ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ

ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ

ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

Σύγκριση τεχνολογιών αποθήκευσης και ανάκτησης γεωχωρικών δεδομένων μεγάλης κλίμακας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Περιβολιώτης Νικόλαος

Επιβλέπων : Ασκούνης Δημήτριος

Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ

ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ

ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

Σύγκριση τεχνολογιών αποθήκευσης και ανάκτησης γεωχωρικών δεδομένων μεγάλης κλίμακας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Περιβολιώτης Νικόλαος

Επιβλέπων : Ασκούνης Δημήτριος

Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Φεβρουαρίου 2020.

.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαρράς	Χάρης Δούκας
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Αναπ. Καθηγητής Ε.Μ.Π

Αθήνα, Φεβρουάριος 2020

.....

Περιβολιώτης Νικόλαος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια παρατηρείται μια εκρηκτική αύξηση στα δεδομένα που παράγονται από τον σύγχρονο άνθρωπο και οφείλεται στην χρήση της ψηφιακής τεχνολογίας. Η εκμετάλλευση του πλήθους αυτού των δεδομένων αποτελεί μια πρόκληση για τις επιχειρήσεις, τον άνθρωπο αλλά και την τεχνολογία. Στόχος της παρούσας διπλωματικής εργασίας είναι να κάνει μια ανασκόπηση στις τεχνολογίες που υπάρχουν για την διαχείριση δεδομένων μεγάλης κλίμακας και να συγκρίνει την απόδοση κάποιων από αυτών σε πραγματικά δεδομένα.

Τα γεωχωρικά δεδομένα που αφορούν θαλάσσια μεγέθη είναι μια κατηγορία που συγκεντρώνει αρκετό ενδιαφέρον τόσο για κλιματολογικούς λόγους όσο και για εμπορικούς αν συμπεριλάβει κανείς την ναυτιλία. Οι τεχνολογίες μεγάλων δεδομένων χωρίζονται σε κατηγορίες με βάση το μοντέλο των δεδομένων που αποθηκεύουν και τον τρόπο επεξεργασίας των δεδομένων. Έτσι, έχουμε τις NoSQL βάσεις (βλέπε mongoDb, Cassandra), τις SQL για μεγάλα δεδομένα (SparkSQL, HiveQL, PrestoDb), επεξεργασίας ροών δεδομένων και επεξεργασίας γράφων. Στα πλαίσια αυτής της εργασίας θα συγκριθεί η απόδοση των mongoDb, SparkSQL και HiveQL σε γεωχωρικά δεδομένα που συλλέχθηκαν από το σύστημα θαλάσσιας επιτήρησης Copernicus.

Η σύγκριση των τεχνολογιών γίνεται πάνω σε ένα σύνολο από 14 ερωτήματα που επιλέχθηκαν και αφορούν κυρίως ανάκτηση δεδομένων. Κριτήριο της απόδοσης αποτελεί ο χρόνος απάντησης σε κάθε ερώτημα. Για την εκτέλεση των ερωτημάτων χρειάστηκε να γίνει χρήση υπηρεσιών υπολογιστικού νέφους και να δημιουργηθεί ένα δίκτυο από 5 υπολογιστές.

Από την συγκριτική αξιολόγηση των αποτελεσμάτων προέκυψε ότι καλύτερο χρόνο στην πλειοψηφία των ερωτημάτων παρουσίασε η SparkSQL. Αυτό οφείλεται κυρίως στην υπολογιστική μηχανή Spark που χρησιμοποιεί, η οποία αποθηκεύει τα δεδομένα της στην μνήμη RAM. Από την άλλη, η mongoDb κυρίως λόγω της αποθήκευσης των δεδομένων της σε BSON αύξανε αρκετά τον όγκο των προς επεξεργασία δεδομένων γεγονός που με τη σειρά του αύξανε τις απαιτήσεις του συστήματος σε RAM. Έτσι, η mongoDb χαρακτηρίζεται ως μια ακριβή λύση. Τέλος, η HiveQL είχε χειρότερους χρόνους από την SparkSQL λόγω των περιορισμών του Map Reduce αλγορίθμου που εφαρμόζεται από το Hadoop, πάνω στο οποίο λειτουργεί η HiveQL.

Λέξεις Κλειδιά:

σύγκριση τεχνολογιών δεδομένων μεγάλης κλίμακας, mongoDb, SparkSQL, HiveQL, γεωχωρικά δεδομένα

Abstract

Over the last years, we have seen a rapid growth in data humans generate and that is mainly caused by the use of digital technology. The exploitation of these data poses a challenge for business, society and technology. The goal of current thesis is to review the state of the art of big data technologies and to compare the performance of some of them over real data.

Geospatial data of sea/oceanic origin are attracting great interest because of their climatological and commercial (maritime industry) impact. Big data technologies fall into categories based on the model of their data and the execution engines they use. Consequently, one category is NoSQL databases (like mongoDb, Cassandra), another category is databases for big SQL systems (SparkSQL, HiveQL, PrestoDb) while others is for processing stream data and processing graph data. In the context of current thesis, we are going to compare the performance of mongoDb, SparkSQL and HiveQL over geospatial data collected from the sea observatory system Copernicus.

The comparison of these technologies was made on a set of 14 custom queries that were mainly consisted of reads. The time needed for each technology to answer the query was the criterion to assess them. In order for the queries to run, it was needed to create a cluster of 5 machines on the cloud.

After comparing the performance of the technologies, it was found that SparkSQL had the best results on most queries. That is mainly due to Spark execution engine, which acts as in memory database. On the other hand, mongoDb stores data as BSON objects and that increases the total amount of data. That means that more RAM memory is needed compared to other technologies, thus making mongoDb an expensive solution. Finally, HiveQL is outperformed from SparkSQL because of the Map Reduce algorithm that the underlying Hadoop engine is using.

Keywords:

Big data technologies, benchmarking, mongoDb, SparkSQL, HiveQL, geospatial data

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στον τομέα Ηλεκτρικών και Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π, στα πλαίσια των ερευνητικών δραστηριοτήτων του Εργαστηρίου Συστημάτων Αποφάσεων και Διοίκησης. Ο κύριος στόχος της εργασίας είναι να κάνει μια ανασκόπηση στις τεχνολογίες που υπάρχουν για την διαχείριση δεδομένων μεγάλης κλίμακας και να συγκρίνει την απόδοση κάποιων από αυτών σε γεωχωρικά δεδομένα.

Επιβλέπων καθηγητής ήταν ο κ. Δημήτριος Ασκούνης τον οποίο και θα ήθελα να ευχαριστήσω για την δυνατότητα που μου έδωσε να εντρυφήσω και να αποκομίσω πολύτιμες γνώσεις σε ένα πάρα πολύ ενδιαφέρον για μένα αντικείμενο.

Ακόμη, οφείλω ένα μεγάλο ευχαριστώ στον υποψήφιο διδάκτορα Γιάννη Τσαπέλα για τη συνεχή υποστήριξη του σε όλα τα στάδια της εκπόνησης της παρούσας διπλωματικής εργασίας και την άψογη συνεργασία που είχαμε όλο αυτό το διάστημα.

Τέλος, δεν θα μπορούσα να μην ευχαριστήσω την οικογένεια μου και καθένα χωριστά από όσους με στήριξαν πολύπλευρα καθόλη την διάρκεια των μεταπτυχιακών μου σπουδών μου.

Πίνακας Περιεχομένων

ΚΕΦΑΛΑΙΟ 1: Αντικείμενο εργασίας, δομή και στόχος 13
1.1 Εισαγωγή – θαλάσσια δεδομένα 14
1.2 Ερευνητική ανάγκη 15
1.3 Σκοπός εργασίας – Ερευνητικό ερώτημα 16
1.4 Φάσεις υλοποίησης 17
1.5 Οργάνωση τόμου 17
ΚΕΦΑΛΑΙΟ 2: Θεωρητικό υπόβαθρο τεχνολογιών δεδομένων μεγάλης κλίμακας και συγκριτική αξιολόγηση 19
2.1 Δεδομένα μεγάλης κλίμακας – Ορισμός 20
2.2 Τεχνολογίες αποθήκευσης και επεξεργασίας δεδομένων μεγάλης κλίμακας 21
2.3 Εργαλεία αξιολόγησης της επίδοσης τεχνολογιών 26
2.4 Σχετικές Μελέτες 28
ΚΕΦΑΛΑΙΟ 3. Τεχνικό υπόβαθρο τεχνολογιών δεδομένων μεγάλης κλίμακας 31
3.1 Αποθήκευση αρχείων δεδομένων 32
3.2 Αλγόριθμος Map-Reduce 34
3.3 Hadoop και Hadoop Distributed File System (HDFS) 36
3.4 Apache Hive 37
3.5 Apache Spark - SparkSQL 38
3.6 MongoDb 40
ΚΕΦΑΛΑΙΟ 4. Εκτέλεση πειράματος 45
4.1 Παρουσίαση δεδομένων 46
4.2 Κριτήρια συγκριτικής αξιολόγησης 48
4.3 Υπολογιστικοί πόροι πειράματος 49
4.4 Παρουσίαση πειράματος 50
4.4.1 Μετρήσεις με mongoDb sharded 50
4.4.2 Μετρήσεις με SparkSQL σε τοπολογία standalone cluster 53
4.4.3 Μετρήσεις με HiveQL, hdfs και Apache Tez 56
ΚΕΦΑΛΑΙΟ 5. Αξιολόγηση αποτελεσμάτων 61
5.1 Σύγκριση αποτελεσμάτων και σχολιασμός 62
5.2 Συμπεράσματα έρευνας 65
5.3 Προτάσεις για μελλοντική έρευνα 66

Βιβλιογραφία 69

Κεφάλαιο 1. Αντικείμενο εργασίας, δομή και στόχος

1.1 Εισαγωγή – θαλάσσια δεδομένα

Δεν υπάρχει καμία αμφιβολία πλέον, ότι ζούμε στην εποχή της πληροφορίας, όπου ολοένα και περισσότερα κομμάτια της ανθρώπινης δραστηριότητας εκτίθενται στο διαδίκτυο. Αυτό έχει ως συνέπεια να παράγεται καθημερινά ένας τεράστιος όγκος δεδομένων, η εκμετάλλευση του οποίου αποτελεί μια πρόκληση για την τεχνολογία της πληροφορικής, τις επιχειρήσεις, το κράτος, την επιστήμη και τον σύγχρονο άνθρωπο εν γένει.

Το μέγεθος των δεδομένων που παράγονται καθημερινά είναι πραγματικά ασύλληπτο. Περίπου 2,5 πεντάκις εκατομμύρια bytes από δεδομένα παράγονται καθημερινώς και ο αριθμός αυτός συνεχώς μεγαλώνει καθώς αυξάνεται η επιρροή του διαδικτύου των πραγμάτων (Internet of Things). Είναι δε εντυπωσιακό ότι την τελευταία διετία παράχθηκε το 90% των δεδομένων που έχει παραγάγει η ανθρωπότητα από την αρχή της καταγεγραμμένης ιστορίας της. Κάποιες από τις κυριότερες πηγές δεδομένων είναι η περιήγηση στο διαδίκτυο (google searches), τα μέσα κοινωνικής δικτύωσης (Facebook, Twitter, YouTube), οι εφαρμογές ανταλλαγής γραπτών μηνυμάτων (What's App, Viber, Skype) και οι διαδικτυακές πλατφόρμες παροχής υπηρεσιών (Spotify, Uber, Airbnb). (1)

Μια ακόμα σημαντική πηγή δεδομένων που συγκεντρώνει το ενδιαφέρον τόσο της επιστημονικής όσο και της επιχειρηματικής κοινότητας αποτελεί η θάλασσα και οι δραστηριότητες γύρω από αυτή.

Ειδικότερα, υπάρχουν παρατηρητήρια ωκεανών τα οποία συλλέγουν γεωχωρικά δεδομένα για μεγάλα χρονικά διαστήματα. Τα παρατηρητήρια αυτά μπορεί να βρίσκονται σε πλοία, δορυφόρους και σε διάφορα Eulerian και Lagrangian συστήματα. Να διευκρινιστεί ότι τα Eulerian και τα Lagrangian συστήματα αποτελούν πλωτούς σταθμούς μέτρησης με τους πρώτους να έχουν σταθερή τοποθεσία ενώ τους δεύτερους να ακολουθούν την ροή του μέσου που βρίσκονται. Τα πλοία αποτελούν βασική πηγή συλλογής ωκεανογραφικών δεδομένων με δυνατότητες πλέον να παίρνουν μετρήσεις από σταθερά και μεταβλητά σημεία αλλά και να χρησιμοποιούν ηχοβολιστικά συστήματα. Οι δορυφόροι αποτελούν ένα καινοτόμο εργαλεία στα χέρια των ωκεανογράφων καθώς δίνουν την δυνατότητα για μελέτη φαινομένων αλληλεπίδρασης γης – αέρα – θάλασσας γεγονός που ήταν αδύνατο παλαιότερα. Ηλεκτροοπτικά καλώδια που τοποθετούνται στην επιφάνεια του βυθού είναι μια σημαντική πηγή δεδομένων και παρέχουν πληροφορία σχετική με την σεισμικότητα του βυθού, τα τσουνάμι, την σύσταση του βυθού ακόμα και για το υποθαλάσσιο οικοσύστημα. Κινούμενες πλατφόρμες τύπου Lagrangian υποβοηθούμενες από μπαταρίες χρησιμοποιούνται για να σχηματίσουν χάρτες με την πορεία των κυμάτων. Τέλος, υπάρχουν αισθητήρες εγκατεστημένοι σε σταθερά βάθη μες τη θάλασσα και παρέχουν δεδομένα υψηλής συχνότητας και συγκεκριμένου βάθους τα οποία υποβοηθούν τις μετρήσεις από πλοία, δορυφόρους και άλλα μέσα. (2)

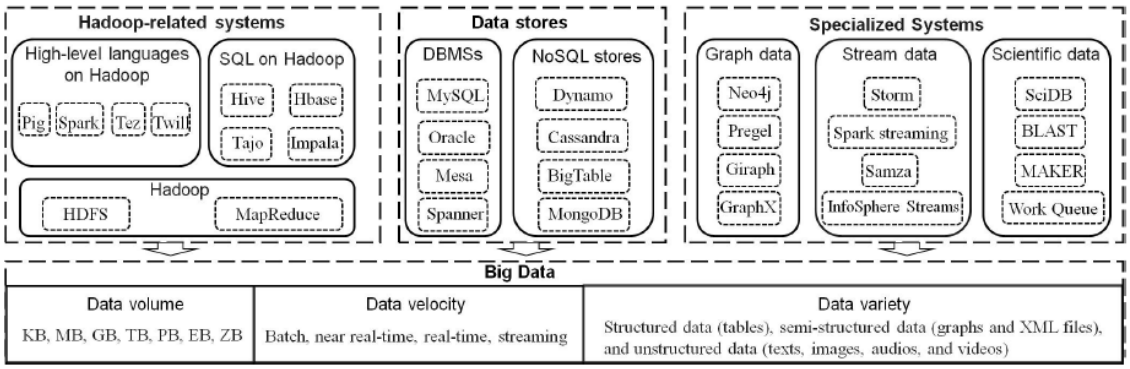
Επιπλέον, η ναυτιλία αποτελεί μια σημαντική πηγή δεδομένων γύρω από την θάλασσα. Μεγάλες ποσότητες δεδομένων παράγονται από συστήματα πλοήγησης πλοίων που

