ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μεταπτυχιακή Διπλωματική Εργασία για το

ΠΜΣ Ψηφιακών Συστημάτων και Υπηρεσιών

Κατεύθυνση: Δικτυοκεντρικών Πληροφοριακών Συστημάτων

Ανάπτυξη Εφαρμογής σε Cloud Computing:

Υπολογισμός Skyline Σημείων Βασισμένη στη Μέθοδο Map-Reduce με Χρήση της Γλώσσας R

Λελιόπουλος Παναγιώτης, ΑΜ: ΜΕ11062
Επιβλέπων: Ανα/της Καθ/της Μαρίνος Θεμιστοκλέους

ΠΕΙΡΑΙΑΣ 2015

Περιεχόμενα

Introduction

In this Master Thesis we study the performance of a computer cluster, based on the developing on a skyline algorithm. Furthermore the particular algorithm is implemented so as to run parallel in distributed systems, with the method of MapReduce.

Specifically, the experimental part of the work consists of a computer cluster from one to four nodes. Also we compared the timing to three different dataset of 2000 points. Hence, we are counting separately the time performance of each dataset from one up to four nodes respectively.

The development of the application is based on the method of MapReduce, and implemented on the R language. The basic architecture is essentially three basic functions such as the random number generator, the classification of items in ascending order by column, and finally the calculation of the skyline points.

Finally our application has been tested locally on a computer by using Virtual Machines, which are based on the Cloudera platform. Furthermore our cluster is able to run on Cloud Computing platform.

1 Εισαγωγή

Στην παρούσα εργασία μελετάμε την απόδοση μιας υπολογιστικής συστάδας (clustering), βασισμένη πάνω στην ανάπτυξη ενός skyline αλγορίθμου. Επιπλέων ο συγκεκριμένος αλγόριθμος είναι υλοποιημένος έτσι ώστε να τρέχει παράλληλα σε κατανεμημένα συστήματα, με την μέθοδο του MapReduce.

Συγκεκριμένα το πειραματικό μέρος της εργασίας αποτελείτε από μια υπολογιστική συστάδα (clustering) από έναν μέχρι τέσσερεις κόμβους, όπου πραγματοποιείτε χρονική σύγκριση με τρία διαφορετικά dataset των 2000 σημείων. Επιπλέων μετράτε ξεχωριστά η χρονική απόδοση του κάθε dataset από έναν μέχρι τέσσερεις κόμβους αντίστοιχα.

Η ανάπτυξη της εφαρμογής βασίζετε πάνω στην μέθοδο του MapReduce, και υλοποιείτε στην γλώσσα R. Η βασική αρχιτεκτονική της είναι ουσιαστικά τρεις βασικές λειτουργίες όπως, η γεννήτρια τυχαίων αριθμών, η ταξινόμηση των σημείων κατά αύξουσα σειρά κατά στήλη και τέλος ο υπολογισμός των skyline σημείων.

Τέλος η εφαρμογή μας, έχει δοκιμαστεί τοπικά σε υπολογιστή με την χρήση Εικονικών Μηχανών (Virtual Machines), όπου αυτά βασίζονται στην πλατφόρμα Cloudera. Επιπλέων το cluster μας είναι σε θέση τρέξει και σε υποδομές Υπολογιστικού Νέφους (Cloud Computing).

1.1 Ποιο το Ερευνητικό Πρόβλημα

Η ανάγκη για την εύρεση των σημείων Skyline είναι πολύ σημαντική για ποικίλα προβλήματα. Όπως όμως φαίνεται και παραπάνω όταν οι διαστάσεις των δεδομένων αυξηθούν τότε οι ανάγκες επεξεργασίας μεγαλώνουν και όσο περισσότερα είναι τα σημεία από τα οποία αποτελείται μια συλλογή τόσο πιο δύσκολη γίνεται η εύρεση των Skyline σημείων. Αν ο αλγόριθμος που θα χρησιμοποιηθεί δεν είναι αποδοτικός στη λειτουργία του, τότε ο χρόνος που χρειάζεται για την επεξεργασία μιας πολύ μεγάλης συλλογής, είναι εύκολο να γίνει απαγορευτικός. Για παράδειγμα ένας απλοϊκός αλγόριθμος θα ήταν:

- Να ελέγχουμε κάθε σημείο σε σχέση με κάθε άλλο ως προς την κυριαρχία τους.
- Να επιστρέφουμε όλα τα σημεία που δεν κυριαρχούνται – δηλαδή εκείνα που είναι μη συγκρίσιμα μεταξύ τους.

Κάτι τέτοιο όμως δεν θα μπορούσε να χρησιμοποιηθεί σε ρεαλιστικά προβλήματα με πολλά δεδομένα καθώς το κόστος αυτού του αλγορίθμου θα ήταν τετραγωνικό. Αυτό έχει ως αποτέλεσμα να γίνει διερευνητική αναζήτηση για την εύρεση αποδοτικών αλγορίθμων που προσανατολίζονται ανάλογα από τη περίπτωση σε διαφορετικού τύπου συλλογές και σε διαφορετικές ανάγκες ή στην υλοποίηση ενός νέου αλγορίθμου.

1.2 Σκοπός

Είναι γνωστό ότι σήμερα οι διάφοροι αλγόριθμοί δουλεύουν σε μικρή κλίμακα, τοπικού χαρακτήρα και στηρίζονται στις όποιες δυναμικές σχέσεις που δημιουργούνται τόσο σε τοπικό γεωγραφικό επίπεδο όσο και στην εργασιακή οντότητα.

Λαμβάνοντας υπ' όψιν τα παραπάνω αντιλαμβανόμαστε ότι η κατασκευή και το «τρέξιμο ενός skyline αλγόριθμου» λογικά να βοηθούσε στην κάλυψη των παραπάνω αναγκών και με στόχο να βελτιωθούν τα ήδη εφαρμοζόμενα και να σχεδιασθούν εκ νέου κοινά αποδεκτά. Για αυτό η παρούσα εργασία προσπαθεί τη(ν):

- Βελτίωση της απόδοσης του skyline στην εκτέλεση επερωτήσεων σε πολυδιάστατα δεδομένα με την χρήση του skyline αλγόριθμου.
- Αξιολόγηση της χρήσης ευρετηρίου για εκτέλεση επερωτήσεων σε πολυδιάστατα δεδομένα με χρήση του skyline αλγόριθμου.

Για να επιτευχθούν οι προαναφερθέντες στόχοι πρέπει να γίνουν τα εξής:

- Προσαρμογή του κώδικα για εκτέλεση επερώτησης εύρους με τη χρήση ευρετηρίου σαν MapReduce εργασία με το skyline αλγόριθμο.
- Τέλος να την αξιολογήσει και να αναφέρει μελλοντικές επεκτάσεις.

Επίσης ο σκοπός της παρούσης εργασίας είναι η εκτελεστική αποδοτικότητα των skyline επερωτήσεων σε πολυδιάστατα δεδομένα πάνω στο Cloud Computing. Όπως επίσης σκοπός της είναι αφενός η ανάπτυξη ενός αλγορίθμου με την κατάλληλη δομή και αφετέρου ο ίδιος αλγόριθμος να είναι σε θέση να τρέξει πάνω στο Υπολογιστικό Νέφος.

1.3 Αντικειμενικοί Στόχοι

Μία κατηγορία ερωτημάτων που εντοπίζουμε συχνά ακόμα και σε καθημερινή βάση είναι τα ερωτήματα κορυφογραμμής στα οποία αναζητούμε τα σημεία της κορυφογραμμής αλλιώς γνωστά και ως Skyline points. Για να εξηγήσουμε τι αντιπροσωπεύουν, θα δώσουμε ένα παράδειγμα. Έστω πως κάποιος αναζητά να διαλέξει ένα από τα δυνατά ξενοδοχεία σε μια παραθαλάσσια περιοχή. Επιπλέον έστω ότι τον ενδιαφέρει η τιμή ανά ημέρα διαμονής στο ξενοδοχείο αυτό, να είναι όσο το δυνατό πιο χαμηλή αλλά ταυτόχρονα και η απόστασή του από την παραλία να μην είναι μεγάλη. Αυτά τα δύο χαρακτηριστικά όμως συχνά είναι αντικρουόμενα. Σε αυτό το σημείο λοιπόν αποκτά αξία η εύρεση του Skyline όλων των δυνατών ξενοδοχείων. Στόχος μας είναι η επίλυση του άνω προβλήματος και σε άλλες ερευνητικές περιοχές.

1.4 Δομή της Εργασίας

Στο 2ο Κεφάλαιο ξεκινάμε την μελέτη μας με βιβλιογραφική ανασκόπηση, όπου εξετάζουμε την θεωρία των Big Data (μεγάλος όγκος δεδομένων). Όπως και τις τρεις βασικές αρχές τους: όγκος, ταχύτητα και ποικιλία.

Στην συνέχεια μελετάμε την θεωρία του Cloud Computing (Υπολογιστικό Νέφος) καθώς και συναφείς τεχνολογίες, όπως το Grid Computing (Τεχνολογία Πλέγματος), το Utility Computing (Υπολογιστική Χρησιμότητα), όπως και το Virtualization (Εικονικό Περιβάλλον). Να επισημάνουμε ότι όσον αφορά στην τεχνολογία του Virtualization είναι ουσιαστικά η συγκεκριμένη τεχνολογία που χρησιμοποιούμε στην παρούσα εργασία για το τρέξιμο των κόμβων (nodes).

Επιπλέων κάνουμε αναφορά στην ιστορική αναδρομή του Cloud Computing και εξετάζουμε τα κύρια χαρακτηριστικά του όπως η On-demand self-service (Αυτοεξυπηρέτηση κατά απαίτηση), η Broad network access (Ευρεία πρόσβαση στο

δίκτυο), η Resource pooling (Συνάθροιση πόρων), η Rapid elasticity (Προσαρμοστική ελαστικότητα) και η Measured Service (Μετρήσιμη υπηρεσια).

Επίσης μελετάμε τα τρία μοντέλα παροχής υπηρεσιών του Cloud Computing, τα οποία εξυπηρετούν διαφορετικές ανάγκες και καλύπτουν ένα φάσμα διαφορετικών υπηρεσιών. Αυτά τα μοντέλα είναι τα Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS) και Cloud Infrastructure as a Service (IaaS). Όπου αναφέρονται σε καθαρές υπηρεσίες λογισμικού, υπηρεσίες πλατφόρμας και υπηρεσίες υπολογιστικών πόρων, αντίστοιχα.

Σημαντικό είναι να αναφέρουμε τα μοντέλα ανάπτυξης όπου αυτά με την σειρά τους είναι το Ιδιωτικό Cloud, το Κοινοτικό Cloud, το Δημόσιο Cloud και το Υβριδικό Cloud.

Τέλος στην τελευταία ενότητα αναφερόμαστε στα πλεονεκτήματα και στα μειονεκτήματα του Cloud Computing. Στα πλεονεκτήματα κατατάσσονται η οικονομία πόρων, η αύξηση παραγωγικότητας εργαζομένων, η διευκόλυνση της συνεργασίας, η ευελιξία, η ανάπτυξη των επιχειρήσεων, ο εντοπισμός πληροφοριών από το δίκτυο δεδομένων, η δυνατότητα ανάκτησης δεδομένων σε περιπτώσεις απώλειας και η ανάπτυξη και φιλοξενία εφαρμογών.

Στα μειονεκτήματα κατατάσσονται, το υψηλό κόστος, η αξιοπιστία της υπηρεσίας, η ασφάλεια, οι πολύπλοκοι κανονισμοί, καθώς επίσης και άλλες απειλές ασφάλειας όπως τα επιβλαβή δίκτυα.

Στο 3ο Κεφάλαιο εξετάζουμε την γλώσσα R όπου και αναπτύξαμε την εφαρμογή μας. Κάνουμε μια γενική επισκόπηση στην R καθώς και στις βιβλιοθήκες που χρησιμοποιούμε για τις MapReduce διαδικασίες.

Επιπλέων κάνουμε μια εκτενή αναφορά στη μέθοδο Map Reduce, όπου πρόκειται για ένα προγραμματιστικό μοντέλο το οποίο προορίζετε για την παράλληλη επεξεργασία μεγάλου όγκου δεδομένων.

Επίσης αναφερόμαστε στο Hadoop το οποίο είναι ένα framework ανοιχτού κώδικα στο οποίο βασίζετε η παρούσα εργασία. Καθώς και στο σύστημα αρχείων που χρησιμοποιεί, το HDFS. Στην ουσία το HDFS είναι ένα σύστημα αρχείων το οποίο

χρησιμοποιεί το Hadoop για να διαβάζει τα αρχεία προς επεξεργασία καθώς και για να γράφει, να διαβάζει και να αποθηκεύει τα αποτελέσματα.

Επιπλέων κάνουμε μια αναφορά στην μέθοδο της εύρεσης των skyline σημείων. Σε αυτή την μέθοδο βασίζετε ως επί το πλείστο η φιλοσοφία του αλγορίθμου που αναπτύξαμε. Η μέθοδος των skyline είναι στην ουσία πρόκειται για ένα τρόπο υπολογισμού σημείων που βρίσκονται ποιο κοντά στον άξονα των X και στον άξονα των Y. Η συγκεκριμένη μέθοδος υπολογισμού των σημείων καθώς και ο συνδυασμός με την μέθοδο του MapReduce είναι στην ουσία ο αλγόριθμος που εξετάζουμε στην παρούσα εργασία.

Τέλος στο παρών Κεφάλαιο αναφερόμαστε στους στόχους και στους σκοπούς της εργασίας. Όπως και στην σχεδίαση της εργασίας σε επίπεδο αρχιτεκτονικής και κώδικα. Η συγκεκριμένη υλοποίηση του αλγορίθμου δημιουργήθηκε σε τρεις βασικές λειτουργιές όπως η δημιουργία τυχαίων σημείων, η ταξινόμηση των σημείων ανά στήλη και τέλος ο υπολογισμός των skyline σημείων. Συγκεκριμένα οι δύο τελευταίες λειτουργίες έχουν ανατηχθεί με αυτό τον τρόπο έτσι ώστε να τρέχουν πάνω στο HDFS και να γίνετε η επεξεργασία των δεδομένων με την μέθοδο του MapReduce.

Στο 4ο Κεφάλαιο μελετάμε και αναλύουμε τους βασικούς Skyline αλγορίθμους, οπού και αυτοί βασίζονται στην μέθοδο του MapReduce. Γίνετε μια εκτενή αναφορά στους αλγόριθμους όπως ο αλγόριθμος Skyline-Baseline, ο αλγόριθμος Skyline-Fifo, ο αλγόριθμος Skyline-Volume, ο αλγόριθμος Skyline-Sampling-Adapt-Fifo, ο αλγόριθμος Skyline-Sampling-Adapt-Volume, ο αλγόριθμος Skyline-Sampling-Baseline και ο αλγόριθμος Skyline-Sampling-Fifo. Τέλος γίνετε ιδιαίτερη αναφορά στον αλγόριθμο Block-Nested-Loop (BNL) οπού αποτέλεσε και την βάση για την ανάπτυξη του δικού μας αλγορίθμου.

Στο 5ο Κεφάλαιο αναλύονται οι προδιαγραφές του πειραματικού μέρους, όπως το συνολικό Hardware που χρησιμοποιήσαμε για το στήσιμο του υπολογιστικού cluster αλλά και ξεχωριστά του κάθε κόμβου. Γίνετε αναφορά στο λογισμικό, όπως στην γλώσσα προγραμματισμού που αναπτύχτηκε ο αλγόριθμος μας και τέλος τα Dataset που χρησιμοποιήθηκαν.

Επιπλέων εστιάζουμε στο στήσιμο του αλγορίθμου και στις επιμέρους λειτουργίες που τον συνθέτουν, απεικονίζοντας τον σε ένα λογικό διάγραμμα. Κάνουμε ιδιαίτερη

αναφορά στα τρία βασικά του μέρη του, όπως στη δημιουργία τυχαίων αριθμών μέχρι 2000 σημείων, στην ταξινόμηση του dataset με την διαδικασία του MapReduce και την εύρεση των skyline σημείων επίσης με την διαδικασία του MapReduce.

Επίσης στο ίδιο Κεφάλαιο ερευνούμε σε όλο το φάσμα του πειραματικού μας μέρους, βήμα προς βήμα, την συμπεριφορά του Map Reduce σε όλους τους κόμβους. Μελετάμε τα συμπεράσματα μας στο πειραματικό μέρος με ένα μόνο κόμβο και καταλήγουμε διαδοχικά στους τέσσερεις κόμβους. Οπού το πειραματικό μέρος με τους τέσσερεις κόμβους ουσιαστικά αποτελεί και την πλήρη ανάπτυξη των πειραμάτων μας.

Μέσα από αυτό το κεφάλαιο βγάζουμε χρήσιμα και πολύτιμα συμπεράσματα, όχι μονό για το τρέξιμο του συγκεκριμένου αλγορίθμου αλλά γενικότερα όλων των Map Reduce υπολογισμών.

Ειδικά σε αυτό το Κεφάλαιο περισσότερο μελετάμε την υποδομή του υλικού μας (Hardware) και πως διανέμετε από κόμβο σε κόμβο, χάρη στο MapReduce. Ενώ κάνουμε συγκρίσεις μεταξύ των κόμβων σε κάθε ξεχωριστό cluster που μελετάμε.

Τέλος βλέπουμε τις αδυναμίες της MapReduce μεθόδου αλλά και των κόμβων μας στην παρούσα εργασία.

Στο 6ο Κεφάλαιο εξετάζουμε τα αποτελέσματα του πειραματικού μας μέρους στα τρία διαφορετικά dataset που έχουμε ορίσει. Σε όλα τα datasets ο μέγιστος αριθμός των σημείων είναι 2000, ενώ η διαφορά τους είναι στον αριθμό των στηλών που αποτελούνται. Συγκεκριμένα το πρώτο dataset αποτελείτε από 2 στήλες, το δεύτερο από 6 στήλες και το τρίτο από 10 στήλες.

Στο ίδιο Κεφάλαιο λαμβάνουμε τις χρονικές μετρήσεις από τον κάθε κόμβο ξεχωριστά και για τα τρία datasets, έπειτα απεικονίζονται τα χρονικά αποτελέσματα σε γραφικές παραστάσεις κάνοντας σύγκριση του κάθε dataset σε κάθε κόμβο.

Επιπλέων κάνουμε αναφορά σε κάποια συγκεκριμένα γραφήματα σχετικά με την απόδοση του cluster μας, μετά το τρέξιμο και την επεξεργασία του αλγορίθμου μας. Συγκεκριμένα αυτά τα γραφήματα αναφέρονται στην συνολική κατανάλωση της CPU του cluster μας στο πεδίο του χρόνου. Την συνολική μεταφορά δεδομένων στο δίκτυο του cluster στο πεδίο του χρόνου. Την συνολική μεταφορά δεδομένων στο Hadoop

file system στο πεδίο του χρόνου. Ο συνολικός αριθμός jobs στο cluster στο πεδίο του χρόνου. Ο συνολικός αριθμός των tasks του cluster στο πεδίο του χρόνου και τελος ο συνολικός αριθμός των maps του cluster στο πεδίο του χρόνου.

Τέλος στο 7ο Κεφάλαιο, βγάζουμε τα τελικά συμπεράσματα για τα αποτελέσματα της εργασίας και προτείνουμε μελλοντικές επεκτάσεις για περαιτέρω ανάπτυξη.

2 Ορισμοί Έννοιες

2.1 Θεωρία των Big Data

Από το 2001, ο αναλυτής Doug Laney, αρθρογραφούσε σχετικά με την παρούσα επικρατούσα τάση των Big Data (Μεγάλων Δεδομένων) μιλώντας για τα 3 V των Big Data: Volume (Όγκος), Velocity (Ταχύτητα) και Variety (Ποικιλία).

Όγκος - Volume. Είναι πολλοί οι παράγοντες που συμβάλλουν στην αύξηση του όγκου των δεδομένων. Δεδομένα συναλλαγών αποθηκευμένα για χρόνια. Αδόμητα δεδομένα συνεχούς ροής από τα Social Media. Αυξημένος αριθμός δεδομένων που συλλέγονται από αισθητήρες (sensors) και επικοινωνία μεταξύ μηχανών (machine-to-machine). Στο παρελθόν ο υπερβολικός όγκος δεδομένων δημιουργούσε προβλήματα στην αποθήκευση. Αλλά με τη μείωση του κόστους αποθήκευσης, προκύπτου νέα προβλήματα, συμπεριλαμβανομένου του πώς να προσδιοριστούν με συνάφεια και σχετικότητα οι μεγάλοι όγκοι δεδομένων και πώς μπορούμε μέσω τεχνολογιών analytics να αντλήσουμε πραγματική αξία από αυτά.

Ταχύτητα - Velocity. Τα δεδομένα «ρέουν» με πρωτοφανή ταχύτητα και πρέπει να τα αντιμετωπίζουμε έγκαιρα. Οι ετικέτες RFID, οι αισθητήρες και τα έξυπνα συστήματα, οδηγούν στην ανάγκη να αντιμετωπίσουμε χειμάρρους δεδομένων σε σχεδόν πραγματικό χρόνο. Η αρκετά γρήγορη αντίδραση μας ώστε να αντιμετωπίσουμε την ταχύτητα των δεδομένων αποτελεί ιδιαίτερη πρόκληση για τους περισσότερους οργανισμούς.

Ποικιλία - Variety. Σήμερα τα δεδομένα έρχονται σε οποιαδήποτε πιθανή μορφή. Δομημένα δεδομένα, αριθμητικά δεδομένα αποθηκευμένα σε παραδοσιακές βάσεις. Πληροφορίες που δημιουργούνται από εμπορικές εφαρμογές (Line-of-business applications). Αδόμητα έγγραφα κειμένου, email, video, ήχου, δεδομένα χρηματιστηριακών συναλλαγών και εμπορικών συναλλαγών. Η διαχείριση, η συγχώνευση και η διακυβέρνηση διαφορετικών ειδών δεδομένων είναι θέματα που απασχολούν σοβαρά πολλούς οργανισμούς και επιχειρήσεις.

Πρέπει να λάβουμε υπόψη δύο επιπλέον διαστάσεις όταν σκεφτόμαστε τα Big Data:

Μεταβλητότητα - Variability. Εκτός από τις αυξανόμενες ταχύτητες και την ποικιλία των δεδομένων, οι ροές των δεδομένων μπορεί να είναι εξαιρετικά ασυνεπείς με

περιοδικές κορυφώσεις . Υπάρχει για παράδειγμα κάποια τάση στα social media; Τα καθημερινά, εποχιακά και τα δεδομένα που πυροδοτούνται από συμβάντα μπορεί να είναι δύσκολο να διαχειριστούν. Ακόμη περισσότερο όταν εμπλέκονται και αδόμητα δεδομένα.

Πολυπλοκότητα - Complexity. Τα δεδομένα στις μέρες μας προέρχονται από πολλές πηγές. Και η σύνδεση, ο καθαρισμός και η μετατροπή των δεδομένων από όλα τα συστήματα εξακολουθεί να είναι ένας συνεχής αγώνας. Ωστόσο, είναι απαραίτητη η σύνδεση και η συσχέτιση σχέσεων, ιεραρχιών και πολλαπλών συνδέσεων δεδομένων, αλλιώς τα δεδομένα γρήγορα θα βγουν εκτός ελέγχου.

2.2 Cloud Computing

Η ονομασία Cloud Computing («Υπολογιστικό Σύννεφο») αποτελεί μια εφαρμογή του διαδικτύου και προέρχεται από την απεικόνιση του διαδικτύου με διαγράμματα - παρουσιάσεις.

Διαφορετικά είδη πληροφοριών και υπηρεσιών προσφέρονται και αλληλεπιδρούν μεταξύ τους, έχοντας τη βάση τους σε μια ενιαία πλατφόρμα. «Το πληροφοριακό μοντέλο Cloud Computing επιτρέπει την αδιάλειπτη, ευέλικτη λειτουργία, κοινόχρηστης δικτυακής πρόσβασης με ανάλογη κατανομή των υπολογιστικών πόρων όπως διακομιστές, δίκτυα, διαχείριση αποθηκευτικών χώρων, εφαρμογές και υπηρεσίες. Οι πόροι μπορούν να πολύ εύκολα να παρακολουθηθούν και να αποδοθούν με πολύ μικρή παρέμβαση της διαχείρισης, ή αλληλεπίδρασης με τον πάροχο των υπηρεσιών. Το κυρίαρχο πλεονέκτημα του cloud computing είναι η άμεση διαθεσιμότητα πόρων, ενώ αποτελείται από πέντε ουσιώδη χαρακτηριστικά, τρία μοντέλα παροχής υπηρεσιών και τέσσερα μοντέλα ανάπτυξης.»

Παραπάνω αναφέρεται ο ορισμός του Cloud Computing όπως αυτός ορίζεται από το Εθνικό Ινστιτούτου Τυποποιήσεων και Τεχνολογίας των Ηνωμένων Πολιτειών N.I.S.T. (National Institute of Standards and Technology) το οποίο είναι ένα παγκοσμίου φήμης ίδρυμα στον Τομέα των τεχνολογιών της πληροφορικής. Στη συνέχεια του κεφαλαίου, θα πραγματοποιηθεί ιστορική αναδρομή της τεχνολογίας και θα παρουσιάσουμε την αρχιτεκτονική του Cloud Computing που αποτελείται από πέντε χαρακτηριστικά, τρία μοντέλα υπηρεσίας και τέσσερα μοντέλα ανάπτυξης, όπως είδαμε παραπάνω.

Στις μέρες μας το Cloud Computing βρίσκεται παντού, από τα email, την ανταλλαγή αρχείων μέσω διαδικτύου, σε εφαρμογές διαφορετικού είδους συσκευές όπως, τηλέφωνα και tablet, μέχρι πολυσύνθετες εφαρμογές οι οποίες φιλοξενούνται σε απομακρυσμένους διακομιστές. Είναι ένα είδος τεχνολογίας το οποίο έχει συμβάλει σημαντικά στην εξέλιξη της τεχνολογίας του διαδικτύου και στον τρόπο λειτουργίας και οργάνωσης των επιχειρήσεων.

2.2.1 Συναφείς τεχνολογίες

Το Cloud Computing συχνά συγκρίνεται με τις με τις ακόλουθες τεχνολογίες, με τις οποίες παρουσιάζει κοινά στοιχεία:

- **Grid Computing (Τεχνολογία πλέγματος)**

Το Grid Computing είναι ένα διαμοιραζόμενο σύστημα πληροφορικής, το οποίο «συντονίζει» τους δικτυακούς πόρους. Στόχος του είναι η περαίωση μεγάλων υπολογιστικών εργασιών, ενώ η συγκεκριμένη τεχνολογία βρήκε αρχικά εφαρμογή στις επιστημονικές εφαρμογές που απαιτούσαν συνήθως μεγάλη υπολογιστική ισχύ.

Το Cloud Computing είναι παρόμοιο με το υπολογιστικό πλέγμα σε ότι αφορά στους κατανεμημένους πόρους για την επίτευξη μιας εφαρμογής. Ωστόσο, το Cloud Computing με τον συνδυασμό περεταίρω τεχνολογιών, όπως η εικονοποίηση (virtualazation), κατανέμει τους πόρους σε πολλαπλά επίπεδα (hardware & software), παρέχοντας τους μεγαλύτερη δυναμική.

- **Utility Computing (Υπολογιστική χρησιμότητα)**

Το Utility computing είναι ένα σύστημα πληροφορικής με αρκετά μεγάλη ιστορία, το οποίο επίσης παρέχει υπολογιστικούς πόρους, με τη διαφορά ότι πραγματοποιείται τιμολόγηση με βάση την χρήση και όχι την κατ' αποκοπή χρέωση των πελατών. Σε διάλεξη του ο John McCarthy, μιλώντας στο Centennial MIT το 1961, ανέφερε σχετικά με το utility computing ότι: "Εάν οι υπολογιστές του είδους που έχω υποστηρίξει (το Utility Computing) γίνουν οι υπολογιστές του μέλλοντος, τότε τα πληροφοριακά συστήματα μπορούν κάποια μέρα να λειτουργούν σαν μέσα κοινής ωφέλειας, όπως ακριβώς το τηλέφωνο. Το βοηθητικό πρόγραμμα υπολογιστή θα μπορούσε να αποτελέσει τη βάση μιας νέας και ισχυρής βιομηχανίας.

Το Cloud Computing μπορεί να κατανοηθεί ως η ιδανική υλοποίηση του utility computing. Με την κατανομή των πόρων της ζήτησης και της χρησιμότητας με βάση τους παρόχους υπηρεσιών τιμολόγησης, μεγιστοποιεί την αξιοποίηση των πόρων και την ελαχιστοποίηση του κόστους λειτουργίας τους.

- **Virtualization (Εικονικό περιβάλλον)**

Το Virtualization είναι μια τεχνολογία που ξεπερνά τους περιορισμούς του φυσικού υλικού και παρέχει εικονικούς πόρους για υψηλού επιπέδου επιστημονικές εφαρμογές. Το Virtualization προσφέρει έναν οικονομικά αποδοτικό και ευέλικτο τρόπο χρήσης και διαχείρισης των υπολογιστικών πόρων. Η συγκεκριμένη τεχνική χρησιμοποιείται τόσο στο Grid Computing, όσο και στο Cloud Computing, για την καλύτερη κατανομή των υπολογιστικών παροχών ανάλογα με τη ζήτηση. Αυτό επιτρέπει στους υφιστάμενους υπολογιστικούς πόρους να τροφοδοτηθούν δυναμικά κατά το χρόνο εκτέλεσης από τους χρήστες, με βάση τις απαιτήσεις των εφαρμογών.

2.2.2 Ιστορική αναδρομή

Η προέλευση του όρου Cloud Computing είναι ασαφής και δεν είναι δυνατόν να οριστεί χρονολογικά με ακρίβεια. Ετυμολογικά, η έκφραση “cloud”, σύννεφο ή νέφος, συναντάται συνήθως στην επιστήμη για να περιγράψει ένα πλήθος αντικειμένων τα οποία εξ αποστάσεως σχηματίζουν έναν όγκο που μοιάζει με σύννεφο, είτε για να περιγράψει ένα σύνολο πραγμάτων του οποίου τα στοιχεία εμπίπτουν σε ένα δεδομένο πλαίσιο. Ο όρος με την σύγχρονη έννοιά του, συναντάται για πρώτη φορά το 1996, σε ένα εσωτερικό αρχείο της εταιρίας Compaq. Η ονομασία του αποδίδεται σε δύο στελέχη της εταιρείας Compaq, στο νεαρό προγραμματιστή Sean O'Sullivan και στον διευθυντή του τμήματος μάρκετινγκ της Compaq, George Favaloro. Το πρόγραμμα που προώθησε η εν λόγω εταιρεία για πώληση servers βρήκε τεράστια απήχηση σε επιχειρήσεις παροχής υπηρεσιών διαδικτύου της εποχής, και απέφερε κέρδη ύψους 2 δισεκατομμυρίων δολαρίων ανά έτος. Το 2006, μεγάλες εταιρείες όπως η Google και η Amazon, εντάσσουν την εφαρμογή του «υπολογιστικού νέφους» στα συστήματά τους, έτσι ώστε να προσφέρουν στους χρήστες τους τη δυνατότητα πρόσβασης σε όλο το λογισμικό και τα αρχεία τους μέσω μιας κοινής πλατφόρμας, αντί για τους καθιερωμένους προσωπικούς ηλεκτρονικούς υπολογιστές.

Σήμερα, το Cloud Computing αποτελεί ένα από τις πιο σύγχρονες και πολυσυζητημένες εφαρμογές στον τομέα της πληροφορικής, γεγονός που επιβεβαιώνεται από τις 48 εκατομμύρια εμφανίσεις που εμφανίζονται κατά την αναζήτησή του στο διαδίκτυο.

Εντούτοις, η έννοια του Cloud Computing έχει την αφετηρία της στη δεκαετία του 1950, όταν μεγάλης κλίμακας υπολογιστές διατέθηκαν σε σχολεία και επιχειρήσεις.

Μεγάλα συστήματα ηλεκτρονικών υπολογιστών (mainframes) δημιουργούσαν κυριολεκτικά ένα "server room". Παράλληλα, πολλοί χρήστες είχαν τη δυνατότητα πρόσβασης στο mainframe μέσω "απλών τερματικών", ενώ επιδίωξη της καινοτόμας λειτουργίας τους αποτέλεσε η διευκόλυνση της πρόσβασης στους υπέρ-υπολογιστές, δηλαδή στο κεντρικό δίκτυο.

Εξαιτίας του υψηλού κόστους που απαιτούσε η αγορά, αλλά και η συντήρηση των υπέρ-υπολογιστών, ένας οργανισμός δεν θα ήταν σε θέση να αντέξει οικονομικά ένα mainframe για κάθε χρήστη. Επομένως, ο οικονομικότερος τρόπος θα ήταν το σύστημα να επιτρέπει πολλαπλούς χρήστες να μοιράζονται την πρόσβαση στο ίδιο ιστότοπο αποθήκευσης δεδομένων και επεξεργαστικής ισχύος από κάθε τερματικό. Με την ενεργοποίηση της κοινής χρήσης του mainframe, ο οργανισμός θα μπορεί να έχει μια καλύτερη απόδοση της επένδυσής του.

Δύο δεκαετίες αργότερα, κατά τη διάρκεια της δεκαετίας του 1970, η IBM κυκλοφόρησε ένα λειτουργικό σύστημα το οποίο ονομάζεται VM. Το ακρωνύμιο προκύπτει από την ονομασία "Virtual Machines" (VMs), καθώς επέτρεψε τους διαχειριστές να χειρίζονται πολλαπλά εικονικά συστήματα σε ένα κοινό φυσικό περιβάλλον. Το VM λειτουργικό σύστημα πήρε το 1950 την εφαρμογή της από κοινού πρόσβασης ενός mainframe στο επόμενο επίπεδο, επιτρέποντας πολλαπλές διαφορετικά περιβάλλοντα να βρίσκονται στον ίδιο υπολογιστή.

Οι περισσότερες από τις βασικές λειτουργίες οποιουδήποτε εικονικού λογισμικού virtualization που συναντώνται σήμερα, μπορούν να αναχθούν σε αυτό το πρώιμο VM OS: κάθε VM θα μπορούσε να χρησιμοποιήσει προσαρμοσμένα λειτουργικά συστήματα και λειτουργικά συστήματα επισκεπτών, που είχαν τη δική τους μνήμη, CPU και σκληρούς δίσκους μαζί με CD -ROMs, πληκτρολόγια και δικτύωση. Η εικονικότητα, ή «Virtualization», έγινε ένας οδηγός της τεχνολογίας, και

λειτούργησε σαν καταλύτης για μερικές από τις μεγαλύτερες εξελίξεις στον τομέα των επικοινωνιών και της πληροφορικής.

Στη δεκαετία του 1990, οι εταιρείες τηλεπικοινωνιών που καθ' όλη τη διάρκεια της λειτουργίας τους προσέφεραν μόνο ατομικές ειδικές point -to -point συνδέσεις δεδομένων, εμπλουτίζουν τις υπηρεσίες τους με την προσφορά εικονικών ιδιωτικών συνδέσεων δικτύου. Το νέο είδος σύνδεσης χαρακτηρίζεται από την ίδια ποιότητα υπηρεσιών όπως στις ειδικές υπηρεσίες, αλλά με χαμηλότερο κόστος. Αντί για μια εφαρμογή που επιτρέπει σε περισσότερους χρήστες να έχουν τις δικές τους ανεξάρτητες συνδέσεις σε μια κοινή φυσική υποδομή, οι εταιρείες τηλεπικοινωνιών ήταν σε θέση να παρέχουν στους χρήστες τους την μοιραζόμενη πρόσβαση στην ίδια φυσική υποδομή. Η αλλαγή αυτή επιτρέπει στις τηλεφωνικές εταιρίες να στραφούν στην καλύτερη ισορροπία και κυκλοφορία του δικτύου και να αποκτήσουν περισσότερο έλεγχο πάνω στο εύρος ζώνης χρήσης. Ταυτόχρονα, ξεκίνησε και η ανάπτυξη του virtualization για συστήματα βασιζόμενα σε ακολουθίες υπολογιστών, το οποίο εφαρμόστηκε ως κάτι περιορισμένο σε μεγάλες εταιρείες, καθώς η σύνδεση στο διαδίκτυο γίνεται ολοένα και πιο προσιτή, το επόμενο βήμα ήταν να παραλάβει απευθείας σύνδεση με το «εικονικό» σύστημα.

