνατή συμπίεση, ώστε να μην ζημιωθούμε εν τέλη από το τη διαδικασία της συμπίεσης.

<u>Θεώρηση της αρχικής μας λύσης:</u> Η αρχική μας ιδέα ήταν ότι αν κάποιο μεμονωμένο προϊόν εμφανίζεται πιο λίγες φορές από το threshold να

αφαιρεθεί αυτό καθώς και όλοι οι άλλοι συνδυασμοι στους οποίους εμφανίζεται, γλυτώνοντας έτσι περιττούς ελέγχους. Αυτό όμως απορρίφθηκε διότι μας δυσκόλεψε στην υλοποίηση. Θεωρήσαμε ότι θα διάβαζε τουλάχιστον 2 φορές τα αρχεία εισόδου και θα έγραφε τα ενδιάμεσα αποτελέσματα άλλες τόσες καθώς θα έπρεπε να μεταφερθούν στο δίκτυο. Αυτό το θεωρήσαμε χρονοβόρο αφού το πρόβλημα μπορούσαμε να το υλοποιήσουμε και με έναν κύκλο MapReduce.

3. Επιπλέον πακέτα που χρησιμοποιήθηκαν

Τα πακέτα αυτά χρησιμοποιήθηκαν στον Mapper για την δημιουργία όλων των δυνατών συνδυασμών των προϊόντων μιας παραγγελίας.

- com.google.common.collect.Sets
- com.google.common.collect.ImmutableSet

4. Παράμετροι του συστήματος

Οι επιπλέον παράμετροι είναι το threshold που το δίνουμε ως args. Επίσης δεν χρειάζεται να δίνεται το όνομα της main.

5. Γραφικές παραστάσεις και πίνακες χρόνων

2 Nodes

Threshold 5000:

1. 00hrs, 07mins, 45sec job_1670238997489_0011

Average Map Time	1mins, 56sec
Average Shuffle Time	2mins, 30sec
Average Merge Time	47sec
Average Reduce Time	2mins, 29sec

2. 00hrs, 08mins, 12sec job_1670238997489_0019

Average Map Time	1mins, 35sec

Average Shuffle Time	2mins, 56sec
Average Merge Time	47sec
Average Reduce Time	2mins, 32sec

3. 00hrs, 09mins, 06sec job_1670238997489_0020

Average Map Time	1mins, 39sec
Average Shuffle Time	3mins, 41sec
Average Merge Time	48sec
Average Reduce Time	2mins, 31sec

4. 00hrs, 08mins, 08sec, job_1670238997489_0021

Average Map Time	1mins, 47sec
Average Shuffle Time	2mins, 49sec
Average Merge Time	47sec
Average Reduce Time	2mins, 34sec

5. 00hrs, 12mins, 47sec job_1670238997489_0022

Average Map Time	1mins, 51sec
Average Shuffle Time	7mins, 23sec
Average Merge Time	47sec
Average Reduce Time	2mins, 31sec

Threshold 10000:

1. 00hrs, 12mins, 45sec job_1670238997489_0024

Average Map Time	1mins, 51sec
Average Shuffle Time	7mins, 18sec
Average Merge Time	48sec
Average Reduce Time	2mins, 32sec

2. 00hrs, 10mins, 34sec job_1670238997489_0025

Average Map Time	1mins, 44sec
Average Shuffle Time	5mins, 7sec
Average Merge Time	48sec

Average Reduce Time	2mins, 34sec

3. 00hrs, 08mins, 02sec job_1670238997489_0027

Average Map Time	1mins, 37sec
Average Shuffle Time	2mins, 14sec
Average Merge Time	47sec
Average Reduce Time	2mins, 33sec

4. 00hrs, 08mins, 12sec job_1670238997489_0028

Average Map Time	1mins, 35sec
Average Shuffle Time	2mins, 53sec
Average Merge Time	48sec
Average Reduce Time	2mins, 33sec