συμπεριλαμβάνουν ραντάρ, αυτόματους πιλότους και διάφορους αισθητήρες. Ακόμη υπάρχουν ειδικά πλοία που χρειάζονται ιδιαίτερο εξοπλισμό για την λειτουργία τους όπως ραντάρ κυμάτων, εντοπιστές πετρελαιοκηλίδων και υψηλής ακρίβειας αισθητήρες πλοήγησης. Πολλά δεδομένα που αφορούν την πλοήγηση του πλοίου αλλά και την κατάσταση λειτουργίας του μεταφέρονται μέσω συστημάτων αυτόματου εντοπισμού πλοίων (Automatic Identification System - AIS) . Συγκεκριμένα αφορούν πληροφορία για τον αριθμό-ταυτότητα του πλοίου, την θέση του, την ταχύτητα και τον προορισμό. Όλα τα πλοία με φορτίο πάνω από 300 τόνους που εκτελούν διεθνή ταξίδια υποχρεούνται να διαθέτουν AIS και να επικοινωνούν με άλλα πλοία ή με επίγειους σταθμούς και δορυφόρους για να αποφεύγονται οι συγκρούσεις. Τέλος, υπάρχει και ο αυτόματος καταγραφέας ταξιδιού (Voyage Data Recorder - VDR) ο οποίος είναι υποχρεωτικός σε όλα τα επιβατικά πλοία και στα εμπορικά με φορτίο μεγαλύτερο από 3000 τόνους. Το VDR καταγράφει όλη την πληροφορία που σχετίζεται με την πορεία του πλοίου και αφορά θέση, ταχύτητα, καταγραφές μικροφώνων από την γέφυρα, επικοινωνία προσωπικού, καιρικές συνθήκες, κατάσταση εξοπλισμού πυρόσβεσης και άλλα. Είναι τόσο μεγάλος ο όγκος της πληροφορίας που καταγράφεται ώστε μετά από κάθε ταξίδι σβήνεται. Μέχρι τώρα η πληροφορία αυτή χρησιμοποιείται για περιπτώσεις ατυχημάτων θα μπορούσε όμως να αξιοποιηθεί αποτελεσματικότερα από τις πλοιοκτήτριες εταιρίες. (3)

1.2 Ερευνητική ανάγκη

Το σύνολο των δεδομένων που παράγεται από τον σύγχρονο άνθρωπο μας βάζει αναμφίβολα στην εποχή των δεδομένων μεγάλης κλίμακας (Big data). Σε αυτή την εποχή, ο αυξανόμενος όγκος των δεδομένων, η απαίτηση για γρήγορη επεξεργασία τους και η μεγάλη ποικιλία στον τύπο, την δομή και την προέλευση των δεδομένων δημιουργούν πολλές και νέες τεχνολογικές προκλήσεις στις οποίες και δεν μπορούν να ανταπεξέλθουν τα παραδοσιακά συστήματα διαχείρισης δεδομένων. Η ανάγκη για αποτελεσματική αποθήκευση και επεξεργασία των δεδομένων οδηγεί στην εμφάνιση και ανάπτυξη νέων τεχνολογιών για Big data οι οποίες χωρίζονται σε τρεις γενικές κατηγορίες (βλ. εικόνα 1.1):

1. Hadoop και συναφείς τεχνολογίες (δηλαδή το οικοσύστημα που έχει αναπτυχθεί γύρω από το Hadoop και περιλαμβάνει γλώσσες υψηλού επιπέδου πάνω από το Hadoop αλλά και SQL στο Hadoop).
2. Συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων και μη σχεσιακές βάσεις δεδομένων (NoSQL) που χρησιμοποιούνται ευρέως από διαδικτυακές εφαρμογές αλλά και για ανάλυση δεδομένων.
3. Εξειδικευμένα συστήματα για την επεξεργασία δεδομένων διασυνδεμένων γράφων, συνεχών ροών δεδομένων (streams) και σύνθετων επιστημονικών δεδομένων.



Εικόνα 1.1: Επισκόπηση τεχνολογιών Big Data

Σε αυτό το πλαίσιο, η αξιολόγηση της απόδοσης των τεχνολογιών αποθήκευσης και ανάκτησης δεδομένων μεγάλης κλίμακας αποκτά καθοριστική σημασία. Και αυτό γιατί έτσι αξιολογούνται οι υπάρχουσες τεχνολογίες μεγάλων δεδομένων και δίνεται το έναυσμα για περεταίρω βελτίωση της απόδοσης τους. Ενώ επιπλέον, εφαρμογές που επεξεργάζονται μεγάλα δεδομένα διευκολύνονται στο να εκτιμήσουν τις ανάγκες τους και με αυτό τον τρόπο προωθείται και από την επιχειρηματική κοινότητα η εξέλιξη των τεχνολογιών αυτών. (4)

1.3 Σκοπός εργασίας – Ερευνητικό ερώτημα

Σε αυτή την εργασία τίθενται τα ακόλουθα ερευνητικά ερωτήματα:

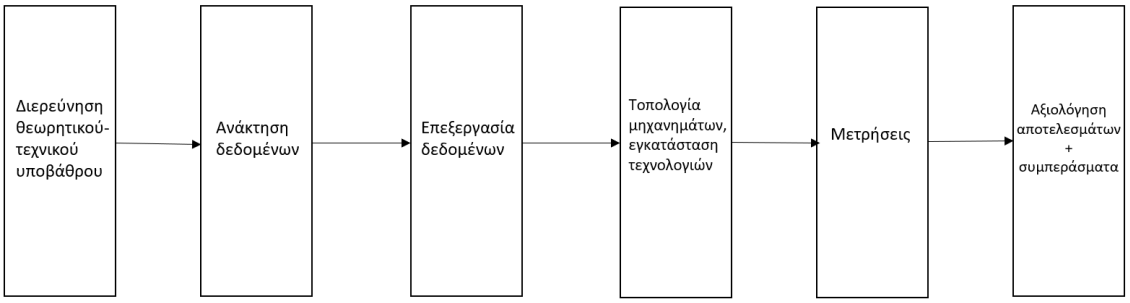
- Δεδομένου ενός συνόλου δεδομένων και μιας γκάμας ερωτημάτων πάνω σε αυτό, ποια τεχνολογία μεγάλων δεδομένων παρουσιάζει καλύτερη απόδοση από άποψη χρόνου απάντησης;
- Πώς ερμηνεύονται οι διαφορές στην απόδοση και από ποιες παραμέτρους επηρεάζεται η κάθε τεχνολογία;

Θα χρησιμοποιηθούν τρεις διαφορετικές τεχνολογίες, οι δύο ανήκουν στο οικοσύστημα του Hadoop και είναι το SparkSQL και το Hive ενώ η τρίτη ανήκει στις μη σχεσιακές βάσεις και είναι η MongoDb. Τα δεδομένα που χρησιμοποιήθηκαν αφορούν τα γεωχωρικά χαρακτηριστικά μιας περιοχής και το μέγεθος τους υπερβαίνει τα 10GB. Κρίνεται απαραίτητη η χρήση πόρων από υπηρεσίες υπολογιστικού νέφους (cloud computing), προκειμένου να δημιουργηθεί ένα διανεμημένο σύστημα που να εκμεταλλεύεται πόρους από περισσότερα από ένα μηχανήματα.

Στόχος της παρούσας διπλωματικής είναι να απαντηθούν τα ερευνητικά ερωτήματα που τέθηκαν χρησιμοποιώντας τις παραπάνω τεχνολογίες και δεδομένα. Παράλληλα θα επιχειρηθεί να δοθεί και το αντίστοιχο θεωρητικό υπόβαθρο.

1.4 Φάσεις υλοποίησης

Η εκπόνηση της διπλωματικής εργασίας πραγματοποιήθηκε μεταξύ Ιουνίου και Φεβρουαρίου 2020 και η πορεία αυτής ακολούθησε τις εξής φάσεις, που παρουσιάζονται παρακάτω στην εικόνα 1.2.



Εικόνα 1.2: Φάσεις υλοποίησης εργασίας

Όπως φαίνεται και στην εικόνα 1.2 αρχικά έγινε μια ανασκόπηση του θεωρητικού υποβάθρου της εργασίας και αναζητήθηκαν όλα τα τεχνικά μέσα που ήταν απαραίτητα για την υλοποίηση της εργασίας. Έπειτα συλλέχθηκαν τα δεδομένα πάνω στα οποία θα εφαρμόζονταν οι μετρήσεις και έγινε η απαραίτητη επεξεργασία τους. Στη συνέχεια αποφασίστηκε η απαραίτητη τοπολογία μηχανημάτων αλλά και οι προδιαγραφές που θα έπρεπε αυτά να πληρούν και εγκαταστάθηκαν οι προς μελέτη τεχνολογίες. Αμέσως μετά διεξήχθησαν τα πειράματα και ελήφθησαν οι μετρήσεις. Τέλος, συγκεντρώθηκαν οι μετρήσεις και αξιολογήθηκαν, εξάγοντας με αυτό τον τρόπο τα απαραίτητα συμπεράσματα.

1.5 Οργάνωση τόμου

Στο 2ο κεφάλαιο της εργασίας γίνεται αναφορά στις έννοιες των Big Data, των διαφορετικών τεχνολογιών μεγάλων δεδομένων και των μεθόδων σύγκρισης αυτών. Επιπλέον, παρουσιάζεται η υπάρχουσα κατάσταση όσο αφορά την συγκριτική αξιολόγηση τεχνολογιών δεδομένων μεγάλης κλίμακας και εντοπίζονται τυχών ερευνητικά κενά του τομέα.

Στο 3ο κεφάλαιο αναλύονται οι τεχνολογίες μεγάλων δεδομένων που χρησιμοποιήθηκαν για την συγκριτική αξιολόγηση. Πρόκειται για το Parquet format, το Apache Hadoop-hadoop filesystem, το Apache Hive, το Apache Spark/SparkSQL και την MongoDb.

Στο 4ο κεφάλαιο περιγράφεται το πείραμα που διεξήχθη. Παρουσιάζονται τα δεδομένα και η προεργασία τους, η τοπολογία μηχανημάτων που χρησιμοποιήθηκε καθώς και η διαδικασία με την οποία κάθε τεχνολογία τέθηκε σε λειτουργία. Τέλος, αναφέρονται

όποιες παραδοχές κρίθηκαν αναγκαίες για την διεξαγωγή του πειράματος και παρατίθενται τα αποτελέσματα των μετρήσεων.

Κεφάλαιο 2. Θεωρητικό υπόβαθρο τεχνολογιών δεδομένων μεγάλης κλίμακας και συγκριτική αξιολόγηση

2.1 Δεδομένα μεγάλης κλίμακας – Ορισμός

Ο όρος δεδομένα μεγάλης κλίμακας ή Big Data , όπως είναι ευρύτερα γνωστός χάρις την αγγλική βιβλιογραφία, χρησιμοποιείται από την δεκαετία του 1990. Με τον όρο Big Data αναφερόμαστε σε σύνολα δεδομένων με μέγεθος που το υπάρχον λογισμικό δεν μπορεί να ανακτήσει, να διαχειριστεί και να επεξεργαστεί μέσα σε ένα ανεκτό χρονικό διάστημα. Το «μέγεθος» των Big Data συνεχώς μεταβάλλεται και για το 2012 ήταν από κάποιες δεκάδες terabyte έως πολλά zettabyte δεδομένων.

Σύμφωνα με έναν ορισμό του 2018, ένα σύνολο δεδομένων χαρακτηρίζεται ως big data όταν χρειάζονται τεχνικές παράλληλου προγραμματισμού για οποιαδήποτε ενέργεια πάνω σε αυτά. Επισημαίνεται δε ότι με την χρήση παράλληλου προγραμματισμού συντελείται μια σαφής και ριζική αλλαγή στο σχεσιακό μοντέλο δεδομένων του Codd. (5)

Τα Big Data χαρακτηρίζονται πολύ εύστοχα από το αποκαλούμενο μοντέλο 3V, όπου οι όροι Volume (όγκος), Velocity (ταχύτητα) και Variety (ποικιλία) έχουν κυρίαρχο ρόλο. Τελευταία, προστίθεται και ένα τέταρτο V στο μοντέλο αυτό και αφορά το Veracity (εγκυρότητα).

Εικόνα 2.1: Τα 3+1V των Big Data: Volume, Variety, Velocity και Veracity

Το *Volume (όγκος)* αναφέρεται στο πλήθος των δεδομένων που είναι διαθέσιμα σε κάθε επιχείρηση/οργανισμό τα οποία δεν είναι απαραίτητο να της ανήκουν όλα αφού μπορεί να έχει πρόσβαση και σε δεδομένα εκτός αυτής. Όσο ο όγκος των δεδομένων αυξάνεται τόσο η αξία του πλήθους των διαφορετικών εγγραφών θα μειώνεται σε σύγκριση με την παλαιότητα, τον τύπο και την ποικιλία.

Με το *Velocity (ταχύτητα)* εννοούμε την ταχύτητα με την οποία τα δεδομένα δημιουργούνται, μεταδίδονται και συλλέγονται για επεξεργασία. Μάλιστα, δεν επαρκεί μόνο να είμαστε σε θέση να επεξεργαζόμαστε τα δεδομένα και να εξάγουμε πληροφορία σε πραγματικό χρόνο, αλλά επιπλέον είναι απαραίτητο να εκτελούμε και όλες τις λειτουργίες που ενεργοποιούνται από αυτά, σε πραγματικό χρόνο, ώστε να μην καθυστερεί η όλη διαδικασία. Για παράδειγμα μια εφαρμογή που παρακολουθεί τις τιμές των εταιρειών στο χρηματιστήριο, δεν αρκεί να μπορεί να τις καταγράφει, αλλά πιθανώς να θέλουμε και να αγοράζει ή να πουλάει μια μετοχή όταν αυτή ξεπερνά κάποια τιμή.