Το κόστος του φυσικού εξοπλισμού (hardware), αν και δεν βρισκόταν στο ίδιο επίπεδο με αυτό των διακομιστών του 1950, ήταν υπερβολικά υψηλό σε σύγκριση με τα σημερινά δεδομένα. Καθώς όλο και περισσότεροι άνθρωποι εξέφρασαν την επιθυμία για απευθείας σύνδεση και η ζήτηση είχε αγγίξει πρωτοφανή επίπεδα, τα χρήματα που θα έπρεπε να δαπανηθούν για την αγορά mainframes, προς υλοποίηση του συγκεκριμένου εγχειρήματος, θα ήταν εξωφρενικά. Η τεχνολογία που κατέστησε το όλο εγχείρημα δυνατό ήταν αυτή της εικονικότητας (virtualization).

Οι διακομιστές τοποθετούνται εικονικά σε κοινόχρηστα περιβάλλοντα φιλοξενίας Virtual Private Servers και Virtual Dedicated Servers, ενώ υιοθετούν τους ίδιους τύπους λειτουργικού που παρείχε το λειτουργικό που υλοποίησε την εικονικότητα στη δεκαετία του 1950. Παραδειγματικά , αν υποθέσουμε ότι μια εταιρεία απαιτεί 13 φυσικά συστήματα για να τρέξει τις ιστοσελίδες και τις εφαρμογές, με τη χρήση του virtualization είναι δυνατόν να διαχωρίσει αυτά τα 13 διαφορετικά συστήματα ανάμεσα σε δύο φυσικούς κόμβους. Με αυτό τον τρόπο, αυτό το είδος του

περιβάλλοντος εξοικονομεί το κόστος των υποδομών και ελαχιστοποιεί την ποσότητα του εξοπλισμού που απαιτείται στις ανάγκες της εκάστοτε επιχείρησής.

Καθώς το κόστος αγοράς του υλικού εξοπλισμού ενός διακομιστή σταδιακά μειώνεται, ο αριθμός των χρηστών αυξανόταν με γεωμετρική πρόοδο και ο καθένας είχε τη δυνατότητα αγοράς ατομικού server. Αυτό το γεγονός δημιούργησε ένα νέο είδος προβλήματος, το οποίο ήταν η ανεπάρκεια ενός και μόνο διακομιστή να παράσχει τους πόρους που ανταποκρίνονται σε αυτή τη ζήτηση. Η αγορά μέχρι πρότινος ακολουθούσε τη φιλοσοφία του μικρού αριθμού διακομιστών, λόγω του υψηλού τους κόστους τους, και τον διαχωρισμού των λειτουργιών τους, ενώ τώρα προκύπτει η ανάγκη για συνδυασμό των διακομιστών, εφόσον ο αριθμός τους έχει αυξηθεί υπερβολικά και απαιτείται συνένωση των εφαρμογών τους. Λόγω αυτής της συγκυρίας, αναπτύχθηκε η βασική εφαρμογή του «υπολογιστικού νέφους».

Με την εγκατάσταση και ρύθμιση ενός λογισμικού «hypervisor» σε πολλούς φυσικούς κόμβους , το σύστημα θα εκμεταλλεύεται το σύνολο των πόρων, όπως αν οι πόροι αυτοί βρίσκονταν σε ένα και μόνο φυσικό κόμβο. Σε μια προσπάθεια απεικόνισης και χαρακτηρισμού αυτού του περίπλοκου περιβάλλοντος, οι τεχνολόγοι εισάγουν όρους όπως «υπολογιστική χρησιμότητα» και «υπολογιστικό νέφος», οι οποίοι αναφέρονται στο σύνολο των τεχνολογιών και των επιμέρους χρήσεων και εφαρμογών. Σε αυτά τα περιβάλλοντα cloud computing, η προσθήκη νέων διακομιστών καθίσταται μια απλή διαδικασία, καθώς η ρύθμισή της είναι εφικτή, και ακολούθως και η ένταξή της στο σύνολο.

Καθώς οι τεχνολογίες και hypervisors κερδίζουν προοδευτικά έδαφος στην αξιόπιστη ανταλλαγή και την παροχή υπηρεσιών, πολλές επιχειρήσεις τολμούν να αυξήσουν το πληροφοριακό τους περιβάλλον, ώστε να λάβουν τα οφέλη του cloud για χρήστες που δεν τυχαίνει να έχουν αφθονία των φυσικών διακομιστών. Οι συγκεκριμένοι χρήστες μπορούσαν να χρησιμοποιήσουν υποδομές «Cloud Computing», παραγγέλνοντας την υπολογιστική ισχύ που χρειάζονται από το μεγαλύτερο δίκτυο παροχής πόρων μέσω της εφαρμογής cloud, και καθώς οι servers βρίσκονται ήδη σε απευθείας σύνδεση, η διαδικασία της «ενεργοποίησης» μιας νέας διαδικασίας είναι σχεδόν στιγμιαία. Η οικονομική επιβάρυνση που συνεπάγεται μια νέα παραγγελία σύνδεσης ή ακύρωσης είναι μικρής τάξεως για τον ιδιοκτήτη του περιβάλλοντος cloud computing, γεγονός που καθιστά τη διαχείριση του περιβάλλοντος σημαντικά

πιο εύκολη και ευέλικτη. Οι περισσότερες εταιρείες έχουν υιοθετήσει την εφαρμογή του " νέφους ", αναφέροντας παραδειγματικά την SoftLayer.

Η εταιρεία SoftLayer υιοθέτησε την ιδέα ενός περιβάλλοντος Cloud Computing και εισήγαγε μια τροποποίηση στη λειτουργία της: αντί να εγκαταστήσει το λογισμικό σε έναν κορμό μηχανημάτων ώστε να επιτρέπει στους χρήστες να χρησιμοποιούν τμήματά του, οικοδομεί μια πλατφόρμα που έχει τη δυνατότητα να αυτοματοποιεί τις διαδικασίες εισαγωγής ενός διακομιστή στο σύστημα απευθείας, χωρίς να είναι απαραίτητη η χρήση hypervisor στο διακομιστή. Αυτό το είδος πλατφόρμας ονομάζεται "IMS". Ουσιαστικά, ο ρόλος που παίζουν οι hypervisors και εικονοποίηση περιβαλλόντων για μία ομάδα διακομιστών, αντικαθίσταται από την πλατφόρμα "IMS" για ένα ολόκληρο κέντρο δεδομένων.

Τα παραπάνω είχαν ως αποτέλεσμα την αμεσότητα και μεγάλη ταχύτητα ενός διακομιστή, ο οποίος μπορεί να διαθέτει όσους πόρους χρειάζεται ο παραλήπτης του χωρίς την εγκατάσταση υλικού software που δεν είναι απαραίτητο για τη λειτουργία του. Χωρίς την ένταση hypervisor μεταξύ του λειτουργικού συστήματος και του βασικού υλικού εξοπλισμού, οι διακομιστές έχουν καλύτερες επιδόσεις. Έχοντας, τέλος, αυτοματοποιήσει σχεδόν τα πάντα στα κέντρα δεδομένων, δίνεται στον καθέναν η δυνατότητα χρήσης εξισορρόπηστών φορτίου, firewalls και συσκευές αποθήκευσης για το χρονικό διάστημα που τους είναι απαραίτητη, έχοντας επιπλέον το δικαίωμα διακοπής της χρήσης τους όποτε το αποφασίσουν.

Οι εταιρείες IBM και SoftLayer δημιουργούν την τάση προς την ευρύτερη υιοθέτηση των καινοτόμων υπηρεσιών cloud, θέτοντας φιλόδοξους στόχους για το μέλλον. Αν αναλογιστεί κανείς την εξέλιξη που μεσολάβησε από το 1950 μέχρι σήμερα, από τη διευκόλυνση ενός δικτύου ολιγάριθμων συσκευών μέχρι την κάλυψη μιας παγκόσμιας αγοράς, συμπεραίνει κανείς ότι οι υπηρεσίες cloud αποτέλεσαν καταλύτη για την πρόοδο της τεχνολογίας και τη σημερινή πραγματικότητα. Αξίζει να σημειωθεί ότι αυτού του είδους η εφαρμογή επηρεάζει όχι μόνο το σύγχρονο μοντέλο διοίκησης των επιχειρήσεων και δόμησης των πολυεθνικών εταιρειών, αλλά και τους μεμονωμένους χρήστες σε ατομικό επίπεδο.

Η υπηρεσία Cloud Computing διαφέρει από τις υπόλοιπες υπηρεσίες του είδους, σε τρεις βασικούς τομείς. Πρώτον, προσφέρει το πλεονέκτημα της ελαστικότητας. Αυτό σημαίνει ότι κοστολογείται με βάση το λεπτό ή την ώρα, και προσαρμόζεται στη

ζήτηση του εκάστοτε χρήστη, ο οποίος χρεώνεται την υπηρεσία για όσο χρονικό διάστημα την χρειάζεται.

Δεύτερον η διαχείριση εκπίπτει πλήρως στη δικαιοδοσία του παρόχου, καθιστώντας εύκολη τη χρήση του, με μόνο ζητούμενο από τον καταναλωτή, έναν προσωπικό υπολογιστή και πρόσβαση στο διαδίκτυο.

2.3 Κύρια Χαρακτηριστικά

Ακολούθως, αναλύονται τα πέντε κύρια χαρακτηριστικά του Cloud computing, τα οποία το κάνουν να διαφέρει από τις υπόλοιπες υπολογιστικές μεθόδους, με βάση το NIST.

- **On-demand self-service (Αυτοεξυπηρέτηση κατά απαίτηση)**

Ο χρήστης μπορεί να δεσμεύει υπολογιστικούς πόρους, όπως αποθηκευτικό χώρο και χρόνο στους διακομιστές, ανάλογα με τις απαιτήσεις του. Οι πόροι δεσμεύονται ή αποδεσμεύονται αυτόματα ανάλογα με τις απαιτήσεις.

- **Broad network access (Ευρεία πρόσβαση στο δίκτυο)**

Οι δυνατότητες του είναι διαθέσιμες σε όλο το δίκτυο και μπορεί να διατεθεί με μηχανισμούς που προωθούν την χρήση ετερογενών συσκευών όπως κινητά τηλέφωνα, φορητούς υπολογιστές, ταμπλέτες.

- **Resource pooling (Συνάθροιση πόρων)**

Οι υπολογιστικοί πόροι του παρόχου χρησιμοποιούνται για την εξυπηρέτηση πολλαπλών καταναλωτών. Με τη χρήση του μοντέλου ενοικιάσεως πόρων (multi-tenant model), είτε φυσικών είτε εικονικών, μπορούμε δυναμικά να διαχειριζόμαστε τους πόρους ανάλογα με τις απαιτήσεις των πελατών.

Με αυτόν τρόπο, δημιουργείται μια αίσθηση ανεξαρτησίας ότι ο πελάτης δεν έχει κανένα έλεγχο ή γνώση της τοποθεσίας των παρεχόμενων πόρων που του παρέχονται, αλλά μπορεί να προσδιορίσει την τοποθεσία σε πιο γενικό επίπεδό, για παράδειγμα ανά κράτος ή ανά κέντρο δεδομένων. Παραδείγματα παρεχόμενων πόρων είναι οι αποθηκευτικοί χώροι, η μνήμη, το εύρος ζώνης δικτύου και τα εικονικά μηχανήματα.

- **Rapid elasticity (Προσαρμοστική ελαστικότητα)**

Οι πόροι μπορούν να διατεθούν ευέλικτα και τις πιο πολλές φορές αυτόματα, ώστε να αποδεσμεύονται ή να δεσμεύονται άμεσα. Όσον αφορά στον καταναλωτή, οι διαθέσιμες δυνατότητες για δέσμευση και χρήση συχνά φαίνεται να είναι απεριόριστες και μπορούν ν αγοραστούν ή να αποδεσμευτούν ανά πάσα στιγμή.

- **Measured Service (Μετρήσιμη)**

Τα συστήματα Cloud μπορούν να ρυθμίζουν και να βελτιστοποιούν την διαθεσιμότητα των προσφερόμενων πόρων με αυτόματο τρόπο, με τη χρήση μιας δυνατότητας μέτρησης ανάλογα με το είδος της υπηρεσίας. π.χ. αποθήκευση, επεξεργασία, εύρος ζώνης, ενεργοί λογαριασμοί. Η χρήση των πόρων προσφέρει τη δυνατότητα ελέγχου και παρακολούθησης, παρέχοντας διαφάνεια τόσο για τον πάροχο, όσο και για τον καταναλωτή της υπηρεσίας.

2.4 Μοντέλα Παροχής Υπηρεσιών

Στη τεχνολογία του Cloud Computing, χρησιμοποιούνται τρία μοντέλα παροχής υπηρεσιών, τα οποία εξυπηρετούν διαφορετικές ανάγκες και καλύπτουν ένα φάσμα διαφορετικών υπηρεσιών. Τα μοντέλα αυτά επιγραμματικά είναι τα Software as a Service, Cloud Platform as a Service και Infrastructure as a Service και περιγράφονται στη συνέχεια.

- **Μοντέλο Cloud Software as a Service (SaaS)**

Ο καταναλωτής έχει πλέον την δυνατότητα να χρησιμοποίει της εφαρμογές οι οποίες δεν βρίσκονται τοπικά αλλά σε μια απομακρυσμένη τοποθεσία (Cloud Infrastructure). Οι εφαρμογές είναι διαθέσιμες από διάφορες "μικρές" συσκευές (client), όπως ένα πρόγραμμα περιήγησης, ηλεκτρονικό ταχυδρομείο κ.α. Ο καταναλωτής δεν έχει δυνατότητα διαχείρισης ή τον έλεγχο της υποδομής του Cloud, όπως διακομιστές, λειτουργικά συστήματα, δικτυακούς πόρους, συστήματα αποθήκευσης ακόμα και μεμονωμένες εφαρμογές. Η μονή εξαίρεση είναι οι μικρές μικρορυθμίσεις που μπορεί να πραγματοποιήσει πάνω στην εφαρμογή.

- **Cloud Platform as a Service (PaaS)**

Ο πελάτης έχει τη δυνατότητα να αναπτύξει την δικιά του Cloud υποδομή, η οποία δημιουργήθηκε βασιζόμενη σε προγραμματισμό και υποστηρίζεται από τον πάροχο. Ο καταναλωτής έχει τον έλεγχο του λειτουργικού κομματιού, δηλαδή της ανάπτυξης, διαμόρφωσης και φιλοξενίας της εφαρμογής και όχι των υλικών πόρων που χρησιμοποιούνται.

- **Cloud Infrastructure as a Service (IaaS)**

Οι υπολογιστικοί πόροι που παρέχονται στον καταναλωτή, παραδειγματικά αναφέρονται η επεξεργαστική ισχύ, η δυνατότητα αποθήκευσης και άλλοι, του δίνουν τη δυνατότητα να αναπτύξει και να εκτελέσει "αυθαίρετο" λογισμικό το οποίο μπορεί να περιλαμβάνει διαφόρου είδους εφαρμογές. Πρέπει να σημειωθεί ότι οι χρήστες δεν έχουν καθόλου έλεγχο των πόρων.

Η υποδομή είναι το χαμηλότερο επίπεδο, και είναι ένα μέσο για να παρέχεται η επεξεργασία, η αποθήκευση, το δίκτυο και άλλοι βασικοί υπολογιστικοί πόροι σαν δεδομένες υπηρεσίες δια μέσω του δικτύου. Οι πάροχοι του Cloud μπορούν να εφαρμόσουν και να τρέχουν λειτουργικά συστήματα και λογισμικό για τους υπολογιστικούς πόρους που τους παρέχονται (hardware).

2.5 Μοντέλα Ανάπτυξης

Τα μοντέλα ανάπτυξης στο Cloud Computing διακρίνονται σε τέσσερις κατηγορίες, ανάλογα με την φύση των χρηστών και τον τρόπο διαχείρισής τους:

- **Ιδιωτικό Cloud**

Η υποδομή του ιδιωτικού Cloud λειτουργεί αποκλειστικά για έναν οργανισμό. Δίνουν στους χρήστες άμεση πρόσβαση σε υπολογιστικούς πόρους, οι οποίοι φιλοξενούνται στην υποδομή ενός συγκεκριμένου οργανισμού. Οι χρήστες μπορούν να ελέγξουν από μόνοι τους και να τροποποιήσουν το μέγεθος από τους πόρους που παίρνουν από το private Cloud, συχνά μέσω μιας διεπαφούς διαδικτυακής υπηρεσίας, όπως ακριβώς και σε ένα δημόσιο (public) Cloud. Οι πόροι στο ιδιωτικό Cloud μπορούν να διαχειρίζονται από την οργάνωση ή από κάποιο τρίτο μέρος, με προαπαιτούμενες ρυθμίσεις ή χωρίς.

Στην περίπτωση που ένας πάροχος υπηρεσιών εκμεταλλεύεται τους πόρους ενός δημόσιου cloud συστήματος "σύννεφου" με στόχο να δημιουργήσει ένα ιδιωτικό, τότε ως αποτέλεσμα προκύπτει ένα ιδεατό ιδιωτικό σύννεφο. Ιδιωτική ή δημόσια, η επιδίωξη του cloud computing είναι η παροχή εύκολης, κλιμακωτής πρόσβασης σε υπολογιστικούς πόρους και υπηρεσίες πληροφορικής.

- **Κοινοτικό Cloud**

Η υποδομή του Cloud είναι διαμοιρασμένη από πολλούς οργανισμούς και εξυπηρετεί μια συγκεκριμένη κοινότητα, η οποία ενοποιείται με κάποιο ενδιαφέρον ή σκοπό, και περιορίζεται με μια σειρά κοινών ρυθμίσεων, π.χ. αποστολή, απαιτήσεις ασφάλειας και εκτιμήσεις συμμόρφωσης. Το κοινοτικό Cloud μπορεί να διαχειρίζεται από κάποιον από τους αναφερθέντες οργανισμούς, να έχει την εποπτεία του ένας τρίτος οργανισμός ή επιχείρηση. Οι πόροι μπορούν να διαχειρίζονται από την οργάνωση ή από κάποιο τρίτο μέρος, με προαπαιτούμενα στοιχεία ή χωρίς.

- **Δημόσιο Cloud**

Τα public Clouds παρέχουν πρόσβαση στους υπολογιστικούς τους πόρους στο γενικό κοινό μέσω του διαδικτύου. Ο οργανισμός παρέχει, με το ανάλογο κόστος, τις υπηρεσίες στο ευρύ κοινό ή σε εταιρείες. Το δημόσιο Cloud προσφέρει υπηρεσίες σε οποιοδήποτε χρήστη του διαδικτύου, με κυρίαρχο φορέα παροχής υπηρεσίας σήμερα, την Amazon Web services.

- **Υβριδικό Cloud**

Οι πόροι (μηχανήματα, δίκτυο, αποθήκευση) δυο ή περισσότερων μοντέλων Cloud (Ιδιωτικά, κοινοτικά, δημόσια), μπορούν να συνδυαστούν μεταξύ τους και να επιτρέψουν εξισορρόπηση δεδομένων και φορητότητα εφαρμογών (π.χ., cloud bursting for load balancing between clouds). Ένας οργανισμός ή μια εταιρεία μπορεί να εφαρμόσει ένα μοντέλο ή να συνδυάσει πολλά διαφορετικά μοντέλα, ανάλογα με το μοντέλο Cloud που του παρέχει την καλύτερη λύση.

2.6 Πλεονεκτήματα και Μειονεκτήματα του Cloud Computing

Αναλύονται τα οφέλη και μειονεκτήματα που επιφέρει η εφαρμογή cloud computing σε επιχειρήσεις, καθώς όλοι οι μεγάλοι οργανισμοί στρέφονται όλο και περισσότερο σε αυτή, ως μια οικονομικά αποδοτική μέθοδος της λειτουργίας των πόρων υψηλής στάθμης. Ο οικονομικός αντίκτυπος της νέας αυτής εφαρμογής έχει ως αποδέκτες του εταιρείες και οργανισμούς που εκμεταλλεύονται τις υπηρεσίες του.

Το Cloud Computing είναι μια ριζική αλλαγή στην ισχύουσα κατάσταση της πληροφορικής. Εδώ και δεκαετίες, οι περισσότερες εταιρείες επενδύουν σημαντικά κεφάλαια για αγορά λογισμικού και υλικού, ώστε να αναβαθμίσουν την παραγωγικότητα τους. Το Cloud Computing προσφέρει ένα καθ' όλα διαφορετικό μοντέλο, όπου οι εταιρείες "φορτώνουν" όλες τις ψηφιακές πληροφορίες ενός μεγάλου τμήματος IT υπ' ενοικίαση στην πλατφόρμα cloud, και όχι σε κάποιο σύστημα της ιδιοκτησίας τους. Στην εργασία αυτή θα προσπαθήσουμε να κατανοήσουμε εάν αυτό αποτελεί μια προσωρινή ή μόνιμη αλλαγή, καθώς το μόνο βέβαιο είναι ότι συνέβη μια ριζική και μη αναστρέψιμη μετατόπιση στο πώς η υπολογιστική ισχύς παράγεται και καταναλώνεται.

2.6.1 Τα οφέλη του Cloud

Όσον αφορά στις επιχειρήσεις, η εισαγωγή του Cloud Computing στα προγράμματα μιας εταιρίας, προσφέρει ένα ολοκαίνουριο φάσμα νέων δυνατοτήτων και λειτουργιών.

- **Οικονομία πόρων**

Αν και η μέχρι τώρα πεποίθηση ήταν ότι η λειτουργία μιας επιχείρησης πραγματοποιείται μέσα στην επιχείρηση, με το σύστημα cloud παρέχεται περισσότερη ελευθερία σε ότι αφορά τη χωροθέτηση και το υλικό εξοπλισμό τους.

Σύμφωνα με έρευνα της Microsoft, το 89% του προϋπολογισμού μιας εταιρείας δαπανάται για τη συντήρηση και τις υποδομές του ηλεκτρονικού υλικού (software και hardware). Επομένως, μέσω της νέας τεχνολογίας θα μπορούσε να πραγματοποιηθεί η πρόσβαση σε όλα τα έγγραφα που χρειάζονται οι εργαζόμενοι, ανεξαρτήτως της συσκευής που διαθέτουν και της τοποθεσίας.

- **Αύξηση παραγωγικότητας εργαζομένων**

Πολλοί επαγγελματίες περνούν το μεγαλύτερο μέρος της του χρόνου τους σε χώρους εκτός εργασίας ή και στο εξωτερικό, όπου χρειάζονται άμεση και αξιόπιστη πρόσβαση στις εκτιμήσεις του κόστους, φωτογραφίες, σχεδιαγράμματα, και άλλα μεγάλα αρχεία. Η διαδικτυακή διαχείριση και την κοινή χρήση αρχείων μιας cloud-based επιχείρησης μπορεί να πραγματοποιηθεί μέσω web browser ή αποθηκευτικές εφαρμογές που έχουν δημιουργηθεί για υπολογιστές και smartphones. Με αυτόν τον τρόπο, έχοντας μια συσκευή συνδεδεμένη με το internet, εξασφαλίζεται η πρόσβαση σε όλα τα αρχεία τους, ακόμη και όταν βρίσκονται καθ' οδόν. Έτσι, εξοικονομείται πολύτιμος χρόνος και βελτιώνεται η παραγωγικότητα του προσωπικού.

- **Διευκόλυνση της συνεργασίας**

Ένα άλλο όφελος του Cloud Computing προσφέρει τη δυνατότητα συνεργασίας σε ομάδες και κοινότητες με τρόπους που προηγουμένως θα ήταν ανέφικτοι. Παραδειγματικά αναφέρεται η περίπτωση της συμβουλευτικής εταιρείας CSC, η στράφηκε προς την εταιρεία πληροφορικής Jive για την απόκτηση cloud-based λογισμικού. Απασχολώντας 90.000 υπαλλήλους της σε όλη την υδρόγειο αγορά υλικού εξοπλισμού και αδειών χρήσης λογισμικού για κάθε εργαζόμενο θα ήταν απαγορευτικά ακριβή, αν η CSC υποχρεούταν να αγοράσει το σύνολο του υλικού η ίδια. Υιοθετώντας την cloud-based τεχνική για οικονομική διαχείριση πόρων που ονομάζεται C3, τα άτομα συνεργάζονται πάνω σε μια κοινή πλατφόρμα, με ταυτόχρονη πρόσβαση στα στοιχεία της επιχείρησης και συνάμα με σημαντικά χαμηλότερο κόστος.

- **Ευελιξία**

Κατά την εφαρμογή του Cloud Computing, αποφεύγονται χωρικοί περιορισμοί, και λαμβάνοντας υπόψη τον παγκόσμιο χαρακτήρα της αγοράς σήμερα, είναι ένα από τα βασικότερα πλεονεκτήματα της νέας τεχνολογίας. Αν στη πάροδο του χρόνου παραστεί η ανάγκη να μετακινηθεί μέρος των δραστηριοτήτων μίας εταιρείας σε άλλο σημείο, η ακόμη και όλη η εταιρεία, ο χρόνος που θα παραμείνει χωρίς πληροφοριακά στοιχεία και ο κίνδυνος απώλειας πληροφοριών είναι μηδενικός αφού όλα τα συστήματα και το λογισμικό παραμένουν συνεχώς διαθέσιμα.

Το Cloud Computing επιτρέπει τη πρόσβαση στις εφαρμογές και στα δεδομένα μίας εταιρείας από οποιοδήποτε σημείο του κόσμου, κρυπτογραφημένα με απόλυτη ασφάλεια, μέσω διαδικτύου. Όταν η σύνδεση με το διαδίκτυο δεν είναι εφικτή οι απαιτήσεις σε ταχύτητα μπορούν να ικανοποιηθούν ακόμη και από μία εφεδρική σύνδεση 3G.

- **Ανάπτυξη των επιχειρήσεων**

Οι εφαρμογές Cloud Computing επιτρέπει στις επιχειρήσεις να αυξήσουν την κλίμακα και την δύναμη των πληροφοριακών συστημάτων τους και της ταχύτητας με την οποία μπορεί να έχει πρόσβαση ο χρήστης. και να αναπτυχθεί. Εξαλείφει υλικοτεχνικές προδιαγραφές, γεωγραφικούς περιορισμούς και οργανωτικά όρια, διευκολύνοντας σημαντικά την διοίκηση και ανάπτυξη τους. Όλα αυτά τα πλεονεκτήματα αναμένεται να αυξηθούν, καθώς προχωρούμε βαθύτερα στην εποχή του Cloud Computing. Αναγνωρίζοντας αυτό, οι εταιρείες στο μέλλον θα υιοθετούν τη χρήση του νέφους, ακόμη και όταν έχουν τους τεχνικούς, οικονομικούς και ανθρώπινους πόρους για να ασκήσουν οποιαδήποτε στρατηγική ανάπτυξης.

Η υπηρεσία βίντεο Netflix, το κοινωνικό δίκτυο παιχνιδιού Zynga, και το eBay, είναι μεταξύ των εταιρειών που έχουν δηλώσει δημόσια ότι το σύννεφο είναι ένα σημαντικό μέρος της στρατηγικής τους με τους υπολογιστές. Έχουν συνειδητοποιήσει ότι δεν χρειάζεται να έχουν τη δική του τεχνολογία ώστε να είναι ανταγωνιστικοί.

- **Εντοπισμός πληροφοριών από το δίκτυο δεδομένων**

Με την ψηφιοποίηση των δεδομένων κάθε επιχείρησης, συγκεντρώνονται τεράστιες ποσότητες δεδομένων. Οι πάροχοι cloud προμηθεύουν υλικό και αλγόριθμους εντοπισμού και ομαδοποίησης στοιχείων, ώστε να βοηθήσει τις επιχειρήσεις να διαχειριστούν και να φιλτράρουν αυτόν τον όγκο πληροφοριών. Τα παραπάνω λειτουργούν με σκοπό την κατανόηση και την πρόβλεψη και εργαζομένων και πελατών.

Σαν παράδειγμα θα αναφερθεί η εφαρμογή ARG point-of-sale της Radiant Systems, η οποία χρησιμοποιείται από χιλιάδες εστιατόρια, διατηρώντας τα δεδομένα τους. Η συγκεκριμένη εφαρμογή, εκτός της βασικής υπηρεσίας της, παρέχει ανίχνευση κλοπής, μέσω μιας σειράς ανάλυσης και σύγκρισης δεδομένων. Με αυτό τον τρόπο,

δεν είναι αναγκαία η εγκατάσταση περεταίρω λογισμικού, η μίσθωση επιπλέον τεχνικών ή αναλυτών, διευκολύνοντας τη λειτουργία και παρέχοντος μια συνολική και ελεγχόμενη αντίληψη των εσόδων και εξόδων κάθε επιχείρησης.

- **Δυνατότητα ανάκτησης δεδομένων σε περιπτώσεις απώλειας**

Πρόσφατες έρευνες έχουν δείξει ότι περίπου το 90% των επιχειρήσεων δεν έχουν επαρκή σχέδια για να αντιμετωπίσουν ολική ή μερική απώλεια της μηχανογραφικής τους υποδομής ώστε να συνεχίσουν αδιάλειπτα τη λειτουργία τους. Με το cloud computing μπορούν να αυτοματοποιηθούν οι διαδικασίες αποκατάστασης καταστροφής μέσω της χρήσης αντιγράφων ασφαλείας, έως την διατήρηση έτοιμων διακομιστών για να λειτουργήσουν ως εικόνα άλλων.

- **Ανάπτυξη και Φιλοξενία Εφαρμογών**

Πριν από την εμφάνιση του Cloud Computing, οι προγραμματιστές λογισμικού ήταν υποχρεωμένοι να αγοράσουν, να διαμορφώσουν και να διατηρήσουν τους δικούς τους διακομιστές, πράγμα ιδιαίτερα χρονοβόρο, για αυτό οι υπηρεσίες που παρέχουν τα cloud networks εισήγαγαν φιλοξενία των εφαρμογών για τους πελάτες τους στους δικούς τους διακομιστές. Παράλληλα παρέχουν και ισχυρό λογισμικό σε εταιρίες και ιδιώτες. Το Google Earth Builder, για παράδειγμα, είναι ένα σύνολο ψηφιακών εργαλείων γεωγραφικής οπτικοποίησης και ανάλυσης δεδομένων, το οποίο επιτρέπει στους οργανισμούς να ανεβάσουν τα δικά τους δεδομένα. Με αυτόν τον τρόπο, προκύπτει η δημιουργία δυναμικών και πολυδιάστατων εφαρμογών, όπως το Google Earth, Google Chart Tools, και το Google Maps, με εξαιρετικά χαμηλότερους οικονομικούς πόρους σε σύγκριση με αυτούς που χρειάζεται μια ανεξάρτητη εταιρεία.

Κάθε οργανισμός που συλλέγει στοιχεία και πληροφορίες για γεωγραφικές εκτάσεις, τοπόσημα, χώρες και ιστορικά μνημεία, καθώς και περιβαλλοντικά ζητήματα, μεταφέρει τα δεδομένα στο Google και τα στοιχεία γίνονται προσβάσιμα τόσο στις σελίδες του εκάστοτε οργανισμού ή εταιρείας, όσο και στην κοινόχρηστη πλατφόρμα του Google.

2.6.2 Μειονεκτήματα του Cloud

Ωστόσο, το Cloud Computing δεν έχει μόνο υποστηρικτές. Τα μειονεκτήματα που παρουσιάζονται είναι τα εξής:

- **Κόστος**

Η διάχυτη αβεβαιότητα που επικρατεί σχετικά με τις δαπάνες του cloud, καθίσταται πιο σαφής όσον αφορά στο συγκριτικό κόστος κάθε επιχείρησης. Πρώτον, οι περισσότερες εταιρείες δεν ξοδεύουν τεράστια ποσά στον τομέα της τεχνολογίας, έτσι ώστε ακόμη και σημαντικές μειώσεις στον προϋπολογισμό της πληροφορικής δεν θα αποτελούν μεγάλη διαφορά στην οικονομική τους αποτίμηση. Δεύτερον, με την πάροδο του χρόνου, τα οικονομικά της κατασκευής και λειτουργίας μιας τεχνολογικής υποδομής θα ωφελούν τους παρόχους cloud computing -τόσο στο συνολικό κόστος, όσο και στην υιοθέτηση των ολοένα βελτιωμένων εφαρμογών.

Αυτό αποδεικνύεται από τις συνεχείς μειώσεις στις τιμές της Amazon Web Services, πάνω από δέκα φορές τα τρία τελευταία χρόνια, ώστε να κάνει τα προγράμματα που προωθεί όλο και πιο προσιτά. Επιπροσθέτως, παρά την υπόσχεση του cloud computing, τα συστήματα παλαιού τύπου τεχνολογιών συνεχίζουν να αποτελούν τροχοπέδη κάθε CIO που επιθυμεί να υιοθετήσει συστήματα cloud και απαιτούνται αποφάσεις για την ενοποίηση και τυποποίηση τους. Οι περισσότεροι οργανισμοί διαθέτουν ένα συνονθύλευμα hardware, τα λειτουργικά συστήματα και εφαρμογές που πρέπει αρχικά να ξετυλιχτούν και να απλοποιηθούν, ώστε η εταιρεία τους να μπορεί να κινηθεί προς αυτήν την κατεύθυνση.

- **Αξιοπιστία**

Ένα βασικό ζήτημα είναι η αξιοπιστία. Η αξιοπιστία του Cloud Computing αμφισβητήθηκε τον Απρίλιο του 2011, όταν μεγάλα τμήματα της υποδομής Web Services της Amazon κατέρρευσαν για τρεις ημέρες. Αυτό αποτέλεσε ένα μεγάλο πλήγμα για πολλές εταιρείες που το χρησιμοποιούσαν, αλλά όχι όλες. Για παράδειγμα, η δημοφιλής υπηρεσία βίντεο Netflix βασίζεται σε μεγάλο βαθμό στην Amazon, αλλά δεν επηρεάστηκε από τη διακοπή. Αυτό συνέβη λόγω της τακτικής που ακολουθούν οι εταιρείες Cloud, να παραμένουν σε λειτουργία ακόμα και σε περίπτωση μεγάλου προβλήματος, δίνοντας όλη την προσοχή τους σε επιλεγμένες εταιρείες. Σε γενικές γραμμές η αξιοπιστία του cloud είναι αξιοθαύμαστη, με

παράδειγμα την υπηρεσία Gmail της Google, η οποία ήταν διαθέσιμη για το 99.984% του 2010, ή για όλους, αλλά επτά λεπτά κάθε μήνα.

Σύμφωνα με την εταιρεία τεχνολογικών ερευνών αγοράς Radicati Group, εκτιμάται ότι αυτό είναι περίπου 32 φορές πιο αξιόπιστο από το μέσο εταιρικό e-mail. Πιο πολύπλοκα συστήματα, όπως αυτά των συναλλαγών των τραπεζών, ενδεχομένως να χρειάζονται μεγαλύτερο χρόνο προσαρμογής (uptime) από αυτό, αλλά για τις περισσότερες λοιπές χρήσεις παραμένει το ίδιο.

- **Ασφάλεια**

Το τρίτο βασικό ζήτημα είναι αυτό της ασφάλειας. Υποκλοπές μεταδόσεων, παραβίαση των firewalls, εισβολή ιών, worms, και άλλων μορφών malware είναι πιθανά. Δεν είναι δυνατόν να υπάρχει 100% ασφάλεια στο διαδίκτυο, ωστόσο, η επόμενη καλύτερη προσέγγιση είναι η διαρκής παρακολούθηση του τοπίου των απειλών. Αυτό μπορεί να επιτευχθεί μέσω καλύτερων τεχνολογιών για την προστασία των συσκευών, δικτύων, και μεταδόσεων, καθώς και η πρόσληψη και να διατηρήσει ειδικών ασφαλείας για την επίβλεψή τους.

Μεγάλη προσοχή πρέπει να δοθεί σε εταιρείες πωλήσεων και συναλλαγών, με πλήρη ψηφιοποίηση των εμπορευμάτων και κωδικοποίηση των προσωπικών στοιχείων. Η κοινότητα cloud computing είναι σε θέση να τις εφαρμόσει με μια συνειδητή και οργανωμένη προσπάθεια.