5. 00hrs, 08mins, 02sec job_1670366157694_0006

Average Map Time	1mins, 47sec
Average Shuffle Time	2mins, 43sec
Average Merge Time	52sec
Average Reduce Time	2mins, 31sec

Threshold 50000:

1. 00hrs, 12mins, 27sec job_1670238997489_0029

Average Map Time	1mins, 51sec
Average Shuffle Time	6mins, 57sec
Average Merge Time	48sec
Average Reduce Time	2mins, 34sec

2. 00hrs, 08mins, 06sec job_1670238997489_0030

Average Map Time	1mins, 46sec
Average Shuffle Time	2mins, 46sec
Average Merge Time	49sec
Average Reduce Time	2mins, 33sec

3. 00hrs, 08mins, 02sec job_1670238997489_0032

Average Map Time	1mins, 38sec
Average Shuffle Time	2mins, 13sec
Average Merge Time	48sec
Average Reduce Time	2mins, 33sec

4. 00hrs, 08mins, 16sec job_1670238997489_0033

Average Map Time	1mins, 35sec
Average Shuffle Time	2mins, 58sec
Average Merge Time	48sec
Average Reduce Time	2mins, 32sec

5. 00hrs, 08mins, 09sec job_1670366157694_0007

Average Map Time	1mins, 38sec
Average Shuffle Time	2mins, 16sec
Average Merge Time	47sec
Average Reduce Time	2mins, 34sec

1 Node

Threshold 5000:

1. 00hrs, 08mins, 36sec job_1670362094916_0001

Average Map Time	1mins, 26sec
Average Shuffle Time	3mins, 4sec
Average Merge Time	48sec
Average Reduce Time	2mins, 34sec

2. 00hrs, 08mins, 35sec job_1670362094916_0002

Average Map Time	1mins, 26sec
Average Shuffle Time	3mins, 6sec
Average Merge Time	47sec
Average Reduce Time	2mins, 31sec

3. 00hrs, 08mins, 37sec job_1670362094916_0003

Average Map Time	1mins, 26sec
Average Shuffle Time	3mins, 8sec
Average Merge Time	48sec
Average Reduce Time	2mins, 31sec

4. 00hrs, 08mins, 39sec job_1670362094916_0004

Average Map Time	1mins, 26sec
Average Shuffle Time	3mins, 7sec
Average Merge Time	48sec
Average Reduce Time	2mins, 35sec

5. 00hrs, 08mins, 36sec job_1670362094916_0005

Average Map Time	1mins, 26sec
Average Shuffle Time	3mins, 2sec
Average Merge Time	48sec
Average Reduce Time	2mins, 34sec

Threshold 10000:

1. 00hrs, 08mins, 43sec job_1670366157694_0001

Average Map Time	1mins, 27sec
Average Shuffle Time	3mins, 11sec
Average Merge Time	48sec
Average Reduce Time	2mins, 32sec

2. 00hrs, 08mins, 39sec job_1670366157694_0002

Average Map Time	1mins, 27sec
Average Shuffle Time	3mins, 4sec
Average Merge Time	47sec
Average Reduce Time	2mins, 34sec

3. 00hrs, 08mins, 20sec job_1670366157694_0003

Average Map Time	1mins, 27sec
Average Shuffle Time	1mins, 52sec
Average Merge Time	47sec
Average Reduce Time	2mins, 30sec

4. 00hrs, 08mins, 32sec job_1670366157694_0004

Average Map Time 1mins, 26sec
Average Shuffle Time 3mins, 1sec
Average Merge Time 46sec
Average Reduce Time 2mins, 32sec

5. 00hrs, 08mins, 34sec job_1670366157694_0005

Average Map Time 1mins, 27sec
Average Shuffle Time 3mins, 7sec
Average Merge Time 46sec
Average Reduce Time 2mins, 31sec

Threshold 50000:

1. 00hrs, 08mins, 25sec job_1670366157694_0008

Average Map Time 1mins, 27sec
Average Shuffle Time 1mins, 57sec
Average Merge Time 47sec
Average Reduce Time 2mins, 34sec