Με το *Variety (ποικιλία)* εννοούμε το μεγάλο εύρος πιθανών διαφορετικών τύπων δεδομένων που καλούμαστε να χειριστούμε (κείμενο, εικόνες, βίντεο, ήχος). Από την πλευρά του αναλυτή, η ποικιλία αποτελεί το μεγαλύτερο εμπόδιο στο να εκμεταλλευτεί μεγάλο όγκο δεδομένων. Ασύμβατοι τύποι δεδομένων, ασύνδετες δομές δεδομένων και ασυνεπείς σημασιολογία θέτουν σημαντικά εμπόδια που μπορεί να οδηγήσουν σε σύγχυση τον αναλυτή. (6)

Τέλος, με τον όρο *Veracity (εγκυρότητα)* αναφερόμαστε στο πόσο ακριβή ή αληθή είναι τα δεδομένα που έχουμε στη διάθεση μας. Η εγκυρότητα των δεδομένων είναι πολύ σημαντική γιατί μας βοηθά να καταλάβουμε τι είναι σημαντικό και τι όχι καθώς και να βγάλουμε χρήσιμα συμπεράσματα. Όταν αναφερόμαστε σε big data η εγκυρότητα δεν έχει να κάνει μόνο με την αξιοπιστία των δεδομένων αλλά και με την σωστή επεξεργασία τους ώστε να αφαιρεθεί ο θόρυβος, διπλότυπα, ασυνεπή δεδομένα και τυχόν προκατάληψη στην μέτρηση. Επιπλέον, η επεξεργασία των δεδομένων θα πρέπει να είναι τέτοια ώστε να αποκτά νόημα ανάλογα με τις ανάγκες της επιχείρησης/οργανισμού. (7)

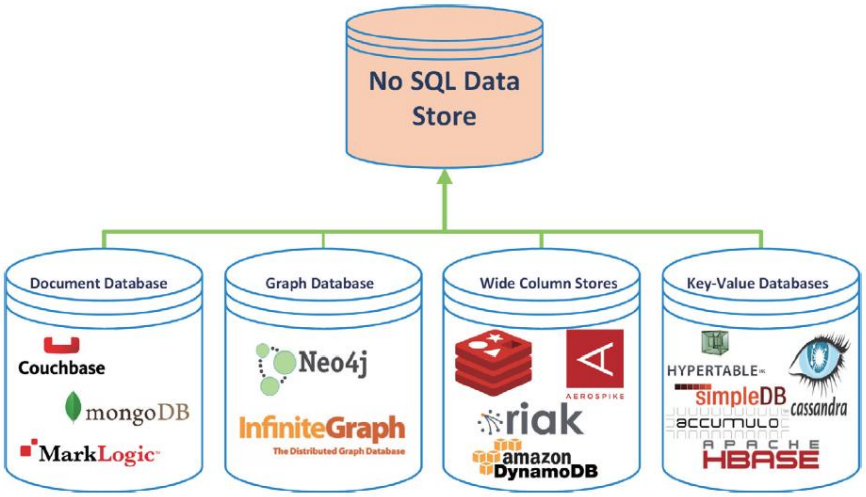
2.2 Τεχνολογίες αποθήκευσης και επεξεργασίας δεδομένων μεγάλης κλίμακας

Σε αυτή την παράγραφο θα αναφερθούμε στις διαφορετικές κατηγορίες τεχνολογιών που έχουν αναπτυχθεί γύρω από τα big data.

1. Μη σχεσιακές βάσεις δεδομένων – NoSQL

Παραδοσιακά το σχεσιακό μοντέλο κυριαρχούσε στην αποθήκευση και ανάκτηση δεδομένων. Οι αυξανόμενες όμως ανάγκες για δυνατότητα επέκτασης της δυναμικότητας των βάσεων δεδομένων μαζί με ανάγκες που προέκυπταν από τις διαδικτυακές εφαρμογές, έθεσαν προκλήσεις στο σχεσιακό μοντέλο. Συγκεκριμένα, η αύξηση των δεδομένων που παράγονται από χρήστες του διαδικτύου δημιούργησε αύξηση στον τύπο και τον όγκο των δεδομένων που πρέπει να διαχειριστεί μια βάση δεδομένων. Τέτοιου είδους δεδομένα δεν έχουν προκαθορισμένο σχήμα και δομή και αυτό δημιουργεί μεγάλο πρόβλημα στις σχεσιακές βάσεις που απαιτούν συγκεκριμένο σχήμα και συσχετίσεις μεταξύ των οντοτήτων. Πλέον σε αρκετές εφαρμογές η γνώση του σχήματος των δεδομένων δεν είναι γνωστή εκ των προτέρων. Προκειμένου να αντιμετωπιστούν αυτά τα προβλήματα εμφανίστηκαν οι μη σχεσιακές βάσεις δεδομένων γνωστές και ως NoSQL (Not Only SQL – όχι μόνο SQL).

Οι μη σχεσιακές βάσεις δεδομένων χωρίζονται σε τέσσερεις κατηγορίες ανάλογα με τον τρόπο αποθήκευσης των δεδομένων. Αναλυτικότερα έχουμε τις key-value pairs (ζεύγη κλειδιού-τιμής), document stores (αποθήκευση εγγράφων), column families stores (κατά στήλη αποθήκευση) και graph stores (αποθήκευση σε μορφή γράφου).



Εικόνα 2.2: Διαφορετικές κατηγορίες NoSQL βάσεων

Στην κατηγορία key-value pairs βάσεων κυριαρχεί η ιδέα της ύπαρξης ενός hashtable όπου ο χρήστης αποθηκεύει δεδομένα-τιμές (values) τα οποία δεικτοδοτούνται από κάποιο μοναδικό κλειδί (key). Δεν είναι απαραίτητη η ύπαρξη ενός αυστηρού σχήματος για τα δεδομένα που αποθηκεύονται, ενώ για την επίτευξη μεγαλύτερης κλιμακωσιμότητας και απόδοσης του συστήματος γίνεται χρήση μηχανισμών caching και δεν υποστηρίζονται συνενώσεις (joins) και συναθροιστικές (aggregate) λειτουργίες. Κάποιες βάσεις τέτοιου είδους είναι οι MemcacheDB[1], Voldemort[2], Redis[3] και Riak[4].

Οι document stores βάσεις μοιάζουν με τις key-value αλλά υποστηρίζουν πιο πολύπλοκα δεδομένα. Τα δεδομένα αυτά (“έγγραφα”) δεν έχουν αυστηρή δομή και έτσι οι χρήστες έχουν τη δυνατότητα να προσθέτουν και να αφαιρούν πεδία κατά βούληση. Όπως και στις key-value βάσεις, κάθε έγγραφο δεικτοδοτείται από ένα κλειδί (key) με το οποίο μπορεί ο χρήστης να το ανακτήσει, και επιπλέον, σε πολλές περιπτώσεις υπάρχει διαθέσιμο API ή query language (γλώσσα ερωτημάτων) ώστε να διευκολύνεται η ανάκτηση εγγράφων με βάση το περιεχόμενό τους. Οι πιο δημοφιλείς βάσεις του είδους είναι οι MongoDB[5], CouchDB[6] και Cassandra[7].

Στις column family stores βάσεις υπερισχύει η ιδέα της αποθήκευσης και επεξεργασίας δεδομένων κατά στήλη και όχι κατά γραμμή. Κύριος στόχος τους είναι να μπορούν να διαχειριστούν πολύ μεγάλο όγκο δεδομένων που βρίσκονται κατανεμημένα σε

[1] http://memcached.org/.

[2] http://www.project-voldemort.com/voldemort/.

[3] http://redis.io/.

[4] http://basho.com/riak/.

[5] http://www.mongodb.org/.

[6] http://couchdb.apache.org/.

[7] http://cassandra.apache.org/.

διάφορους εξυπηρετητές. Και εδώ γίνεται χρήση κλειδιών (rows), που δείχνουν όμως σε μία ή περισσότερες οικογένειες κολώνων (column families). Τυπικά παραδείγματα τέτοιων συστημάτων είναι τα BigTable[8], Hbase[9] και Accumulo[10].

Στις graph stores βάσεις γίνεται χρήση κόμβων (nodes) και ακμών (edges) για την αναπαράσταση και αποθήκευση των δεδομένων. Οι γράφοι δηλαδή αντικαθιστούν το κλασσικό μοντέλο των πινάκων. Το μοντέλο γραφημάτων που ακολουθείται ευνοεί την οριζόντια κλιμάκωση αφού μπορεί να χρησιμοποιηθεί παράλληλα σε πολλούς εξυπηρετητές. (8) (9)

2. Συστήματα επεξεργασίας κατανεμημένων δεδομένων (Batch processing systems)

Το Hadoop[11] κυριαρχεί στην περιοχή της επεξεργασίας δεδομένων σε κατανεμημένα συστήματα (distributed systems). Επικεντρώνεται κυρίως στο να επιτύχει κλιμακωτή απόδοση σε μεγάλα διασυνδεδεμένα υπολογιστικά συστήματα (cluster) με πολλούς κόμβους (nodes). Η υπολογιστική μηχανή του Hadoop εκτελεί υπολογισμούς σε δύο φάσεις, την συλλογή (Map) και την ελαχιστοποίηση (Reduce), ενώ σε κάθε φάση από ένα ζεύγος κλειδιού-τιμής παράγεται ένα ζεύγος με το αποτέλεσμα. Επειδή σε κάθε υπολογιστικό στάδιο κρατούνται στην μνήμη οι ενδιάμεσοι υπολογισμοί, δημιουργείται πρόβλημα στην απόδοση του Hadoop. Για αυτό τον λόγο έχουν εμφανιστεί και άλλα συστήματα που στοχεύουν στο να ξεπεράσουν αυτούς τους περιορισμούς στην απόδοση.

Ένα παράδειγμα είναι το Spark[12], το οποίο έχει δανειστεί από το Hadoop την επεξεργασία δεδομένων στην κύρια μνήμη αλλά χρησιμοποιεί υπολογιστικά φθηνότερες διαδικασίες ανακατάταξης των δεδομένων που επεξεργάζεται. Η βασική ιδέα γύρω από το Spark είναι το Resilient Distributed Dataset (RDD) που αναπαριστά μια συλλογή δεδομένων που είναι διανεμημένη στους κόμβους ενός υπολογιστικού cluster.

3. Συστήματα SQL για big data (Big SQL systems)

Όπως αναφέρθηκε και προηγουμένως τα συστήματα επεξεργασίας κατανεμημένων δεδομένων στα οποία κυριαρχεί το Hadoop παρουσιάζουν σημαντικά προβλήματα απόδοσης. Επίσης έρευνες έχουν δείξει ότι το Hadoop δεν αποτελεί το κατάλληλο εργαλείο για ερωτήματα τύπου SQL όταν ο χρόνος απάντησης πρέπει να είναι από

[8] https://cloud.google.com/bigtable/.

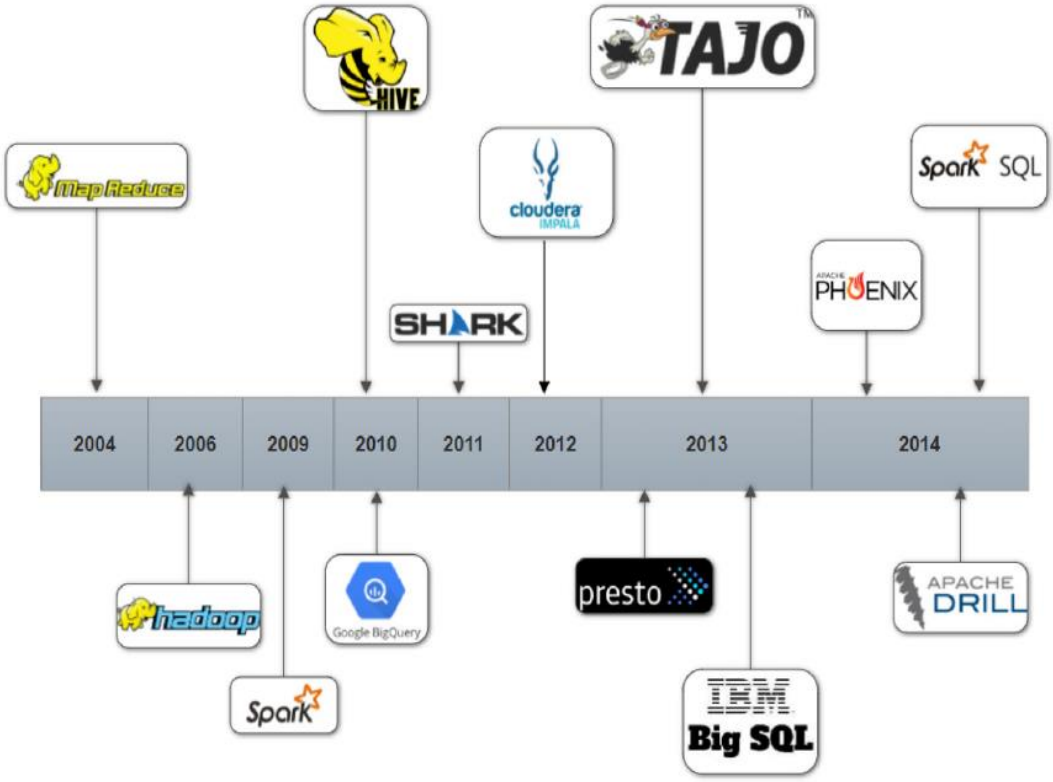
[9] https://hbase.apache.org/.

[10] https://accumulo.apache.org/.

[11] https://hadoop.apache.org/.

[12] https://spark.apache.org/.

κάποια χιλιοστά του δευτερολέπτου έως κάποια δευτερόλεπτα. Συν τοις άλλοις, οι προγραμματιστές είναι εξοικειωμένοι με την χρήση της SQL για ερωτήματα πάνω σε σύνολα δεδομένων. Έτσι, έκαναν την εμφάνιση τους διάφορα συστήματα, που είναι ευρέως γνωστά ως Big SQL, τα οποία δίνουν στον χρήστη τους μια διεπαφή σε μορφή SQL και χρησιμοποιούν είτε γενικές πλατφόρμες επεξεργασίας μεγάλων δεδομένων όπως το Hadoop και το Spark είτε δικές τους μηχανές επεξεργασίας δεδομένων ώστε να παρέχουν καλύτερη απόδοση. Στην εικόνα 2.3 φαίνεται η ιστορική εξέλιξη των τεχνολογιών αυτών.



Εικόνα 2.3: Ιστορική εξέλιξη Big SQL συστημάτων

Η πρώτη Big SQL τεχνολογία που αναπτύχθηκε ήταν το Apache Hive[13] και είχε ως σκοπό να διαχειρίζεται δεδομένα που βρίσκονται στο Hadoop. Με το Hive διευκολύνεται η επεξεργασία μεγάλου όγκου δεδομένων και γίνεται χρήση και μιας παρόμοιας με την SQL γλώσσας ερωτημάτων, της HiveQL.

Πιο πρόσφατα αναπτύχθηκε η SparkSQL[14] η οποία και αποτελεί τμήμα του Apache Spark δίνοντας σε αυτό μια SQL διεπαφή. Χρησιμοποιεί την έννοια του dataframe το οποίο είναι ισοδύναμο με τον πίνακα του σχεσιακού μοντέλου και πάνω σε αυτό αναπτύσσει μια SQL γλώσσα.