- **Κανονισμοί**

Τέλος, το ρυθμιστικό νομικό καθεστώς βρίσκεται σε αρχικό στάδιο. Κάποιοι κανονισμοί είναι ασαφείς, και η νομολογία δεν έχει ακόμη οριστεί, καθώς όλο και περισσότερες λειτουργίες και εφαρμογές στις πολιτικές των επιχειρήσεων συσσωρεύονται. Επίσης, αρκετοί οργανισμοί λαμβάνουν μια επιθετική προσέγγιση προς το φαινόμενο cloud. Για ένα παράδειγμα, ένας μεγάλος οργανισμός που παρά τις πολυάριθμες ρυθμιστικές απαιτήσεις, κινήθηκε προς το cloud computing, είναι η κυβέρνηση των ΗΠΑ. Το 2011, Vive Kendra, ο οποίος ήταν ο υπεύθυνος του τεχνολογικού τμήματος της κυβέρνησης κατά τη χρονική στιγμή, ανακοίνωσε ένα πλάνο 20 δισ. δολαρίων, ή περίπου το ένα τέταρτο του συνόλου των ομοσπονδιακών δαπανών πληροφορικής, ώστε να προχωρήσουν στη νέα τεχνολογία.

- **Επιβλαβή δίκτυα**

Από τη μεριά των επιχειρήσεων, θα πρέπει να γίνεται έρευνα ανάμεσα στους παρόδους υπηρεσιών lab computing και στις υπηρεσίες που αυτοί προσφέρουν, προκειμένου να καταλήξουν στην πιο αξιόπιστη επιλογή. Επιπλέον οι επιχειρήσεις πρέπει να γνωρίζουν τα τρωτά σημεία που αν προσβληθούν, θα εμποδίσουν την αξιόπιστη χρήση των υπηρεσιών cloud computing. Για παράδειγμα, οι επιχειρήσεις πρέπει να είναι ενήμερες γύρω από προγράμματα και πρακτικές κατάχρησης ή επιβλαβούς χρήσης του cloud computing, όπως το κακόβουλο δίκτυο Zeus (Το Zeus είναι εξαιρετικά δημοφιλές κακόβουλο λογισμικό για την υποκλοπή προσωπικών στοιχείων και ιδιαίτερα τραπεζικών κωδικών) και τα trojan horses Infostealing (κακόβουλο λογισμικό που ενώ φαίνεται να κάνει την επιθυμητή από τον χρήστη λειτουργία πριν το τρέξεις ή το εγκαταστήσεις, αντί για αυτό "μπαίνει" στο σύστημα του χρήστη και αντλεί ιδιωτικά στοιχεία), τα οποία εκμεταλλεύονται με ιδιαίτερη επιτυχία ευαίσθητους ιδιωτικούς πόρους και δεδομένα που υπάρχουν σε περιβάλλοντα cloud computing.

Αξίζει να σημειωθεί ότι οι απειλές ασφάλειας δεν είναι απαραίτητα προϊόν κακόβουλων προθέσεων. Καθώς οι μηχανισμοί κοινωνικής δικτύωσης εξελίσσονται, όλο και περισσότερες ιστοσελίδες στηρίζονται σε interfaces προγραμματισμού εφαρμογών (APIs), δηλαδή σε ένα σετ λειτουργιών που επιτρέπουν την αλληλεπίδραση μεταξύ προγραμμάτων λογισμικού για την ανταλλαγή δεδομένων από διαφορετικές πηγές. Οι ιστοσελίδες που στηρίζονται σε πολλά APIs συχνά αντιμετωπίζουν προβλήματα λόγω του φαινομένου του πιο αδύναμου κρίκου, στο οποίο ένα μη ασφαλές στέλεχος μπορεί να έχει αρνητική επίδραση στο ευρύτερο σύνολο των APIs. Έτσι, μπορεί να δημιουργηθεί ένας συνδυασμός τρωτών σημείων και να αναπτυχθούν κακόβουλες τεχνικές που θέτουν σε κίνδυνο την ασφάλεια των συστημάτων και των δεδομένων.

2.7 Συμπεράσματα

Εν κατακλείδι, η αβεβαιότητα σχετικά με τα οφέλη του και τις ανησυχίες Cloud Computing στις εταιρείες μπορεί να εκτιμηθεί από τις ίδιες. Αν πυροδοτούσε αλλαγές μόνο στον προϋπολογισμό, οι επιπτώσεις θα ήταν ήσσονος σημασίας, αλλά πόσο πολύτιμη είναι η βελτίωση στην παραγωγικότητα, τη συνεργασία, την ανάλυση και την ανάπτυξη εφαρμογών; Συνειδητοποιούμε ότι πρόκειται για έναν ολοκαίνουριο τρόπο εργασίας και αλληλεπίδρασης.

Το Cloud Computing ίσως μειώσει το χάσμα μεταξύ των μικρών και μεγάλων επιχειρήσεων. Επιτρέπει στις μικρές και μεσαίες επιχειρήσεις να ανταγωνίζονται με ίσους όρους τις ανταγωνιστικές επιχειρήσεις τους, δημιουργώντας ένα κοινό περιβάλλον για όλες τις τάξεις επιχειρήσεων και οργανισμών, ώστε να εκμεταλλεύονται τα οφέλη του. Μειώνοντας το κόστος προγραμματιστικού υλικού, αφαίρεσε την ανάγκη για υψηλές επενδύσεις αρχικού κεφαλαίου, πράγμα δύσκολο έως ανέφικτο για τις περισσότερες που μικρές επιχειρήσεις.

Ωστόσο, παρέχοντας τις ίδιες δυνατότητες σε όλους. Η κάθε εταιρεία δεν μπορεί πια να βασίζεται στην τεχνολογική της υποδομή ως πλεονέκτημα σε σύγκριση με τις όμοιές τις. Πλέον, οι βασικοί ανταγωνιστές κάθε επιχείρησης είναι δυνατόν να αναβαθμίσουν τα δικά τους κριτήρια, απλά με την αλλαγή των υπολογιστικών υποδομών τους.

Ένα κοινό χαρακτηριστικό των μεγάλων τεχνολογικών μεταβολών είναι ότι τα πλήρη αποτελέσματά τους δεν είναι ορατά από την αρχή. Καθώς η υπηρεσία cloud μεγαλώνει και ωριμάζει, οι προμηθευτές της θα συνεχίζουν να καινοτομούν και να διαφοροποιούν τις προσφορές. Τα αποτελέσματα μπορώ να προβλέψω ότι θα οδηγήσει σε εταιρικό υπολογιστών περιβάλλοντα πολύ διαφορετικές από αυτές στις σήμερα.

3 Αναφορά στην Γλώσσα R, στο HDFS και στο Map Reduce - Σχεδιασμός και Ανάλυση του Προβλήματος

3.1 Εισαγωγη

Σε αυτό το κεφάλαιο θα μελετήσουμε τις τεχνολογίες καθώς επίσης και τις τεχνικές που χρησιμοποιούμε στο πειραματικό μέρος της παρούσης διπλωματικής. Όπως θα δούμε και παρακάτω η γλώσσα που τρέχουμε τον skyline map reduce αλγόριθμο είναι η R όπως επίσης κάνουμε χρήση και επιπλέων βιβλιοθηκών στην R για την τεχνική MapReduce.

Επιπλέων στον παρών κεφάλαιο μελετούμε σε βάθος τις τεχνικές του MapReduce καθώς και τον συνδυασμό τρέξιμο των skyline αλγορίθμων με την τεχνική του MapReduce.

Τέλος αναφερόμαστε στον σχεδιασμό και στην ανάλυση του προβλήματος τόσο σε επίπεδο αρχιτεκτονικής όσο και σε κώδικα. Επίσης αναλύονται όλες οι φάσεις της λειτουργίας του κώδικα καθώς και πως αλληλεπιδρά με το Σύστημα Αρχείων του Hadoop (HDFS).

3.2 Η Γλώσσα R

Η R αποτελείτε από τρία πράγματα: ένα έργο, μια γλώσσα και ένα περιβάλλον. Η R προσφέρει ένα ολοκληρωμένο σύνολο υπηρεσιών λογισμικού για ανάλυση δεδομένων, υπολογισμών και γραφημάτων. Το λογισμικό R γράφτηκε αρχικά από τους Ross Ihaka και Robert Gentleman στα μέσα της δεκαετίας του 90 (σε αυτούς οφείλει το όνομά του). Από το 1997 αναπτύσσεται από το R Devel- opment Core Team. Η R είναι λογισμικό ανοικτού κώδικα (open source) και αποτελεί μέρος του έργου GNU.

Σαν γλώσσα η R μπορεί να θεωρηθεί ότι αποτελεί μια εφαρμογή της γλώσσας προγραμματισμού S. Η γλώσσα και το περιβάλλον S αναπτύχθηκε στα Bell Laboratories (πρώην AT&T, τώρα Lucent Technologies) από τους Rick Becker, John Champers και Allan Wilks (το έργο ξεκίνησε το 1976).

Άλλες δύο (εμπορικές) εφαρμογές της γλώσσας S είναι η “παλαιά S μηχανή” (S version 3; S-P LUS 3.x and 4.x) και η “νέα S μηχανή” (S version 4; S-P LUS 5.x και

άνω). Όταν ρωτάμε για διαφορές μεταξύ R και S ουσιαστικά αναφερόμαστε στις διαφορές μεταξύ R και των δύο S μηχανών. Οι διαφορές είναι ελάχιστες και έτσι κώδικας που γράφεται για την R τρέχει σχεδόν αμετάβλητος και στις δύο S μηχανές.

Η ιστοσελίδα της R είναι η http://www.r-project.org και αποτελεί την κύρια πηγή πληροφόρησής του.

3.3 Βιβλιοθήκες της R για Map Reduce

Η R διαθέτει πολλές χρήσιμες συναρτήσεις χωρίς ειδική αναζήτηση, αυτόματα. Υπάρχουν όμως πάρα πολλές διαφορετικές συναρτήσεις, που χρησιμοποιούνται για διαφορετικά είδη αναλύσεων. Οι περισσότερες είναι οργανωμένες σε εξωτερικά «πακέτα» συναρτήσεων, που ονομάζονται «βιβλιοθήκες». Τα πακέτα αυτά πρέπει να τα εγκαταστήσουμε στον υπολογιστή μας ξεχωριστά από την ίδια την R. Η εγκατάσταση κατεβάζει αυτομάτως το πακέτο από το διαδίκτυο, άρα πρέπει πρώτα να εξασφαλίσουμε ότι η σύνδεσή μας είναι ενεργή. Για να εγκαταστήσουμε ένα πακέτο, π.χ. το psych (για αναλύσεις που χρησιμοποιούνται συχνά στην ψυχολογία), χρησιμοποιούμε την κατάλληλη συνάρτηση μέσα από την R:

```
>install.packages("psych",depend=T).
```

Την πρώτη φορά που θα χρησιμοποιήσουμε αυτή τη συνάρτηση, η R θα μας ρωτήσει από πού να κατεβάσει το πακέτο. Επιλέγουμε την Ελλάδα ή άλλη ευρωπαϊκή χώρα, π.χ. Αυστρία. Στη συνέχεια η R κατεβάζει και εγκαθιστά ότι είναι απαραίτητο για τη λειτουργία του πακέτου. Αν θέλουμε να εγκαταστήσουμε περισσότερα πακέτα μονομιάς, χρησιμοποιούμε την ακολουθία:

```
>install.packages(c("e1071","nortest","Hmisc"),depend=T)
```

Η R εγκαθιστά τα πακέτα μαζί με τυχόν προ απαιτούμενα. Στο εξής θα είναι διαθέσιμα στον υπολογιστή μας ανεξάρτητα από το αν είμαστε συνδεδεμένοι στο διαδίκτυο ή όχι.

Για να χρησιμοποιήσουμε συναρτήσεις από τα νέα πακέτα, πρέπει πρώτα να «φορτώσουμε» τη βιβλιοθήκη που περιλαμβάνει το πακέτο, χρησιμοποιώντας τη συνάρτηση library. Έτσι, αν έχουμε ένα πλαίσιο δεδομένων ch και θέλουμε να χρησιμοποιήσουμε τη συνάρτηση describe που περιλαμβάνεται στη βιβλιοθήκη του πακέτου psych: >library(psych).

Η R εγκαθιστά τα πακέτα μαζί με τυχόν προ απαιτούμενα. Στο εξής θα είναι διαθέσιμα στον υπολογιστή μας ανεξάρτητα από το αν είμαστε συνδεδεμένοι στο διαδίκτυο ή όχι. Για να χρησιμοποιήσουμε συναρτήσεις από τα νέα πακέτα, πρέπει πρώτα να «φορτώσουμε» τη βιβλιοθήκη που περιλαμβάνει το πακέτο, χρησιμοποιώντας τη συνάρτηση library. Έτσι, αν έχουμε ένα πλαίσιο δεδομένων ch και θέλουμε να χρησιμοποιήσουμε τη συνάρτηση describe που περιλαμβάνεται στη βιβλιοθήκη του πακέτου psych: >library(psych).

>describe(ch) Η ενεργοποίηση της βιβλιοθήκης πρέπει να γίνεται κάθε φορά που ξεκινάμε την R.

3.4 Το Map Reduce

Καθώς οι ολο ένα αυξανόμενοι χρήστες «παράγουν» συνεχώς δεδομένα, ανεβάζοντας στο διαδίκτυο βίντεο, φωτογραφίες, ενημερώνοντας τα προφίλ τους στις σελίδες κοινωνικής δικτύωσης, τα υπάρχοντα εργαλεία καθίστανται όλο και πιο ανεπαρκή να επεξεργαστούν τόσο μεγάλο όγκο δεδομένων. Αυτή την ανεπάρκεια, έρχεται να καλύψει το MapReduce.

Το MapReduce, είναι ένα προγραμματιστικό μοντέλο, το όποιο προορίζεται για την παράλληλη επεξεργασία τεράστιου όγκου δεδομένων, του οποίου η κύρια ιδέα είναι να κρύψει τις λεπτομέρειες της παράλληλης εκτέλεσης και να επιτρέψει στους χρήστες να επικεντρωθούν στις στρατηγικές επεξεργασίας δεδομένων.

Το Hadoop, στο οποίο βασίζεται η εργασία, είναι μια open source υλοποίηση του MapReduce, η οποία έχει χρησιμοποιηθεί ευρέως από ένα πλήθος οργανισμών. Πιο συγκεκριμένα, το Hadoop είναι ένα framework που διευκολύνει την ανάπτυξη εφαρμογών οι οποίες επεξεργάζονται τεράστιο όγκο δεδομένων παράλληλα σε μεγάλες συστάδες (χιλιάδες κόμβους) με αξιοπιστία και ανοχή σε σφάλματα.

Το Hadoop αποτελείται από δύο επίπεδα: το επίπεδο αποθήκευσης δεδομένων που λέγεται Hadoop DFS (HDFS) και το επίπεδο επεξεργασίας δεδομένων που λέγεται Hadoop MapReduce Framework. Το HDFS είναι ένα σύστημα αρχείων το οποίο χρησιμοποιεί το Hadoop για να διαβάζει τα αρχεία προς επεξεργασία καθώς και για να ‘γράφει’ εκεί τα αποτελέσματα.

3.4.1 Το Σύστημα Αρχείων του Hadoop

Το HDFS έχει master/slave αρχιτεκτονική. Μια συστάδα HDFS αποτελείται από έναν μόνο NameNode, έναν master server που διαχειρίζεται το namespace του συστήματος αρχείων και ρυθμίζει την πρόσβαση των πελατών στα αρχεία. Συμπληρωματικά υπάρχει ένας αριθμός DataNodes, συνήθως ένας ανά κάθε κόμβο της συστάδας, που ελέγχουν την αποθήκευση στον κόμβο πάνω στον οποίο τρέχουν. Το HDFS εκθέτει ένα file system namespace και επιτρέπει στα δεδομένα των χρηστών να αποθηκευτούν σε αρχεία. Εσωτερικά κάθε αρχείο σπάει σε ένα ή περισσότερα μπλοκς τα οποία με τη σειρά τους αποθηκεύονται στους datanodes. Ο NameNode εκτελεί εργασίες του namespace του συστήματος αρχείων όπως άνοιγμα, κλείσιμο και μετονομασία αρχείων και φακέλων. Επιπρόσθετα καθορίζει την αντιστοίχιση των μπλοκς στους datanodes. Οι DataNodes είναι υπεύθυνοι για την εξυπηρέτηση αιτημάτων ανάγνωσης και εγγραφής από τους πελάτες του συστήματος αρχείων. Επιπλέον δημιουργούν, διαγράφουν και αντιγράφουν μπλοκς μετά από οδηγίες του NameNode.

Η ύπαρξη ενός και μοναδικού NameNode στη συστάδα απλοποιεί σημαντικά την αρχιτεκτονική του συστήματος. Ο NameNode κρατά και ελέγχει όλα τα μεταδεδομένα του HDFS. Με τον τρόπο αυτό τα μεταδεδομένα είναι αποκομμένα από τα δεδομένα του συστήματος το οποίο έχει σχεδιαστεί με τέτοιο τρόπο ώστε τα δεδομένα των χρηστών να μην περνάνε πότε μέσα από το NameNode.

Μια πλήρως διαμορφωμένη HDFS συστάδα τρέχει ένα σετ από daemons που περιλαμβάνουν:

1. NameNode: Είναι ο master του HDFS ο οποίος διευθύνει τους σκλάβους DataNode daemons.

2. DataNode: είναι ο εργάτης του κατανεμημένου συστήματος αρχείων ο οποίος γράφει και διαβάζει HDFS μπλοκς σε πραγματικά αρχεία στο τοπικό σύστημα αρχείων.

3. SecondaryNameNode: Είναι ένας βοηθητικός daemon για την παρακολούθηση της κατάστασης της HDFS συστάδας.

4. JobTracker: Είναι ο σύνδεσμος ανάμεσα στις εφαρμογές-πελάτες και στο Hadoop, αποφασίζει το πλάνο εκτέλεσης για τις καταχωρημένες εργασίες, αναθέτει

διαφορετικές εργασίες στους κόμβους και παρακολουθεί όλες τις εργασίες που τρέχουν.

5. TaskTracker: Είναι υπεύθυνος για την εκτέλεση των μεμονωμένων εργασιών που αναθέτει ο JobTracker.

3.4.2 Περιγραφή του MapReduce Framework

Ένα MapReduce job εκτελείται σε δύο φάσεις: την map και τη reduce. Επίσης μπορεί να χωριστεί σε πολλά map tasks και reduce tasks, ανάλογα με τον αριθμό των Mappers και Reducers που θα ορίσει ο χρήστης. Σημαντικό ρόλο παίζει ο master node, ο οποίος διαλέγει κενούς «εργάτες» και τους αναθέτει ένα map ή reduce task ανάλογα με το στάδιο. Πριν ξεκινήσει ένα Map task, ένα αρχείο εισόδου φορτώνεται στο σύστημα αρχείων του Hadoop. Κατά τη διάρκεια του φορτώματος, το αρχείο χωρίζεται σε πολλαπλά blocks δεδομένων τα οποία έχουν το ίδιο μέγεθος και κάθε block αντιγράφεται τρείς φορές ώστε να είναι εγγυημένη η ανοχή σφαλμάτων. Στη συνέχεια, κάθε block ανατίθεται σ' ένα mapper και αυτός εφαρμόζει την Map σε κάθε εγγραφή του block δεδομένων. Τα ενδιάμεσα αποτελέσματα που παράγονται από τους mappers, ταξινομούνται τοπικά σε ζεύγη κλειδιού-τιμής με βάση το κλειδί και στη συνέχεια, τα αποτελέσματα αποθηκεύονται στους τοπικούς δίσκους των mappers όπου χωρίζονται σε r κομμάτια, όπου r είναι ο αριθμός των Reducers που χρησιμοποιούνται.

Όταν όλα τα Map tasks ολοκληρωθούν, θα ξεκινήσει η reduce φάση όπου θα ανατεθούν στους «εργάτες» τα Reduce tasks. Ο Reducer διαβάζει τα ενδιάμεσα αποτελέσματα και τα συγχωνεύει με βάση τα ενδιάμεσα κλειδιά έτσι ώστε τιμές του ίδιου κλειδιού να ομαδοποιούνται με βάση αυτό. Στη συνέχεια, ο Reducer εφαρμόζει την μέθοδο Reduce στις ενδιάμεσες τιμές για κάθε κλειδί που συναντά και τα τελικά αποτελέσματα αποθηκεύονται και αντιγράφονται τρείς φορές στο HDFS.

Σημαντικό πλεονέκτημα του MapReduce είναι η ανοχή σφαλμάτων και αυτό επιτυγχάνεται ανιχνεύοντας κόμβους που δε λειτουργούν πια και αναθέτοντας ξανά τα tasks τους σε άλλους υγιείς. Επίσης, κόμβοι που έχουν ολοκληρώσει τα tasks τους μπορούν να δεχτούν νέα κόμβοι δεδομένων και να ξεκινήσουν νέα tasks.

Ακόμα, tasks που εκτελούνται σε αργούς κόμβους μετατίθενται σε άλλους κενούς που έχουν ολοκληρώσει τα προηγούμενα tasks τους, χωρίς βέβαια αυτό να σημαίνει ότι θα εκτελεστούν γρηγορότερα.

3.4.3 Βελτιώσεις του MapReduce

Για όλα τα παραπάνω χαρακτηριστικά, το MapReduce και γενικότερα τα υπολογιστικά νέφη έχουν κερδίσει έδαφος και έχουν γίνει διάφορες μελέτες και πειράματα σε σχέση με την απόδοσή τους συγκριτικά με άλλα συστήματα. Παρακάτω θα δούμε κάποιες από αυτές τις έρευνες.

Οι C.Doulkeridis et al. παρουσιάζουν μια έρευνα στην οποία συγκεντρώνουν όλες τις προσπάθειες βελτίωσης της απόδοσης της παράλληλης επεξεργασίας των επερωτήσεων χρησιμοποιώντας το MapReduce. Πιο συγκεκριμένα, καταγράφουν τις πιο σημαντικές αδυναμίες και περιορισμούς του MapReduce, προτείνοντας παράλληλα τεχνικές επίλυσής τους. Τέλος, παρουσιάζουν μια ταξινόμηση των ήδη υπαρχουσών βελτιώσεων με βάση το πρόβλημα στου οποίου την επίλυση στοχεύουν.

Οι J.Schad et al έχουν κάνει runtime μετρήσεις σε υπολογιστικό νέφος και συγκεκριμένα στο Amazon Elastic Computing Cloud (EC2), που αφορούν στο instance startup, CPU απόδοση, ταχύτητα μνήμης, disk I/O, network bandwidth και S3 access. Ο λόγος αυτής της έρευνας είναι η απρόβλεπτη απόδοση στο Cloud Computing η οποία επηρεάζει τους ερευνητές βάσεων δεδομένων, οι οποίοι εκτελούν wall clock πειράματα, και τις εφαρμογές βάσεων δεδομένων που παρέχουν service level agreements. Για την έρευνα χρησιμοποίησαν μια multimode MapReduce εφαρμογή για να ποσοτικοποιήσουν τον αντίκτυπο σε πραγματικές εφαρμογές ευαίσθητων δεδομένων. Συγκέντρωσαν δεδομένα για ένα ολόκληρο μήνα και μετά τα συνέκριναν με τα αποτελέσματα που απέκτησαν σε ένα τοπικό cluster. Η ανάλυσή τους έδειξε ότι μικρά και μεγάλα instances πλήττονται από μεγάλη διακύμανση στην επίδοση. Επίσης παρατηρούν ότι ένας από τους λόγους αυτής της διακύμανσης είναι οι διαφορετικοί τύποι συστημάτων που χρησιμοποιούνται σε virtual nodes, π.χ: τα Xeon-based system έχουν καλύτερη επίδοση από τα Opteon-based systems. Επίσης, συνέκριναν τη διακύμανση σε υπολογιστικό νέφος με τη διακύμανση σε φυσική συστάδα και τα αποτελέσματα έδειξαν ότι ένα MapReduce job είχε πολύ περισσότερη διακύμανση στο EC2. Ένα σημαντικό συμπέρασμα αυτής της εργασίας

είναι ότι η διακύμανση στο EC2 είναι προς το παρόν τόσο μεγάλη που τα wall clock πειράματα μπορούν να εκτελεστούν μόνο με πολύ προσοχή.

Οι J.Dean et al. και οι M.Stonebraker et al. παρουσιάζουν το MapReduce, περιγράφουν την αρχιτεκτονική του και κυρίως το συγκρίνουν με παράλληλες βάσεις δεδομένων παραθέτοντας τα πλεονεκτήματα και τα μειονεκτήματά του. Στις έρευνές τους αναφέρουν ότι έχουν παρατηρηθεί διάφορες παρανοήσεις για το MapReduce σε σχετική βιβλιογραφία. Κάποιες από αυτές αναφέρουν ότι το MapReduce δεν μπορεί να χρησιμοποιήσει δείκτες και εφαρμόζει πλήρες σκανάρισμα σε όλα τα δεδομένα εισόδου, ότι η είσοδος και η έξοδος είναι πάντα απλά αρχεία σ' ένα σύστημα αρχείων και ότι απαιτεί τη χρήση ανεπαρκών textual data formats. Επίσης ασχολούνται και με άλλα σημαντικά ζητήματα όπως το ότι το MapReduce είναι ανεξάρτητο από συστήματα αποθήκευσης και μπορεί να επεξεργαστεί τα δεδομένα χωρίς προηγουμένως να απαιτεί να φορτώνονται σε βάση δεδομένων. Ακόμα αναφέρουν ότι περίπλοκοι μετασχηματισμοί είναι συχνά ευκολότερο να εκφραστούν σε MapReduce απ' ότι σε SQL. Τέλος, καταλήγουν στα εξής συμπεράσματα για το MapReduce. Πρώτον, είναι ένα μοντέλο εύκολο να χρησιμοποιηθεί. Δεύτερον, μια ευρεία γκάμα προβλημάτων εκφράζονται εύκολα με υπολογισμούς MapReduce. Τρίτον, η υλοποίηση του MapReduce είναι εύρωστη σε μεγάλες συστάδες που αποτελούνται από χιλιάδες μηχανήματα.

Στη συνέχεια, οι J.Dean et al. παρατηρούν ότι τα ζητήματα του πώς να παραλληλιστούν οι υπολογισμοί, να κατανεμηθούν τα δεδομένα και να διαχειρίζονται οι αποτυχίες συντελούν στο να υποβαθμιστεί ο αρχικός απλός υπολογισμός χρησιμοποιώντας μεγάλο πολύπλοκο κώδικα για να τα αντιμετωπίσουν. Εξαιτίας αυτής της πολυπλοκότητας σχεδίασαν ένα abstraction που τους επιτρέπει να εκφράζουν απλούς υπολογισμούς που προσπαθούν να εφαρμόσουν κρύβοντας όμως τις λεπτομέρειες του παραλληλισμού, της ανοχής σφαλμάτων, της κατανομής των δεδομένων και την ισορροπία φορτίου σε μια βιβλιοθήκη. Το abstraction είναι εμπνευσμένο από την map και την reduce μέθοδο σε Lisp αλλά και από άλλες λειτουργικές γλώσσες. Η χρήση ενός λειτουργικού μοντέλου με map και reduce λειτουργίες καθορισμένες από το χρήστη, τους επιτρέπει να παραλληλίζουν εύκολα μεγάλους υπολογισμούς και να χρησιμοποιούν την επανεκτέλεση ως τον βασικό μηχανισμό για την ανοχή σφαλμάτων. Οι σημαντικότερες προσφορές αυτής της εργασίας είναι ένα απλό και δυνατό interface που επιτρέπει τον αυτόματο

παραλληλισμό και την κατανομή υπολογισμών, συνδυασμένο με μια υλοποίηση του interface που επιτυγχάνει υψηλή απόδοση σε μεγάλα clusters. Από αυτή την εργασία έφτασαν στο συμπέρασμα ότι ο περιορισμός του προγραμματιστικού μοντέλου κάνει ευκολότερο τον παραλληλισμό και την κατανομή υπολογισμών καθώς επίσης κάνει τους υπολογισμούς ανεκτικούς σε σφάλματα. Αρκετές βελτιώσεις στην εργασία τους είχαν στόχο να μειώσουν τα δεδομένα που στέλνονται στο δίκτυο. Η τοπική βελτιστοποίηση τούς επέτρεψε να διαβάζουν δεδομένα από τοπικούς δίσκους και το να γράφουν ένα αντίγραφο από τα ενδιάμεσα δεδομένα σε τοπικούς δίσκους, εξοικονομεί εύρος ζώνης του δικτύου. Ακόμα, οι περιττές εκτελέσεις μπορούν να χρησιμοποιηθούν για να μειώσουν το φαινόμενο της ύπαρξης αργών μηχανημάτων και για να διαχειρίζονται τις αποτυχίες και την απώλεια δεδομένων. Μετά τις μετρήσεις που έκαναν έφτασαν στο συμπέρασμα ότι το MapReduce είναι ένα άκρως αποτελεσματικό και αποδοτικό εργαλείο για εύρωστη και ανεκτική σε σφάλματα ανάλυση δεδομένων. Επίσης, παρέχει ανοχή σε σφάλματα για μεγάλες εργασίες και η αποτυχία στην μέση μιας πολύωρης εκτέλεσης δεν απαιτεί επανεκκίνηση της εργασίας από την αρχή.

Οι K.Lee et al. κάνουν στην ουσία μια επισκόπηση και παραθέτουν τα πλεονεκτήματα και μειονεκτήματα του MapReduce, ταξινομούν τις βελτιώσεις που έχουν γίνει και καταλήγουν στο συμπέρασμα ότι το MapReduce είναι απλό παρέχοντας επεκτασιμότητα και ανοχή σε σφάλματα για την επεξεργασία μεγάλου όγκου δεδομένων. Ωστόσο, το MapReduce είναι απίθανο να αντικαταστήσει τα ΣΔΒΔ αλλά αναμένεται να λειτουργήσουν συμπληρωματικά με εύρωστη και ευέλικτη, παράλληλη επεξεργασία για δεδομένα διαφόρων τύπων.

Οι J.Dittrich et al. υποστηρίζουν ότι το MapReduce επεξεργάζεται τα tasks σε scan-oriented fashion και αυτό έχει σαν αποτέλεσμα η επίδοσή του να μην φτάνει αυτή ενός καλά ρυθμισμένου παράλληλου ΣΔΒΔ. Έτσι προτείνουν ένα νέο τύπο συστήματος που λέγεται Hadoop++ το οποίο δίνει σημαντική ώθηση στην επίδοση ενός task χωρίς ν' αλλάζει καθόλου το framework του Hadoop. Για να επιτευχθεί αυτό, εισάγουν την τεχνολογία τους στα κατάλληλα σημεία με UDFs επηρεάζοντας έτσι το Hadoop από μέσα. Αυτό έχει σαν αποτέλεσμα, πρώτον το Hadoop++ να υπερτερεί σημαντικά του Hadoop. Δεύτερον, οποιεσδήποτε μελλοντικές αλλαγές του Hadoop μπορούν να χρησιμοποιηθούν κατευθείαν στο Hadoop++ χωρίς να ξαναγραφτεί κώδικας. Τρίτον, το Hadoop++ δεν απαιτεί την αλλαγή του Hadoop

interface. Επίσης, είδαν ότι όσον αφορά στην επεξεργασία επερωτήσεων το Hadoop++ φτάνει και καμιά φορά βελτιώνει τα runtimes των επερωτήσεων του HadoopDB και αλλάζει το εσωτερικό layout ενός split - ένα μεγάλο partition δεδομένων — και/ή τροφοδοτεί το Hadoop με τα κατάλληλα UDFs χωρίς ν' αλλάξει καθόλου το framework του Hadoop. Για τις μετρήσεις τους υλοποίησαν εργασίες όπως data loading, selection task και join task. Επίσης υλοποίησαν fault tolerance πειράματα (node failures και straggler nodes πειράματα).

Τα πειραματικά αποτελέσματα έδειξαν ότι σε γενικές γραμμές το Hadoop++ υπερτερεί του Hadoop. Ακόμα σε tasks που σχετίζονται με indexing και join processing το Hadoop++ υπερτερεί και του HadoopDB – χωρίς να απαιτείται ένα ΣΔΒΔ ή μεγάλες αλλαγές στο framework της εκτέλεσης ή το interface του Hadoop. Επίσης, παρατήρησαν ότι όσο αυξάνεται το μέγεθος του split, τόσο το Hadoop++ βελτιώνεται στα selection και join tasks ενώ η επίδοση στην ανοχή σφαλμάτων μειώνεται. Αυτό δείχνει ότι υπάρχει μια σύνδεση μεταξύ των runtime και fault tolerance jobs του MapReduce.

3.4.4 Διεργασίες του Map Reduce Framework

Μία εργασία MapReduce χωρίζεται σε δύο στάδια, ένα map και ένα reduce. Ο master κόμβος βρίσκει ελεύθερους κόμβους – εργάτες και τους αναθέτει από ένα map ή reduce task ανάλογα με το στάδιο. Πριν ξεκινήσει η εργασία map ένα αρχείο εισόδου φορτώνεται στο hdfs.

Καθώς φορτώνεται σπάει σε πολλαπλά μπλοκ δεδομένων που έχουν το ίδιο μέγεθος και κάθε μπλοκ αντιγράφεται 3 φορές εξασφαλίζοντας έτσι αντοχή σε σφάλματα. Στη συνέχεια κάθε μπλοκ ανατίθεται σε ένα mapper (κόμβος εργάτης στον οποίο έχει ανατεθεί ένα map task) που εφαρμόζει τη συνάρτηση Map σε κάθε εγγραφή του μπλοκ δεδομένων. Τα ενδιάμεσα αποτελέσματα που προέκυψαν από τους mappers ταξινομούνται τοπικά για να ομαδοποιηθούν τα ζεύγη κλειδιού-τιμής που μοιράζονται κοινό κλειδί. Μετά από την τοπική ταξινόμηση η συνάρτηση combine προαιρετικά εφαρμόζεται για να βρει ένα πρώιμο άθροισμα στα ομαδοποιημένα ζεύγη κλειδιού – τιμής ώστε να ελαχιστοποιηθεί το κόστος της μεταφοράς όλων των ενδιάμεσων αποτελεσμάτων στο reducer. Στη συνέχεια τα mapped αποτελέσματα αποθηκεύονται στους τοπικούς δίσκους στους mappers και χωρίζονται σε τόσα τμήματα όσοι και οι reducers της MapReduce εργασίας. Όταν ολοκληρωθούν όλα τα

map tasks ανατίθενται reduce tasks στους εργάτες κόμβους. Επειδή όλα τα mapped αποτελέσματα είναι ήδη χωρισμένα και αποθηκευμένα σε τοπικούς δίσκους κάθε reducer πραγματοποιεί το ανακάτεμα (shuffling) απλά τραβώντας το τμήμα(partition) των mapped αποτελεσμάτων που του αντιστοιχεί από τους mappers. Ένας reducer διαβάζει τα ενδιάμεσα αποτελέσματα και τα συγχωνεύει με βάση τα ενδιάμεσα κλειδιά ώστε όλες οι τιμές με το ίδιο κλειδί να ομαδοποιηθούν. Στη συνέχεια κάθε reducer εφαρμόζει τη συνάρτηση Reduce στις ενδιάμεσες τιμές για κάθε κλειδί που συναντά. Τα αποτελέσματα των reducers αποθηκεύονται και αντιγράφονται 3 φορές στο hdfs. Το MapReduce framework εκτελεί όλες τις εργασίες βασιζόμενο σε ένα σχήμα προγραμματισμού που διαμορφώνεται στο χρόνο εκτέλεσης. Σε αντίθεση με τα ΣΔΒΔ το MapReduce framework δεν καθορίζει ποιες εργασίες θα τρέξουν σε ποιους κόμβους πριν την εκτέλεση αλλά το αποφασίζει ολοκληρωτικά στο χρόνο εκτέλεσης (runtime) και επιτυγχάνει με αυτόν τον τρόπο αντοχή στα σφάλματα εντοπίζοντας τις αποτυχίες και αναθέτοντας τις εργασίες των αποτυχημένων κόμβων σε υγιείς. Σε κόμβους που ολοκλήρωσαν τις εργασίες τους αναθέτονται νέα μπλοκς εισόδου. Με τον τρόπο αυτό επιτυγχάνεται εξισορρόπηση του φόρτου με τους γρηγορότερους κόμβους να επεξεργάζονται περισσότερα δεδομένα εισόδου και τις εργασίες κόμβων που αντιμετωπίζουν δυσκολίες να αναθέτονται σε ελεύθερους κόμβους που ολοκλήρωσαν τις δικές τους εργασίες. Τέλος οι εργασίες map reduce εκτελούνται χωρίς να απαιτείται επικοινωνία με άλλες εργασίες και έτσι δε δημιουργείται καμία διένεξη στο συγχρονισμό και κανένα κόστος επικοινωνίας ανάμεσα σε εργασίες κατά τη διάρκεια μιας εκτέλεσης MR.