2. 00hrs, 08mins, 39sec job_1670366157694_0009

Average Map Time 1mins, 27sec
Average Shuffle Time 3mins, 10sec
Average Merge Time 48sec
Average Reduce Time 2mins, 32sec

3. 00hrs, 08mins, 38sec job_1670366157694_0010

Average Map Time 1mins, 27sec
Average Shuffle Time 3mins, 6sec
Average Merge Time 47sec
Average Reduce Time 2mins, 32sec

4. 00hrs, 08mins, 40sec job_1670366157694_0011

Average Map Time 1mins, 26sec
Average Shuffle Time 3mins, 8sec
Average Merge Time 48sec
Average Reduce Time 2mins, 33sec

5. 00hrs, 08mins, 28sec job_1670366157694_0012

Average Map Time 1mins, 27sec

Average Shuffle Time	1mins, 58sec
Average Merge Time	47sec
Average Reduce Time	2mins, 35sec

Average *Elapsed Time*

- 2 Nodes:
  threshold 5000: 00:08:29
  threshold 10000: 00:08:56
  threshold 50000: 00:08:10

- 1 Node:
  threshold 5000: 00:08:36
  threshold 10000: 00:08:35
  threshold 50000: 00:08:35

Average Map Time

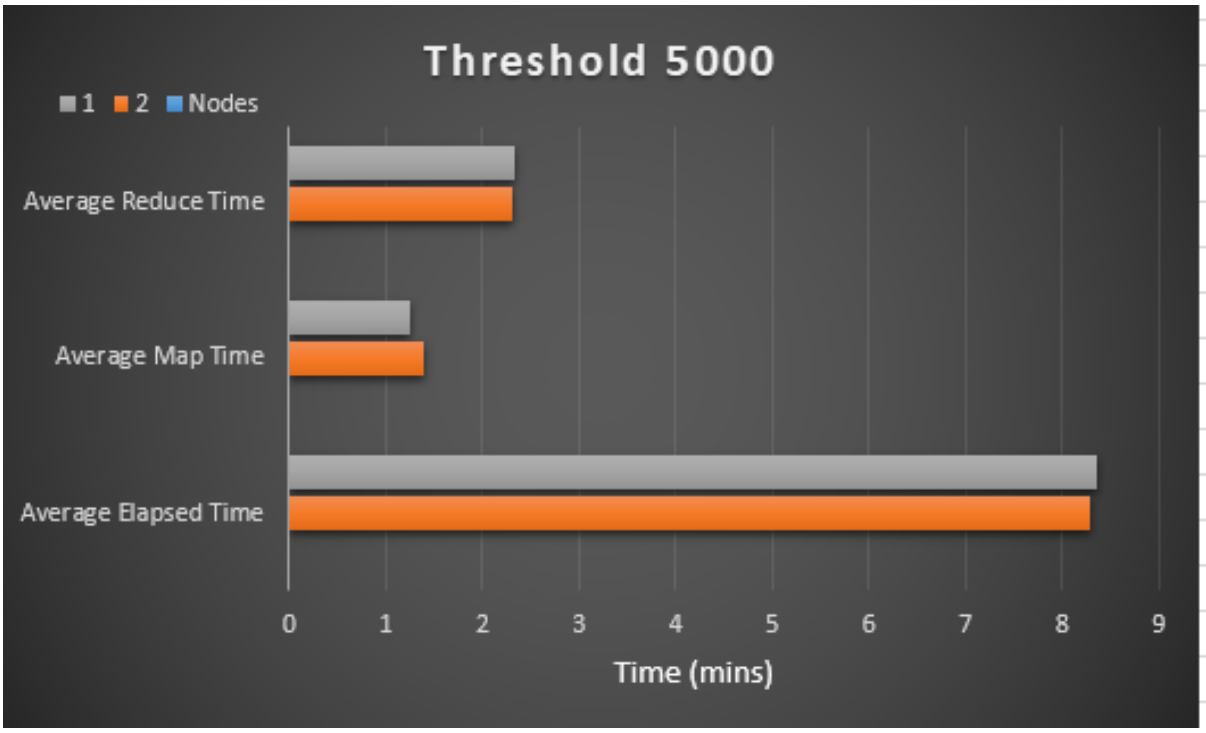
- 2 Nodes:
  threshold 5000: 00:01:40
  threshold 10000: 00:01:39
  threshold 50000: 00:01:40