[13] https://cwiki.apache.org/confluence/display/Hive/LanguageManual.

[14] https://spark.apache.org/sql/.

Μια ακόμα big SQL τεχνολογία είναι και το Apache Impala, το οποίο χρησιμοποιεί μια δική του μηχανή επεξεργασίας δεδομένων, την Passively Parallel Processing (MPP) και εφαρμόζεται πάνω στο Hadoop. Επίσης υποστηρίζει ερωτήματα σε SQL πάνω σε δεδομένα που είναι αποθηκευμένα είτε στο Hadoop filesystem είτε στο Apache HBase.

Τέλος υπάρχει και το PrestoDb[15], το οποίο εισήχθη ως big SQL τεχνολογία από την Facebook και μπορεί να αναλύσει γρήγορα και άμεσα μεγάλο όγκο δεδομένων.

4. Συστήματα επεξεργασίας ροών δεδομένων για big data (Big stream processing systems)

Ένα παράδειγμα επεξεργασίας δεδομένων μεγάλης κλίμακας που κερδίζει έντονο ενδιαφέρον τελευταία λόγω της ανάπτυξης αντίστοιχων εφαρμογών είναι η επεξεργασία ροών δεδομένων. Το Apache Storm[16] είναι ένα τέτοιο παράδειγμα και έχει δώσει το έναυσμα για δημιουργία μηχανών επεξεργασίας ροών δεδομένων ανθεκτικών σε λάθη. Άλλο ένα παράδειγμα είναι το Apache Flink[17], που έχει την δυνατότητα να επεξεργάζεται ροές δεδομένων που προέρχονται από διαφορετικές πηγές χρησιμοποιώντας έναν σχεσιακό πίνακα ο οποίος και γεμίζει με δεδομένα συνεχόμενα. Τέλος και το Apache Kafka[18] ανήκει σε αυτή την κατηγορία.

5. Συστήματα επεξεργασίας δεδομένων γράφων για big data (Big graph processing systems)

Οι αλγόριθμοι που χρησιμοποιούνται στην επεξεργασία γράφων έχουν περιηγητική και επαναληπτική φύση αφού γενικά μιλώντας πρέπει να διασχίζουν ένα γράφο. Θα μπορούσαν να εκτελεστούν ως διεργασίες στο Hadoop η μια μετά την άλλη, αλλά θα απαιτούντο να ανταλλάσσεται όλος ο γράφος κατά τα στάδια της επεξεργασίας. Αυτός ο τρόπος όπως είναι φανερό είναι αργός και αναποτελεσματικός. Για αυτόν τον λόγο αναπτύχθηκε λογισμικό που μπορεί να εκτελεί αποτελεσματικά επαναληπτικούς αλγόριθμους πάνω σε γράφους σε ένα cluster υπολογιστών. Συγκεκριμένα το 2010 η Google παρουσίασε το Pregel, το οποίο έδινε στους χρήστες του ένα API που τους ενθάρρυνε να γράφουν κώδικα σκεπτόμενοι με «κόμβους» και «συνδέσεις». Άλλα τέτοια παραδείγματα είναι το Giraph[19] και το Apache Hama[20]. (10)

[15] https://prestodb.io/.

[16] https://storm.apache.org/.

[17] https://flink.apache.org/.

[18] https://kafka.apache.org/.

[19] http://giraph.apache.org/.

[20] http://hama.apache.org/.

2.3 Εργαλεία αξιολόγησης της επίδοσης τεχνολογιών

Σε αυτή την παράγραφο πρόκειται να γίνει αναφορά στα σημαντικότερα εργαλεία που έχουν αναπτυχθεί για συγκριτική αξιολόγηση (benchmarking) των διαφορετικών big data τεχνολογιών.

Στην κατηγορία των NoSQL βάσεων, το πρώτο benchmark που εμφανίστηκε ήταν το ***YCSB (Yahoo! Cloud Serving Benchmark)*** το 2010 από τους Cooper et al. Στο benchmark περιλαμβάνεται γεννήτρια φορτίου και ένα σύνολο από προκαθορισμένες ενέργειες όπως διάβασμα πολλών δεδομένων (read heavy), ενημέρωση πολλών δεδομένων (update heavy), ταξινόμηση, διάβασμα των πιο πρόσφατων δεδομένων, διάβασμα – επεξεργασία – γράψιμο και άλλα. Με αυτό το benchmark αξιολογούνται τα χαρακτηριστικά των τεχνολογιών αυτών και αφορούν την δυνατότητα κλιμάκωσης (scalability), την διαθεσιμότητα (availability) και την δυνατότητα δημιουργίας αντιγράφων (replication). Χρησιμοποιήθηκε δε από τους Cooper et al για την σύγκριση των HBase, Cassandra, Yahoo! PNUTS και μιας υλοποίησης της MySQL που εμπεριέχει την δυνατότητα διαμοιρασμού φορτίου. (11)

Έχουν κυκλοφορήσει επίσης παραλλαγές του YCSB με βελτιωμένα χαρακτηριστικά.

Οι Patil et al παρουσίασαν το ***YCSB++*** benchmark το οποίο εστιάζει στην βελτίωση της διαδικασίας εκσφαλμάτωσης (debugging) του benchmark και στην καλύτερη κατανόηση κάποιων σύνθετων χαρακτηριστικών των NoSQL βάσεων. Επιπλέον υποστηρίζει φορτίο σε παραπάνω από ένα μηχανήματα. Συγκεκριμένα, για την διευκόλυνση του debugging τα αρχεία καταγραφής (logs) του benchmark συλλέγονται σε ένα κεντρικό εργαλείο και συνδυάζονται με αντίστοιχα logs της βάσης αλλά και του υπολογιστικού συστήματος, δίνοντας έτσι μια πιο καθαρή ματιά στον χρήστη του benchmark. Σχεδιάστηκε από τους Patil et al για να εφαρμοστεί στις Accumulo και Hbase, διαθέτει όμως API που καλύπτει και αρκετές άλλες NoSQL βάσεις. (12)

Οι Dey et al επέκτειναν το YCSB με την δημιουργία του ***YCSB+T***, το οποίο εστιάζει στην ορθότητα των δεδομένων και στην υποστήριξη συναλλαγών (transactions) για κάθε συνδιαλλαγή με την βάση. Αναλυτικότερα, επιτρέπει επιπρόσθετες ενέργειες πέρα από τις κλασσικές του YCSB για διάβασμα, διαγραφή, ανανέωση και γράψιμο. Όλες οι ενέργειες γίνονται μέσα στα πλαίσια ενός transaction. Τέλος υπάρχει και ένα στάδιο όπου γίνονται έλεγχοι συνέπειας των δεδομένων μετά από την εκτέλεση κάθε ενέργειας για να ανιχνευτούν και να ποσοτικοποιηθούν τυχών ανωμαλίες στα transaction. (13)

Για *συγκριτική αξιολόγηση σε συστήματα επεξεργασίας κατανεμημένων δεδομένων* υπάρχει μεταξύ των άλλων το HiBench για το Hadoop και Sparkbench για το Spark.

Το ***Hibench*** περιλαμβάνει ένα σύνολο από εφαρμογές Hadoop είτε ειδικά φτιαγμένες για το benchmark όπως Sort, WordCount, TeraSort είτε από πραγματικές εφαρμογές όπως αναζήτηση στο διαδίκτυο και μηχανική μάθηση. Με την εκτέλεση αυτών των σεναρίων αξιολογείται η απόδοση των διαφόρων συστημάτων με όρους ταχύτητας, ικανότητας και χρησιμοποίησης πόρων. (14)

Το ***Sparkbench*** είναι ειδικά φτιαγμένο ώστε να καλύπτει ένα μεγάλο σύνολο από Spark εφαρμογές. Συγκεκριμένα περιλαμβάνει εφαρμογές για μηχανική μάθηση, υπολογισμό σε γράφους, SQL ερωτήματα και εφαρμογές που διαχειρίζονται ροές δεδομένων. Επιπλέον, διαθέτει μια γκάμα από κριτήρια για να βοηθήσει τους χρήστες να συγκρίνουν χρησιμοποιώντας τις διαφορετικές παραμετροποιήσεις, βελτιστοποιήσεις και cluster τοπολογίες του Spark. Μερικά από τα κριτήρια είναι ο χρόνος εκτέλεσης μιας εργασίας και ο ρυθμός επεξεργασίας δεδομένων σε MB/second. (15)

Όσο αφορά την *συγκριτική αξιολόγηση στα big SQL* συστήματα κυριαρχούν τα benchmarks της TCP (Transactions Processing Performance Council), ενός μη κοινωφελούς οργανισμού που στόχο έχει την δημιουργία πιστοποιημένων benchmark για συστήματα βάσεων δεδομένων. Συγκεκριμένα, θα αναφερθούμε στα TPC-H, TPC-DS και TPCx-BB.

Το ***TPC-H*** είναι πολύ δημοφιλές στον χώρο των βάσεων δεδομένων και της επεξεργασίας συναλλαγών (transaction processing). Περιέχει ένα σύνολο από πιθανά ερωτήματα που θα ενδιέφεραν μια επιχείρηση καθώς και ένα σύνολο από ταυτόχρονες ενέργειες πάνω στα δεδομένα. Αναλυτικότερα, τα δεδομένα οργανώνονται σε 8 σχεσιακού τύπου πίνακες και πάνω σε αυτά εκτελούνται 22 SQL ερωτήματα. Επιπλέον, παρέχεται μια γεννήτρια δεδομένων που παράγει τα δεδομένα που είναι απαραίτητα για την εκτέλεση του benchmark. (16)

Το ***TPC-DS*** επικεντρώνεται στα συστήματα επιχειρηματικής ευφυίας, όπου τα δεδομένα που παράγονται από τις διεργασίες πρέπει να αναλύονται και να καθοδηγούν τις επιχειρήσεις να λάβουν αποφάσεις σε πραγματικό χρόνο. Μερικές φορές χρειάζεται εξίσου καλά η ανάλυση των δεδομένων να δίνει και μια καθοδήγηση για μακροπρόθεσμες αποφάσεις. Τα δεδομένα του benchmark είναι οργανωμένα σε 7 πίνακες γεγονότων, 1 πίνακα 17 διαστάσεων και 99 ερωτήματα επ' αυτών. Και εδώ συμπεριλαμβάνεται μια γεννήτρια δεδομένων με δυνατότητα επιλογής του τελικού όγκου των δεδομένων. Η αξιολόγηση της απόδοσης της βάσης γίνεται μετρώντας τον χρόνο απάντησης ερωτημάτων για ένα χρήστη, την ικανότητα εξυπηρέτησης ερωτημάτων από πολλούς χρήστες και η απόδοση όταν εκτελείται συντήρηση δεδομένων δοθέντων των τεχνικών προδιαγραφών των υπολογιστικών συστημάτων που χρησιμοποιούνται. (17)

Το ***TPCx-BB*** δημιουργήθηκε ως συνδυασμός των TPC-DS και BigBench το 2013. Το συγκεκριμένο benchmark υποστηρίζει αδόμητα, ημιδομημένα και πλήρως δομημένα δεδομένα. Το κομμάτι των δομημένων δεδομένων του TPCx-BB βασίζεται στο TPC-DS και προσομοιάζει μια εφαρμογή επιχείρησης για υποβοήθηση στη λήψη αποφάσεων. Το κομμάτι με τα ημιδομημένα δεδομένα χρησιμοποιεί ως μοντέλο τα κλικ ενός χρήστη σε μια ιστοσελίδα και επιδιώκει να αναλύσει την συμπεριφορά του χρήστη. Ενώ, το κομμάτι με τα αδόμητα δεδομένα έχει ως μοντέλο τις αξιολογήσεις πελατών σε προϊόντα και στοχεύει στην ανάλυση συναισθήματος. Τα δομημένα δεδομένα αποτελούν το 20% του συνόλου. Το TPCx-BB benchmark αποτελείται από τρία ανεξάρτητα στάδια εκτέλεσης τα οποία και εκτελούνται διαδοχικά χωρίς εξωτερικές διακοπές. Στο πρώτο στάδιο δημιουργείται η βάση, παράγονται τα

δεδομένα και γεμίζουν οι πίνακες της βάσης, στο δεύτερο στάδιο εκτελούνται όλες οι εργασίες στο μέγιστο φορτίο και στο τρίτο στάδιο γίνεται έλεγχος της ικανότητας του συστήματος να ανταπεξέρχεται σε πολλαπλούς χρήστες την ίδια χρονική στιγμή. (18)

Τέλος, για *συγκριτική αξιολόγηση σε συστήματα επεξεργασίας ροών δεδομένων* αναφέρεται ότι υπάρχει το StreamBench και το Yahoo! Streaming Benchmark.

2.4 Σχετικές Μελέτες

Σε αυτή την παράγραφο θα επικεντρωθούμε στα συμπεράσματα μελετών που έχουν διεξαχθεί και συμπεριλαμβάνουν τις τεχνολογίες που θα χρησιμοποιήσουμε στην έρευνα μας, δηλαδή τις mongoDb, SparkSQL και Hive. Η πρώτη ανήκει στην κατηγορία των NoSQL οπότε στην βιβλιογραφία συγκρίνεται με άλλες της κατηγορίας, ενώ οι άλλες δύο βρίσκονται στην ίδια κατηγορία των big SQL και συγκρίνονται μεταξύ τους.