3.4.5 Κριτική για το MapReduce framework

Το MapReduce και ιδιαίτερα το Hadoop, κέρδισε έδαφος και θεωρήθηκε το επόμενο βήμα στη διαχείριση μεγάλου όγκου δεδομένων, παράλληλα όμως δέχτηκε και έντονη κριτική καθώς θεωρήθηκε ότι αποτελεί ένα βήμα πίσω στην παράλληλη επεξεργασία όπως αυτή επιτυγχάνεται στα ΣΔΒΔ. Το MapReduce υστερεί σε πολλά από τα χαρακτηριστικά που έχουν αποδειχτεί πολύτιμα για την ανάλυση δομημένων δεδομένων κυρίως γιατί αρχικά δε σχεδιάστηκε για να πραγματοποιεί ανάλυση δομημένων δεδομένων.

Έρευνες που πραγματοποιήθηκαν έδειξαν μια καθαρή ανταλλαγή ανάμεσα στην απόδοση και την ανοχή σε σφάλματα. Το MapReduce αυξάνει την ανοχή σε σφάλματα μέσω της ανάλυσης με συχνούς ελέγχους των ολοκληρωμένων εργασιών

και της αντιγραφής των δεδομένων. Οι συχνοί αυτοί όμως έλεγχοι έχουν τίμημα στην αποδοτικότητα. Από τη μεριά τους τα ΣΔΒΔ στοχεύουν στην απόδοση περισσότερο παρά στην αντοχή στα σφάλματα. Τα ΣΔΒΔ εκμεταλλεύονται τη σύνδεση ενδιάμεσων αποτελεσμάτων ανάμεσα στους εκτελεστές επερωτήσεων με πιθανό κίνδυνο όμως να απαιτηθεί μεγάλος αριθμός επανάληψης εργασιών σε περίπτωση αποτυχίας.

Τα πλεονεκτήματα του Hadoop συνοψίζονται στα εξής :

- Είναι απλό και εύκολο στη χρήση καθώς ο προγραμματιστής χρειάζεται να ορίσει μόνο συναρτήσεις map – reduce.
- Είναι ευέλικτο καθώς δεν εξαρτάται από κανένα μοντέλο ή σχήμα και ο προγραμματιστής μπορεί να διαχειριστεί αδόμητα δεδομένα ευκολότερα από ότι με τα ΣΔΒΔ.
- Είναι ανεξάρτητο από το υποκείμενο επίπεδο μνήμης – αποθήκευσης και δυνατότητα να συνεργαστεί με διαφορετικούς τύπους.
- Παρέχει αντοχή στα σφάλματα.
- Προσφέρει υψηλή επεκτασιμότητα.

Αντίστοιχα παρουσιάζει και σημαντικά μειονεκτήματα σε σύγκριση με τα ΣΔΒΔ και συχνά παρουσιάζεται σαν ένα Extract-Transform-Load(ETL) εργαλείο. Τα σημαντικότερα μειονεκτήματα του συνοψίζονται στα εξής:

- Δεν υποστηρίζει καμία γλώσσα υψηλού επιπέδου και καμία τεχνική βελτιστοποίησης επερωτήσεων.
- Δε διαθέτει σχήμα ή ευρετήριο (index) με αποτέλεσμα μην εκμεταλλεύεται τα πλεονεκτήματα της μοντελοποίησης δεδομένων και να προκαλεί πτώση στην απόδοση.
- Δυσκολία να χρησιμοποιηθεί για πολύπλοκους αλγορίθμους μόνο με χρήση MR εργασιών.

- Χαμηλή αποδοτικότητα καθώς δίνει πρωτεύουσα σημασία στην αντοχή στα σφάλματα και στην επεκτασιμότητα οι λειτουργίες του δεν είναι βελτιστοποιημένες για αποδοτικότητα I/O.

- Είναι πολύ νέο σε σχέση με τα ΣΔΒΔ .

3.4.6 Skyline Επερωτήσεις

Τα τελευταία χρόνια, η διαχείριση και η αποθήκευση των δεδομένων έχει γίνει εξαιρετικά κατανεμημένη. Αυτό έχει σαν αποτέλεσμα, να είναι απαραίτητη η ύπαρξη περισσότερο προηγμένων διαχειριστών ερωτημάτων σε πολυδιάστατα δεδομένα, όπως οι skyline επερωτήσεις, οι οποίες θα διευκολύνουν τους χρήστες στη διαχείριση αυτού του τεράστιου όγκου δεδομένων.

Ο εκάστοτε χρήστης πρέπει, μέσα από ένα σύνολο πολυδιάστατων αντικειμένων, να επιλέξει εκείνα τα αντικείμενα τα οποία είναι καλύτερα από κάποια άλλα, όμως, δεν ξέρει με βάση ποιό κριτήριο ή ποια διάσταση να τα κατατάξει και να τα αξιολογήσει ώστε να καταλήξει στα καλύτερα. Το Skyline του διασφαλίζει ότι θα βρει τα αντικείμενα τα οποία ικανοποιούν καλύτερα τις συνθήκες που έχει ορίσει ο χρήστης και δεν υπάρχουν άλλα αντικείμενα καλύτερα από αυτά. Για τον υπολογισμό των Skyline σημείων πραγματοποιείται μια σειρά από συγκρίσεις μεταξύ των σημείων με τη βοήθεια των Skyline Επερωτήσεων.

Πιο συγκεκριμένα, για ένα δεδομένο σετ από σημεία p1, p2, .. pN, η skyline επερώτηση επιστρέφει ένα σύνολο σημείων P, τέτοια ώστε να μην υπάρχει κανένα άλλο σημείο στο σύνολο των αρχικών δεδομένων που να επικρατεί έναντι των σημείων αυτών, δηλαδή των σημείων που ανήκουν στο σύνολο P.

Επικράτηση Σημείου σημαίνει το εξής: ένα σημείο pi επικρατεί ενός άλλου σημείου pj αν και μόνο αν η τιμή της συντεταγμένης του pi σε οποιοδήποτε άξονα δεν είναι μεγαλύτερη από την αντίστοιχη τιμή της συντεταγμένης του σημείου pj ή η τιμή μιας τουλάχιστον συντεταγμένης του pi να είναι μικρότερη από την τιμή της αντίστοιχης συντεταγμένης του σημείου pj.

Για παράδειγμα, έστω ότι υπάρχει ένα σύνολο σημείων, όπως φαίνεται στο παρακάτω σχήμα, το οποίο περιέχει πληροφορίες για ξενοδοχεία και κάθε τελεία αντιστοιχεί σ' ένα ξενοδοχείο. Δηλαδή, απεικονίζεται η απόσταση από την παραλία και η τιμή για κάθε σημείο.

3.4.7 Skyline επεξεργασία στο MapReduce

Σύμφωνα με τους K.Mullesgaard et al. προηγούμενες εργασίες έχουν παρουσιάσει αρκετούς τρόπους να εκτελεστούν οι skyline επερωτήσεις με τη χρήση του MapReduce, χωρίς όμως να γίνεται χρήση των πλεονεκτημάτων της παράλληλης επεξεργασίας αφού ένα σημαντικό μέρος της επερώτησης τρέχει πάντα σειριακά και χρησιμοποιείται ένας μόνο reducer για τον υπολογισμό των τελικών skyline σημείων.

Έτσι, προτείνουν ένα νέο αλγόριθμο, τον MapReduce - Grid partitioning Multiple Reducers Skyline (MR - GPMRS), ο οποίος τρέχει όλη την επερώτηση παράλληλα. Η βάση αυτού του αλγόριθμου είναι ο διαμοιρασμός του αρχικού συνόλου δεδομένων χρησιμοποιώντας μια grid τεχνική. Χρησιμοποιείται ένα bitstring, το οποίο καταγράφει ποιοι διαμερισμοί δεν είναι κενοί, πράγμα το οποίο καθιστά ικανή τη λήψη αποφάσεων βασισμένων στην κατανομή ολόκληρου του συνόλου δεδομένων. Ο τρόπος που συνδυάζεται η λειτουργία της επικράτειας σημείων έναντι άλλων στις skyline επερωτήσεις με το διαμοιρασμό χώρου με βάση grid τεχνικές, επιτρέπει το διαμοιρασμό των δεδομένων σε μέρη τα οποία μπορούν να επεξεργαστούν ανεξάρτητα το ένα από το άλλο. Αυτό σημαίνει ότι αυτά τα μέρη δεδομένων μπορούν να επεξεργαστούν από διαφορετικούς reducers. Αυτό σημαίνει ότι ο MR-GPMRS λειτουργεί πολύ καλά για μεγάλα σετ δεδομένων και μεγάλες συστάδες. Με τα πειράματά τους, δείχνουν ότι ο MR-GPMRS τρέχει αρκετές φορές καλύτερα από τους αντίστοιχους αλγόριθμους σε μεγάλα σετ δεδομένων. Ωστόσο, το γεγονός ότι οι reducers πρέπει να λαμβάνουν αντίγραφα των διαμερίσεις αυξάνει σημαντικά το κόστος επικοινωνίας. Η πειραματική μελέτη τους, δείχνει ότι η χρήση πολλαπλών reducers δεν είναι η καλύτερη επιλογή όταν το skyline ποσοστό είναι χαμηλό. ·τσι, για τη βελτιστοποίηση του αλγόριθμου για οποιοδήποτε σύνολο δεδομένων, είναι απαραίτητο να βρεθεί τρόπος, ο MR-GPMRS να επιλέγει έξυπνα τον αριθμό των reducers που θα χρησιμοποιήσει.

Οι L.Chen et al. παρουσιάζουν μια νέα MapReduce Skyline μέθοδο για παράλληλη επεξεργασία των skyline επερωτήσεων. Η νέα αυτή μέθοδος μειώνει σημαντικά το χρόνο της φάσης του Reduce εξαιτίας του περιορισμού των περιττών υπολογισμών για την ύπαρξη dominated σημείων. Τα πειράματα που διεξήγαν στο Hadoop έδειξαν ότι η μέθοδός τους λειτουργεί το ίδιο καλά με την αύξηση των διαστάσεων και την πληθικότητα των δεδομένων. Επίσης, έδειξαν ότι η νέα angular partitioned

MapReduce μέθοδος τρέχει 1.7 και 2.3 φορές γρηγορότερα από τις dimensional και grid partitioning μεθόδους.

Οι B.Zhang et al. ορίζουν το πρόβλημα της επεξεργασίας skyline επερωτήσεων στο MapReduce framework. Βασιζόμενοι σε διάφορες στρατηγικές διαμερισμού των δεδομένων, αναπτύσσουν τρείς αλγόριθμους για τον υπολογισμό skyline στο MapReduce: MapReduce based BNL (MR-BNL), MapReduce based SFS (MR-SFS) και MapReduce based Bitmap (MR-Bitmap). Εκτεταμένα πειράματα έλαβαν χώρα για την αξιολόγηση και τη σύγκριση των αλγορίθμων αυτών υπό διαφορετικές κάθε φορά συνθήκες, όπως κατανομή των δεδομένων, διαστάσεις, μέγεθος buffer και μέγεθος συστάδας. Τα πειράματα έδειξαν ότι οι αλγόριθμοι MR- BNL και MR-SFS λειτουργούν καλά στις περισσότερες περιπτώσεις ενώ αντιμετωπίζουν προβλήματα με τις διαστάσεις σε παράλληλα περιβάλλοντα. Ο αλγόριθμος MR-Bitmap λειτουργεί το ίδιο καλά ανεξάρτητα από τις κατανομές των δεδομένων, ειδικά όταν το bitmap ταιριάζει ακριβώς στην μνήμη ενός μόνο κόμβου.

Στη συγκεκριμένη μελέτη ο skyline αλγόριθμος που βασίζετε στη μέθοδο Map Reduce που εξετάζουμε είναι δικής μας υλοποίησης και είναι ποιο απλουστευμένος στην υλοποίηση του σε σχέση με τους παραπάνω αλγορίθμους. Παρακάτω θα δούμε εκτενέστερα τη λειτουργία του αλγορίθμου.

3.5 Αναλύοντας τα Δεδομένα και Σχεδιάζοντας το Πρόβλημα

Για την επίτευξη των στόχων και σκοπών της παρούσας εργασίας χρειάστηκε να γίνει μια σχεδίαση σε επίπεδο αρχιτεκτονικής αλλά και κώδικα, έτσι ώστε οι τρεις φάσεις της παράλληλης επεξεργασίας των skyline επερωτήσεων, δηλαδή διαμοιρασμός του χώρου, τοπικός υπολογισμός των skyline σημείων και συγχώνευση αυτών, να χωριστούν σε MapReduce jobs. Έτσι λοιπόν, ο κώδικας σχεδιάστηκε έτσι ώστε να εκτελείται σε δυο jobs, με σκοπό να εκμεταλλευτούν οι δυνατότητες που προσφέρει έτσι κι αλλιώς το MapReduce, όπως είναι η παραγωγή διαμερίσεων και η συγχώνευση των σημείων στην μέθοδο reduce().

Έτσι, χρειάστηκε μόνο να οριστεί ο τρόπος που θα εκτελεστεί ο διαμερισμός του χώρου και να καθοριστούν τα κριτήρια σύμφωνα με τα οποία τα σημεία θα πάνε στις αντίστοιχες διαμερίσεις. Με αυτό τον τρόπο διευκολύνθηκε σε σημαντικό βαθμό η ανάπτυξη του κώδικα. Ωστόσο, πριν από την εκτέλεση του κώδικα θα πρέπει να δημιουργηθούν τα αρχικά αρχεία ή το αρχείο προς επεξεργασία με τη βοήθεια

γεννητριών παραγωγής τυχαίων αριθμών. Επίσης, για τη σωστή εκτέλεση του κώδικα χρειάζεται να γίνουν κάποιες ρυθμίσεις σε μεταβλητές που χρησιμοποιούνται από τον κώδικα. Οι μεταβλητές αυτές μπορεί να αφορούν στο framework του Hadoop, στον κώδικα ή και στο σύστημα αρχείων του Hadoop (HDFS).

Σε επίπεδο λειτουργειών του αλγοριθμου, μπωρουμε να τον χωρισουμε σε γενικες γραμμες σε τρεις βασικες λειτουργιες. Όπου πρώτη είναι η λειτουργία της γεννήτριας των τυχαίων σημείων, η δεύτερη λειτουργεία είναι η ταξινόμηση των σημείων κατά αύξουσα σειρά ανά στήλη, ενώ για την τρίτη και τελευταία φάση του υπολογισμού των skyline σημείων δημιουργήθηκε μια επιπλέων λειτουργία. Κάθε μια λειτουργία που υλοποιεί το διαμοιρασμό του χώρου μετά την ολοκλήρωσή της και πριν την έξοδο από το σύστημα, καλεί την λειτουργία που υπολογίζει εντέλει τα skyline σημεία, χρησιμοποιώντας την μέθοδο του δεσίματος των MapReduce jobs (chaining MR jobs), με την οποία τα jobs φαίνεται να τρέχουν σαν ένα job αποκρύπτοντας από το χρήστη τις τεχνικές λεπτομέρειες. Άρα, η έξοδος της πρώτης λειτουργίας είναι η είσοδος της δεύτερης.

Όλες οι λειτουργίες του κώδικα είναι δομημένες σύμφωνα με το MapReduce πρότυπο όπως έχει περιγραφεί σε προηγούμενο κεφάλαιο. Πιο συγκεκριμένα, όλες οι λειτουργίες περιλαμβάνουν κλάσεις Mapper, Partitioner και Reducer οι οποίες εκτελούνται η μια μετά την άλλη με τη σειρά που αναφέρονται. Μέσα στις κλάσεις Mapper και Reducer υλοποιούνται οι μέθοδοι map() και reduce(), οι οποίες παίρνουν ορίσματα συγκεκριμένου τύπου. Όπως έχει αναφερθεί και νωρίτερα η είσοδος για το MapReduce είναι μια λίστα από ζεύγη (key1, value1) συγκεκριμένου τύπου, οι οποίοι θα αναλυθούν στο επόμενο κεφάλαιο.

3.6 Παραγωγή Αρχείων Τυχαίων Αριθμών

Τα δεδομένα τα οποία διαχειρίζονται από τον κώδικα είναι πολυδιάστατα, όπως περιγράφεται και στο κεφάλαιο 2. Οι βάσεις δεδομένων περιέχουν εγγραφές οι οποίες αποτελούνται από πολλές διαστάσεις. Στο παράδειγμα των ξενοδοχείων, το οποίο είναι παράδειγμα με δεδομένα δυο διαστάσεων, η τιμή του δωματίου αποτελεί τη μια διάσταση και η απόστασή του από τη θάλασσα την άλλη. Σ' ένα σύστημα αξόνων, κάθε διάσταση αναπαρίσταται σ' έναν άξονα. Στη συγκεκριμένη περίπτωση, το πείραμα μας διεξάγετε με τρία διαφορετικά dataset. Έτσι δημιουργούνται και επεξεργάζονται δεδομένα τριών διαφορετικών datasets, το Dataset.1 αποτελείτε από

δυο άξονες x, y, το Dataset.2 αποτελείτε από έξι άξονες x,y,z,X,Y,Z και το Dataset.3 αποτελείτε από δέκα άξονες x,y,z,X,Y,Z,a,b,c,d.

Στην παρούσα εργασία, η παραγωγή του αρχικού αρχείου προς επεξεργασία γίνεται από γεννήτριες τυχαίων αριθμών η οποία καλείτε μέσα από την εντολή «sample.int» μέσα στον αλγόριθμο, τις οποίες διαστάσεις τις έχουμε ήδη ορίσει στον αλγόριθμο μας, οι τυχαίες τιμές ξεκινούν από το 1 έως το 2000. Επίσης, στην παραγωγή του αρχικού αρχείου μπορεί να καθοριστεί και η κατανομή των σημείων. Ενδεικτικά, παρατίθεται μέρος του αρχικού αρχείου προς επεξεργασία:

1099 1362 1 1636 818 1042 655 1024 780 972 1414 1919 270 1227 902 1981
1059 21 307 1893 1776 449 318 1836 516 1894 1728 1057 1718 569 1906 510
89 1526 1043 1640 133 724 313 861 409 616 945 1631 1646 218 333 1665 179
551 543 1191 1277 1696 1248 376 1331 462 320 975 1998 1822 1835 1853 905
39 871 1630 433 354 1702 1947 327 1267 146 1660 461 1607 650 1169 559 971

3.7 Το Σύστημα αρχείων του Hadoop (HDFS) και Ρύθμιση Παραμέτρων (Βιβλιοθήκες R)

Σημαντικό ρόλο στη σωστή εκτέλεση του κώδικα παίζει η ρύθμιση των παραμέτρων που χρησιμοποιούνται είτε για τη ρύθμιση του συστήματος αρχείων είτε για τη ρύθμιση των παραμέτρων που χρησιμοποιούνται από το framework του Hadoop. Μια σωστή και ακριβής ρύθμιση παραμέτρων εξασφαλίζει την εύρυθμη λειτουργία του κώδικα ενώ ταυτόχρονα ελαχιστοποιεί το περιθώριο σφάλματος.

3.7.1 Ρύθμιση παραμέτρων

Στην αρχή της εκτέλεσης του κώδικα και πριν την εκτέλεση του πρώτου Job, πραγματοποιείται η ρύθμιση των παραμέτρων των jobs του MapReduce καθώς και η αρχικοποίηση των παραμέτρων που χρησιμοποιούνται αργότερα στον κώδικα. Η ρύθμιση όλων των παραμέτρων πραγματοποιείται μέσα από ένα instance της κλάσης JobConf, όπως αυτή ορίζεται στις βιβλιοθήκες του hadoop-0.20. Ο χρήστης μπορεί να αρχικοποιήσει τις παραμέτρους του μέσα από ένα συγκεκριμένο αρχείο, το οποίο διαβάζεται πριν την εκτέλεση του βασικού κώδικα. Στο αρχείο αυτό, ο χρήστης μπορεί να παραμετροποιήσει τον αριθμό των διαστάσεων των σημείων, τον αριθμό των σημείων, την ανώτατη τιμή που μπορούν οι διαστάσεις, τον αριθμό των διαμερίσεων του χώρου, τα partitions δηλαδή, καθώς και οποιαδήποτε άλλη

παράμετρο θέλει να χρησιμοποιήσει για την εκτέλεση του κώδικα. Στην ουσία, μέσα από αυτό το αρχείο ο χρήστης θα έχει τη δυνατότητα να αλληλεπιδρά με τον κώδικα και ν' αλλάζει τις συνθήκες εκτέλεσής του εάν το επιθυμεί, χωρίς να επεξεργάζεται τον ίδιο τον κώδικα.

Επίσης, κατά τη διάρκεια της ρύθμισης των παραμέτρων και πριν την εκτέλεση του κώδικα, αρχικοποιούνται και κάποιες μεταβλητές του framework του Hadoop. Πιο αναλυτικά, ορίζεται από ποια κλάση θα δημιουργηθεί το εκτελέσιμο αρχείο, πόσα reduce tasks θα χρησιμοποιηθούν, το όνομα των κλάσεων που θα λειτουργήσουν ως Mapper, Partitioner και Reducer αντίστοιχα. Επίσης, ορίζεται ο τύπος του τελικού key και value, που στην προκειμένη περίπτωση θα είναι τύπου Text.

3.7.2 Σύστημα αρχείων του Hadoop

Η κατάλληλη ρύθμιση του συστήματος αρχείων του Hadoop (HDFS) αποτελεί μια πολύ σημαντική διαδικασία καθώς το HDFS είναι ένα κομμάτι με το οποίο αλληλεπιδρά συνεχώς ο κώδικας είτε για να διαβάσει αρχεία είτε για να γράψει αρχεία σε αυτό.

Όπως είναι αναμενόμενο, η ρύθμιση του συστήματος αρχείων πραγματοποιείται πριν την έναρξη της εκτέλεσης του κώδικα. Σε αυτή τη φάση λοιπόν, δημιουργούνται ή ορίζονται στο File System του Hadoop (HDFS) συγκεκριμένα μονοπάτια για κάθε κλάση, η ονομασία των οποίων καθορίζεται από το χρήστη. Συγκεκριμένα δημιουργείται από το ίδιο το framework ένα μονοπάτι εξόδου στο HDFS πριν ξεκινήσει η φάση του mapping, ενώ το μονοπάτι εισόδου δημιουργείται από το χρήστη μέσω command prompt, οπότε κατά τη διάρκεια της ρύθμισης των παραμέτρων ορίζεται μόνο τα ονόματά τους. Επίσης, ο χρήστης θα πρέπει να έχει 'ανεβάσει' στο μονοπάτι εισόδου του HDFS το αρχικό αρχείο προς επεξεργασία πριν την εκτέλεση του κώδικα.

3.7.3 Επεξεργασία Skyline επερωτήσεων

Πρόσφατα έχει παρατηρηθεί αυξημένο ενδιαφέρον για τα Skyline queries. Το Skyline ενός συνόλου σημείων ορίζεται ως τα σημεία που δεν κυριαρχούνται από κανένα άλλο σημείο. Ένα σημείο x κυριαρχεί ενός άλλου σημείου y αν το x δεν είναι χειρότερο σε καμία διάσταση (από αυτές που μας ενδιαφέρουν) από το y και είναι καλύτερο σε τουλάχιστον μια. Για παράδειγμα αν κάποιος ενδιαφέρεται για ένα ξενοδοχείο το οποίο να βρίσκεται κοντά στην παραλία και να είναι φθηνό, τότε το

Skyline θα περιέχει το σύνολο των απαντήσεων, το σύνολο αυτό δεν θα περιέχει ούτε το φθηνότερο ξενοδοχείο αν βρίσκεται πολύ μακριά ούτε και το κοντινότερο στην παραλία αν είναι πολύ ακριβό. Ήδη έχουνε προταθεί ικανοποιητικοί αλγόριθμοι για την εφαρμογή Skyline ερωτήσεων σε μια παραδοσιακή βάση. Δυστυχώς όμως οι περισσότεροι από αυτούς βασίζονται σε παραδοχές που δεν ισχύουν στα P2P συστήματα.

Μελετητές προτείνουν να επεκτείνουν την SQL έτσι ώστε να μπορεί να υποστηρίξει Skyline queries, παρουσιάζουν και μετρούν την αποδοτικότητα εναλλακτικών αλγόριθμων για την υλοποίηση της Skyline λειτουργίας, και δείχνουν πως αυτή η λειτουργία μπορεί να συνδυαστεί με άλλες λειτουργίες στη βάση (π.χ join και Top N). Εμεις βασιζόμενοι σε αυτές τις απόψεις καθώς και σε άλλων μελετητών θα προσπαθήσουμε να βελτίωσουμε της απόδοση του skyline στην εκτέλεση επερωτήσεων σε πολυδιάστατα δεδομένα με την χρήση του skyline αλγόριθμου.

4 Ανάλυση των Αλγορίθμων Skyline

4.1 Εισαγωγή

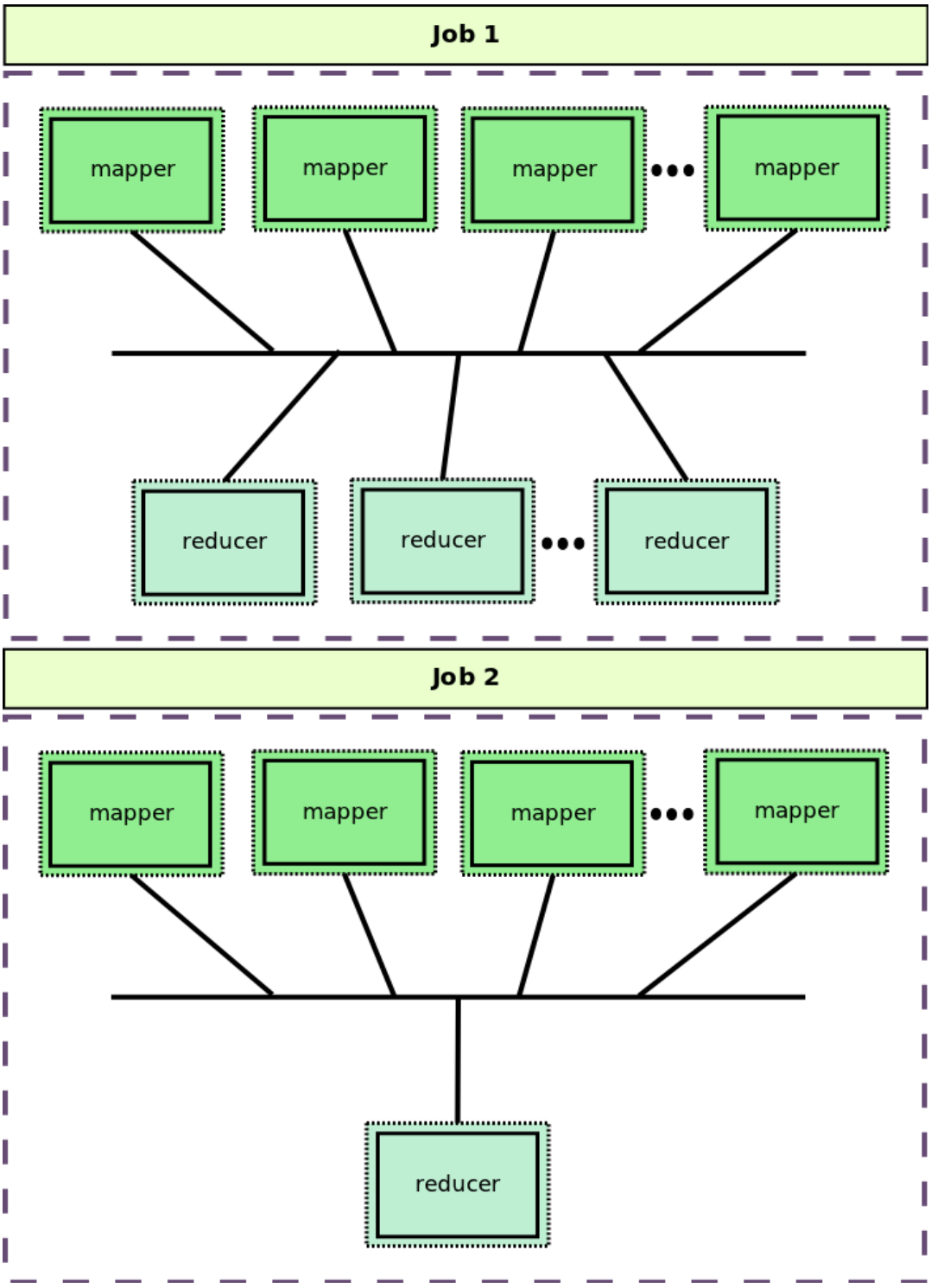
Σε αυτό το Κεφάλαιο αναλύονται κάποιοι βασικοί Skyline αλγόριθμοι, οι οποίοι λειτουργούν με βάση την μέθοδο του MapReduce. Οι αλγοριθμοι οι οποιοι αναλυονται είναι οι Skyline-Baseline, Skyline-Fifo, Skyline-Volume, Skyline-Sampling-Adapt-Fifo, Skyline-Sampling-Adapt-Volume, Skyline-Sampling-Baseline και Skyline-Sampling-Fifo. Τέλος γίνετε ιδιαίτερη αναφορά στον αλγόριθμο Block-Nested-Loop (BNL) οπού αποτέλεσε και την βάση για την ανάπτυξη του δικού μας αλγορίθμου.

4.1.1 Ο Αλγόριθμος Skyline-Baseline

Στη μέθοδο αυτή ο υπολογισμός της skyline συλλογής δεδομένων ολοκληρώνεται σε δύο MapReduce διαδικασίες. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στην εικόνα 1.

Στην πρώτη διεργασία, πολλοί Mappers διαβάζουν την συλλογή δεδομένων. Για κάθε διάνυσμα που δέχονται υπολογίζουν το άθροισμα των συντεταγμένων του και στη συνέχεια το προωθούν στους Reducers. Στη ενδιάμεση φάση, στη φάση του Shuffling χρησιμοποιείται η τεχνική Secondary Sorting έτσι ώστε τα δεδομένα που θα φτάσουν σε κάθε Reducer να είναι ταξινομημένα ως προς το άθροισμα συντεταγμένων για να μπορεί άμεσα να εκτελεστεί ο αλγόριθμος SFS. Στη φάση του Reduce, ο κάθε Reducer με βάση τον αλγόριθμο SFS υπολογίζει το skyline των τοπικών δεδομένων που δέχτηκε και ετσι η διαδικασία ολοκληρώνεται.

Στη δεύτερη διαδικασία, πολλοί Mappers διαβάζουν τα δεδομένα που προέκυψαν από πριν, δηλαδή τα τοπικά skyline σημεία. Όπως και πριν, υπολογίζουν το άθροισμα των συντεταγμένων τους και τα προωθούν στη φάση του Reduce. Στη φάση Shuffling με Secondary Sorting τα δεδομένα φτάνουν ταξινομημένα ως προς το άθροισμα των συντεταγμένων. Τέλος στη φάση του Reduce ένας τελικός Reducer εκτελεί τον αλγόριθμο SFS και υπολογίζει το ολικό skyline της συλλογής δεδομένων.



Εικόνα 1: Αλγόριθμος Skyline-Baseline με Map Reduce διεργασία

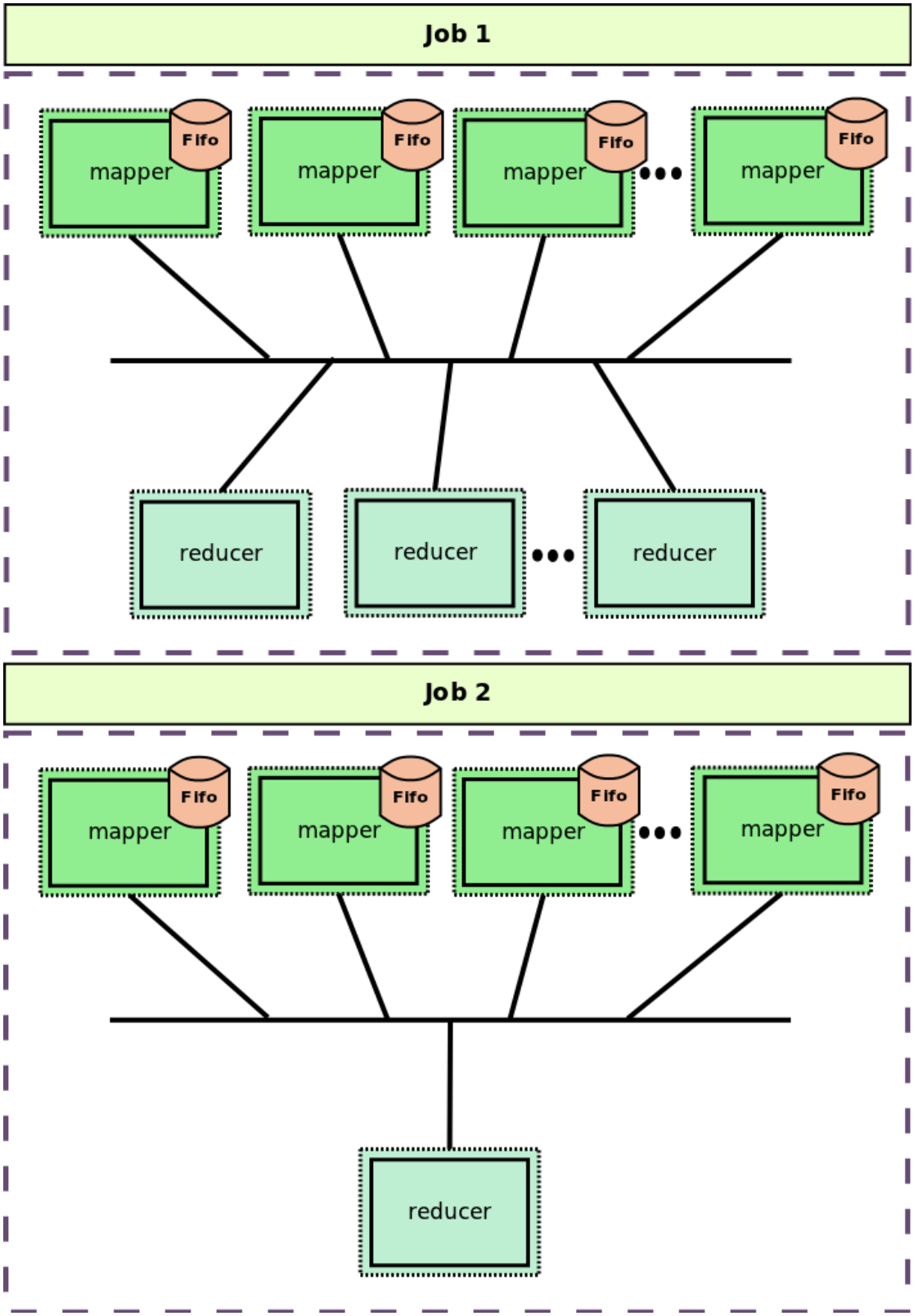
4.1.2 Ο Αλγόριθμος Skyline-Fifo

Στη μέθοδο αυτή ο υπολογισμός του skyline αλγορίθμου, ολοκληρώνεται σε δύο MapReduce διαδικασίες. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στην εικόνα 2.