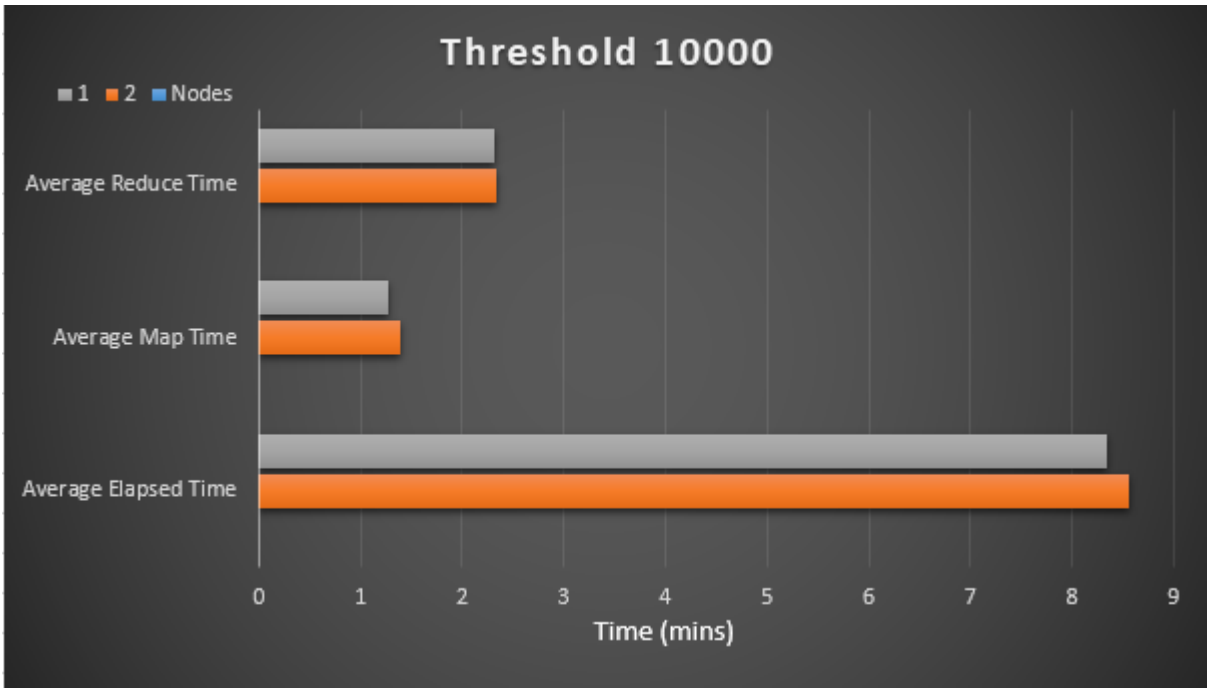
- 1 Node:
  threshold 5000: 00:01:26
  threshold 10000: 00:01:27
  threshold 50000: 00:01:27