Οι Abramova και Bernardino στην έρευνα τους συνέκριναν την mongoDb με την Cassandra χρησιμοποιώντας το YCSB benchmark. Τα αποτελέσματα έδειξαν ότι με την αύξηση του μεγέθους των δεδομένων η mongoDb άρχισε να δείχνει μειωμένη απόδοση και μερικές φορές τα αποτελέσματα ήταν ιδιαίτερα φτωχά. Από την άλλη, η Cassandra έδειξε πιο γρήγορες αποκρίσεις με την αύξηση των δεδομένων. Επιπλέον και στις ενέργειες που αφορούν την ενημέρωση δεδομένων (update) η Cassandra είναι γρηγορότερη παρουσιάζοντας μειωμένους χρόνους εκτέλεσης και μάλιστα ανεξάρτητα του συνολικού μεγέθους των δεδομένων. Συνεπώς, η Cassandra υπερτερεί σε όλα τα σενάρια της mongoDb σε ότι αφορά την απόδοση. (19)

Οι Swaminathan και Elmasri χρησιμοποίησαν και αυτοί το YCSB benchmark για να συγκρίνουν τις mongoDb, Cassandra και HBase. Στις περισσότερες ενέργειες η Cassandra παρουσιάζει την καλύτερη απόδοση. Εξαίρεση αποτελεί η περίπτωση που απαιτείται μόνο ανάκτηση δεδομένων (read) όπου η mongoDb υπερτερεί και η περίπτωση που γίνονται τυχαίες εγγραφές (blind writes) όπου η HBase έχει καλύτερη απόδοση. Επίσης, σε σχετικά μικρές βάσεις (κάποια GB) η Cassandra έχει καλύτερη συμπεριφορά σε σχέση με την HBase. (20)

Οι Aluko και Sakr με την έρευνα τους συνέκριναν Hive, SparkSQL, Impala και PrestoDb χρησιμοποιώντας και τα τρία benchmark της TPC (TPC-H, TPC-DS, TPCx-BB). Τα αποτελέσματα του TPC-H έδειξαν ότι το Hive και το SparkSQL είχαν την χειρότερη επίδοση για όλα τα ερωτήματα όταν είχαν τα δεδομένα τους αποθηκευμένα ως αρχεία κειμένου. Αξίζει να σημειωθεί ότι το Hive ήταν σημαντικά αργότερο από SparkSQL, φανερώνοντας έτσι τα προβλήματα απόδοσης που αντιμετωπίζει το Hadoop (η μηχανή εκτέλεσης του Hive) σε σύγκριση με το Spark (η μηχανή εκτέλεσης του SparkSQL). Με την χρήση ORC αρχείων το Hive είχε χειρότερη απόδοση ενώ το SparkSQL σε κάποιες περιπτώσεις έδειξε αρκετά βελτιωμένη απόδοση. Εκτελώντας το TPC-DS συμπεραίνεται ότι το Impala έχει την πιο γρήγορη απόκριση ενώ το Hive είναι το πιο αργό. Το SparkSQL και το Presto παρουσιάζουν μια παρόμοια καλή

συμπεριφορά. Με κατάλληλη παραμετροποίηση κατάφεραν να ολοκληρωθούν και τα 99 ερωτήματα του benchmark. Τέλος, το TPCx-BB benchmark είναι συμβατό μόνο με το Hive και το SparkSQL και για αυτό τον λόγο εκτελέστηκε στα δύο αυτά συστήματα μόνο. Τα αποτελέσματα και εδώ έδειξαν ότι το SparkSQL έχει καλύτερη απόδοση από το Hive. Ένας σημαντικός περιορισμός που τίθενται στο Hive είναι ότι κατά την διάρκεια εκτέλεσης των διαφόρων υπολογιστικών σταδίων του τα ενδιάμεσα αποτελέσματα γράφονται στον δίσκο εισάγοντας έτσι μια σημαντική χρονική καθυστέρηση. (21)

Αξίζει να σημειωθεί ότι δεν υπάρχει κάποιο απόλυτο benchmark που να αξιολογεί όλες τις τεχνολογίες το ίδιο καλά και να ταιριάζει στις ανάγκες όλων των διαφορετικών εφαρμογών. Έτσι κατά περίπτωση και ανάλογα με τις εκάστοτε απαιτήσεις επιλέγεται κάποιο benchmark. Αυτό που κατά την άποψη του συγγραφέα λείπει από την βιβλιογραφία είναι ένα benchmark που να συγκρίνει NoSQL βάσεις με big SQL τεχνολογίες. Αυτό το κενό θα προσπαθήσουμε να καλύψουμε στην παρούσα διπλωματική εργασία.

Κεφάλαιο 3. Τεχνικό υπόβαθρο τεχνολογιών δεδομένων μεγάλης κλίμακας

Σκοπός του κεφαλαίου αυτού είναι ο αναγνώστης να αποκτήσει μια καλύτερη άποψη για την αρχιτεκτονική και λειτουργία όλων των τεχνολογιών που χρησιμοποιήθηκαν στην παρούσα διπλωματική.

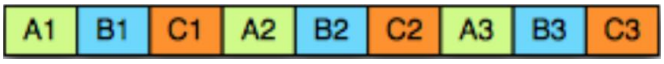
3.1 Αποθήκευση αρχείων δεδομένων

Η αποθήκευση αρχείων κατά στήλη είναι μια πολύ δημοφιλής τεχνική για βελτιστοποίηση ανάλυσης δεδομένων σε σχεσιακές βάσεις δεδομένων που λειτουργούν σε πολλά μηχανήματα παράλληλα. Είναι επίσης γνωστό ότι ως τρόπος αποθήκευσης πλεονεκτεί σε συμπίεση δεδομένων και σε απόδοση στην επεξεργασία μεγάλου όγκου δεδομένων.

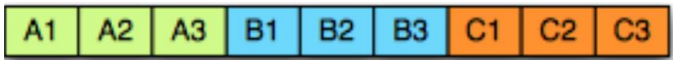
Η ιδέα πίσω από την κατά στήλη αποθήκευση είναι σχετικά απλή, αντί του παραδοσιακού τρόπου που τα δεδομένα γράφονται κατά γραμμή τώρα γράφεται κάθε στήλη ολόκληρη η μια μετά την άλλη. Για να γίνει καλύτερα κατανοητό στις εικόνες 3.1, 3.2, 3.3 φαίνεται ένας πίνακας τριών στηλών και οι αντίστοιχες διατάξεις για αποθήκευση κατά γραμμή και κατά στήλη.

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Εικόνα 3.1: Πίνακας με τρείς στήλες



Εικόνα 3.2: Αποθήκευση κατά γραμμή



Εικόνα 3.3: Αποθήκευση κατά στήλη

Οργανώνοντας τα δεδομένα κατά στήλη αυτά αποκτούν πιο ομοιογενή μορφή γεγονός που επιτρέπει την καλύτερη συμπίεση τους. Η χρήση του δίσκου μειώνεται σημαντικά καθώς πλέον για ένα ερώτημα ανάκτησης δεδομένων δεν είναι απαραίτητο να σαρωθούν όλα τα δεδομένα αλλά ένα υποσύνολο των στηλών. Επίσης, το γεγονός ότι τα δεδομένα στις στήλες είναι ίδιου τύπου μπορούν να εφαρμοστούν αλγόριθμοι κωδικοποίησης που ταιριάζουν καλύτερα με τους σύγχρονους επεξεργαστές καθιστώντας έτσι την επεξεργασία των δεδομένων πιο γρήγορη.

Το Parquet είναι ένας τύπος αποθήκευσης αρχείων σε κατά στήλη μορφή (columnar) που προορίζεται για χρήση στο Hadoop. Είναι ένα έργο ανοιχτού κώδικα και χρησιμοποιήθηκε αρχικά από το Twitter. Δημιουργήθηκε χάρις στην συνεργασία του Twitter και της Cloudera, ενώ τώρα έχει αρκετούς συνεισφέροντες. Το Parquet αποθηκεύει δεδομένα που έχουν μορφή εμφωλευμένων αντικειμένων σε απλή κατά στήλη μορφή ακολουθώντας την τεχνική που περιγράφεται στο άρθρο της Google για το Dremel.

Για να αποθηκευτούν τα δεδομένα κατά στήλη χρειάζεται να έχουν κάποια περιγραφή σε μορφή σχήματος. Το μοντέλο που χρησιμοποιείται αναπαριστά εμφωλευμένα αντικείμενα ως ομάδες από πεδία και αντικείμενα με πολλαπλά στοιχεία, όπως λίστες, ως επαναλαμβανόμενα πεδία.

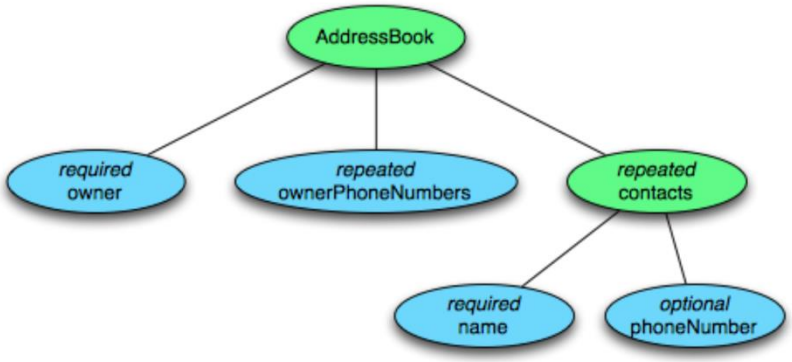
Το βασικό στοιχείο του σχήματος είναι μια ομάδα από πεδία (field) που ονομάζεται μήνυμα (message). Κάθε πεδίο έχει τρία χαρακτηριστικά, την επαναληψιμότητα, τον τύπο και το όνομα. Ο τύπος μπορεί να είναι είτε ομάδα (group) είτε κάποιος πρωτεύον χαρακτήρας (π.χ. int, float, boolean, string). Η επαναληψιμότητα έχει τρείς δυνατές τιμές required (ακριβώς μια εμφάνιση), optional (καμία ή μια εμφάνιση) και repeated (καμία ή παραπάνω εμφανίσεις). Στην εικόνα 3.4 φαίνεται ένα σχήμα για μια λίστα επαφών (address book).

```
message AddressBook {
required string owner;
repeated string ownerPhoneNumbers;
repeated group contacts {
required string name;
optional string phoneNumber;
}
}
```

Εικόνα 3.4: Σχήμα parquet για address book

Το σχήμα που περιγράφτηκε προηγουμένως πρέπει να μετασχηματιστεί σε μια λίστα από στήλες ώστε να αποθηκευτεί σε κατά στήλη μορφή και μετά να μπορεί να γυρίσει πίσω στην αρχική εμφωλευμένη μορφή του. Στο Parquet όλοι οι πρωτεύοντες τύποι πεδίων μετατρέπονται σε στήλες. Αν απεικονίσουμε το σχήμα με μορφή δένδρου, οι πρωτεύοντες χαρακτήρες είναι τα φύλλα (βλέπε εικόνα 3.5).

Κάθε εγγραφή στο Parquet συνοδεύεται από δύο ακεραίους αριθμούς, ο ένας αφορά τον βαθμό επαναληψιμότητας (repetition level) και ο άλλος τον βαθμό ορισμού (definition level). Με αυτά τα δύο νούμερα μπορούμε πλήρως να ανακατασκευάσουμε το αρχικό αντικείμενο που γενικά μπορεί να περιέχει και άλλα αντικείμενα εμφωλευμένα.



Εικόνα 3.5: Σχήμα parquet για address book σε μορφή δένδρου

Το definition level αποσκοπεί στο να καταγράψει το επίπεδο σε ένα εμφωλευμένο αντικείμενο όπου εμφανίστηκε για πρώτη φορά η τιμή null. Όταν υπάρχει καταγεγραμμένη τιμή ο αριθμός αυτός είναι άχρηστος αφού όλα τα προηγούμενα πεδία είναι ορισμένα. Με αυτό τον τρόπο στην ανακατασκευή του αντικειμένου τοποθετείται το null εκεί που ήταν πραγματικά.

Το repetition level υποδηλώνει το σημείο της στήλης από το οποίο ξεκινά μια καινούργια λίστα. Είναι δηλαδή ένας αριθμός που δείχνει πότε ξεκινά μια καινούργια λίστα και σε ποιο επίπεδο μέσα στο εμφωλευμένο αντικείμενο βρίσκεται. Repetition level ορίζεται μόνο όταν ο τύπος του πεδίου είναι repeated.

Με αυτή την πληροφορία είναι δυνατή η ανακατασκευή του αρχικού αντικειμένου. Το κόστος σε bit αποθήκευσης που εισάγουν αυτά τα δυο παραπάνω νούμερα δεν είναι ιδιαίτερα σημαντικό καθώς συνήθως τα αντικείμενα δεν έχουν μεγάλο βαθμό εμφώλευσης και με 2 bit λόγου χάρη καλύπτονται μέχρι 7 επίπεδα. Επίσης, οι στήλες έχουν λιγότερες τιμές null και με αυτό τον τρόπο κωδικοποιούνται και συμπιέζονται καλύτερα. (22)

3.2 Αλγόριθμος Map-Reduce

Ο αλγόριθμος Map-Reduce είναι κατάλληλος στο να επεξεργάζεται δεδομένα με μεγάλο όγκο. Αποτελείται από δύο βασικά στάδια, το στάδιο Map και το στάδιο Reduce. Σε κάθε στάδιο η είσοδος είναι ένα ζεύγος κλειδιού-τιμής. Ο προγραμματιστής πρέπει να ορίσει τη συνάρτηση του Map και του Reduce.

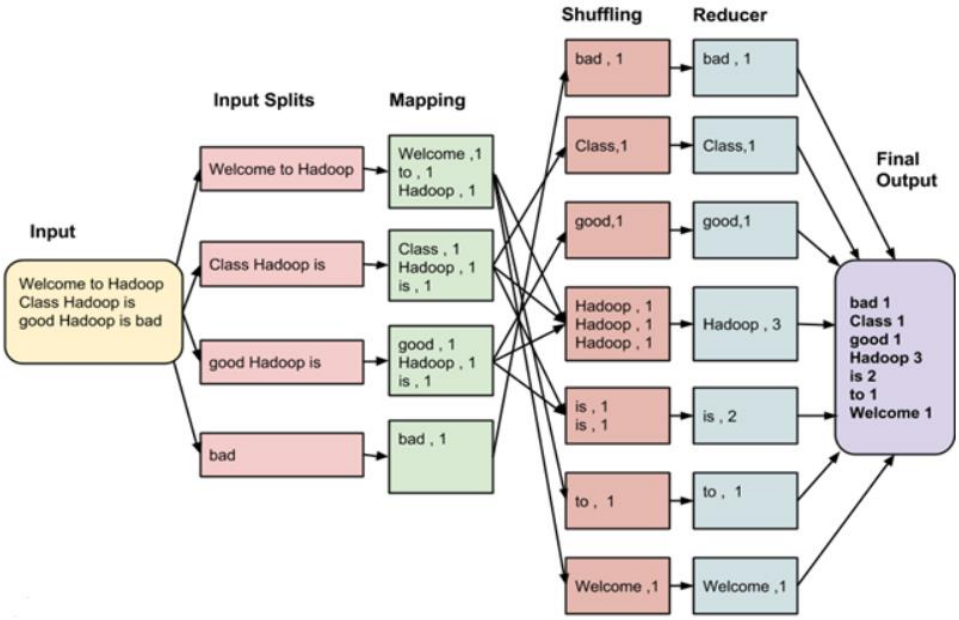
Αναλυτικότερα, ο αλγόριθμος εκτέλεσης περνά από 4 φάσεις, τον διαχωρισμό (*splitting*), την απεικόνιση (*mapping*), την ομαδοποίηση (*shuffling*) και την ελαχιστοποίηση (*reducing*). Οι δύο πρώτες φάσεις αποτελούν το Map στάδιο και οι δύο τελευταίες το Reduce. Κατά το splitting τα δεδομένα χωρίζονται σε μέρη σταθερού μεγέθους που είναι τέτοιο ώστε να μπορεί να το διαχειριστεί ένας και μόνο map. Μετά ακολουθεί η φάση του mapping όπου από τα δεδομένα που παρελήφθησαν από το προηγούμενο στάδιο, με χρήση της συνάρτησης για map παράγονται ζεύγη τιμής-

κλειδιού εξόδου. Τα ζεύγη αυτά δίδονται στη φάση του shuffling, όπου εγγραφές με ίδια κλειδιά ομαδοποιούνται. Τέλος, κατά τη φάση του reducing, οι ομάδες που έχουν δημιουργηθεί με βάση τα κλειδιά, συνδυάζονται και χρησιμοποιώντας τη συνάρτηση για reduce δημιουργείται ένα τελικό αποτέλεσμα ανά κλειδί.