Στην πρώτη διαδικασία, πολλοί Mappers διαβάζουν την συλλογή δεδομενων. Κάθε Mapper διαθέτει μία μνήμη fifo το μέγεθος της οποίας ορίζεται από τον χρήστη. Καθώς σταδιακά ένα προς ένα τα διανύσματα καταφθάνουν ως είσοδος, αυτά δεν συλλέγονται κατευθείαν αλλά περνούν από μία ενδιάμεση φάση. Συγκεκριμένα, το πρώτο διάνυσμα που θα διαβαστεί θα εισαχθεί στη μνήμη fifo. Το επόμενο που θα έρθει θα συγκριθεί με το υπάρχον και αν κυριαρχεί σε αυτό τότε το υπάρχον θα σβηστεί και το νέο θα εισαχθεί, αν είναι ανεξάρτητο θα εισαχθεί στο υπόλοιπο της μνήμης και αν κυριαρχείται τότε απορρίπτεται και η επεξεργασία του σταματά εκεί. Αυτή η διαδικασία επαναλαμβάνεται καθώς τα διανύσματα έρχονται ένα προς ένα, ετσι κάθε νέο διάνυσμα συγκρίνεται με όλα τα υπόλοιπα στη μνήμη με το ίδιο σκεπτικό, έως ότου η μνήμη γεμίσει.

Μέχρι στιγμής κανένα σημείο δεν έχει προωθηθεί στους Reducers. Όταν δεν θα υπάρχει άλλος χώρος στη μνήμη fifo ένα σημείο αφαιρείται από αυτήν και συλλέγεται για την φάση του Reduce. Το σημείο που επιλέγεται είναι το πιο παλιό από τα υπόλοιπα, ενώ υπάρχει η λογίκη “First In First Out”. Με αυτό το σκεπτικό το κέρδος είναι σημαντικό καθώς ένα κομμάτι της συλλογής δεδομένων απορρίπτεται και δεν ταξιδεύει πάνω στο δίκτυο κατά την φάση του Shuffling. Για τα σημεία τα οποία συλλέγονται, πριν προωθηθούν υπολογίζεται το άθροισμα των συντεταγμένων τους. Στη συνέχεια με την τεχνική του Secondary Sorting φτάνουν στους Reducers ταξινομημένα ως προς αυτό το άθροισμα για να ξεκινήσει η φάση Reduce. Σε αυτήν, η εκτέλεση είναι ομοιότυπη με εκείνη του αλγορίθμου Skyline-Baseline. Εκτελείται δηλαδή ο αλγόριθμος SFS και υπολογίζονται τα τοπικά skylines.

Η δεύτερη διαδικασία είναι ίδια με την πρώτη με μόνη διαφορά ότι στη φάση Reduce τα δεδομένα συλλέγονται σε έναν μόνο Reducer ώστε να υπολογιστεί το ολικό skyline. Τονίζεται πως ως μέγεθος μνήμης fifo παραμένει το ίδιο μέγεθος που ορίστηκε και για την πρώτη διαδικασία.



Εικόνα 2: Αλγόριθμος Skyline-Fifo με Map Reduce διεργασία

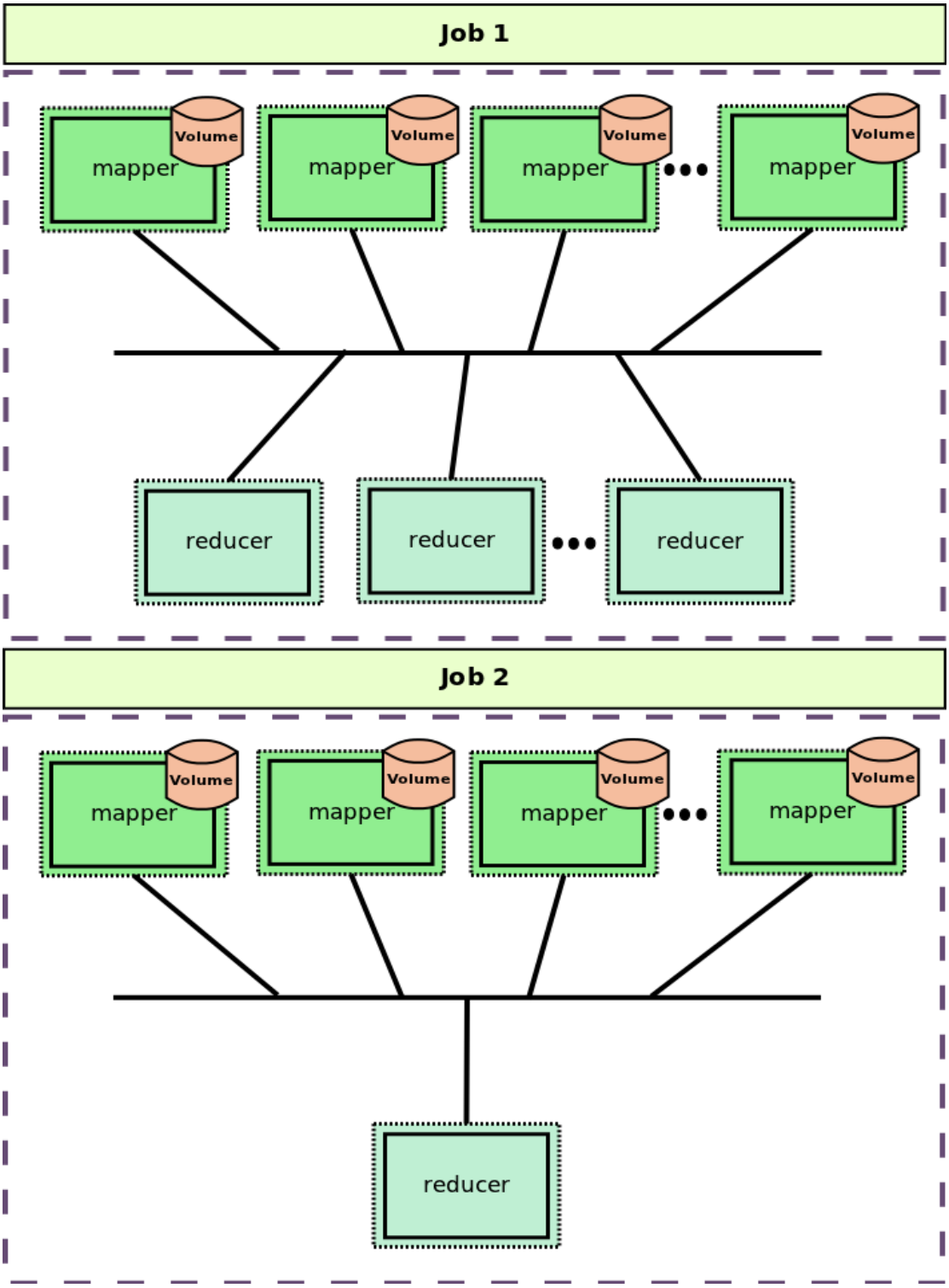
4.1.3 Ο Αλγόριθμος Skyline-Volume

Στη μέθοδο αυτή ο υπολογισμός του skyline της συλλογής δεδομένων, ολοκληρώνεται σε δύο MapReduce διαδικασίες με σκεπτικό ανάλογο με αυτό του αλγορίθμου Skyline-Fifo. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στο παρακάτω στην εικόνα 3.

Στην πρώτη διαδικασία, πολλοί Mappers διαβάζουν την συλλογή δεδομενων. Κάθε Mapper διαθέτει μία μνήμη volume το μέγεθος της οποίας ορίζεται από τον χρήστη. Καθώς σταδιακά ένα προς ένα τα διανύσματα καταφθάνουν ως είσοδος, αυτά δεν συλλέγονται κατευθείαν αλλά περνούν από μία ενδιάμεση φάση. Όπως στον αλγόριθμο Skyline-Fifo έτσι και εδώ υπάρχει μία ενδιάμεση μνήμη, η μνήμη volume το μέγεθος της οποίας δίνεται από τον χρήστη. Όταν ένα νέο στοιχείο καταφθάνει, συγκρίνεται με όλα τα υπόλοιπα στοιχεία που βρίσκονται σε αυτήν και αν κυριαρχεί σε κάποια διανύσματα τότε αυτά σβήνονται. Αν είναι ανεξάρτητο από αυτά τότε εισάγεται στη μνήμη και τέλος αν κυριαρχείται το ίδιο από άλλο στοιχείο, τότε, απορρίπτεται. Η διαφορά αυτού του αλγορίθμου από τον Skyline-Fifo είναι η οργάνωση των στοιχείων μέσα στην μνήμη. Εδώ δεν επικρατεί το σκεπτικό “First In First Out” όταν η μνήμη γεμίσει και ένα στοιχείο πρέπει να προωθηθεί στη φάση Reduce. Αλλά, τα στοιχεία βρίσκονται διαρκώς ταξινομημένα ως προς το volume τους. Δηλαδή ως προς το εμβαδό της περιοχής στην οποία κυριαρχούν. Πρώτο στοιχείο στη μνήμη είναι εκείνο με το μεγαλύτερο volume και τελευταίο εκείνο με το μικρότερο. Ένα στοιχείο με μεγάλο volume έχει σημαντικές πιθανότητες να κυριαρχήσει σε ένα ερχόμενο στοιχείο και έτσι αυτό να απορριφθεί γρήγορα.

Έχοντας τα στοιχεία με μεγάλο volume “ψηλά” μέσα στη μνήμη volume, μειώνουμε τις απαιτούμενες συγκρίσεις που χρειάζονται για τα σημεία που κυριαρχούνται ώστε να τα απορρίψουμε. Όταν η μνήμη γεμίσει και ένα στοιχείο πρέπει να αφαιρεθεί, θα επιλεχτεί εκείνο με το μικρότερο volume και θα συνεχίσει τη πορεία του προς την φάση Reduce. Με αυτό το σκεπτικό η μνήμη με την πάροδο του χρόνου έχει όλο και πιο “δυνατά” στοιχεία που θα διώχνουν όλο και περισσότερα από εκείνα που έρχονται. Στην συνέχεια το σκεπτικό παραμένει το ίδιο με εκείνο τον υπόλοιπων αλγορίθμων. Για τα στοιχεία υπολογίζεται το άθροισμα των συντεταγμένων τους και προωθούνται στη φάση Shuffling. Με την τεχνική Secondary Sorting φτάνουν ταξινομημένα στους Reducers και εκεί εκτελείται ο SFS για τον υπολογισμό των τοπικών skylines.

Η δεύτερη διαδικασία είναι η ίδια με την πρώτη. Χρησιμοποιείται με τον ίδιο τρόπο η μνήμη volume και έχει το ίδιο μέγεθος, ουσιαστικά αυτό που δόθηκε εξ αρχής από τον χρήστη. Η είσοδος σε αυτή τη διαδικασία είναι η έξοδος της προηγούμενης, δηλαδή, τα τοπικά skylines και η έξοδος είναι το ολικό skyline. Για αυτό στη φάση Reduce ενεργοποιείται μόνο ένας Reducer για να εκτελέσει τον τελικό SFS.



Εικόνα 3: Αλγόριθμος Skyline-Volume με Map Reduce διεργασία

Το κέρδος αυτού του αλγορίθμου σε σχέση με τον Skyline-Baseline αλγόριθμο είναι πως με τις συγκρίσεις που εκτελούνται στη φάση Map με την μνήμη volume απορρίπτονται σημεία που έτσι και αλλιώς δεν ανήκουν στο τελικό skyline αποτέλεσμα και δεν περνούν από τη φάση του Shuffling κάτι που θα σήμαινε επιπλέον κόστος μεταφοράς στο δίκτυο καθώς και περισσότερα I/Os.

4.1.4 Ο Αλγόριθμος Skyline-Sampling-Adapt-Fifo

Στον αλγόριθμο αυτό καθώς και στους υπόλοιπους που ακολουθούν εισάγεται η έννοια της δειγματοληψίας. Στόχος αυτής είναι να προσθέσουμε ένα πέρασμα της συλλογής στον αλγόριθμό μας με αντάλλαγμα να προσδιορίσουμε τη μορφή της συλλογής και τα χαρακτηριστικά της π.χ. το μέγεθος του skyline της σε σχέση με το πλήθος των στοιχείων της, ώστε να κάνουμε καλύτερες επιλογές σε παραμέτρους που χρησιμοποιούμε για την αποδοτικότερη λειτουργία στο υπόλοιπο του αλγορίθμου. Λόγω του επιπλέον περάσματος που εκτελούμε στη συλλογή δεδομένων, προστίθεται μία έξτρα διαδικασία MapReduce στον αλγόριθμο. Συγκεκριμένα για τον Skyline-Sampling-Adapt-Fifo το ολικό skyline υπολογίζεται σε τρείς διαδικασίες όπως παρουσιάζεται και στο παρακάτω εικόνα 4.

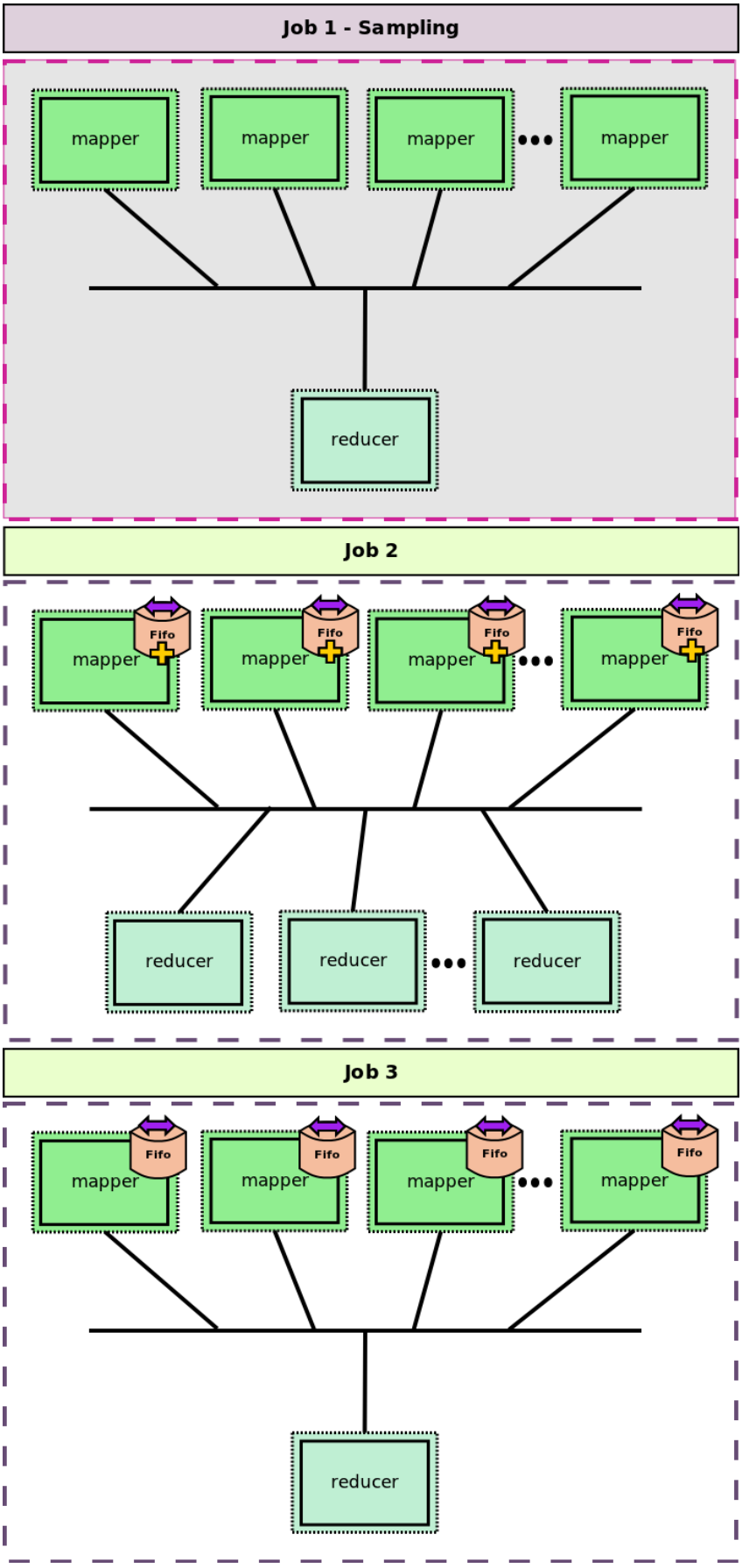
Στην πρώτη διαδικασία πολλοί Mappers διαβάζουν τη συλλογή δεδομένων. Για κάθε διάνυσμα που επεξεργάζονται, με ορισμένη πιθανότητα, η οποία δίνεται από τον χρήστη, επιλέγουν αν θα κρατήσουν αυτό το διάνυσμα ή όχι. Για τα διανύσματα που κρατούνται, υπολογίζεται το άθροισμα των συντεταγμένων τους και προωθούνται στη συνέχεια στη φάση του Shuffling. Εκεί με την τεχνική Secondary Sorting φτάνουν ταξινομημένα σε έναν Reducer όπου εκτελείται ο αλγόριθμος SFS και υπολογίζεται το skyline του δείγματος. Επιπλέον, υπολογίζεται με εκτίμηση το ποσοστό της συλλογής που αποτελεί το skyline. Τόσο αυτό το εκτιμώμενο ποσοστό όσο και το skyline του δείγματος αποθηκεύονται στο κατανεμημένο σύστημα αρχείων του Hadoop.

Στη δεύτερη διαδικασία χρησιμοποιείται η μνήμη fifo όπως και στον αλγόριθμο Skyline-Fifo. Εδώ όμως το σκεπτικό διαφέρει διότι διαθέτουμε επιπλέον πληροφορίες που αποκτήθηκαν από τη διαδικασία της δειγματοληψίας. Έτσι τώρα το μέγεθος της μνήμης fifo δεν ορίζεται πια από τον χρήστη αλλά προσδιορίζεται ανάλογα από το πόσο μεγάλο είναι το skyline σε σχέση με το μέγεθος της συλλογής. Η μόνη παράμετρος που ορίζεται από τον χρήστη είναι το μέγιστο επιτρεπόμενο μέγεθος μνήμης που μπορεί να χρησιμοποιηθεί. Το ποσοστό αυτού που τελικά χρησιμοποιείται θα είναι ίσο με το ποσοστό του skyline στο μέγεθος της συλλογής που εκτιμήθηκε. Έτσι αν μια συλλογή έχει μικρό skyline αποτέλεσμα δεν χρειάζεται να "ξοδεύεται" πολλή μνήμη για αυτήν και να γίνονται πολλές συγκρίσεις ενώ όταν μια συλλογή έχει μεγάλο skyline αποτέλεσμα θα μπορεί να έχει καλή απόδοση καθώς

η μνήμη fifo ορίζεται στις ανάγκες της. Έτσι στη συνέχεια ο αλγόριθμος δουλεύει σε αυτή τη διαδικασία μαζί με την μνήμη fifo όπως και ο αλγόριθμος Skyline-fifo.

Το μέγεθος της μνήμης ορίστηκε όπως αναφέραμε. Για επιπλέον αποτελεσματικότητα και μεγαλύτερο κέρδος στις απαιτούμενες συγκρίσεις, κατά την εκκίνηση της δεύτερης διαδικασίας, σε κάθε Mapper η μνήμη fifo δεν είναι άδεια αλλά γεμάτη με τα σημεία του skyline του δείγματος. Αν τα σημεία αυτά είναι περισσότερα από όσα χωράει η μνήμη τότε κρατούνται εκείνα με το μικρότερο άθροισμα συντεταγμένων ενώ αν ισχύει το ανάποδο και ο χώρος την μνήμης είναι μεγαλύτερος, κρατούνται όλα. Έτσι καθώς οι Mappers διαβάζουν την είσοδο, το κάθε νέο διάνυσμα συγκρίνεται με στοιχεία που έχουν ήδη βρεθεί ως skyline ανάμεσα σε άλλα. Τονίζεται πως τα στοιχεία της εισόδου δεν αναμειγνύονται με εκείνα του skyline του δείγματος ώστε να μην δημιουργηθούν τυχόν διπλότυπα που δεν υπάρχουν πραγματικά στη συλλογή δεδομένων μας, αλλά το skyline του δείγματος χρησιμοποιείται μόνο ώστε να χρειαστούν λιγότερες συγκρίσεις και να αξιοποιηθεί η διαθέσιμη μνήμη με τον καλύτερο δυνατό τρόπο. Στη συνέχεια εφαρμόζεται το σκεπτικό fifo όπως και στον αλγόριθμο Skyline-Fifo. Για κάθε σημείο που συλλέγεται υπολογίζεται το άθροισμα των συντεταγμένων του, προωθείται στη φάση Shuffling και φτάνει με ταξινομημένο τρόπο ως προς το άθροισμα αυτό για να περάσει στη φάση Reduce. Εκεί με τον αλγόριθμο SFS υπολογίζονται τα τοπικά skylines στους Reducers τα οποία αποθηκεύονται για την επόμενη διαδικασία.

Η τρίτη διαδικασία είναι ομοιότυπη με τη δεύτερη διαδικασία του Skyline-Fifo με μόνη διαφορά ότι το μέγεθος της μνήμης fifo δεν δίνεται από τον χρήστη αλλά είναι αυτό το οποίο υπολογίστηκε βάση της δειγματοληψίας. Αναφέρεται πως σε αυτή τη φάση η μνήμη fifo κατά την εκκίνηση των Mappers είναι άδεια.



Εικόνα 4: Skyline-Sampling-Adapt-Fifo με Map Reduce διεργασία

4.1.5 Ο Αλγόριθμος Skyline-Sampling-Adapt-Volume

Στη μέθοδο αυτή ο υπολογισμός του skyline της συλλογής ολοκληρώνεται σε τρεις MapReduce διαδικασίες με σκεπτικό ανάλογο με αυτό του αλγορίθμου Skyline-Sampling-Adapt-Fifo. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στο παρακάτω στην εικόνα 5. Η μόνη διαφορά των δύο αυτών αλγορίθμων είναι ότι στη μνήμη που χρησιμοποιείται στους Mappers η οργάνωση δεν γίνεται βάση της λογικής fifo αλλά βάση του volume όπως και στον Skyline-Volume.

4.1.6 Ο Αλγόριθμος Skyline-Sampling-Baseline

Στη μέθοδο αυτή ο υπολογισμός του skyline της συλλογής ολοκληρώνεται σε τρεις MapReduce διαδικασίες με σκεπτικό ανάλογο με αυτό του αλγορίθμου Skyline-Sampling-Adapt-Fifo. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στην εικόνα 6.

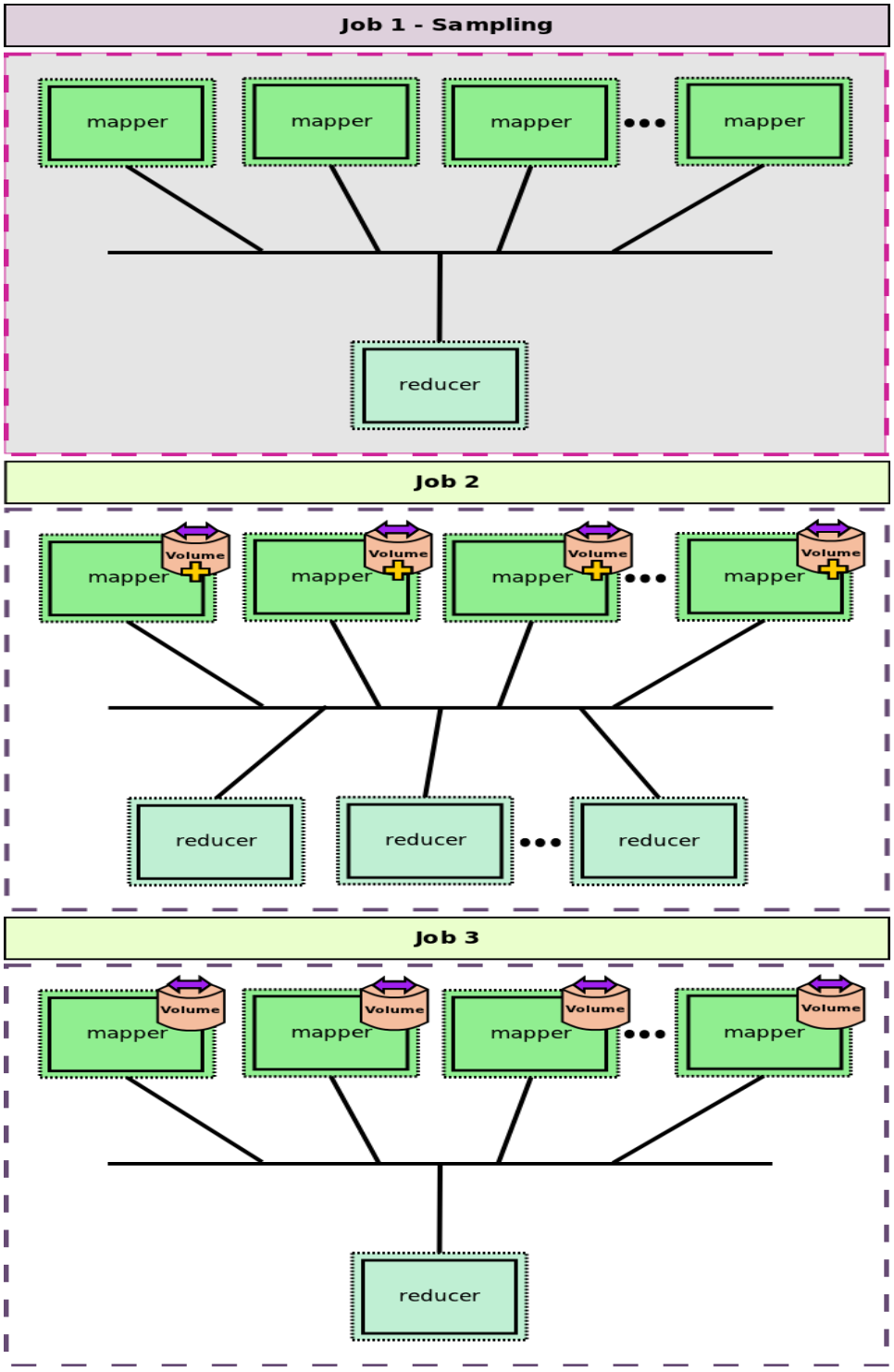
Στη πρώτη διαδικασία εκτελείται η δειγματοληψία όπως αυτή περιγράφηκε στον αλγόριθμο Skyline-Sampling-Adapt-Fifo. Η μόνη διαφορά εδώ είναι πως δεν χρειάζεται να εκτιμηθεί το ποσοστό της συλλογής δεδομένων που αποτελεί το skyline αλλά απλά να βρεθεί το skyline του δείγματος. Η δεύτερη διαδικασία είναι όπως η πρώτη του αλγορίθμου Skyline-Baseline. Υπάρχει μόνο μία διαφορά, εδώ αξιοποιείται το skyline του δείγματος και λειτουργεί σαν φίλτρο. Κάθε διάνυσμα που έρχεται ως είσοδος ελέγχεται με κάθε σημείο του. Αν κυριαρχηθεί από κάποιο δεν εξετάζεται περισσότερο. Έτσι ένα σημαντικό κομμάτι της συλλογής απορρίπτεται από την αρχή. Η τρίτη διαδικασία είναι ομοιότυπη με την δεύτερη του αλγορίθμου Skyline-Baseline.

4.1.7 Ο Αλγόριθμος Skyline-Sampling-Fifo

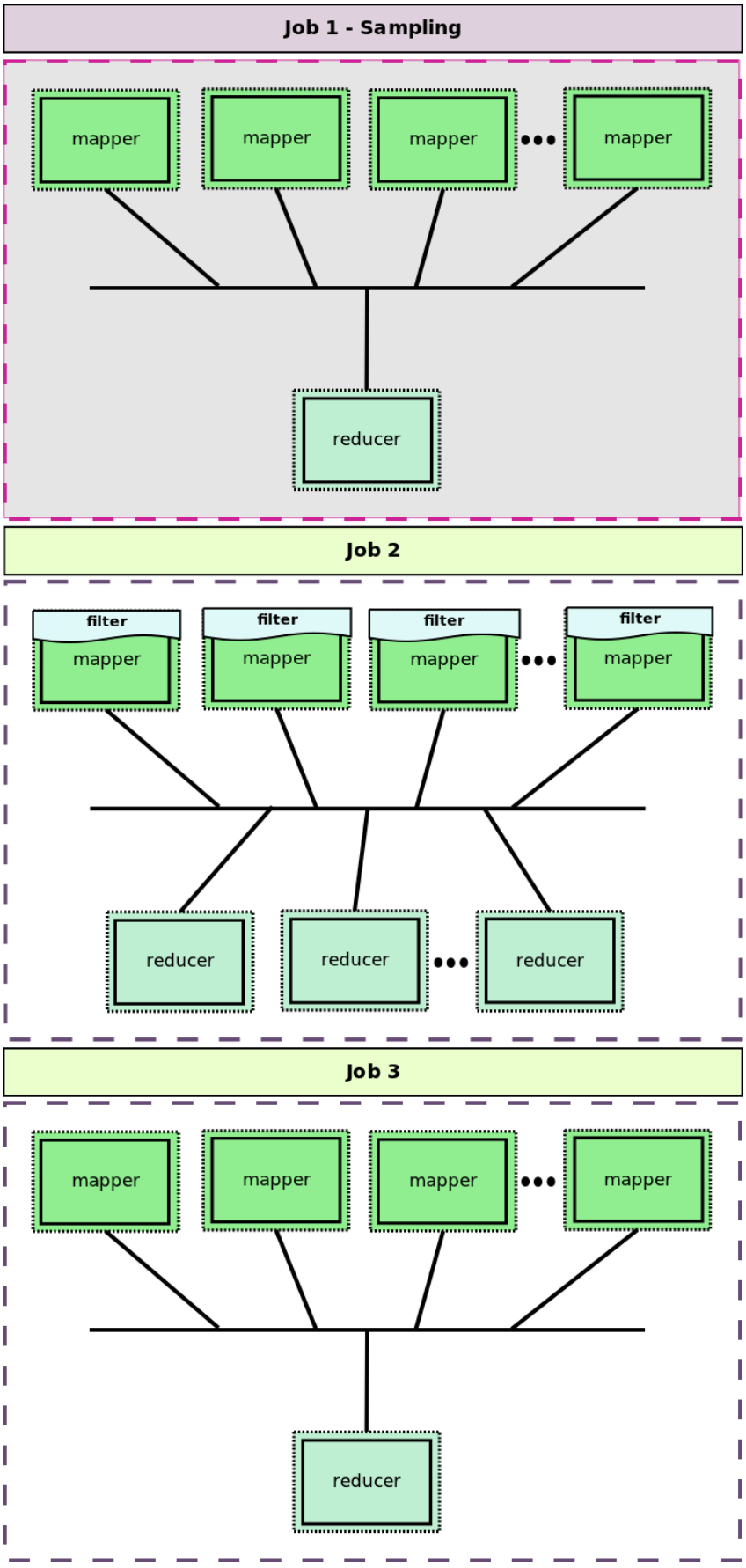
Στη μέθοδο αυτή ο υπολογισμός του skyline της συλλογής ολοκληρώνεται σε τρεις MapReduce διαδικασίες με σκεπτικό ανάλογο με αυτό του αλγορίθμου Skyline-Sampling-Adapt-Fifo. Μια οπτική αναπαράσταση του αλγορίθμου φαίνεται στην εικόνα 7.

Στη πρώτη διαδικασία εκτελείται η δειγματοληψία όπως αυτή περιγράφηκε στον αλγόριθμο Skyline-Sampling-Adapt-Fifo με μόνη διαφορά ότι εδώ δεν χρειάζεται να εκτιμηθεί το ποσοστό της συλλογής που αποτελεί το skyline αλλά απλά να βρεθεί το skyline του δείγματος μας. Η δεύτερη διαδικασία είναι όπως η πρώτη του αλγορίθμου Skyline-Fifo με μία επιπρόσθετη λειτουργία. Εδώ αξιοποιείται το skyline του

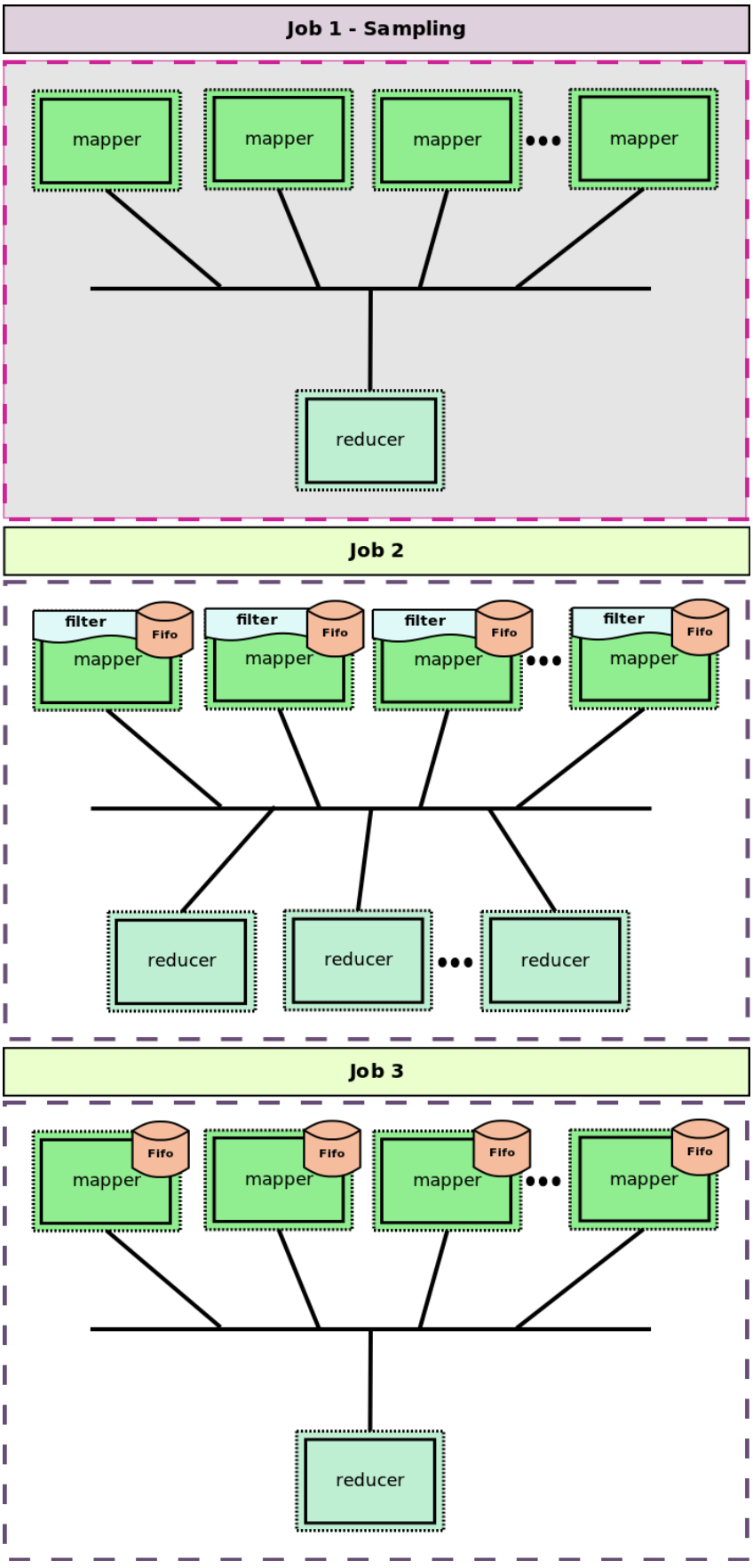
δείγματος μας και λειτουργεί σαν φίλτρο. Κάθε διάνυσμα που έρχεται ως είσοδος ελέγχεται με κάθε σημείο του. Αν κυριαρχηθεί από κάποιο δεν εξετάζεται περισσότερο. Έτσι ένα σημαντικό κομμάτι της συλλογής απορρίπτεται από την αρχή. Η τρίτη διαδικασία είναι ίδια με την δεύτερη του αλγορίθμου Skyline-Fifo βάση της οποίας προκύπτει και το ολικό skyline της συλλογής δεδομένων.