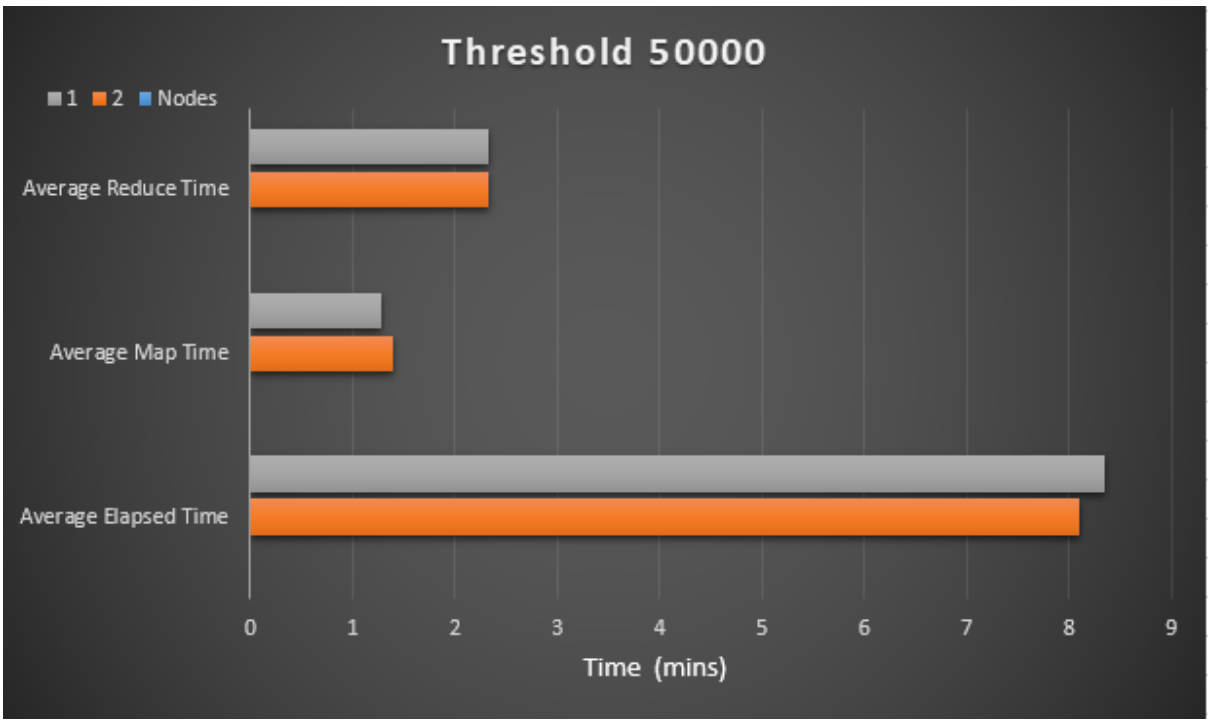
Average Reduce Time

- 2 Nodes:
  threshold 5000: 00:02:32
  threshold 10000: 00:02:33
  threshold 50000: 00:02:33

- 1 Node:
  threshold 5000: 00:02:33
  threshold 10000: 00:02:32
  threshold 50000: 00:02:33