Ένα τυπικό παράδειγμα εφαρμογής του αλγορίθμου είναι το πρόβλημα της μέτρησης των φορών εμφάνισης λέξεων σε ένα κείμενο (WordCount). Έστω το κείμενο της εικόνας 3.6, στην εικόνα 3.7 φαίνονται όλες οι φάσεις εκτέλεσης του αλγορίθμου όπως περιγράφτηκαν προηγουμένως.

```
Welcome to Hadoop Class
Hadoop is good
Hadoop is bad
```

Εικόνα 3.6: Αρχικό κείμενο για WordCount



Εικόνα 3.7: Υλοποίηση του WordCount με map – reduce

Επιγραμματικά, το κείμενο χωρίζεται σε γραμμές, για κάθε γραμμή βρίσκονται οι λέξεις, οι λέξεις ομαδοποιούνται αν είναι ίδιες και τέλος αθροίζονται οι φορές εμφάνισης της κάθε μιας και δημιουργείται ο πίνακας εξόδου με κλειδιά τις διαφορετικές λέξεις και τιμή το πλήθος εμφάνισης τους. (23)

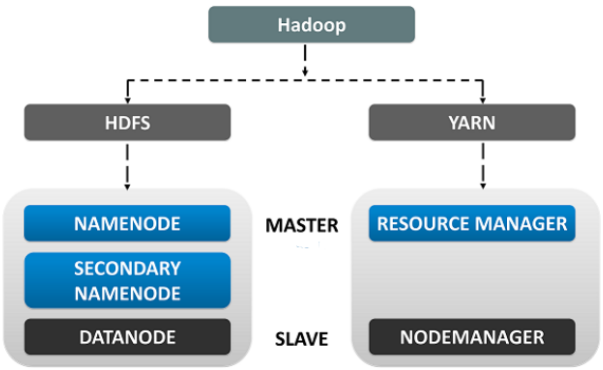
3.3 Hadoop και Hadoop Distributed File System (HDFS)

Το Hadoop ανήκει στην Apache και είναι λογισμικό ανοιχτού κώδικα που χρησιμοποιείται για να διευκολύνει ένα δίκτυο από υπολογιστές να επιλύει προβλήματα που απαιτούν την διαχείριση και επεξεργασία μεγάλου όγκου δεδομένων. Το λογισμικό που διαθέτει παρέχει δυνατότητες για διανεμημένη αποθήκευση δεδομένων και ικανότητα για επεξεργασία μεγάλου όγκου δεδομένων χρησιμοποιώντας αλγορίθμους τύπου Map Reduce. Το Hadoop είναι σχεδιασμένο για χρήση σε cluster υπολογιστών που πολλές φορές η ποιότητα του υλικού δεν είναι καλή. Έτσι, λαμβάνεται υπόψη η πιθανότητα κάποιο μηχάνημα να βγει εκτός λειτουργίας και το Hadoop είναι υπεύθυνο να τη διαχειριστεί.

Ο κορμός του Apache Hadoop αποτελείται από ένα σύστημα αποθήκευσης αρχείων, το Hadoop Distributed File System (HDFS), και ένα σύστημα επεξεργασίας δεδομένων τύπου Map Reduce. Το Hadoop χωρίζει τα δεδομένα σε μεγάλα κομμάτια και τα διανέμει σε όλους τους κόμβους του cluster. Έπειτα, αρχίζει η παράλληλη επεξεργασία των δεδομένων από κάθε κόμβο. Με αυτόν τον τρόπο, το Hadoop εκμεταλλεύεται την τοπικότητα των δεδομένων αφού κάθε κόμβος επεξεργάζεται τελικά τα δεδομένα που έχει αποθηκευμένα σε αυτόν. Το γεγονός αυτό επιτρέπει να γίνεται γρηγορότερη και αποτελεσματικότερη επεξεργασία αρχείων σε σύγκριση με συμβατικές αρχιτεκτονικές που βασίζονται σε παράλληλα συστήματα αποθήκευσης αρχείων και τα δεδομένα και οι υπολογισμοί γίνονται μέσω του δικτύου.

Το Apache Hadoop περιλαμβάνει τα ακόλουθα συστατικά:

- ✓ *Hadoop Common* – περιλαμβάνει βιβλιοθήκες και κώδικα που χρησιμοποιούνται από τα υπόλοιπα στοιχεία
- ✓ *Hadoop Distributed Filesystem (HDFS)* – το διανεμημένο σύστημα αποθήκευσης αρχείων
- ✓ *Hadoop YARN* – μια πλατφόρμα που είναι υπεύθυνη για την διαχείριση των υπολογιστικών πόρων του cluster
- ✓ *Hadoop MapReduce* – υλοποίηση του MapReduce αλγορίθμου για δεδομένα μεγάλης κλίμακας (24)



Εικόνα 3.8: Αρχιτεκτονική του Hadoop

Αξίζει να αναφερθούμε αναλυτικότερα στην αρχιτεκτονική του διανεμημένου συστήματος αποθήκευσης αρχείων - hdfs, καθώς αποτελεί τον κύριο λόγο για την καλή επίδοση του Hadoop στην επεξεργασία δεδομένων και την δυνατότητα κλιμάκωσης. Τα δομικά στοιχεία του hdfs είναι:

- ✓ *Namenode*: Αποτελεί το πιο κεντρικό κομμάτι του hdfs. Έχει ρόλο master και είναι σχεδιασμένος για να κρατά τα metadata του συστήματος. Είναι επίσης υπεύθυνος για να παρακολουθεί την «υγεία» των υπολοίπων κόμβων και να αναθέτει εργασίες στους datanodes.
- ✓ *Datanode*: Αποτελεί το κομμάτι του συστήματος που αποθηκεύει τα δεδομένα. Έχει ρόλο slave και δίνει αναφορά στον master για την κατάσταση της υγείας του και την πρόοδο των εργασιών που εκτελεί. Αν ο datanode δεν απαντήσει στον namenode τότε αυτός θεωρείται νεκρός και οι εργασίες του ανατίθενται στον αμέσως επόμενο διαθέσιμο datanode.
- ✓ *Secondary Namenode*: Δρα βοηθητικά του namenode και κύριος ρόλος του είναι να κρατά αρχείο με τα σημεία που είναι αποθηκευμένα τα metadata του namenode. (25)

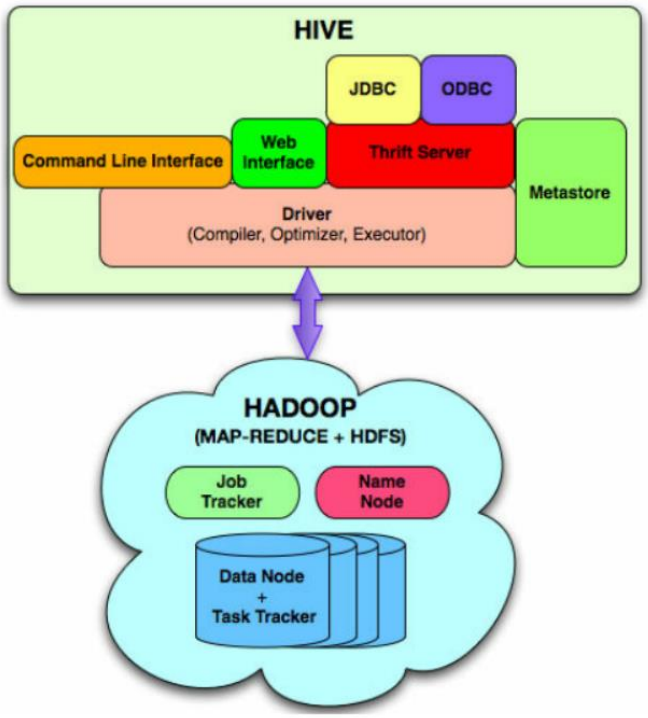
3.4 Apache Hive

Το Apache Hive εμφανίστηκε ως μια λύση για αποθήκευση δεδομένων χρησιμοποιώντας την υποδομή του Hadoop. Το Hive υποστηρίζει την ανάλυση μεγάλου όγκου δεδομένων και παρέχει μια δική του γλώσσα τύπου SQL την HiveQL για ανάκτηση και ανάλυση δεδομένων. Χρησιμοποιεί το hdfs για να αποθηκεύει τα δεδομένα του και υποστηρίζει διαφορετικούς τύπους αρχείων στους πίνακες του. Επιπλέον, μπορεί να έχει ως μηχανή εκτέλεσης (execution engine) των εργασιών του το Hadoop, το Tez και το Spark. Αυτές οι μηχανές με την σειρά τους χρησιμοποιούν το YARN για τον διαμοιρασμό των εργασιών, την διαχείριση των υπολογιστικών πόρων και την διαχείριση μνήμης. Τα ερωτήματα σε HiveQL μετατρέπονται αυτόματα σε εργασίες που εκτελούνται από την εκάστοτε μηχανή εκτέλεσης. Τα αποτελέσματα συλλέγονται από το YARN και αποστέλλονται στον Hive client.

Στην εικόνα 3.9 φαίνονται τα βασικά κομμάτια που συναποτελούν το Hive καθώς και η εξάρτηση από το Hadoop. Αναλυτικότερα, πρόκειται για τα ακόλουθα:

- ✓ *Metastore*: Αυτό το κομμάτι είναι υπεύθυνο για την αποθήκευση του σχήματος της βάσης, των πινάκων του Hive και της τοποθεσίας των αρχείων των πινάκων.
- ✓ *Driver*: Εδώ οργανώνεται η εκτέλεση του HiveQL ερωτήματος. Πιο περιγραφικά, ο driver ξεκινά την εκτέλεση του ερωτήματος, συλλέγει και αποθηκεύει τα ενδιάμεσα και τελικά αποτελέσματα και σηματοδοτεί το τελείωμα της εκτέλεσης του ερωτήματος.
- ✓ *Compiler*: Αυτό το κομμάτι μεταφράζει το HiveQL ερώτημα και παράγει το σχέδιο εκτέλεσης του ερωτήματος. Το σχέδιο εκτέλεσης περιλαμβάνει τα στάδια και τις εργασίες που πρέπει να εκτελέσει η μηχανή εκτέλεσης

- ✓ *Optimizer*: Ρόλος του είναι να εκτελεί όλους τους απαραίτητους μετασχηματισμούς/αλλαγές στο σχέδιο εκτέλεσης ώστε να βελτιστοποιηθεί η εκτέλεση του ερωτήματος.
- ✓ *Executor*: Εκτελεί την εργασία που του ανατέθηκε και επικοινωνεί με το Hadoop ώστε να ελέγχει την διαδικασία εκτέλεσης.
- ✓ *User Interface*: Για επικοινωνία με τον χρήστη υπάρχει το Hive CLI και ο Thrift Server από όπου μπορεί φτιάξει κάποιος έναν JDBC client λόγου χάρη. (26)



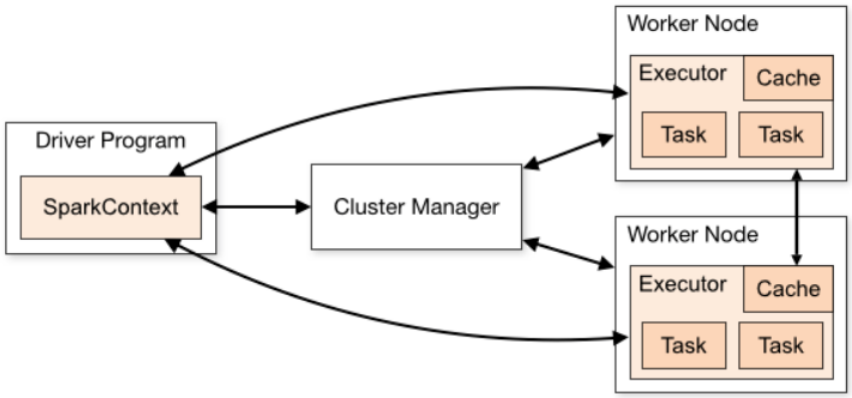
Εικόνα 3.9: Αρχιτεκτονική Hive και Hadoop

3.5 Apache Spark - SparkSQL

Το Apache Spark χρησιμοποιείται για την επεξεργασία μεγάλου όγκου δεδομένων σε υπολογιστικά cluster. Προσφέρει ένα σύνολο από βιβλιοθήκες σε τρεις γλώσσες προγραμματισμού (Java, Scala, Python) και αποτελεί μια ενοποιημένη υπολογιστική μηχανή.

Αναλυτικότερα, το Spark είναι μια υπολογιστική μηχανή καθώς διαχειρίζεται την εισαγωγή δεδομένων από διάφορα συστήματα αποθήκευσης αρχείων και πραγματοποιεί υπολογισμούς σε αυτά. Χαρακτηριστικό του Spark είναι ότι δεν αποθηκεύει εσωτερικά τα δεδομένα που διαχειρίζεται, χρησιμοποιώντας έτσι πόρους από την μνήμη RAM (in memory system). Επίσης, χαρακτηρίζεται ενοποιημένη γιατί προσφέρει ένα αρκετά ευρύ API μέσω του οποίου μπορεί να δομηθεί μια εφαρμογή χωρίς να χρειάζεται να σπάει σε περισσότερα συστήματα/APIs. Τέλος, το Spark διαθέτει βιβλιοθήκες για SQL (SparkSQL), μηχανική μάθηση (MLib), επεξεργασία ροών δεδομένων (Spark Streaming) και ανάλυση γράφων (GraphX).

Κάθε εφαρμογή που προορίζεται να εκτελεστεί από το Spark, αναλύεται σε μια διεργασία *οδηγητής* (driver) και σε ένα σύνολο διεργασιών *εργατών* (workers/executors) που είναι διανεμημένες στο υπολογιστικό cluster. Στην εικόνα 3.10 παρουσιάζεται ένα σχήμα με την αλληλεπίδραση των εργασιών κατά την εκτέλεση μιας εφαρμογής.