Εικόνα 5: Skyline-Sampling-Adapt-Volume με Map Reduce διεργασία



Εικόνα 6: Skyline-Sampling-Baseline με Map Reduce διεργασία



Εικόνα 7: Skyline-Sampling-Fifo με Map Reduce διεργασία

4.2 Ο Αλγόριθμος Block Nested Loops (BNL)

Ειδική μνεία θα κάνουμε για έναν από τους πρώτους αλγορίθμους που χρησιμοποιήθηκαν για τον υπολογισμό της κορυφογραμμής ενός συνόλου δεδομένων είναι ο Block Nested Loops Algorithm (BNL), ο οποίος αναφέρεται σε αρκετά επιστημονικά άρθρα. Η πιο χαρακτηριστική περιγραφή του δίνεται από τους Borzsonyi et al. (2001). Πρόκειται πρακτικά για έναν επαναληπτικό αλγόριθμο, ο οποίος σε κάθε του επανάληψη διαβάζει ένα στοιχείο εισόδου από το σύνολο των δεδομένων της εισόδου. Η βασική ιδέα του αλγορίθμου είναι η διατήρηση ενός χώρου στην κύρια μνήμη που περιέχει ασύγκριτες τιμες, σε σχέση με τα δεδομένα που δεν έχουν ακόμη διαβαστεί. Ο χώρος αυτός συνήθως αναφέρεται ως "παράθυρο". Προκύπτει εύκολα το συμπέρασμα ότι το παράθυρο αρχικά θα είναι άδειο και ότι με το τέλος της εκτέλεσης του αλγορίθμου, τα στοιχεία που θα έχουν απομείνει σε αυτό θα αποτελούν και τα στοιχεία της ζητούμενης κορυφογραμμής. Το συμπέρασμα που αναφέρθηκε παραπάνω είναι αποτέλεσμα των βημάτων που ακολουθεί ο BNL, τα οποία είναι τα εξής:

a) Όταν διαβάζεται ένα στοιχείο εισόδου, έστω p, τότε συγκρίνεται με όλες τις τιμές που περιέχονται στο παράθυρο και ανάλογα με το αποτέλεσμα της σύγκρισης προκύπτουν τρία ενδεχόμενα. Το p είτε τοποθετείται στο παράθυρο, είτε απορρίπτεται και δεν εξετάζεται ξανά, είτε τοποθετείται σε ένα προσωρινό αρχείο για να εξεταστεί ξανά σε κάποιο επόμενο πέρασμα του αλγορίθμου. Αυτές οι τρεις καταστάσεις προκύπτουν ως εξής:

i. Το στοιχείο p γίνεται "dominated" από κάποιο στοιχείο που βρίσκεται ήδη στο παράθυρο. Τότε το στοιχείο p είναι σίγουρο πως δεν αποτελεί μέρος της κορυφογραμμής. Συνεπώς, απορρίπτεται και δεν εξετάζεται ποτέ ξανά.

iii. Το στοιχείο p δεν γίνεται "dominated" από κάποιο άλλο στοιχείο του παραθύρου αλλά ταυτόχρονα δεν "κυριαρχεί" ("dominates") επί κάποιου άλλου στοιχείου του παραθύρου. Τότε, το στοιχείο p εισέρχεται στο παράθυρο, εφόσον υπάρχει διαθέσιμος χώρος. Σε διαφορετική περίπτωση, τοποθετείται σε ένα προσωρινό αρχείο. Οι τιμές που περιέχονται σε αυτό το αρχείο λαμβάνονται υπόψη σε κάποια επόμενη επανάληψη του αλγορίθμου.

b) Στο τέλος της κάθε επανάληψης, οι τιμές του παραθύρου, που έχουν συγκριθεί και με όλες τις τιμές του προσωρινού αρχείου, μπορούν να εξαχθούν από το παράθυρο, καθώς θα αποτελούν σίγουρα μέρος της κορυφογραμμής.

Όσον αφορά την πολυπλοκότητα του αλγορίθμου BNL, αυτή είναι της τάξεως O(n) στην καλύτερη περίπτωση (best case) και O(n2) στη χειρότερη (worst case). Προφανώς, ο BNL κρίνεται ιδανικός για περιπτώσεις όπου η κορυφογραμμή είναι μικρή.

4.2.1 The Skyline Operator

Στον παρακάτω πίνακα 1 βλέπουμε σε μορφή ψευδοκώδικα τον αλγόριθμο BNL. Οπού η παρακάτω μορφή βρίσκει εφαρμογή σε γλωσσά T-SQL.

```
 1:  M – Input of dimensional points
 2:  R output of dimensional points
 3:  T temporary file for dimensional points
 4:  S set of dimensional points
 5:  p < q point p is dominated by point q
 6:
 7:  function SkylineBNL(Input)
 8:  {
 9:  Output = 0;
10:  Temp = 0;
11:  Set=0;
12:  CountIn = 0;
13:  CountOut = 0;
14:
15:  while(input)
16:  do
17:  {
18:  foreach p in Set
19:  {
20:  if(TimeStamp(p)=countin
21:  {
22:  save(Output, p), Release(p)
23:  }
24:  Load(Input,p)
25:  TimeStamp(p) = CountOut;
26:  CountIn++
27:  foreach(q in Set)
28:  {
29:  if(p>q) release (p) break;
30:  if(p<q) then reelase(q);
31:  }
32:  if(MemoryAvailable)
33:  {
34:  save(Temp,p), release(p)
35:  CountOut ++;
36:  }
37:  //keep it running to iterate through entire sets.
38:  if(Main == empty)
39:  {
40:  Main = Temp;
```

```
41:  Temp=0;CountIn=0;CountOut=0;
42:  }
43:  }
44:  }
45:
46:  //flush
47:  foreach(p in Set)
48:  {
49:  save(Output,p), release(p)
50:  }
51:
52:  return Output;
```

Πίνακας 1: Ανάπτυξη σε ψευδοκώδικα του Block – Nested Loop αλγορίθμου. (http://codingbliss.com/page/3/, 2012)

4.2.2 Μεταφορά του BNL σε περιβάλλον MapReduce (MR - BNL)

Η χρήση του BNL σε περιβάλλον MapReduce (MR - BNL) βασίζεται στα ίδια βήματα που αναφέρθηκαν παραπάνω, με μια διαφορετική βέβαια φιλοσοφία υλοποίησης ώστε να ανταποκρίνεται στα πρότυπα που ορίζει το μοντέλο του MapReduce. Όπως φαίνεται και στον ψευδοκώδικα, η διαδικασία χωρίζεται σε δύο Jobs, την Division Job και την Merging Job, με την κάθε Job να διαθέτει από έναν Mapper και έναν Reducer.

Αρχικά, στην Division Job, ο DivisionMapper "διαβάζει" ένα προς ένα τα σημεία του συνόλου δεδομένων εισόδου. Για κάθε σημείο υπολογίζει την subspace flg και εξάγει ένα ζεύγος με την subspace flg ως κλειδί (key) και το σημείο ως τιμή (value). Εδώ, πρέπει να αναφερθεί ότι οι subspace flgs δείχνουν σε ποια υποπεριοχή ανήκει κάθε σημείο, όπως αυτές ορίζονται από τις διαμέσους κάθε διάστασης. Για να γίνει πιο εύκολα αντιληπτό, εάν για παράδειγμα έχουμε ένα πεδίο τιμών δύο διαστάσεων, όπου η μέγιστη τιμή που παρατηρείται για τη μία διάσταση είναι η τιμή 100, ενώ για την άλλη η τιμή 50, τότε προκύπτουν τέσσερις υποπεριοχές οριζόμενες από τις αντίστοιχες διαμέσους, δηλαδή την τιμή 50 για την πρώτη διάσταση και 25 για τη δεύτερη. Επομένως, η υποπεριοχή "A" περιέχει σημεία των οποίων η τιμή της πρώτης διάστασης βρίσκεται στο (0,50) και της δεύτερης στο (0,25), η υποπεριοχή "B" περιέχει σημεία των οποίων η τιμήτης πρώτης διάστασης βρίσκεται στο (0,50) και της δεύτερης στο (25,50), κ.ο.κ.

Στη συνέχεια, τα ζεύγη εξόδου του Division Mapper, αφού πρώτα ομαδοποιηθούν με βάση το κλειδί εξόδου (key out), διαδικασία που έτσι και αλλιώς γίνεται αυτόματα λόγω της φύσης του MapReduce όπως έχει ήδη αναφερθεί, δίνονται ως είσοδος στον Division Reducer. Αυτός υπολογίζει, με χρήση του απλού BNL την τοπική κορυφογραμμή (local skyline) για κάθε υποπεριοχή. Έτσι, ο DivisionReducer δίνει ως έξοδο ζεύγη της μορφής <subspace flg , local skyline>, τα οποία και τοποθετούνται σε ένα αρχείο εξόδου. Συμπερασματικά, μπορούμε κατά κάποιο τρόπο να πούμε πως με την Division Job μοιράζουμε το μέγεθος του προβλήματος βρίσκοντας τα τοπικά αποτελέσματα για κάθε υποπεριοχή, τα οποία και αργότερα συνδυάζονται για να βρούμε το ζητούμενοτελικό - συνολικό αποτέλεσμα.

Το αρχείο εξόδου της Division Job γίνεται αρχείο εισόδου για την Merging Job. Η διαδικασία αυτή γίνεται μέσα στην main του προγράμματος μας με την κατάλληλη παραμετροποίηση των δύο Job. Ο Merging Mapper διαβάζει κάθε σημείο που βρίσκεται στο αρχείο, ανεξαρτήτως subspace flg, και δίνει ως έξοδο ζεύγη της μορφής <null, (subspace flg , σημείο P)>. Εδώ, να αναφέρουμε πως στη θέση του "null" μπορεί να μπει οποιοσδήποτε άλλος αριθμός ή σύμβολο, καθώς η χρήση του αποσκοπεί απλά στο να έχουν όλα τα ζεύγη εξόδου το ίδιο κλειδί (key out) ώστε στη συνέχεια να ομαδοποιηθούν και να δοθούν ως είσοδος στον MergingReducer.

Algorithm 1. MR-BNL

INPUT: the original data set S
OUTPUT: the skyline of data set S
Division Job
  Map Task
  for each point P_i **in** dataset S
    compute P_i's subspace flag F_i
    output (F_i,P_i)
  Reduce Task
  for each subspace flag F_i
    compute local Skyline SP_k using **BNL**
    output (F_i, SP_k) **in** file t

Merging Job
  Map Task
  for each point P_i **in** file t
    output (**null**, (F_i, P_i))
  Reduce Task
  compute global Skyline SP_k using **BNL** with **pre-comparison**
  output (SP_k, **null**)

Εικόνα 8: Ψευδοκώδικας αλγορίθμου του MapReduce – BNL (Zhang *et al.*, 2011)

5 Μέθοδος και Πραγματοποίηση του Πειραματικού Μέρους

5.1 Μέθοδος Προσέγγισης του Προβλήματος

Σε προηγούμενα κεφάλαια προβήκαμε σε βιβλιογραφική έρευνα σε διεθνείς και ελληνικές πηγές (όπως μελέτες, έρευνες, εξειδικευμένα δημοσιεύματα). Έτσι στο παρών κεφάλαιο επρόκειτο να προχωρήσουμε σε καταγραφή των απαιτήσεων, στην ανάλυση απαιτήσεων, τον σχεδιασμό, την υλοποίηση, τον έλεγχο σχεδίασης και τέλος να προβούμε σε κατασκευή της εφαρμογής.

5.1.1 Εισαγωγή και σύνοψη υπάρχουσας κατάστασης

Από τη βιβλιογραφική έρευνα, η οποία παρουσιάζεται σε προηγούμενα κεφάλαια, είναι προφανή τα πλεονεκτήματα του Υπολογιστικού Νέφους σε διάφορες υπηρεσίες αλλά και της παράλληλης επεξεργασίας πολυδιάστατων δεδομένων μέσω των Skyline Επερωτήσεων. Στην πορεία αναπτύχθηκαν διάφορες τεχνικές, οι οποίες αφορούν στη βελτίωση των Skyline Επερωτήσεων και κυρίως στο διαμοιρασμό του χώρου. Σύντομα, λοιπόν, προέκυψε η ιδέα του συνδυασμού αυτών των δυο τεχνολογιών, δηλαδή του Υπολογιστικού Νέφους και των Skyline Επερωτήσεων. Κεντρική ιδέα της εργασίας είναι να επεξεργαστούν παράλληλα διάφορα πολυδιάστατα δεδομένα με τη χρήση Skyline Επερωτήσεων αλλά αυτή τη φορά να είναι δυνατή η μεταφορά τους σε περιβάλλον Υπολογιστικού Νέφους.

Εν ολίγοις, οι ειδικοί στόχοι που πρέπει να έχουν επιτευχθεί μετά το πέρας της παρούσης εργασίας είναι:

- Να εκτελεστούν αποδοτικά Skyline Επερωτήσεις πολυδιάστατων δεδομένων στο Υπολογιστικό Νέφος.

Επιπλέων οι τεχνικοί στόχοι που οδηγούν στην επίτευξη του βασικού στόχου είναι:

- Να αναπτυχθεί κώδικας με συγκεκριμένη δομή, κατάλληλη έτσι ώστε να μπορέσει να εκτελεστεί στο περιβάλλον του Υπολογιστικού Νέφους.
- Να προσαρμοστεί ο ήδη υπάρχων κώδικας έτσι ώστε να μπορέσει να εκτελεστεί στο περιβάλλον του Υπολογιστικού Νέφους.

5.2 Προδιαγραφές Πειραματικού Μέρους

Για την εκτέλεση του πειραματικού μέρους χρησημοποιήσαμε τα παρακάτω τεχνικά χαρακτηριστικά.

Hardware
- Συνολική μοιραζόμενη Μνήμη: 14GB
- Logic CPUs: 6
- SSD Hard Disk
- 4 VM PC Linux nodes

Software
- R language
- Map-Reduce Skyline algorithm
- Cloudera CDH 4.7

Datasets
- Dataset with 2000 random points
- Dataset.1 : x,y (2 columns)
- Dataset.2: x,y,z,X,Y,Z (6 columns)
- Dataset.3 x,y,z,X,Y,Z,a,b,c,d (10 columns)

5.3 Υλοποίηση

Επόμενο βήμα μετά τη σχεδίαση του κώδικα προς εκτέλεση είναι η ανάπτυξή του. Η ανάπτυξη του κώδικα έγινε με τη βοήθεια του R studio και σε γλώσσα προγραμματισμού R. Επίσης πολύ βασικό είναι η εγκατάσταση των βιβλιοθηκών των πακέτων rmr2, rJava και rhdfs καθώς και πριν το τρέξιμο του κώδικα να ορίσουμε τα σωστά paths όπου καλούνται οι βιβλιοθήκες του Hadoop framework οπου με την σειρα τους είναι απαραίτητες για τη σωστή λειτουργία του κώδικα.

Στον ίδιο αλγόριθμο περιλαμβάνετε και η λειτουργία της παραγωγής του κώδικα της γεννήτριας παραγωγής τυχαίων αριθμών. Στο συγκεκριμένο κώδικα, ο χρήστης ορίζει το πλήθος των σημείων που πρέπει να δημιουργηθούν, την ανώτατη τιμή που μπορούν να πάρουν καθώς και το όνομα του αρχείου. Μετά την εκτέλεση του κώδικα, τα αρχεία που δημιουργούνται τοποθετούνται στο τοπικό σύστημα αρχείων.

Μετά την ολοκλήρωση της συγγραφής του κώδικα χρειάστηκε να δημιουργηθεί το κατάλληλο περιβάλλον στο οποίο θα εκτελεστεί. Για το λόγο αυτό, χρειάστηκε να εγκατασταθεί virtual machine με λειτουργικό Linux στο οποίο υπάρχει ήδη

εγκατεστημένο το περιβάλλον του Hadoop. Το virtual machine παρέχεται δωρεάν από την Cloudera και η εγκατάστασή του είναι εξαιρετικά εύκολη.

5.3.1 Ρυθμίσεις Παραμέτρων και Σύστημα Αρχείων του Hadoop

Κατά τη συγγραφή του κώδικα, ένας στόχος που χρειάστηκε να επιτευχθεί ήταν η ευελιξία του κατά την εκτέλεσή του σε διαφορετικά σενάρια, όπως θα δούμε στο επόμενο κεφάλαιο, χωρίς να χρειάζεται ο χρήστης να επεξεργάζεται κάθε φορά τον κώδικα. Στη συγκεκριμένη περίπτωση οι μεταβλητές που αρχικοποιούνται είναι η ανώτερη τιμή που μπορεί να πάρει μια συντεταγμένη, ο αριθμός των διαμερίσεων και ο αριθμός των διαστάσεων των σημείων. Έτσι, κάθε φορά που ο χρήστης θέλει ν' αλλάξει κάποια παράμετρο αρκεί να επεξεργαστεί μόνο το συγκεκριμένο αρχείο και όχι ολόκληρο τον κώδικα.

Το συγκεκριμένο αρχείο πρέπει να βρίσκεται σε κάποιο τοπικό φάκελο του virtual machine έτσι ώστε να μπορεί να διαβαστεί από τον κώδικα. Για την αξιοποίηση του συγκεκριμένου αρχείου δημιουργήθηκε μια λειτουργία, η οποία διαβάζει το αρχείο και θα αναλυθεί περισσότερο παρακάτω. Εκτός από τις ρυθμίσεις των παραμέτρων του κώδικα, πρέπει να ρυθμιστούν και οι παράμετροι που αφορούν στο HDFS. Αρχικά, ο χρήστης πρέπει να δημιουργήσει στο HDFS το μονοπάτι εισόδου, στο οποίο θα ανέβουν τα αρχικά αρχεία προς επεξεργασία.

5.4 Το Λογικό Διάγραμμα του Αλγορίθμου

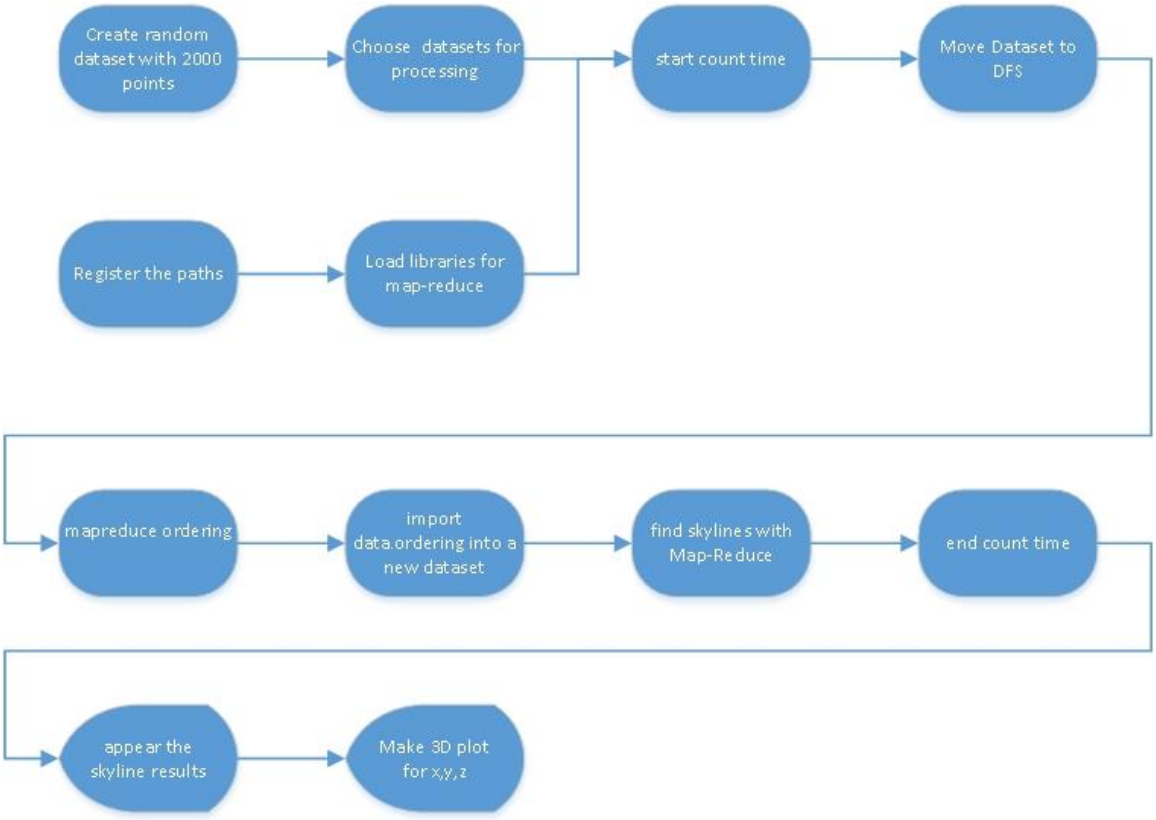
Στην εικόνα 9 βλέπουμε το λογικό διάγραμμα του αλγορίθμου που έχουμε αναπτύξει. Όπως παρατηρούμε είναι σε πλήρη ανάπτυξη και δεν περιλαμβάνει μόνο τα τρία βασικά μέρη του αλγορίθμου που έχουμε αναφέρει σε προηγούμενα κεφάλαια, αλλά όλη τη ροή του κώδικα μας. Όπως παρατηρούμε ο αλγόριθμος μας ξεκινάει με την λειτουργία της δημιουργίας κάποιου random dataset 2000 σημείων που του έχουμε ορίσει, στην συνέχεια επιλέγουμε το dataset με το οποίο θέλουμε να ξεκινήσουμε την διεργασία μας, παράλληλα και ανεξάρτητα από τις προηγούμενες δυο διεργασίες ορίζουμε τα paths για τα Hadoop libraries και φορτώνουμε τις κατάλληλες βιβλιοθήκες που συνεργάζονται με την R για να είμαστε σε θέση να τρέξουμε τις MapReduce διεργασίες.

Μετά από αυτά, ξεκινάει η λειτουργία της μέτρησης χρόνου, όπου μας είναι χρήσιμη για την μέτρηση της επεξεργαστικής απόδοσης αλλά και σύγκρισης του κάθε dataset

σε κάθε συστάδα κόμβων. Στην συνέχεια μεταφέρετε το dataset μας στο DFS και έπειτα ξεκινάει η πρώτη MapReduce διεργασία (job), όπου είναι η ταξινόμηση (ordering) του dataset μας σε αύξουσα σειρά κατά τα σημεία σε συνδυασμό με τις στήλες. Η συγκεκριμένη διεργασία είναι σημαντική λόγο ότι αφενός πρέπει να προηγηθεί πριν την τελική εύρεση των σημείων από την επόμενη διεργασία εύρεσης των skyline σημείων και αφετέρου ότι εάν δεν πραγματοποιηθεί η εν λόγο διεργασία τα skyline σημεία τα οποία θα βρεθούν δεν θα είναι τα σωστά.

Συνεχίζοντας ξεκινάει η δεύτερη διεργασία (job), που ως είσοδο αυτή την φορά παίρνει τα αποτελέσματα της πρώτης διεργασίας (job). Αυτό που συμβαίνει σε αυτή την φάση είναι η εύρεση των skyline σημείων. Η συγκεκριμένη φάση είναι πάλι μια MapReduce διεργασία (job).

Αφού βρεθούν τα skyline σημεία τερματίζουμε την λειτουργία της χρονομέτρησης, εμφανίζουμε τα skyline σημεία που βρεθήκανε όπου είναι και τα αποτελέσματα μας και τέλος προαιρετικά εμφανίζουμε τις ανάλογες γραφικές παραστάσεις, έτσι ώστε να έχουμε και μια γραφική απεικόνιση των αποτελεσμάτων μας, στους άξονες τους οποίους ορίσαμε στην αρχή.



Εικόνα 9: Το λογικό διάγραμμα του αλγορίθμου

5.5 Επεξήγηση Βασικών Μερών του Skyline Αλγορίθμου

Όπως παρακάτω και με την μορφή ψευδοκώδικα βλέπουμε την βασική λειτουργική δομή του αλγορίθμου που αναπτύξαμε. Παρατηρούμε τα τρία βασικά μέρη του αλγορίθμου, όπως τη (ν):

1. Δημιουργία τυχαίων αριθμών μέχρι 2000 σημείων

   Μια γεννήτρια τυχαίων αριθμών είναι μια υπολογιστική ή μηχανική συσκευή που έχει σχεδιαστεί για να παράγει μια ακολουθία αριθμών ή συμβόλων που δεν ακολουθούν κάποιο μοτίβο, δηλαδή εμφανίζονται τυχαία. Οι πολλές εφαρμογές της τυχαιότητας έχουν οδηγήσει στην ανάπτυξη πολλών διαφορετικών μεθόδων για την παραγωγή τυχαίων δεδομένων. Πολλές από αυτές υπάρχουν από τους αρχαίους χρόνους, όπως τα ζάρια, το ρίξιμο ενός νομίσματος, το ανακάτεμα της τράπουλας και πολλές άλλες τεχνικές. Λόγω της μηχανικής φύσεως αυτών των τεχνικών, η δημιουργία μεγάλων ακολουθιών πραγματικά τυχαίων αριθμών (κάτι σημαντικό για την στατιστική) απαιτεί πολλή εργασία και χρόνο. Έτσι, τα αποτελέσματα πολλές φορές συλλέγονταν και διανέμονταν ως πίνακες τυχαίων αριθμών. Εμείς κάναμε χρήση των άνω με σκοπό την δημιουργία κατάλληλου dataset για τον αλγόριθμο μας.

2. Ταξινόμηση του dataset με την διαδικασία του MapReduce.

   Η διαδικασία που ακολουθείτε από το MapReduce ταξινομεί με βέλτιστους αλγόριθμούς, συγκριτικά με άλλους άλγόριθμούς, το dataset με σκοπό την γρήγορη ευρετηρίαση των skyline σημείων.

3. Εύρεση skyline σημείων με την διαδικασία του MapReduce

   Η διαδικασία που ακολουθείτε από το MapReduce βοηθάει στην αποτελεσματική έυρεση των skyline σημείων και στην βέλτιστη από αποψη χρόνου παρουσίαση του αποτελέσματος.

Pseudocode Structure Algorithm

Input: The generated dataset **Dataset.points**
Output: The skyline dataset **nondominated**

Create random dataset with 2000 points
x = Make sample 2000 points
y = Make sample 2000 points

Map-reduce ordering
Map task

For each point k in **Dataset.points**
Compute v subspace flag keyval
Output(v,1)

Reduce task

For each subspace flag keyval
Compute **order** (v)
Output (**Skyline.points)**

Map-Reduce skylines
Map task

For each point k in dataset **Skyline.points**
Compute v subspace flag keyval
{
Compute skylines points in y
}
Output (**nondominated**)

Appear the skyline results

Nondominated

Make 2D plot for x,y

Plot **nondominated**

5.6 Το Πειραματικό Μέρος

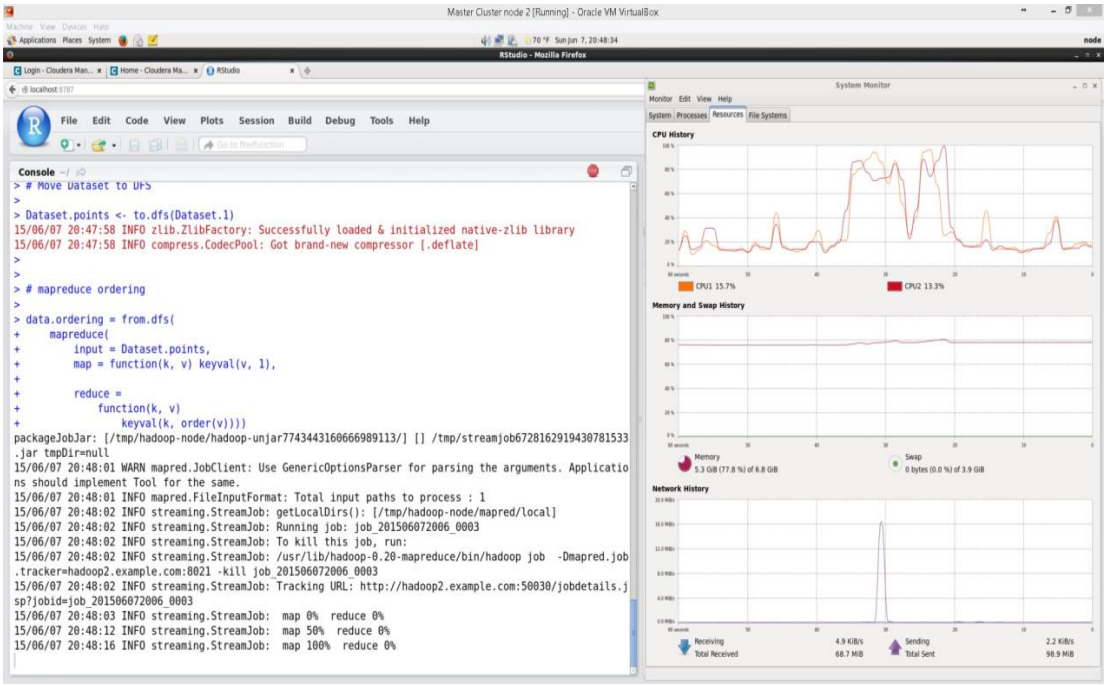
Το πειραματικό μέρος της εργασίας ξεκινά με την εισαγωγή όλου του αλγορίθμου στην κονσόλα του R Studio και ολοκληρώνετε σε σειρά ανά λειτουργιά με τελικό αποτέλεσμα την εύρεση των skyline σημείων και την εμφάνιση γραφημάτων. Όπως θα δούμε παρακάτω, κατά την διάρκεια του πειραματικού μέρους μπορούμε να βγάλουμε νέα συμπεράσματα. Διότι επρόκειτο να δούμε όχι μονό τους τελικούς χρόνους της κάθε υπολογιστικής διαδικασίας, αλλά και την κατανομή της υπολογιστικής ισχύος και κυρίας μνήμης σε κάθε κόμβο ξεχωριστά. Αυτό φυσικά οφείλετε καθαρά στην μέθοδο του MapReduce, αφού κατά κάποιο τρόπο θα λέγαμε ότι αποφασίζει πως, που αλλά και ποσό θα κατανείμει τις υπολογιστικές διαδικασίες και πόσους πόρους θα εκμεταλλευτεί σε κάθε κόμβο. Επίσης σε αυτή την ενότητα δεν θα εξετάσουμε ξεχωριστά την συμπεριφορά του cluster μας σε κάθε dataset αλλά, θα κάνουμε παρατηρήσεις και θα βγάλουμε συμπεράσματα συνολικά.

5.7 Το Πειραματικό Μέρος με ένα Κόμβο

Σε αυτή την περίπτωση δεν πρόκειται να δούμε κάποια γραφήματα που να παρουσιάζουν ιδιαίτερο ενδιαφέρων, μιας και στο πειραματικό μέρος λαμβάνει μέρος μονό ένας κόμβος. Έτσι όλη η επεξεργαστική ισχύς των υπολογισμών μας κατανέμονται αποκλειστικά και μονό σε αυτό τον κόμβο.

5.8 Το Πειραματικό Μέρος με δυο Κόμβους

Σε αυτή την περίπτωση επρόκειτο να δούμε το πειραματικό μέρος της εργασίας μας με δυο κόμβους.



Εικόνα 10: Πειραματικό μέρος με δυο κόμβους, έναρξη

Όπως μπορούμε να δούμε στην εικόνα 10, μετά το φόρτωμα του dataset μας στο HDFS, αρχίζει αμέσως μετά η διαδικασία της MapReduce ταξινόμησης. Όπως μπορούμε να δούμε κατά την διάρκεια του mapping, την συγκεκριμένη διαδικασία την αναλαμβάνει και την ολοκληρώνει ο Master node, ενώ την διαδικασία του Reducing την αναλαμβάνει ο Slave node.

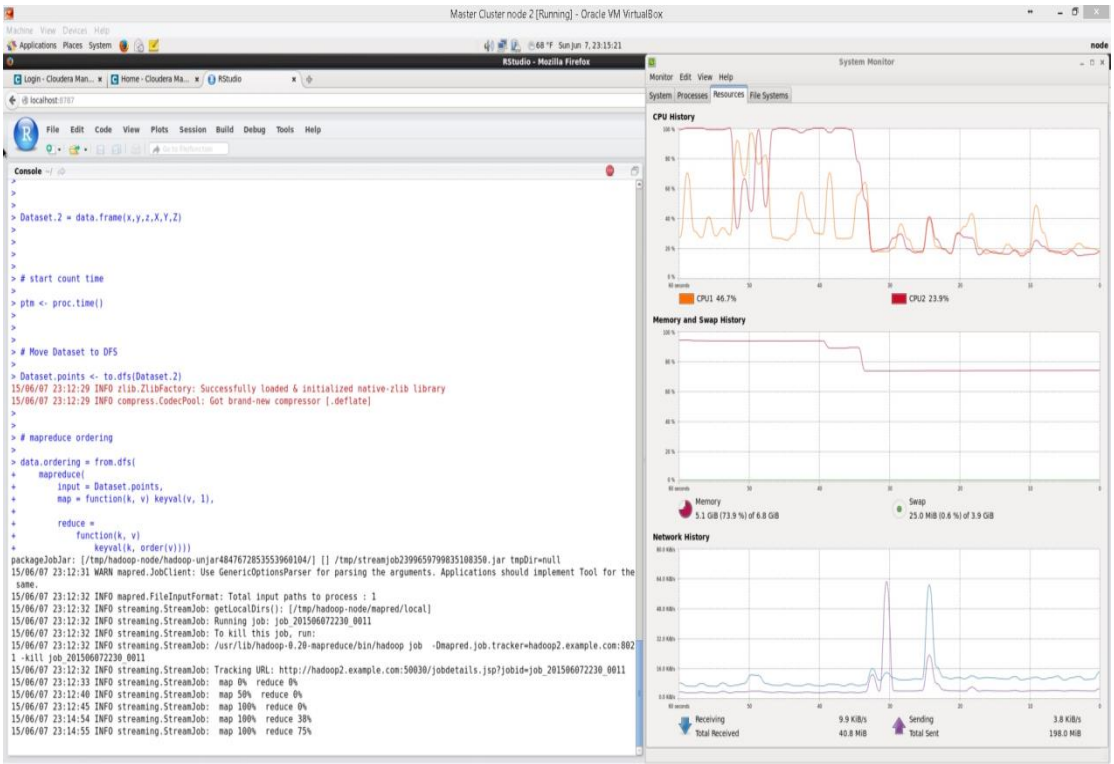
Εικόνα 11: Πειραματικό μέρος με δυο κόμβους, υπολογισμός από τον slave node.

Έτσι όπως παρατηρούμε, όταν ολοκληρώνετε το mapping process από τον Master node αμέσως μετά αναλαμβάνει το reduce process ο slave node (εικόνα 11).

5.9 Το Πειραματικό Μέρος με τρεις Κόμβους

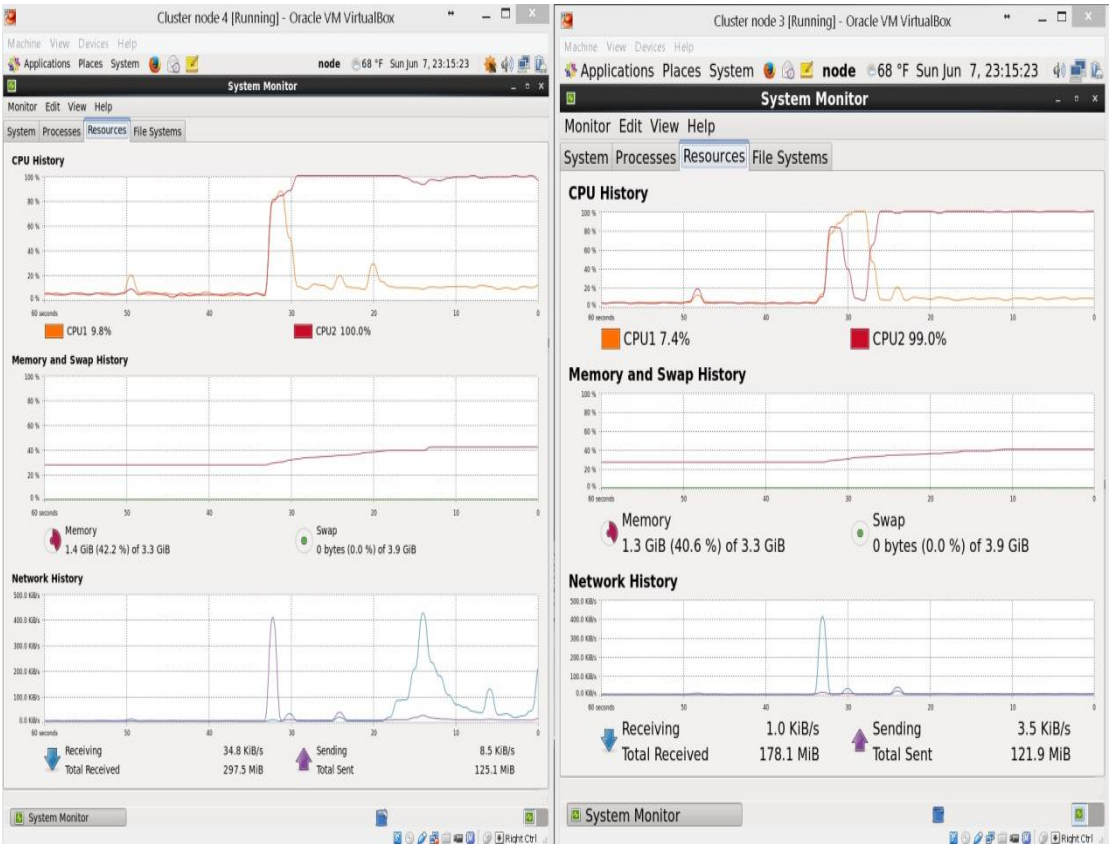
Στο πειραματικό μέρος της εργασίας μας με τρεις κόμβους, μπορούμε να αναφέρουμε πως παρουσιάζει ένα ιδιαίτερο ενδιαφέρων μιας και έχουμε ένα κατάλληλο αριθμό κόμβων να αλληλεπιδρούν και είμαστε σε θέση να δούμε καλύτερα και να μελετήσουμε την κατανομή της υπολογιστικής ισχύος.