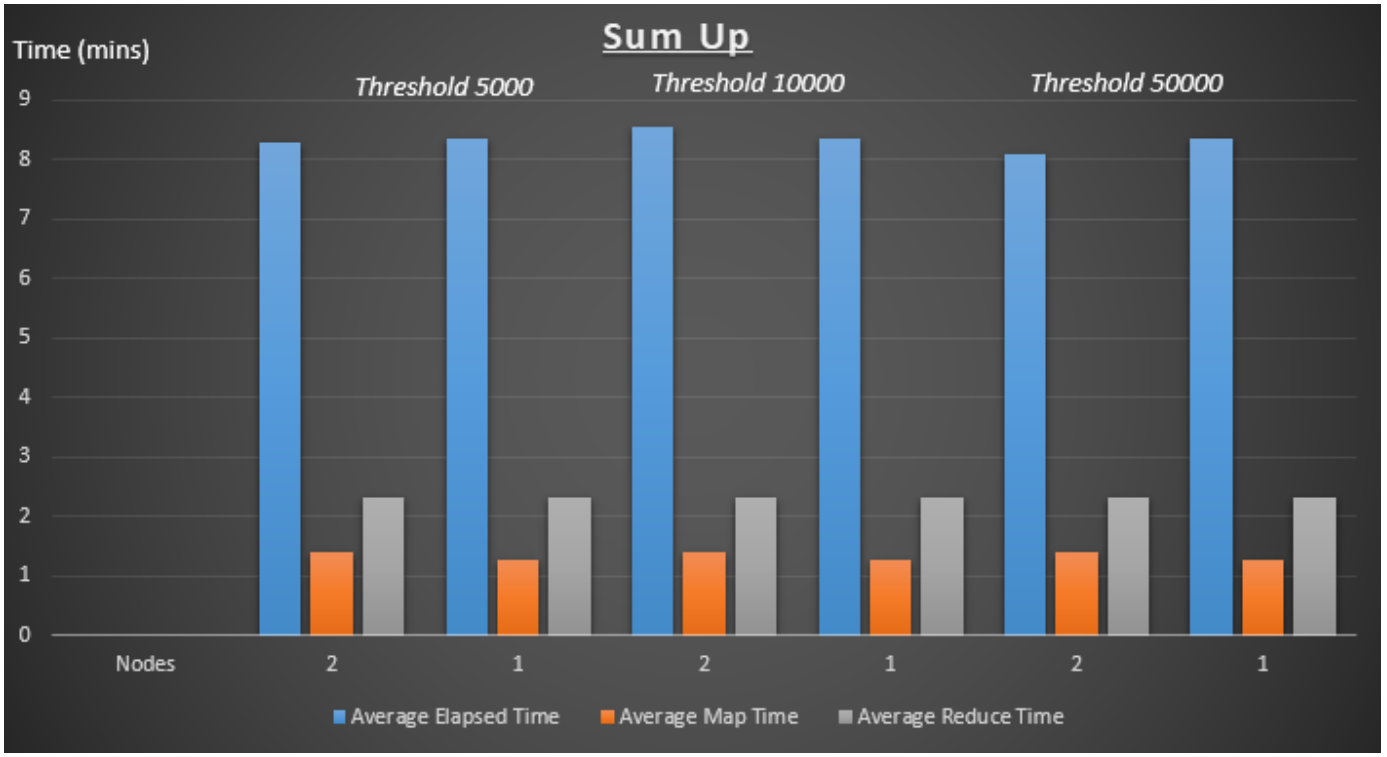

Threshold 5000
1
2
Nodes
Average Reduce Time
Average Map Time
Average Elapsed Time
0
1
2
3
4
5
6
7
8
9
Time (mins)


Threshold 10000
1
2
Nodes
Average Reduce Time
Average Map Time
Average Elapsed Time
0
1
2
3
4
5
6
7
8
9
Time (mins)



Συνολικά

Nodes	Average Elapsed Time	Average Map Time	Average Reduce Time	Threshold
2	00:08:29	00:01:40	00:02:32	5000
1	00:08:36	00:01:26	00:02:33	5000
2	00:08:56	00:01:39	00:02:33	10000
1	00:08:35	00:01:27	00:02:32	10000
2	00:08:10	00:01:40	00:02:33	50000
1	00:08:35	00:01:27	00:02:33	50000



6. Σχολιασμός χρόνων

Γενικά παρατηρούμε πως δεν υπάρχουν μεγάλες διακυμάνσεις στους χρόνους. Ίσως θα περιμέναμε με 2 κόμβους να είναι αρκετά πιο γρήγορο, αλλά τελικά δεν φάνηκε κάτι τέτοιο. Επιπλέον παρατηρούμε πως το Average Map Time βγήκε λίγο μικρότερο κατά μέσο όρο με 1 κόμβο.Αυτό ίσως οφείλεται στο γεγονός ότι με 2 κόμβους απαιτείται κάποιος επιπλέον χρόνος για να μεταφερθούν τα δεδομένα μέσω δικτύου. Επίσης το Average Reduce Time είναι σταθερό για οποιεσδήποτε παραμέτρους. Τέλος οι μετρήσεις σε 1 Node παρουσιάζουν μικρότερες αποκλίσεις μεταξύ τους, κάτι που οφείλεται στο ότι δεν χρησιμοποιείται το δίκτυο.