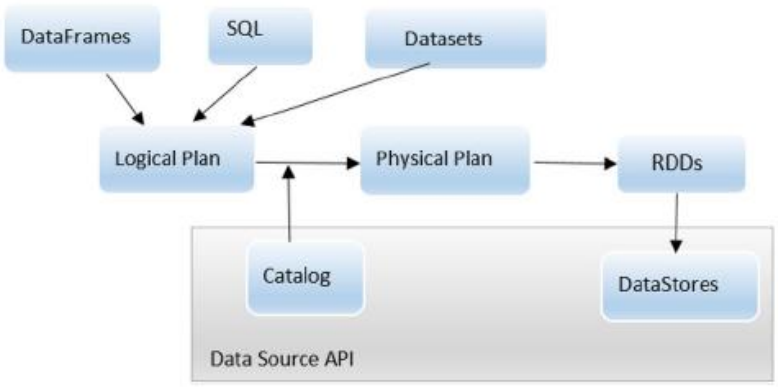
Εικόνα 3.10: Εκτέλεση εφαρμογών από το Spark

Η διεργασία οδηγητής είναι υπεύθυνη να εκκινεί την εφαρμογή, τρέχει σε ένα κόμβο του συστήματος και προγραμματίζει την εκτέλεση των διαφόρων εργασιών σε συνεργασία με τον διαχειριστή του cluster. Επιπλέον, αναλύει, προγραμματίζει και διανέμει το σύνολο των εργασιών στους εργάτες. Τέλος, αποθηκεύει metadata σχετικά με την εκτέλεση της εφαρμογής και τα εκθέτει μέσω webUI.

Οι διεργασίες εργάτες κατανέμονται σε όλο το cluster και είναι υπεύθυνες για την εκτέλεση των εργασιών που τους έχουν ανατεθεί. Με λίγα λόγια εκτελούν την όποια επεξεργασία δεδομένων χρειάζεται και επιστρέφουν τα αποτελέσματα στην διεργασία οδηγητή. Κάθε κόμβος του cluster μπορεί να έχει από μια διεργασία εργάτη έως μια ανά υπολογιστικό πυρήνα (core).

Στην αρχιτεκτονική του Spark σημαντικό ρόλο παίζει το Resilient Distributed Dataset – RDDs και βρίσκεται πίσω από APIs όπως το Dataframe/Dataset. Στην πραγματικότητα όλα τα δεδομένα που διαχειριζόμαστε με το Spark αποθηκεύονται εσωτερικά ως RDDs. Το RDD προγραμματιστικά είναι ανάλογο με μια κατανεμημένη δομή δεδομένων μπορούμε να το φανταστούμε δηλαδή ως μια λίστα αντικειμένων που είναι διαμοιρασμένη σε πολλά μηχανήματα παράλληλα. Εμβαθύνοντας στην λειτουργία των RDD, τα δεδομένα χωρίζονται σε μέρη τα οποία και αποθηκεύονται στους κόμβους που εκτελούν διεργασίες εργάτες. Κάθε κόμβος λοιπόν εκτελεί εργασίες με τα δεδομένα που είναι αποθηκευμένα σε αυτόν. Έτσι αν ένας κόμβος βγει εκτός λειτουργίας το Spark μπορεί να ανασυνθέσει το τα δεδομένα που χάθηκαν από την αρχική πηγή και να υποβάλει πάλι την εργασία για εκτέλεση. (27)

Η SparkSQL είναι μια βιβλιοθήκη του Apache Spark και δίνει την δυνατότητα στους χρήστες του να γράψουν SQL. Παρέχει τρείς βασικές δυνατότητες στη διαχείριση δομημένων και ημιδομημένων δεδομένων. Πρώτον, χρησιμοποιεί την αφαιρετική δομή του Dataframe που είναι αντίστοιχη του πίνακα στις σχεσιακές βάσεις δεδομένων διευκολύνοντας έτσι την διαχείριση δεδομένων. Δεύτερον, μπορεί να διαβάζει και να γράφει δεδομένα που είναι αποθηκευμένα σε διαφορετικές μορφές, όπως JSON, πίνακες Hive και Parquet. Και τρίτον, δίνει μια διεπαφή SQL για δεδομένα που είναι αποθηκευμένα στο Spark. Αυτή η διεπαφή είναι διαθέσιμη είτε μέσω της γραμμής εντολών είτε προγραμματιστικά μέσω JDBC/ODBC.



Εικόνα 3.11: Αρχιτεκτονική SparkSQL

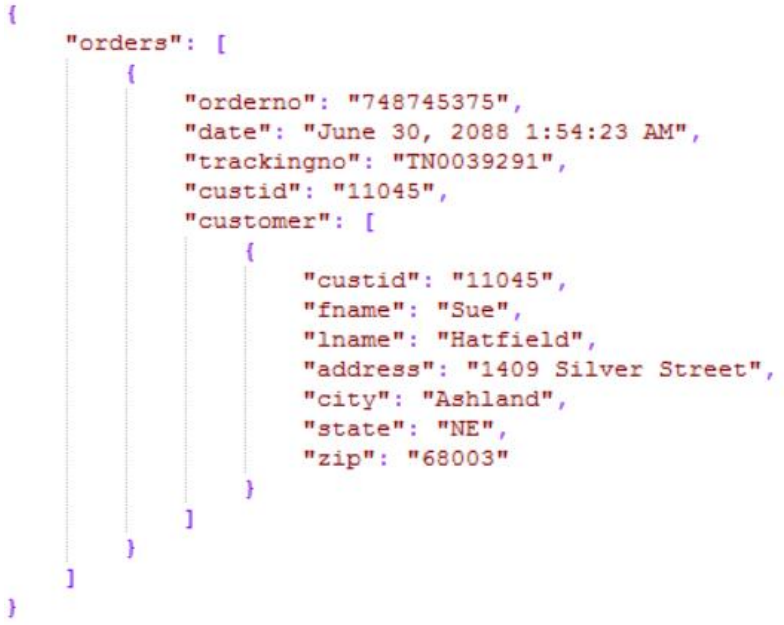
Στην εικόνα 3.11 φαίνεται η αρχιτεκτονική του SparkSQL. Πίσω από την SparkSQL υπάρχει ο Catalyst Optimizer που είναι υπεύθυνος για την δημιουργία και βελτιστοποίηση του πλάνου εκτέλεσης των εκάστοτε ερωτημάτων. Τα βασικά αρχιτεκτονικά επίπεδα του SparkSQL είναι:

- ✓ Το SQL API που υποστηρίζει SQL και HiveQL.
- ✓ Το μοντέλο των RDD που διευκολύνει το Spark να δουλεύει με πίνακες και εγγραφές.
- ✓ Οι πηγές δεδομένων που επιτρέπουν στο Spark να δουλεύει και με άλλου είδους αρχεία εκτός από αρχεία κειμένου και Avro. (28)

3.6 MongoDb

Η mongoDb (από το “humongous” - τεράστιος) είναι η πιο διαδεδομένη Document Database και ίσως και η πιο διαδεδομένη NoSQL Database γενικότερα. Αναπτύχθηκε από την εταιρεία 10gen στα πρότυπα του ανοιχτού κώδικα και δόθηκε για πρώτη φορά σε κυκλοφορία το 2009. Είναι υλοποιημένη στη γλώσσα προγραμματισμού C++, και κύριος στόχος της είναι η επίτευξη υψηλής ταχύτητας και κλιμακωσιμότητας. Ο προκαθορισμένος τρόπος για τη διαχείριση mongoDb βάσεων είναι μέσω του διαδραστικού Javascript shell που παρέχει. Επιπλέον, σε πολλές γλώσσες προγραμματισμού (C, C#, C++, Haskell, Java, JavaScript, Perl, PHP, Python, και Ruby) έχουν αναπτυχθεί βιβλιοθήκες και drivers για την υποστήριξή της.