Όπως μπορούμε να διαπιστώσουμε από την εικόνα 12, το mapping της ταξινόμησης πραγματοποιείτε και πάλι από τον Master node. Μάλιστα βλέπουμε καθαρά πως όταν ολοκληρώνει το process, απελευθερώνει τους υπολογιστικούς πόρους, τόσο σε CPU οόσο και σε RAM που είχε δεσμεύσει κατά την διάρκεια της διαδικασίας.



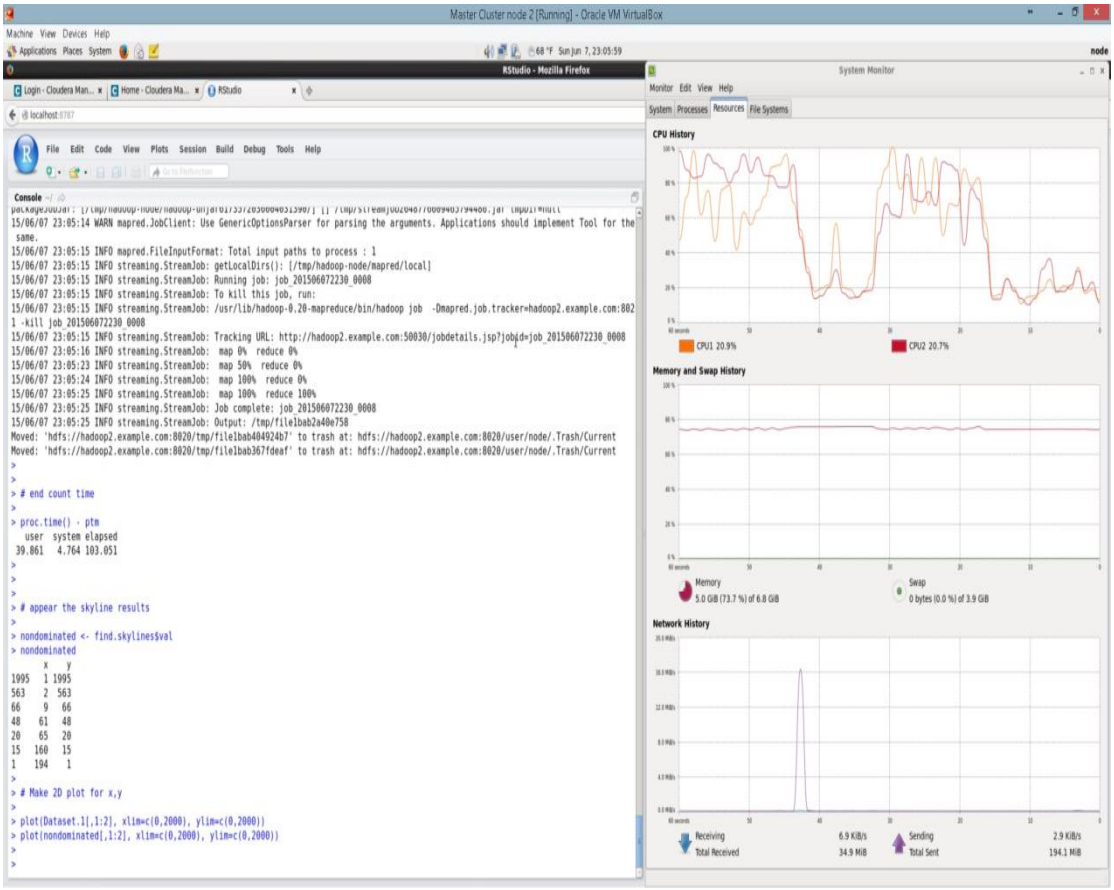
Εικόνα 12: Πειραματικό μέρος με τρεις κόμβους, έναρξη

Στην εικόνα 13, παρατηρούμε ότι μετά το πέρας της διαδικασίας του mapping αρχίζει η διαδικασία του reducing. Εδώ βλέπουμε να ξεκινούν και οι δυο slave κόμβοι σχεδόν ταυτόχρονα και αναλαμβάνουν το process. Επίσης παρατηρούμε ότι την απότομη μεταβολή της CPU σε ποσοστό σχεδόν 100% και την σταδιακή άνοδο από την δέσμευση της διαθέσιμης RAM.

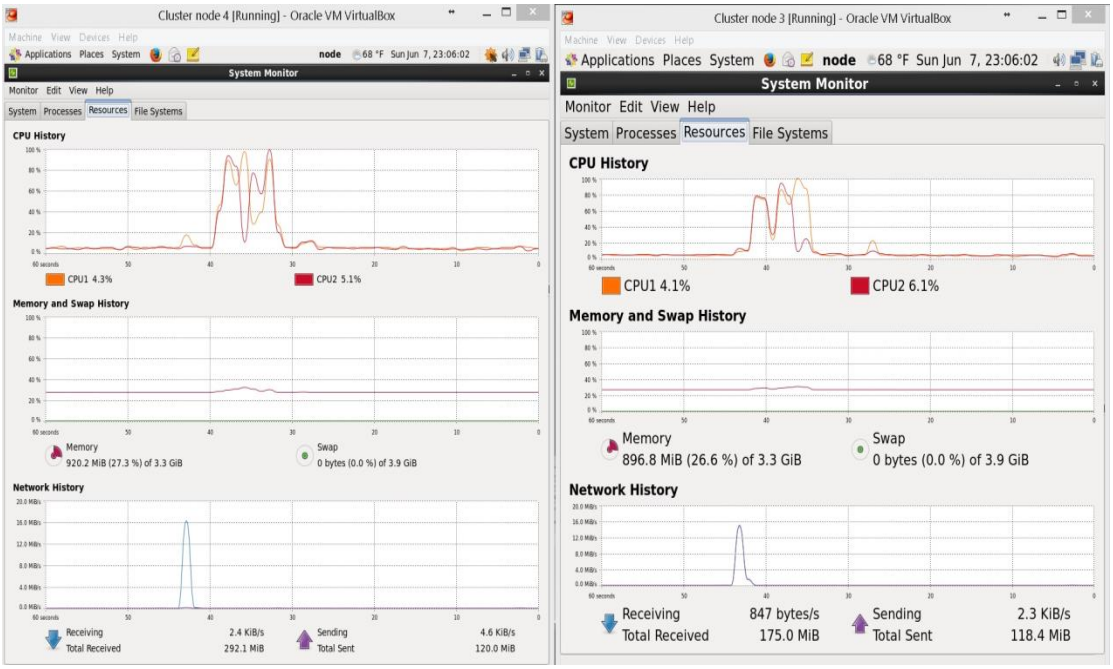


Εικόνα 13: Πειραματικό μέρος με τρεις κόμβους, υπολογισμός από δυο slave nodes.

Επίσης όπως μπορούμε να δούμε στην εικόνα 14 και 15, παρατηρούμε την ιδία διαδικασία του MapReduce αλλά για τον υπολογισμό των skyline σημείων, αυτή την φορά. Επίσης η συγκεκριμένη διαδικασία δεν διαφέρει από την προηγούμενη Map Reduce διαδικασία, που είναι η ταξινόμηση του dataset μας.



Εικόνα 14: Πειραματικό μέρος με τρεις κόμβους, υπολογισμός skyline σημείων από τον Master node.



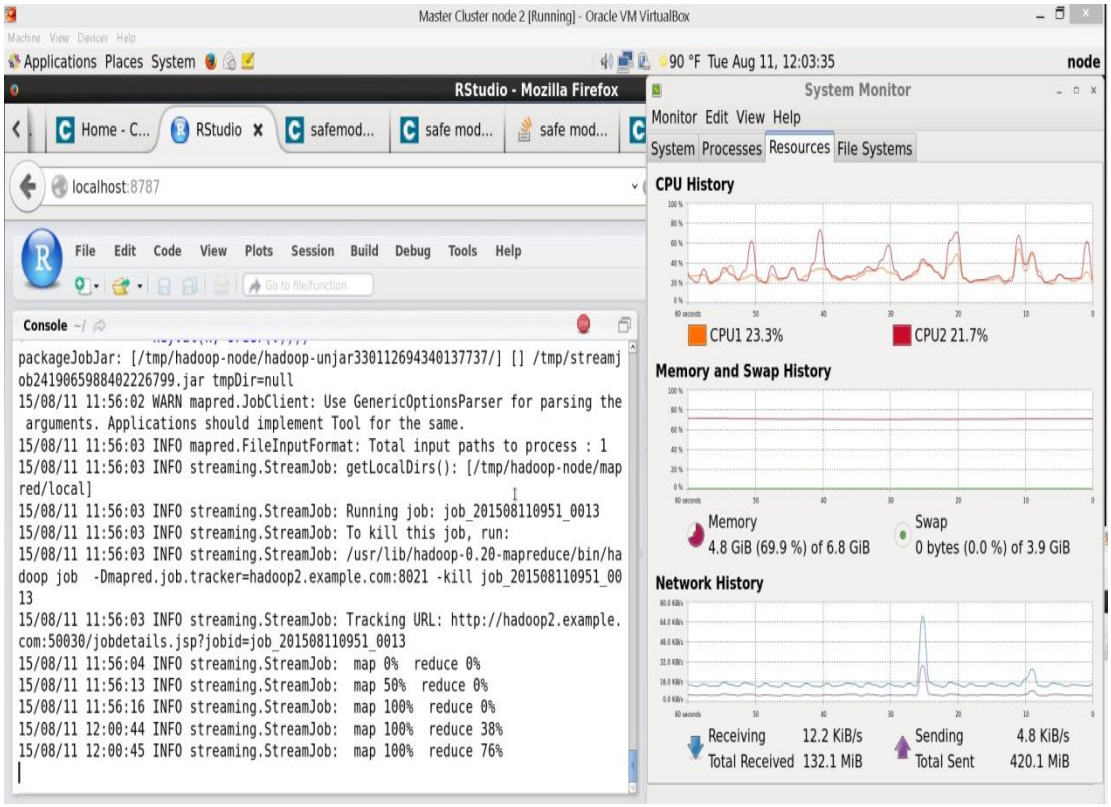
Εικόνα 15: Πειραματικό μέρος με τρεις κόμβους, υπολογισμός skyline σημείων από τους δυο slave nodes.

Τέλος όπως βλέπουμε στην εικόνα 14, στο τέλος της MapReduce διαδικασίας αλλά και του αλγορίθμου γενικότερα, παρατηρούμε τον τελικό συνολικό χρόνο, οπού παίρνουμε τις μετρήσεις μας και βλέπουμε τα υπολογισμένα skyline σημεία του dataset μας. Να διευκρινίσουμε πως πάντα τα σημεία θα είναι διαφορετικά λόγο ότι το dataset που λαμβάνουμε είναι πάντοτε τυχαίο.

5.10 Το Πειραματικό Μέρος με τέσσερεις Κόμβους

Στο 4ο και τελικό πειραματικό μέρος της εργασίας μας με τέσσερεις κόμβους, επρόκειτο να δούμε ακουστικά το πείραμα μας σε πλήρη ανάπτυξη. Εκτιμούμε πως η συμπεριφορά της MapReduce διαδικασίας πιθανών να μην διαφέρει δραματικά σε πειράματα με περισσότερους από τέσσερεις κόμβους. Με την προϋπόθεση βεβαία πως οι ρυθμίσεις στην πλατφόρμα μας (cloudera) συνεχίζουν να είναι όπως έχουν και επιπλέων πως πάντα θα εκτελείτε ένα job κάθε φορά και όχι άλλο job παράλληλα. Οτιδήποτε άλλες δόκιμες πέραν αυτών που έχουν πραγματοποιηθεί, επρόκειτο να επηρεάσουν καταλυτικά τα συμπεράσματα και τα αποτελέσματα του πειραματικού μας μέρους.

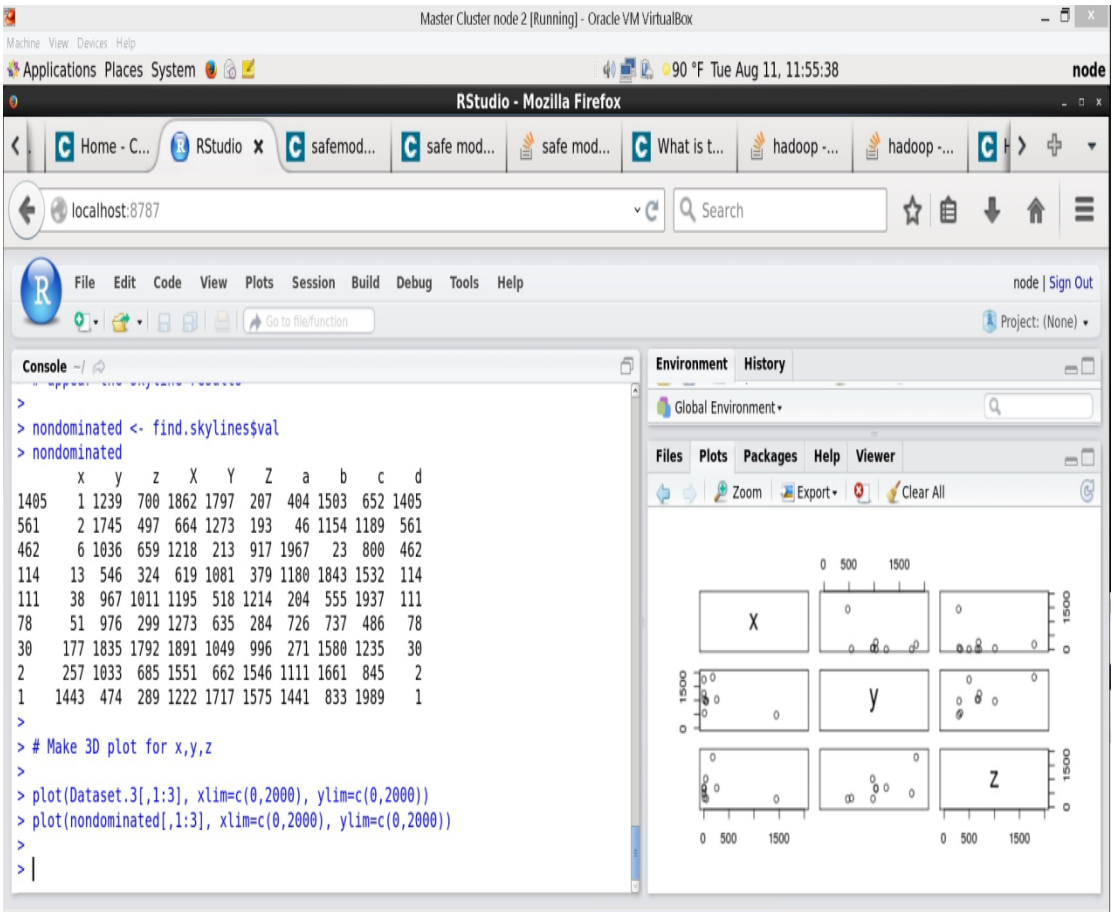
Όπως βλέπουμε στην εικόνα 16, παρατηρούμε μια σταθερότητα των process στην CPU, του Master node ενώ όπως βλέπουμε στην εικόνα 17 οι κόμβοι 5 και 3 έχουν αναλάβει το process της reduce διαδικασίας. Αξιοσημείωτο είναι ότι ο slave node 4 είναι ανενεργός και δεν έχει αναλάβει κανένα process. Σε αυτή την περίπτωση εκτιμούμε πως είναι πολλοί οι λόγοι που ο συγκεκριμένος κόμβος δεν έχει αναλάβει κάποιο process επί της παρούσης. Όπως και γενικότερα παρατηρούμε ότι πάντα κάποιος κόμβος δεν αναλαμβάνει κάποιο process (όχι πάντα ο ίδιος κόμβος αλλά κάθε φορά ένας διαφορετικός).



Εικόνα 16: Πειραματικό μέρος με τέσσερεις κόμβους, Master node.

Εικόνα 17: Πειραματικό μέρος με τέσσερεις κόμβους, slave nodes.

Τέλος στην εικόνα 18 βλέπουμε τα τελικά αποτελέσματα. Με τα υπολογισμένα skyline σημεία, καθώς και τις γραφικές παραστάσεις για τους άξονες x,y και z.



Εικόνα 18: Τελικά αποτελέσματα και γραφικές παραστάσεις.

Όπως συμπεραίνουμε από την ποιο πάνω ανάλυση, βλέπουμε ότι για μικρά cluster (από 1 έως 3 κόμβους), οι υπολογιστικοί πόροι διανέμονται σε όλα τα nodes, ενώ για μεγάλα clusters (από 4 κόμβους και πάνω), παρατηρούμε ότι κάποιοι κόμβοι θα μένουν ανενεργοί κατά την διάρκεια των processes.

Τέλος να επισημάνουμε όπως είχαμε παρατηρήσει κατά την διάρκεια των πειραμάτων μας, είδαμε πως εάν διεξήγαμε τα πειράματα μας με datasets άνω των 2000 σημείων (πχ. με 5000 σημεία), παρατηρούσαμε ότι πολύ γρήγορα η διαθέσιμη κυρία μνήμη εξαντλούνταν και το σύστημα αντλούσε από την swap memory. Αυτό είχε ως σκοπό τον πολύ αργό υπολογισμό των διαδικασιών με αποτέλεσμα την απόρριψη τους μετά από κάποιο μεγάλο χρονικό διάστημα και φυσικά την κατάρρευση του MapReduce process.

6 Αποτελέσματα του Πειραματικού Μέρους

Για την εξαγωγή των αποτελεσμάτων κάναμε χρήση Dataset με 2000 random points σε 4 διαφορετικά σενάρια σε τριών ειδών διαφορετικά Dataset που είχαν τα εξής χαρακτηριστικά:

- Dataset.1: x,y (2 columns)
- Dataset.2: x,y,z,X,Y,Z (6 columns)
- Dataset.3: x,y,z,X,Y,Z,a,b,c,d (10 columns)

6.1 Σενάριο 1

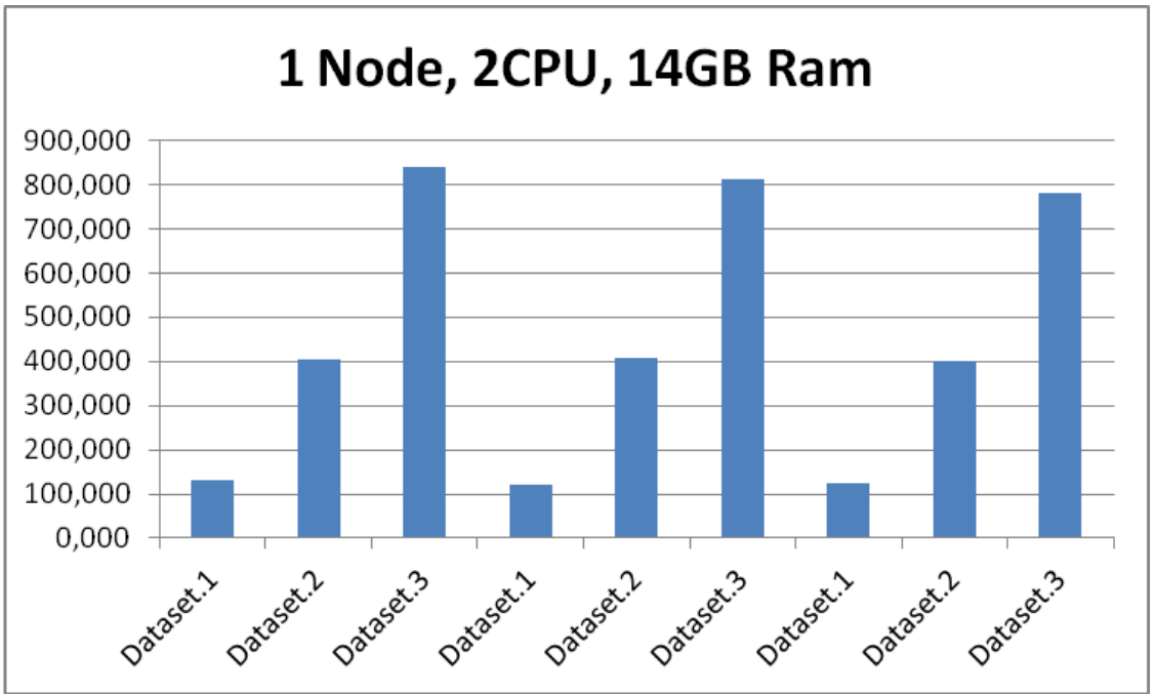
Χρησιμοποιήσαμε υπολογιστική ισχύ 1 Node, 2CPU, 14GB Ram για Dataset με 2000 random points. Τα αποτελέσματα είχαν ως εξής:

1 Node, 2CPU, 14GB Ram			
Dataset.1	131.336 sec	121.131 sec	125.381 sec
Dataset.2	405.812 sec	407.046 sec	399.625 sec
Dataset.3	839.885 sec	810.677 sec	782.112 sec

Πίνακας 2: Μετρήσεις χρόνου για 1 node

Παρατηρούμε ότι υπολογιστική ισχύ 1 Node, 2CPU, 14GB Ram για 2000 random points τα εξής:

- Για το πρωτο Dataset οι χρόνοι έχουν μέσο όρο 125,949 sec
- Για το δεύτερο Dataset οι χρόνοι έχουν μέσο όρο 404,161sec
- Για το τρίτο Dataset οι χρόνοι έχουν μέσο όρο 810,8913sec



Εικόνα 19: Αποτελέσματα 1ου πειραματικού μέρους

Παρατηρούμε ότι η διαφορετικότητα της πολυπλοκότητας των Dataset (Εικόνα 19) δεν επηρεάζει τους χρόνους με αποτέλεσμα να έχουν ομοιομορφία.

6.2 Σενάριο 2

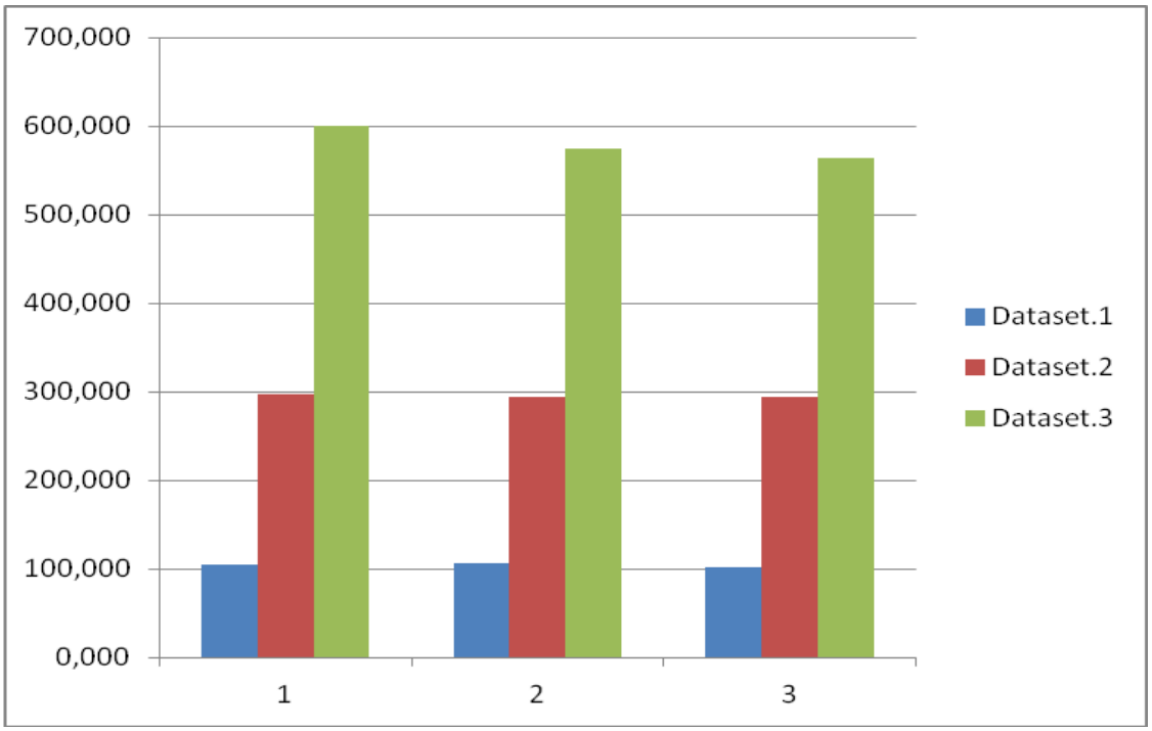
Χρησιμοποιήσαμε υπολογιστική ισχύ 2 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 7GB Ram για Dataset με 2000 random points. Τα αποτελέσματα είχαν ως εξής:

2 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 7GB Ram			
Dataset.1	104,889 sec	106,802 sec	102,095 sec
Dataset.2	297,439 sec	294,040 sec	293,879 sec
Dataset.3	600,448 sec	574,937 sec	564,129 sec

Πίνακας 3: Μετρήσεις χρόνου για 2 nodes

Παρατηρούμε ότι υπολογιστική ισχύ 2 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 7GB Ram για 2000 random points τα εξής:

- Για το πρωτο Dataset οι χρόνοι έχουν μέσο όρο 104,591sec
- Για το δεύτερο Dataset οι χρόνοι έχουν μέσο όρο 295,11sec
- Για το τρίτο Dataset οι χρόνοι έχουν μέσο όρο 579,838sec



Εικόνα 20: Αποτελέσματα 2ου πειραματικού μέρους

Αντίστοιχα κι εδώ παρατηρούμε (Εικόνα 20) ότι η διαφορετικότητα της πολυπλοκότητας των Dataset δεν επηρεάζει τους χρόνους με αποτέλεσμα να έχουν ομοιομορφία. Επίσης με την διπλή επεξεργαστική ισχύ έχουμε μείωση των χρόνων σε όλα τα επίπεδα.

6.3 Σενάριο 3

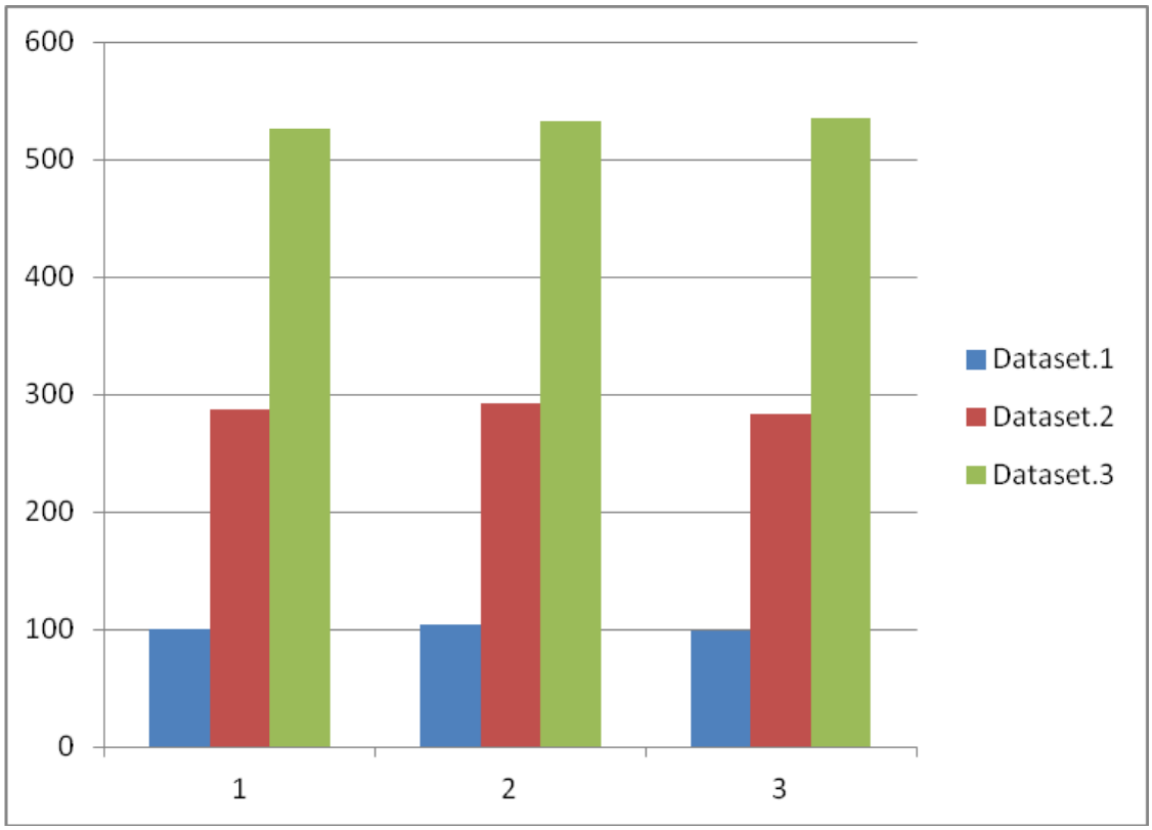
Χρησιμοποιήσαμε υπολογιστική ισχύ 3 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 3.5GB Ram, 3rd Node: 3.5GB Ram για Dataset με 2000 random points. Τα αποτελέσματα είχαν ως εξής:

3 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 3.5GB Ram, 3rd Node: 3.5GB Ram			
Dataset.1	100,331 sec	104,519 sec	98,848 sec
Dataset.2	287,134 sec	291,986 sec	283,684 sec
Dataset.3	525,668 sec	532,506 sec	535,829 sec

Πίνακας 4: Μετρήσεις χρόνου για 3 nodes

Παρατηρούμε ότι υπολογιστική ισχύ 3 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 3.5GB Ram, 3rd Node: 3.5GB Ram για 2000 random points τα εξής:

- Για το πρωτο Dataset οι χρόνοι έχουν μέσο όρο 101,237 sec
- Για το δεύτερο Dataset οι χρόνοι έχουν μέσο όρο 285,601 sec
- Για το τρίτο Dataset οι χρόνοι έχουν μέσο όρο 531,334 sec



Εικόνα 21: Αποτελέσματα 3ου πειραματικού μέρους

Παρατηρούμε ότι η διαφορετικότητα της πολυπλοκότητας των Dataset (Εικόνα 21) δεν επηρεάζει τους χρόνους με αποτέλεσμα να έχουν ομοιομορφία. Επίσης με την τριπλή επεξεργαστική ισχύ έχουμε μείωση των χρόνων σε όλα τα επίπεδα.

6.4 Σενάριο 4

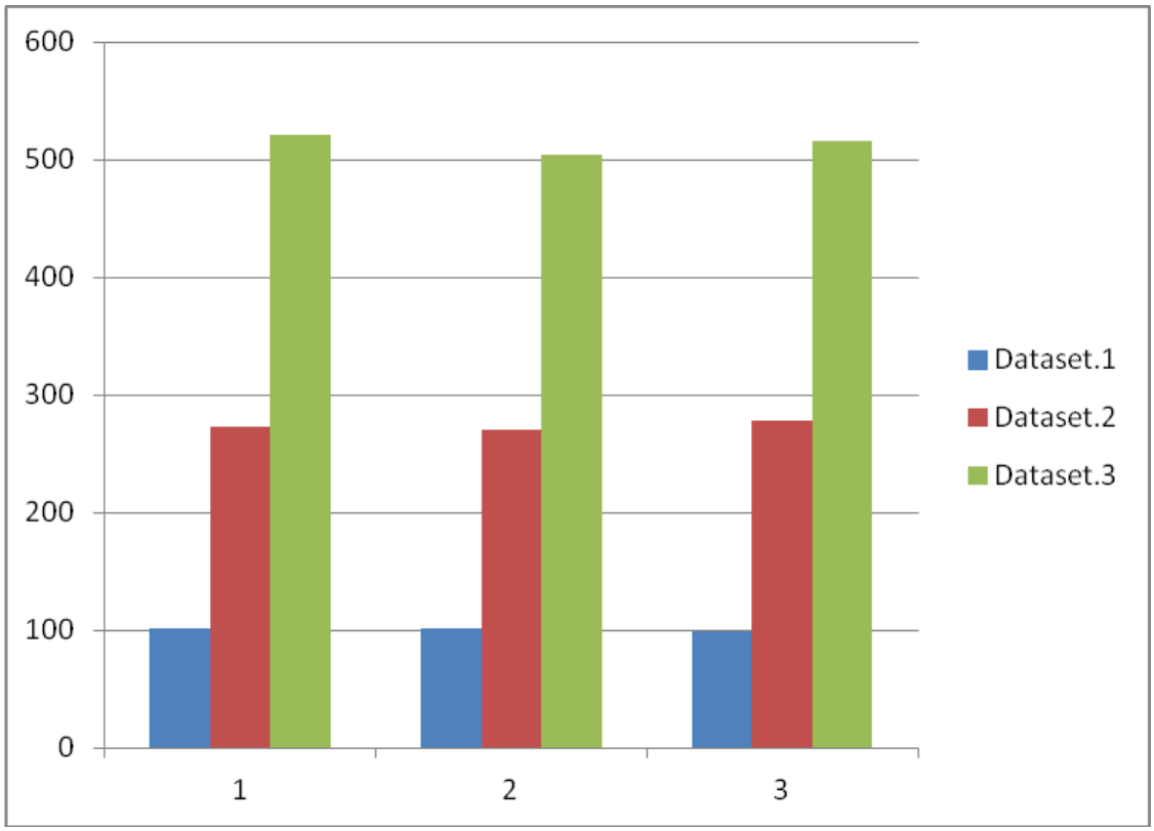
Χρησιμοποιήσαμε υπολογιστική ισχύ 4 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 2.3GB Ram, 3rd Node: 2.3GB Ram, 4th Node: 2.3GB Ram, για Dataset με 2000 random points. Τα αποτελέσματα είχαν ως εξής:

4 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 2.3GB Ram, 3rd Node: 2.3GB Ram, 4th Node: 2.3GB Ram,			
Dataset.1	102,079 sec	101,665 sec	99,577 sec
Dataset.2	272,912 sec	270,010 sec	278,549 sec
Dataset.3	520,630 sec	503,739 sec	516,259 sec

Πίνακας 5: Μετρήσεις χρόνου για 4 nodes

Παρατηρούμε ότι υπολογιστική ισχύ 4 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 2.3GB Ram, 3rd Node: 2.3GB Ram, 4th Node: 2.3GB Ram για 2000 random points τα εξής:

- Για το πρωτο Dataset οι χρόνοι έχουν μέσο όρο 101,107 sec
- Για το δεύτερο Dataset οι χρόνοι έχουν μέσο όρο 273,823 sec
- Για το τρίτο Dataset οι χρόνοι έχουν μέσο όρο 513,542 sec



Εικόνα 22: Αποτελέσματα 4ου πειραματικού μέρους

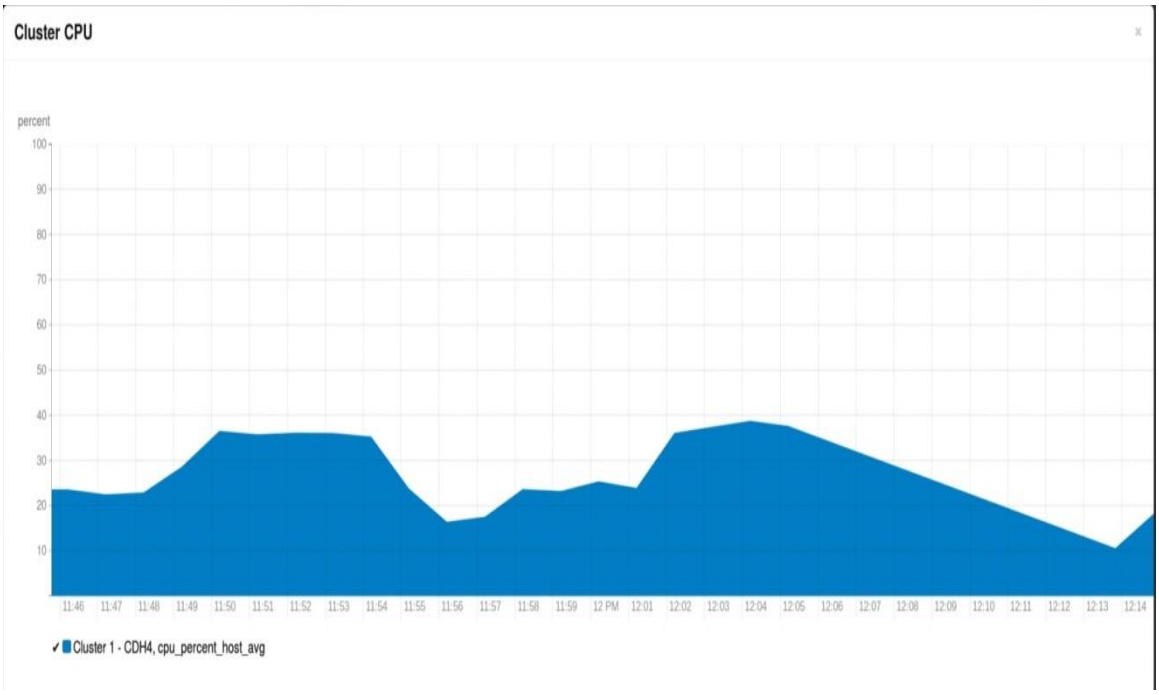
Παρατηρούμε ότι η διαφορετικότητα της πολυπλοκότητας των Dataset (Εικόνα 22) δεν επηρεάζει τους χρόνους με αποτέλεσμα να έχουν ομοιομορφία. Επίσης με την τετραπλή επεξεργαστική ισχύ έχουμε μείωση των χρόνων σε όλα τα επίπεδα.

6.5 Σχετικά Γραφήματα και Μετρήσεις του Cluster

Παρακάτω ακολουθούν ορισμένα ενδεικτικά γραφήματα σχετικά με την καταγραφή των αντιδράσεων των πόρων τους του cluster μας, μετά από την ολοκλήρωση των πειραμάτων μας. Στα συγκεκριμένα γραφήματα μετράμε τις μεταβλητές του cluster που θεωρούμε ως σημαντικές και τις σχολιάζουμε με βάση τα εικονιζόμενων γραφημάτων μας.

6.5.1 Cluster CPU

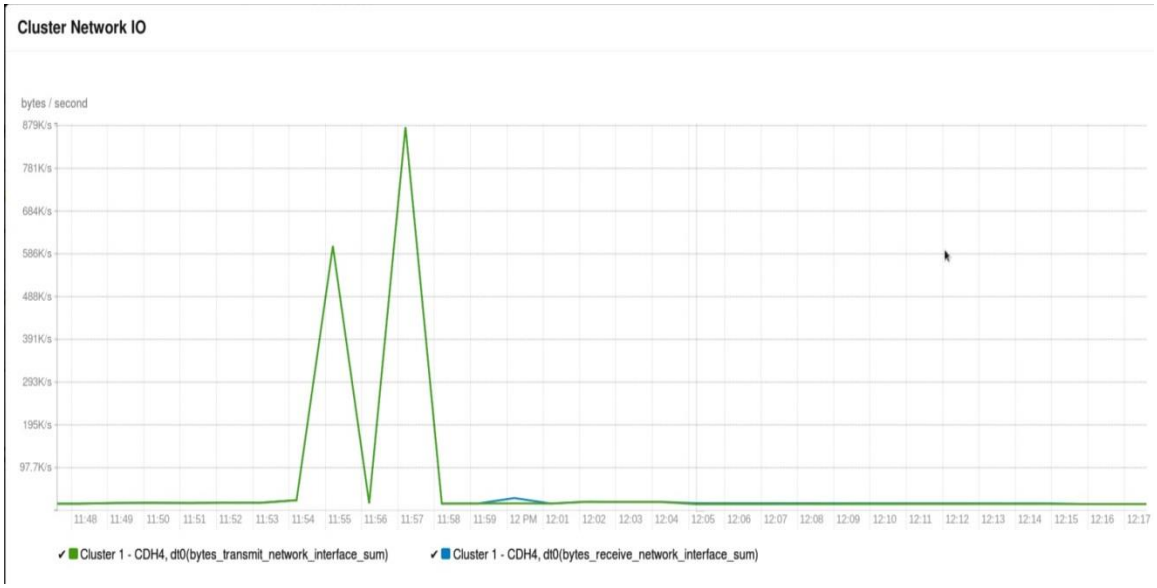
Στο γράφημα της εικόνας 23, παρατηρούμε την συνολική κατανομή της υπολογιστικής ισχύος κατά την διάρκεια των υπολογισμών. Επρόκειτο να παρατηρήσουμε αυξομειώσεις στην υπολογιστική ισχύ σε συνάρτηση με τον χρόνο καθώς και το μέγεθος του dataset που έχουμε τρέξει.



Εικόνα 23: Cluster CPU - Η συνολική κατανάλωση υπολογιστικής ισχύος του cluster στο πεδίο του χρόνου.

6.5.2 Cluster Network IO

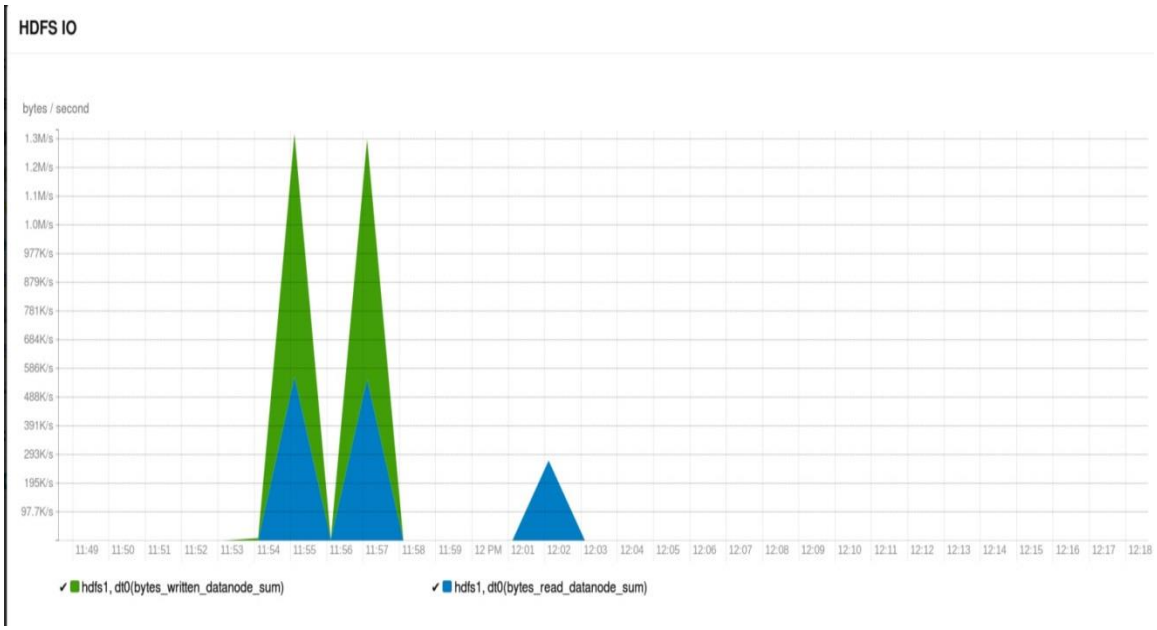
Στο παρακάτω γράφημα (Εικόνα 24), παρατηρούμε την συνολική κίνηση των δεδομένων μας κατά την επεξεργασία των dataset μας, μέσα στο cluster, στο πεδίο του χρόνου. Επίσης παρατηρούμε μια έντονη κινητικότητα (με πράσινο χρώμα) σε κάποια συγκεκριμένα χρονικά διαστήματα, οπού έλαβε μέρος η επεξεργασία μας. Τέλος παρατηρούμε ότι υπάρχει μια κατά πολύ μεγαλύτερη τάξη μεγέθους στα δεδομένα που στέλνονται, παρά που λαμβάνονται μέσα στο cluster μας.



Εικόνα 24: Cluster network IO - Η συνολική μεταφορά δεδομένων στο δίκτυο του cluster στο πεδίο του χρόνου.

6.5.3 HDFS IO

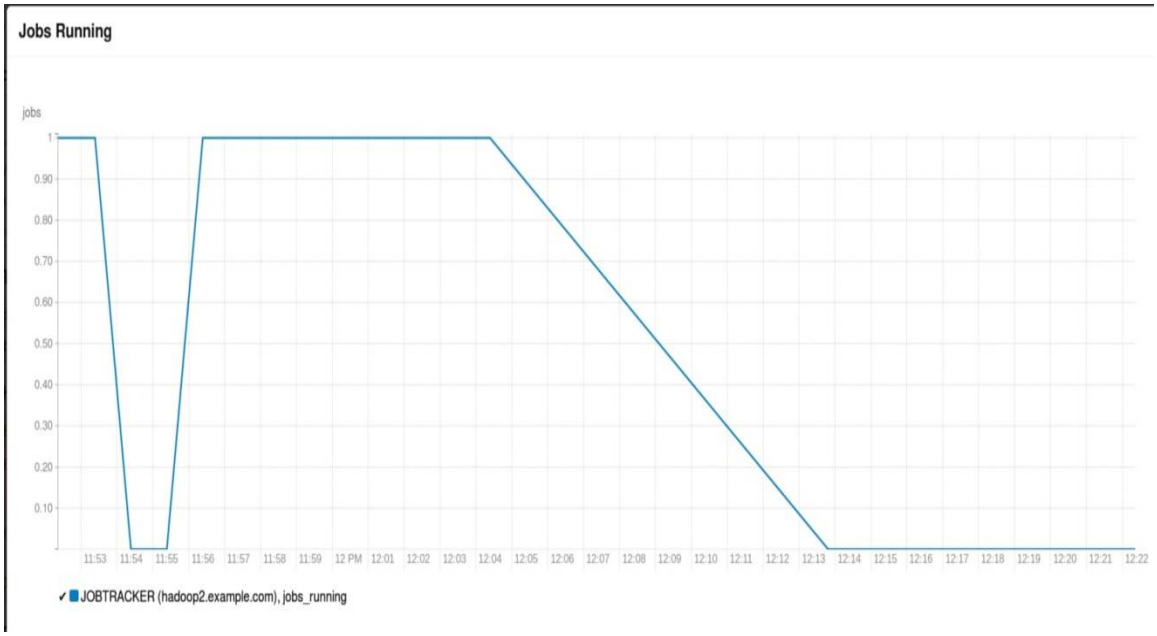
Στο παρακάτω γράφημα (Εικόνα 25), παρατηρούμε την συνολική μεταφορά των δεδομένων στο Hadoop file system στο πεδίο του χρόνου. Το θεωρούμε σημαντικό μιας και το HDFS κατέχει σημαντικό ρόλο στην λειτουργιά και επεξεργασία των δεδομένων μας, επειδή εκεί ουσιαστικά γίνονται οι MapReduce επεξεργασίες και αποθήκευση των αποτελεσμάτων μας. Επιπλέων ως input ορίζετε η πράσινη περιοχή (written) και ως output ορίζετε η μπλε περιοχή (read). Τέλος όπως βλέπουμε οι περιοχές με πράσινο χρώμα (written) είναι κατά πολύ μεγαλύτερες από τις περιοχές με μπλε χρώμα (written).



Εικόνα 25: HDFS IO - Η συνολική μεταφορά δεδομένων στο Hadoop file system στο πεδίο του χρόνου. Ως input ορίζετε η πράσινη περιοχή (written) και ως output ορίζετε η μπλέ περιοχή (read)

6.5.4 Jobs Running

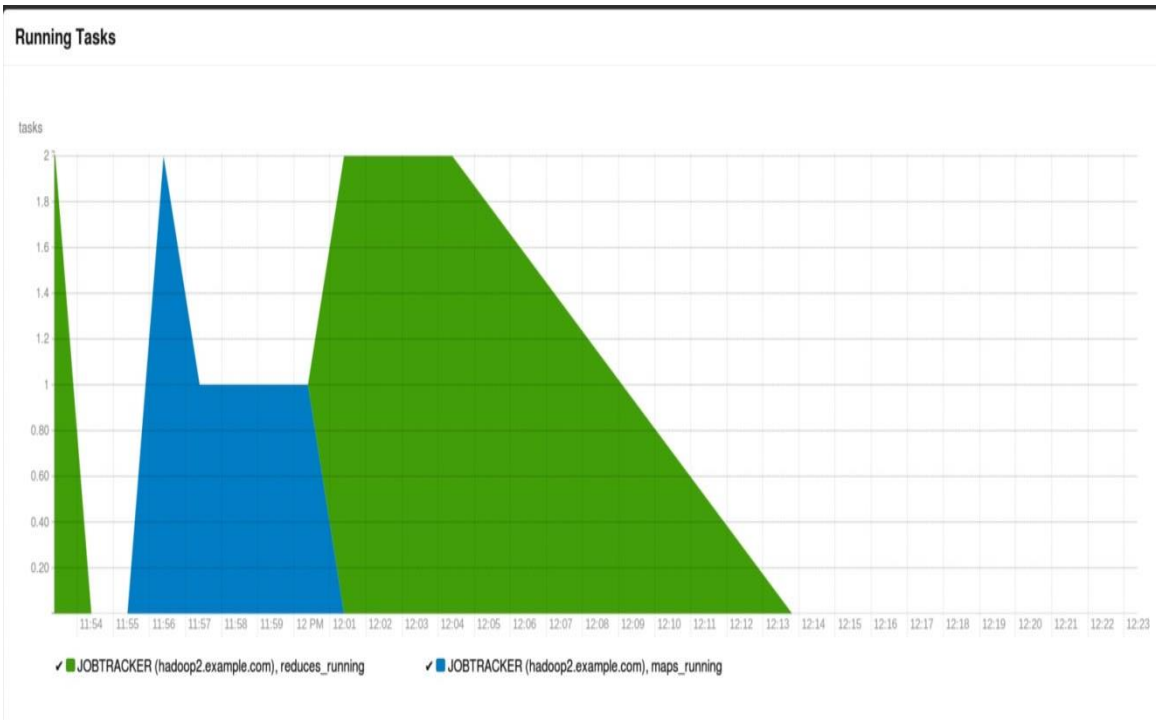
Στο παρακάτω γράφημα (Εικόνα 26), παρατηρούμε τον συνολικό αριθμό jobs στο cluster μας στο πεδίο του χρόνου. Από το γράφημα μπορούμε εύκολα να δούμε πότε ακριβώς ξεκίνησε η διεργασία του MapReduce, πόσο χρόνο κράτησε η επεξεργασία, όπως επίσης και τον αριθμό των jobs που εκτελούνται. Στην προκειμένη περίπτωση εκτελείται μόνο ένα job.



Εικόνα 26: Jobs Running - Ο συνολικός αριθμός jobs στο cluster στο πεδίο του χρόνου.

6.5.5 Running Tasks

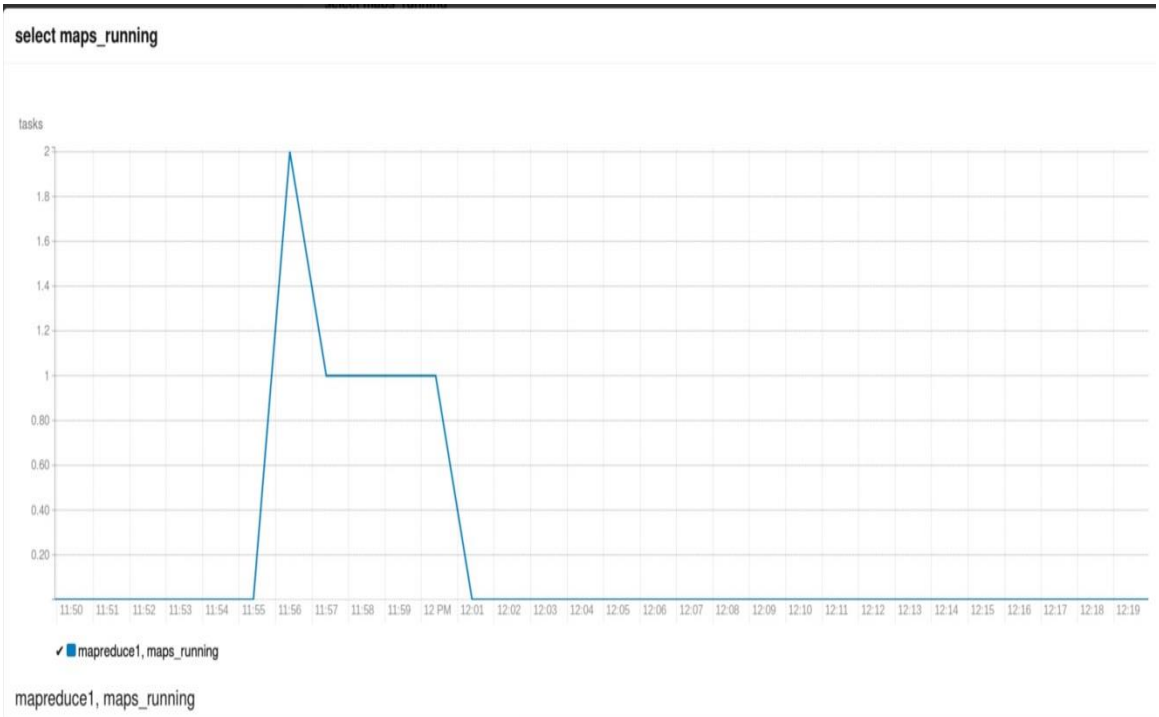
Στο παρακάτω γράφημα (Εικόνα 27), παρατηρούμε τον συνολικό αριθμό των tasks του cluster στο πεδίο του χρόνου. Σε αντίθεση με το προηγούμενο γράφημα όπου παρίστανε τον συνολικό αριθμό των jobs, σε αυτό το γράφημα παρατηρούμε με πράσινο χρώμα τις reduce διαδικασίες και με μπλε χρώμα τις map διεργασίες. Αυτό το γράφημα μας βοηθά να παρατηρήσομε πότε ξεκινάει μια map και πότε μια reduce διαδικασία στο πεδίο του χρόνου. Σε αυτό το γράφημα οι δυο αυτές διαδικασίες υπολογίζονται ως κάτι ξεχωριστό και όχι ως κάτι ενιαίο.



Εικόνα 27: Running Tasks - Ο συνολικός αριθμός των tasks του cluster στο πεδίο του χρόνου.

6.5.6 Maps Running

Στο παρακάτω και τελευταίο γράφημα (Εικόνα 28), παρατηρούμε τον συνολικό αριθμό των maps του cluster στο πεδίο του χρόνου. Ουσιαστικά το γράφημα είναι αποκλειστικά και μόνο για τις maps διεργασίες. Αν το παρατηρήσουμε θα διαπιστώσουμε πως το γράφημα απεικονίζει το περίγραμμα του προηγούμενου στην μπλε περιοχή.



Εικόνα 28: Maps Running - Ο συνολικός αριθμός των maps του cluster στο πεδίο του χρόνου.

7 Συζήτηση και συμπεράσματα

7.1 Συζήτηση

Στο κεφάλαιο αυτό θα αναφερθούν τα βασικά συμπεράσματα που προκύπτουν από το θέμα που μελετήθηκε. Στην παρούσα εργασία αρχικά παρουσιάστηκαν και περιγράφηκαν τα Skyline Queries και το Hadoop – MapReduce. Περιγράφηκε αναλυτικά η αρχιτεκτονική τους, ο τρόπος λειτουργίας τους καθώς και η χρησιμότητά τους. Στη συνέχεια παρουσιάστηκαν προγενέστερες εργασίες που είχαν ασχοληθεί είτε με Skyline Επερωτήσεις είτε με το MapReduce είτε και με τις δυο τεχνολογίες.

Σημαντικό βάρος δόθηκε στην περιγραφή των τριών τεχνικών για το διαμοιρασμό του χώρου αλλά και στην μέθοδο που χρησιμοποιείται για τον παράλληλο υπολογισμό των skyline σημείων. Ακολούθως τέθηκαν οι στόχοι της εργασίας και ακολούθησε η περιγραφή της σχεδίασης της εργασίας σε επίπεδο αρχιτεκτονικής. Στη συνέχεια, περιγράφηκε η υλοποίηση του κώδικα σε επίπεδο κλάσεων, αναλύοντας τη χρησιμότητα της κάθε λειτουργίας.

Τέλος, παρουσιάστηκε η πειραματική μελέτη μέσω γραφημάτων στα οποία αναπαρίστανται οι μετρήσεις που έγιναν.

7.2 Συμπεράσματα

Όπως μπορούμε να παρατηρήσουμε κατά την επεξεργασία των τριών διαφορετικών Dataset 2.000 σημείων αλλά με διαφορετικό αριθμό στηλών το καθένα.

Από τις μετρήσεις που πραγματοποιήσαμε, παρατηρούμε παρακάτω τέσσερεις πίνακες όπου κάθε ένας από αυτούς αντιστοιχεί σε κάθε υπολογιστική συστάδα (cluster), με 1,2,3 και 4 nodes αντίστοιχα και περιλαμβάνει ο κάθε πίνακας, τους χρόνους του κάθε Dataset, τους μέσους όρους, καθώς και την αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση, την οποία την συγκρίνουμε για να λάβουμε τα τελικά συμπεράσματα.

Ουσιαστικά κάνουμε μια σύγκριση των μέσων όρων των χρόνων του κάθε dataset, σύμφωνα με τα μετρήσιμα στοιχεία που έχουμε. Αυτή την διαδικασία την κάνουμε σε κάθε περίπτωση και για όλες τις ομάδες clustering ξεχωριστά.

7.2.1 Μετρήσιμα συμπεράσματα για ένα κόμβο

Στον παρακάτω πίνακα (Πίνακας 6), παρατηρούμε την αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για ένα κόμβο. Όπως μπορούμε να δούμε το Dataset.2 αυξήθηκε 3,2 φορές χρονικά, σε σχέση με το Dataset.1. Ενώ το Dataset.3 αυξήθηκε χρονικά 2 φορές σε σχέση με το Dataset.2. Οι υπολογιστικοί πόροι που χρησιμοποιήθηκαν για το cluster είναι: 2CPU και 14GB Ram.

1 Node, 2CPU, 14GB Ram				Μέσος όρος	Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση
Dataset.1	131.336 sec	121.131 sec	125.381 sec	125,949 sec	N/A
Dataset.2	405.812 sec	407.046 sec	399.625 sec	404,161 sec	3,2
Dataset.3	839.885 sec	810.677 sec	782.112 sec	810,891 sec	2

Πίνακας 6: Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για 1 node

7.2.2 Μετρήσιμα συμπεράσματα για δυο κόμβους.

Στον παρακάτω πίνακα (Πίνακας 7), παρατηρούμε την αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για δυο κόμβους. Όπως μπορούμε να δούμε το Dataset.2 αυξήθηκε 2,8 φορές χρονικά, σε σχέση με το Dataset.1. Ενώ το Dataset.3 αυξήθηκε χρονικά 1,9 φορές σε σχέση με το Dataset.2. Οι υπολογιστικοί πόροι που χρησιμοποιήθηκαν για το cluster είναι: 2CPUs και 7GB Ram για τον κάθε κόμβο.

2 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 7GB Ram				Μέσος όρος	Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση
Dataset.1	104,889 sec	106,802 sec	102,095 sec	104,595 sec	N/A

Dataset.2	297,439 sec	294,040 sec	293,879 sec	295,119 sec	2,8
Dataset.3	600,448 sec	574,937 sec	564,129 sec	579,838 sec	1,9

Πίνακας 7: Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για 2 nodes

7.2.3 Μετρήσιμα συμπεράσματα για τρεις κόμβους.

Στον παρακάτω πίνακα (Πίνακας 8), παρατηρούμε την αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για τρεις κόμβους. Όπως μπορούμε να δούμε το Dataset.2 αυξήθηκε 2,8 φορές χρονικά, σε σχέση με το Dataset.1. Ενώ το Dataset.3 αυξήθηκε χρονικά 1,8 φορές σε σχέση με το Dataset.2. Οι υπολογιστικοί πόροι που χρησιμοποιήθηκαν για το cluster είναι: 2CPUs και 7GB Ram για τον Master κόμβο και 2CPUs και 3.5GB Ram για τον κάθε Slave κόμβο.

3 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 3.5GB Ram, 3rd Node: 3.5GB Ram				Μέσος όρος	Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση
Dataset.1	100,331 sec	104,519 sec	98,848 sec	101,232 sec	N/A
Dataset.2	287,134 sec	291,986 sec	283,684 sec	287,601 sec	2,8
Dataset.3	525,668 sec	532,506 sec	535,829 sec	531,334 sec	1,8

Πίνακας 8: Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για 3 nodes

7.2.4 Μετρήσιμα συμπεράσματα για τέσσερεις κόμβους.

Στον παρακάτω πίνακα (Πίνακας 9), παρατηρούμε την αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για τρεις κόμβους. Όπως μπορούμε να δούμε το Dataset.2 αυξήθηκε 2,7 φορές χρονικά, σε σχέση με το Dataset.1. Ενώ το Dataset.3 αυξήθηκε χρονικά 1,8 φορές σε σχέση με το Dataset.2. Οι υπολογιστικοί πόροι που

χρησιμοποιήθηκαν για το cluster είναι: 2CPUs και 7GB Ram για τον Master κόμβο και 2CPUs και 2,3GB Ram για τον κάθε Slave κόμβο.

4 Nodes, 2CPUs each ,1st Node: 7GB Ram, 2nd Node: 2.3GB Ram, 3rd Node: 2.3GB Ram, 4th Node: 2.3GB Ram,				Μέσος όρος	Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση
Dataset.1	102,079 sec	101,665 sec	99,577 sec	101,107 sec	N/A
Dataset.2	272,912 sec	270,010 sec	278,549 sec	273,823 sec	2,7
Dataset.3	520,630 sec	503,739 sec	516,259 sec	513,542 sec	1,8

Πίνακας 9: Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση για 4 nodes

Έτσι σύμφωνα με τους παραπάνω πίνακες βγάζουμε τα εξής συμπεράσματα:

- Αύξηση χρόνου σε σχέση με την προηγουμένη μέτρηση είναι σχεδόν αμετάβλητο σε όλα τα datasets.
- Όσο αυξάνονται οι κόμβοι τόσο μειώνονται οι χρόνοι.
- Στα μικρά datasets η μείωση των χρόνων όσο αυξάνονται οι κόμβοι, είναι μικρή έως αμελητέα.
- Στα μεγάλα datasets η μείωση των χρόνων όσο αυξάνονται οι κόμβοι, είναι μεγάλη.
- Όσο ποιο μεγάλο dataset τόσο μεγαλύτερο κέρδος σε χρόνους σε cluster με 4 nodes και πάνω.

7.3 Προοπτικές Χρήσης

Οι προοπτικές χρήσης του θέματος και του αλγορίθμου της παρούσης εργασίας είναι πολλές, όπως π.χ. για εμπορική χρήση:

- Την ανάπτυξη του αλγορίθμου για την εύρεση κοντινών λιμανιών σε μια ακτογραμμή.
- Την γρήγορη και άμεση εύρεση αεροπορικών εισιτήριων με σύγκριση τιμών και υπηρεσιών.
- Την εύρεση κάποιου ιατρικού φακέλου με κάποια συγκεκριμένα χαρακτηριστικά που πληρούνται.

Επίσης μπορούμε να επισημάνουμε την εφαρμογή του αλγορίθμου μας για ακαδημαϊκή χρήση όπως:

- Την χρήση αυτούσιου του κώδικα μας για να χρησιμοποιηθεί ως σημείο αναφοράς για την χρονική μέτρηση πολλαπλών κόμβων και hardware.
- Την χρονική μέτρηση και απόδοση διαφορετικών server συστημάτων.
- Την σύγκριση της μεθοδολογίας με διαφορετικές τεχνολογίες και προσεγγίσεις παράλληλου προγραμματισμού.
- Την σύγκριση των αποδόσεων μεταξύ διαφορετικών εκδόσεων του λογισμικού που βασίστηκε.
- Την σύγκριση μεταξύ διαφορετικών προσεγγίσεων και αποδόσεων skyline αλγορίθμων με την υλοποίηση του MapReduce.

7.4 Μελλοντικές επεκτάσεις

Όπως είδαμε σε προηγούμενα κεφάλαια το πειραματικό μέρος της εργασία βασίστηκε κατεξοχήν σε VMs όπου έτρεχαν την όλη στημένη υποδομή και τον αλγόριθμο. Ενώ όπως είδαμε δεν ξεπεράσαμε τους 4 κόμβους. Επίσης είδαμε ότι το πειραματικό μέρος έτρεξε με κάποιο συγκεκριμένο Hardware, σε τοπικό Η/Υ. Επιπλέων, όσον αφορά στα datasets που τρέξαμε είχαμε ένα περιορισμό 2000 σημείων.

Σύμφωνα με τα παραπάνω, οι μελλοντικές επεκτάσεις της παρούσας εργασίας θα μπορούσαν να είναι αρκετές, όπως:

- Το στήσιμο και τρέξιμο της όλης υποδομής σε υπηρεσία Cloud Computing.
- Την αύξηση των κόμβων.
- Το τρέξιμο πολλαπλών διεργασιών.
- Το τρέξιμο του αλγορίθμου για πάνω από 2000 σημεία.
- Το τρέξιμο του αλγορίθμου και την χρονομέτρηση του σε νέες εκδόσεις λογισμικών.
- Το στήσιμο και τρέξιμο του αλγορίθμου σε διαφορετικές γλώσσες προγραμματισμού.
- Την περεταίρω βελτίωση του ιδίου του αλγορίθμου με άλλες τεχνικές, όπως για παράδειγμα την εγγραφή και συλλογή των αποτελεσμάτων σε κάποιο τοπικό αρχείο και όχι τόσο εκτενώς στην κυριά μνήμη του υπολογιστικού cluster.

Βιβλιογραφία

[1] Bo Dong, Qinghua Zheng, Jie Yang, Haifei Li, Mu Qiao: An E-learning Ecosystem Based on Cloud Computing Infrastructure. ICALT 2009: 125-127

[2] Yasuhiko Morimoto; Mohammad Shamsul Arefin; Mohammad Anisuzzaman Siddique: Agent - Based Convex Skyline Set Query for Cloud Computing Environment. (2012)

[3] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, Bongki Moon: Parallel data processing with MapReduce: a survey. SIGMOD Record 40(4): 11-20 (2011)

[4] Jörg Schad, Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz: Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance.PVLDB 3(1): 460-471 (2010)

[5] «Power of Cloud» [Ηλεκτρονικό]. Available: http://transformcloud.blogspot.gr. [Πρόσβαση 2013].

[6] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis: Angle-based space partitioning for efficient parallel skyline computation. SIGMOD Conference 2008: 227-238

[7] Chuck Lam: Hadoop in Action. Manning Publications (2011)

[8] Jeffrey Dean, Sanjay Ghemawat: MapReduce: a flexible data processing tool. Commun. ACM 53(1): 72-77 (2010)

[9] Tom White: Hadoop The Definitive Guide.O'Reilly | Yahoo Press.

[10] Christos Doulkeridis, Kjetil Nørväg: A Survey of Large-Scale Analytical Query Processing in MapReduce. VLDB Journal (to appear 2013)

[11] Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004: 137-150

[12] Michael Stonebraker, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Erik Paulson, Andrew Pavlo, Alexander Rasin: MapReduce and parallel DBMSs: friends or foes? Commun. ACM 53(1): 64-71 (2010)

[13] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J. Abadi, Alexander Rasin, Avi Silberschatz: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2(1): 922-933 (2009)

[14] Mohamed Y. Eltabakh, Yuanyuan Tian, Fatma Özcan, Rainer Gemulla, Aljoscha Krettek, John McPherson: CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop. PVLDB 4(9): 575-585 (2011)

[15] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, Raghotham Murthy: Hive - A Warehousing Solution Over a Map-Reduce Framework. PVLDB 2(2): 1626- 1629 (2009)

[16] Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, Jörg Schad: Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). PVLDB 3(1): 518-529 (2010)

[17] Eaman Jahani, Michael J. Cafarella, Christopher Ré: Automatic Optimization for MapReduce Programs. PVLDB 4(6): 385-396 (2011)

[18] Katja Hose, Akrivi Vlachou: A survey of skyline processing in highly distributed environments. VLDB J. 21(3): 359-384 (2012)

[19] Henning Köhler, Jing Yang, Xiaofang Zhou: Efficient parallel skyline processing using hyperplane projections. SIGMOD Conference 2011: 85-96

[20] Yufei Tao, Greg Fu, Bernhard Seeger, Dimitris Papadias: Skyline queries and its variations - An Optimal and Progressive Algorithm for Skyline Queries. SIGMOD Conference 2003: 467-478

[21] Dimitris Papadias, Yufei Tao, Greg Fu, Bernhard Seeger: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1): 41-82 (2005)

[22] Jan Chomicki, Parke Godfrey, Jarek Gryz, Dongming Liang: Skyline with Presorting. ICDE 2003: 717-719

[23] «Wikipedia - The Free Encyclopedia» [Ηλεκτρονικό]. Available: http://www.wikipedia.org.

[24] «Coding Bliss» [Ηλεκτρονικό]. Available: http://codingbliss.com/page/3/.

[25] Liang Chen, Kai Hwang, Jian Wu: MapReduce Skyline Query Processing with a New Angular Partitioning Approach. IPDPS Workshops 2012: 2262-2270

[26] Mullesgaard, Kasper et al. "Efficient Skyline Computation in MapReduce". Proc. 17th International Conference on Extending Database Technology (EDBT). 2014: 37-48.

[27] Zhang, Boliang, Shuigeng Zhou, and Jihong Guan. "Adapting Skyline Computation to the MapReduce Framework: Algorithms and Experiments." DASFAA Workshops. 2011: 403-414.

[28] Borzsonyi, S., Kossamn, D., & Stocker, K., The Skyline operator. ICDE, 2001: 421 – 430.

Παράρτημα 1

Ενδεικτικά η υλοποίηση του MapReduce Skyline αλγορίθμου σε γλωσσα R, σε τρεις διαστάσεις με μέγεθος 2000 σημείων.

```
Sys.setenv(HADOOP_HOME="/usr/lib/hadoop-0.20-mapreduce")
Sys.setenv(HADOOP_CMD="/usr/bin/hadoop")

Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-0.20-
mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh4.7.1.jar")
#Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-mapreduce/hadoop-
streaming.jar")

Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.35.x86_64")



library(rmr2)
library(rJava)
library(rhdfs)
hdfs.init()



# Create random dataset

x = sample.int(2000, 2000)
y = sample.int(2000, 2000)
z = sample.int(2000, 2000)

Dataset = data.frame(x,y,z)



 # start count time
ptm <- proc.time()



# Move Dataset to DFS
Dataset.points <- to.dfs(Dataset)
```

```
# mapreduce ordering

data.ordering = from.dfs(
   mapreduce(
      input = Dataset.points,
      map = function(k, v) keyval(v, 1),
      reduce =
         function(k, v)
            keyval(k, order(v))))


Skyline.points <- to.dfs(data.ordering$key)


# find skylines with Map-Reduce

find.skylines = from.dfs(
   mapreduce(
      input = Skyline.points,
      map = function(k, v)

      {
         nondom <- v[which(!duplicated(cummin(v$z))), ]

         keyval(k, nondom )}))


# end count time
proc.time() - ptm


# appear the skyline results
nondominated <- find.skylines$val
nondominated



#3D plot
plot(Dataset[,1:3], xlim=c(0,2000), ylim=c(0,2000))
plot(nondominated[,1:3], xlim=c(0,2000), ylim=c(0,2000))
```