Στην mongoDb ακολουθείται η νοοτροπία data as documents. Τα έγγραφα που αποθηκεύονται είναι στη μορφή BSON αντικειμένων, δηλαδή δυαδικά κωδικοποιημένα JSON αντικείμενα. Τα BSON είναι σχεδιασμένα έτσι ώστε να γίνεται εύκολα η διάσχιση και η ανάλυσή τους. Οι χρήστες εισάγουν τα δεδομένα τους σε μορφή JSON, τα οποία στη συνέχεια μετατρέπονται σε BSON ώστε να αποθηκευτούν στη βάση, και αντίστοιχα όταν παίρνουν κάποιο BSON αντικείμενο από τη βάση, το βλέπουν σε JSON μορφή. Στην εικόνα 3.12 φαίνεται μια τυπική δομή ενός JSON αρχείου.



```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```


Εικόνα 3.12: Παράδειγμα ενός JSON αρχείου

Ένα JSON document έχει μηδέν ή περισσότερα ζεύγη κλειδιού – τιμής (key-value pairs). Σαν κλειδί αποθηκεύεται το όνομα του πεδίου ως συμβολοσειρά και σαν τιμή, μια τιμή ενός από τους υποστηριζόμενους JSON datatypes (Number, String, Boolean, Array, Object, Whitespace, null, ή JSON value) είτε κάποιος άλλος όπως ημερομηνία. Τα αντικείμενα αυτά είναι schema less, που σημαίνει πως δεν υπάρχει κάποια ομαδοποίηση μεταξύ εγγράφων που περιέχουν ίδια κλειδιά, κάτι που συναντάμε στο σχεσιακό μοντέλο, όπου όλες οι εγγραφές ενός πίνακα έχουν την ίδια δομή. Αντιθέτως, έγγραφα με παρόμοια δομή που περιλαμβάνουν δεδομένα σχετικά με το ίδιο αντικείμενο, αλλά με διαφορετικά ζεύγη κλειδιού – τιμής, αποθηκεύονται μαζί, σε μια "συλλογή".

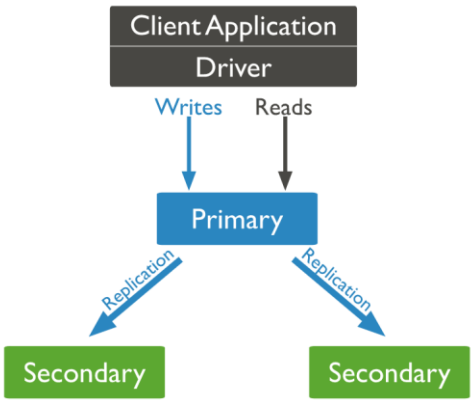
Το πλεονέκτημα των NoSQL που έχει να κάνει με τη μη ύπαρξη αυστηρά καθορισμένου σχήματος των δεδομένων, φαίνεται και στη mongoDb αφού υπάρχει απόλυτη ελευθερία ως προς τα πεδία, τη δομή και τους τύπους δεδομένων κάθε εγγράφου μιας συλλογής. Από την άλλη όμως, ένα πολύ βασικό στοιχείο είναι πως δεν υπάρχει δυνατότητα για joins μεταξύ συλλογών όπως στις σχεσιακές βάσεις δεδομένων. Έτσι, ο σχεδιαστής καλείται να διαμορφώσει το κάθε έγγραφο ξεχωριστά, με τέτοιο τρόπο ώστε να ελαχιστοποιεί τον απαιτούμενο αποθηκευτικό χώρο αλλά και

να διευκολύνει τις αναζητήσεις του, μοντελοποιώντας τα δεδομένα με γνώμονα τον τρόπο χρήσης τους και όχι τον τρόπο αποθήκευσης. (29)

Η mongoDb διαθέτει δική της διάλεκτο για την πραγματοποίηση CRUD (create-read-update-delete) λειτουργιών. Στην εικόνα 3.13 φαίνεται ένα παράδειγμα ερωτήματος για εύρεση εγγράφου στην συλλογή users.

Εικόνα 3.13: Ερώτημα τύπου select στη mongoDb

Η mongoDb υποστηρίζει, επίσης, την δημιουργία δεικτών (indexes) οι οποίοι μπορούν να ταξινομούν τα δεδομένα είτε με αύξουσα σειρά, είτε με φθίνουσα και είτε με μοναδικότητα. Οι δείκτες αυτοί υλοποιούνται ως δείκτες δυαδικών δένδρων (B-trees). Το πεδίο _id που υπάρχει αυτόματα σε κάθε αντικείμενο υπάρχει κιόλας ως δείκτης.

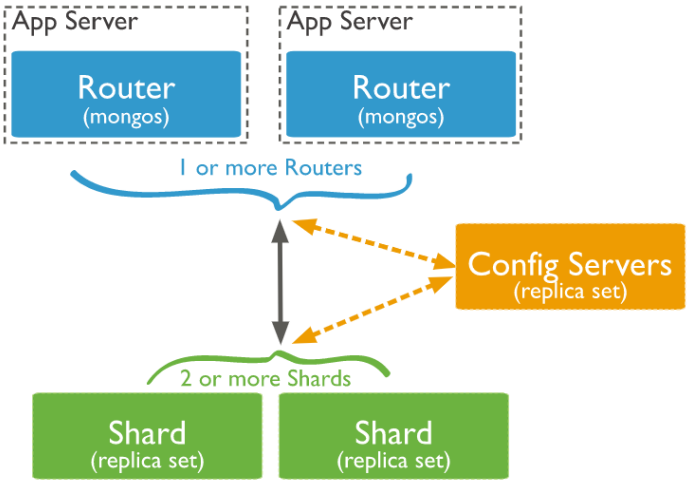


Εικόνα 3.14: Σχηματική αναπαράσταση δημιουργίας αντιγράφων στη mongoDb

Ένα χαρακτηριστικό της mongoDb είναι ότι υποστηρίζει έμφυτα την δημιουργία αντιγράφων (replication), που σημαίνει πως η βάση κλωνοποιείται και συγχρονίζεται σε τουλάχιστον 2 υπολογιστές. Όπως φαίνεται και στην εικόνα 3.14 υπάρχει ένας πρωτεύον κόμβος που έχει πλήρη έλεγχο των δεδομένων και μπορεί να διαβάζει και να γράφει σε αυτά. Επίσης, υπάρχει και ένα σύνολο από δευτερεύοντες κόμβους που καθένας τους έχει ένα αντίγραφο του πρωτεύοντος. Κάθε γράψιμο στον πρωτεύοντα κόμβο μεταδίδεται στους επόμενους. Αν για κάποιο λόγο ό πρωτεύον κόμβος είναι εκτός λειτουργίας τότε υπάρχει διαδικασία εκλογής επόμενου πρωτεύοντα κόμβου. Έτσι κάποιος από τους δευτερεύοντες γίνεται πρωτεύοντας και το σύστημα παραμένει πάντοτε διαθέσιμο.

Άλλο ένα χαρακτηριστικό της mongoDb είναι ότι υποστηρίζει τον αυτόματο διαμοιρασμό δεδομένων σε ένα cluster (sharding). Αυτό το χαρακτηριστικό είναι ιδιαίτερα χρήσιμο στην διαχείριση μεγάλου όγκου δεδομένων όπου ένας υπολογιστής δεν είναι αρκετός για να διαχειριστεί όλα τα δεδομένα. Για να γίνει ο διαμοιρασμός των δεδομένων στους κόμβους του συστήματος, πρέπει να οριστεί ένα κλειδί (shard key) με βάση το οποίο θα καθορίζεται σε ποιο μηχάνημα θα πάει η κάθε εγγραφή. Ο

χρήστης της βάσης συνδέεται στον κύριο (master) κόμβο και υποβάλλει τα ερωτήματα ή οποιαδήποτε άλλη ενέργεια θέλει να εκτελέσει. Όπως φαίνεται στην εικόνα 3.15 υπάρχουν τρείς κατηγορίες κόμβων σε μια τοπολογία με διαμοιρασμό φορτίου. Πρόκειται για τους shard servers, configuration servers και query routers. Για να αποφευχθεί η απώλεια δεδομένων κάθε κατηγορία server διατηρεί αντίγραφα (replication) σε άλλα φυσικά μηχανήματα.



Εικόνα 3.15: Τοπολογία mongoDb με διαμοιρασμό φορτίου σε cluster (sharding)

Αναλυτικότερα, οι *shard servers* είναι τα σημεία που αποθηκεύονται στην πράξη τα δεδομένα. Οι *configuration servers* συγκρατούν πληροφορία σχετική με την αποθήκευση των δεδομένων στο cluster. Αυτή η πληροφορία χρησιμοποιείται από τους query routers για να εντοπιστεί το shard στο οποίο βρίσκονται τα δεδομένα που αναζητούνται. Τέλος, οι *query routers* είναι μια διεπαφή της mongo μεταξύ χρήστη και shard η οποία κατευθύνει στον κατάλληλο shard την εργασία που της διατίθεται. (30)

Κεφάλαιο 4. Εκτέλεση πειράματος

Στόχος του κεφαλαίου αυτού είναι να παρουσιαστούν τα δεδομένα που χρησιμοποιήθηκαν, η προεργασία που υπέστησαν, η μέθοδος σύγκρισης των τεχνολογιών, το περιβάλλον που δημιουργήθηκε καθώς και τα αποτελέσματα των μετρήσεων.

4.1 Παρουσίαση δεδομένων

Τα δεδομένα που επιλέχθηκαν για την σύγκριση των τεχνολογιών είναι γεωχωρικά δεδομένα από την πλατφόρμα copernicus.eu[21]. Αφορούν μετρήσεις βαθυμετρίας για μια περιοχή του ωκεανού καθώς και μετεωρολογικά δεδομένα μιας χρονικής περιόδου σε μια θαλάσσια περιοχή. Στους πίνακες 4.1 και 4.2 παρουσιάζεται το σχήμα των δεδομένων.

Πίνακας 4.1: Σχήμα δεδομένων βαθυμετρίας

Όνομα Πεδίου	*Τύπος Πεδίου*	*Περιγραφή*
latitude	*Double*	*Γεωγραφικό μήκος*
longitude	*Double*	*Γεωγραφικό πλάτος*
depth	*Double*	*Βάθος*
source	*String*	*Πηγή δεδομένων*
observation	*String*	*Παρατηρητήριο*
metadata_repository_id	*String*	*Στοιχεία ηλεκτρονικού αποθετηρίου*

Πίνακας 4.2: Σχήμα μετεωρολογικών δεδομένων

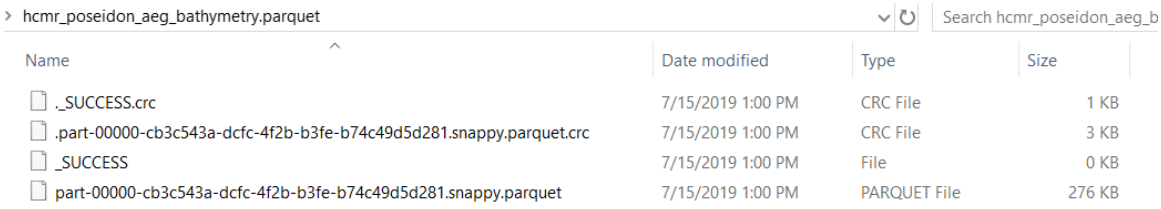
Όνομα Πεδίου	*Τύπος Πεδίου*	*Περιγραφή*
latitude	*Double*	*Γεωγραφικό μήκος*
longitude	*Double*	*Γεωγραφικό πλάτος*
air_temperature_2m	*Double*	*Θερμοκρασία αέρα*
x_wind	*Double*	*Κατεύθυνση ανέμου κατά τον χ άξονα*
y_wind	*Double*	*Κατεύθυνση ανέμου κατά τον ψ άξονα*
air_pressure_at_sea_level	*Double*	*Πίεση αέρα στο ύψος της θάλασσας*

[21] https://www.copernicus.eu/en. Πρόκειται για το παρατηρητήριο της Ευρωπαϊκής Ένωσης για τον πλανήτη, συλλέγει δεδομένα μέσω δορυφορικής παρατήρησης και επιτόπιων σταθμών μετρήσεων.

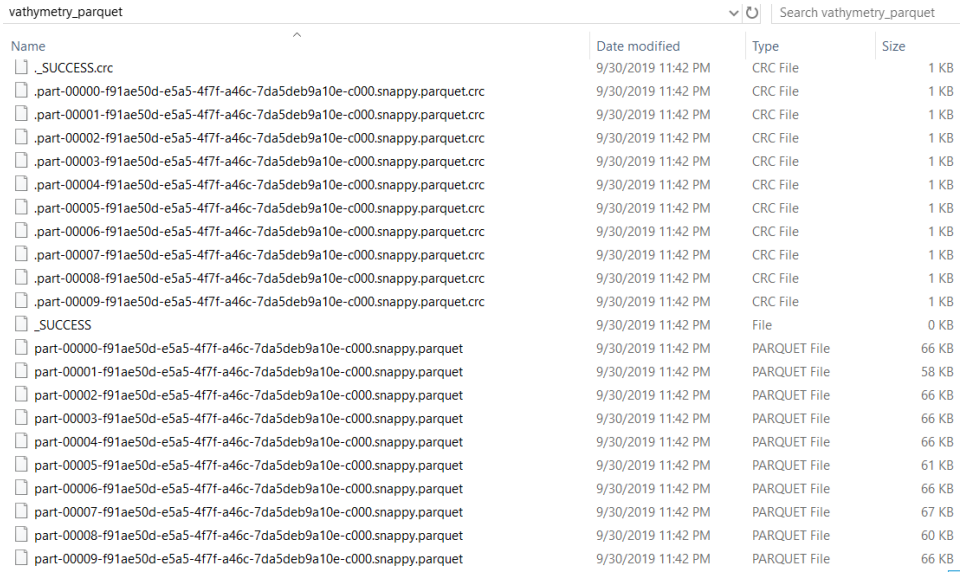
time	*Timestamp*	*Χρονική στιγμή*
source	*String*	*Πηγή δεδομένων*
observation	*String*	*Παρατηρητήριο*
metadata_repository_id	*String*	*Στοιχεία ηλεκτρονικού αποθετηρίου*

Τα δύο αρχεία με τα παραπάνω δεδομένα είναι αποθηκευμένα σε μορφή ***Parquet***. Το αρχείο με τα δεδομένα βαθυμετρίας (πίνακας vathymetry) καταλαμβάνει χώρο ***280 KB*** ενώ το αρχείο με τα δεδομένα μετεωρολογίας (πίνακας forecast) ***10,6 GB***.

Μιας και μιλάμε για γεωχωρικά δεδομένα, η ταξινόμηση των εγγραφών με βάση το γεωγραφικό μήκος και πλάτος είναι μια φυσιολογική επιλογή. Προκειμένου να πετύχουμε αυτή την ταξινόμηση χρειάστηκε κάποια προεργασία. Συγκεκριμένα κατασκευάστηκε ένα πρόγραμμα σε Java το οποίο με την βοήθεια του Spark διάβαζε τα αρχεία και τα φόρτωνε σε dataset. Μετά για κάθε dataset έγινε στρογγυλοποίηση στα δύο δεκαδικά ψηφία για τα πεδία latitude και longitude και εφαρμόστηκε ταξινόμηση κατά latitude και longitude. Επειδή, ειδικά για τα δεδομένα μετεωρολογίας το συνολικό μέγεθος είναι μεγάλο αποφασίστηκε το καινούργιο αρχείο που θα δημιουργούταν να έχει περισσότερα από ένα αυτοτελή αρχεία Parquet. Το μέγεθος του κάθε χωριστού αρχείου ορίστηκε στα ~50KB, έτσι τα δεδομένα βαθυμετρίας χωρίστηκαν σε 10 διαφορετικά αρχεία ενώ τα δεδομένα μετεωρολογίας σε 330. Συνεπώς τα ταξινομημένα dataset χωρίζονται προγραμματιστικά στα αντίστοιχα κομμάτια και με το Spark αποθηκεύονται στον νέο φάκελο. Στις εικόνες 4.1 και 4.2 φαίνονται τα αρχεία βαθυμετρίας πριν και μετά την επεξεργασία.



Εικόνα 4.1: Δεδομένα βαθυμετρίας πριν την επεξεργασία



vathymetry_parquet — Search vathymetry_parquet

Name	Date modified	Type	Size
._SUCCESS.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00000-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00001-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00002-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00003-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00004-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00005-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00006-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00007-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00008-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
.part-00009-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet.crc	9/30/2019 11:42 PM	CRC File	1 KB
_SUCCESS	9/30/2019 11:42 PM	File	0 KB
part-00000-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB
part-00001-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	58 KB
part-00002-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB
part-00003-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB
part-00004-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB
part-00005-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	61 KB
part-00006-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB
part-00007-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	67 KB
part-00008-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	60 KB
part-00009-f91ae50d-e5a5-4f7f-a46c-7da5deb9a10e-c000.snappy.parquet	9/30/2019 11:42 PM	PARQUET File	66 KB

Εικόνα 4.2: Δεδομένα βαθυμετρίας μετά την επεξεργασία (10 αρχεία Parquet)

Το γεγονός ότι το αρχικό αρχείο έχει χωριστεί σε πολλά μικρότερα βοήθησε κατά την διαδικασία λήψης των μετρήσεων καθώς ήταν δυνατό να ξεκινήσουν με ένα μικρό μέγεθος δεδομένων και σταδιακά να κλιμακώνεται. Επίσης, τα ταξινομημένα δεδομένα διευκολύνουν την αποθήκευση σε βάση δεδομένων που λειτουργεί στα πλαίσια ενός cluster όταν η βάση με την οποία γίνεται ο διαχωρισμός των δεδομένων είναι τα πεδία στα οποία έχει γίνει και η ταξινόμηση.

4.2 Κριτήρια συγκριτικής αξιολόγησης

Για την σύγκριση των διαφορετικών τεχνολογιών αποθήκευσης και ανάκτησης δεδομένων μεγάλης κλίμακας επιλέχθηκε ένα σύνολο από ερωτήματα τα οποία έτρεξαν τα ίδια σε όλες τις βάσεις. Είναι συνολικά 14 ερωτήματα τα οποία αφορούν όλα ανάκτηση δεδομένων (reads). Στον πίνακα 4.3 φαίνονται τα ερωτήματα με την σύνταξη τους σε SQL.

Πίνακας 4.3: Λίστα ερωτημάτων για συγκριτική αξιολόγηση

Ερώτημα	***Περιγραφή σε SQL***
Q1	*SELECT * FROM vathymetry;*
Q2	*SELECT * FROM forecast;*
Q3	*SELECT * FROM forecast WHERE latitude=42;*
Q4	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5;*
Q5	*SELECT * FROM forecast WHERE air_pressure_at_sea_level>1010 AND air_pressure_at_sea_level<1010.5;*

Q6	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q7	*SELECT * FROM forecast WHERE latitude>0 AND latitude<1500 GROUP BY time;*
Q8	*SELECT * FROM vathymetry GROUP BY depth;*
Q9	*SELECT * FROM forecast f JOIN vathymetry v ON f.latitude = v.latitude AND f.longitude = v.longitude;*
Q10	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5 AND air_temperature_2m>29 ORDER BY air_temperature_2m;*
Q11	*SELECT COUNT(time) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q12	*SELECT AVG(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q13	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>35 AND latitude<45 GROUP BY time;*
Q14	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*

Τα ερωτήματα Q1, Q2 αφορούν μαζική ανάκτηση δεδομένων, τα Q3-Q5 αφορούν ανάκτηση δεδομένων με κριτήρια, τα Q6-Q8 χρησιμοποιούν group by μέθοδο στα αποτελέσματα τους, το Q9 είναι ένα απλό join, το Q10 κάνει ταξινόμηση με order by ενώ τα Q11-Q14 χρησιμοποιούν συναθροιστικές συναρτήσεις τύπου sum (άθροισμα), avg (μέσος όρος) και count (πλήθος).

Για καθένα από τα παραπάνω ερωτήματα μετρήθηκε ο χρόνος απόκρισης και το σύνολο των εγγραφών που επεστράφησαν. Με βάση αυτά τα δύο στοιχεία θα γίνει και η σύγκριση των τεχνολογιών αργότερα. Οι τεχνολογίες που συγκρίνονται είναι οι Apache Hive, SparkSQL και mongoDb.

4.3 Υπολογιστικοί πόροι πειράματος

Προκειμένου να αξιολογηθούν οι τεχνολογίες που επιλέχθηκαν σε ένα συνολικό όγκο δεδομένων ~11GB χρειάζεται να εφαρμοστούν τοπολογίες κατανεμημένων υπολογιστικών συστημάτων (cluster). Για την επιλογή του κατάλληλου υλικού πρέπει να ληφθεί υπόψη ότι το ίδιο υλικό θα χρησιμοποιηθεί και στις τρεις τεχνολογίες.

Όλες οι τεχνολογίες που χρησιμοποιήθηκαν έχουν υψηλές απαιτήσεις όσο αφορά την μνήμη RAM, ενώ συγκεκριμένα το Spark έχει ιδιαίτερα αυξημένες απαιτήσεις καθώς την χρησιμοποιεί ως αποθηκευτικό μέσο. Οπότε κατά την διαστασιολόγηση των μηχανημάτων τέθηκε η απαίτηση το σύνολο των δεδομένων να μπορεί θεωρητικά να βρίσκεται όλο αποθηκευμένο στην μνήμη RAM. Έτσι η ελάχιστη δυνατή τιμή της

RAM είναι 16GB. Επίσης, η κύρια μνήμη πρέπει να μπορεί κατά ελάχιστο να κρατά αποθηκευμένα τα δεδομένα μαζί με το λειτουργικό σύστημα και το λογισμικό που απαιτείται.

Για την δημιουργία του cluster επιλέχθηκε πάροχος υπηρεσιών υπολογιστικού νέφους (cloud provider) και συγκεκριμένα χρησιμοποιήθηκε το digitalocean.com[22]. Τα χαρακτηριστικά του cluster φαίνονται παρακάτω:

- ✓ 5 ubuntu servers
- ✓ 16GB RAM
- ✓ 50GB SSD
- ✓ AMS3 - Ubuntu 18.04.3 (LTS) x64
- ✓ 2 core ανά μηχάνημα

Τα μηχανήματα τόσο μεταξύ τους όσο και με τον τοπικό υπολογιστή συνδέονται μέσω του πρωτοκόλλου *ssh*. Σε όλα τα μηχανήματα δόθηκε σταθερή διεύθυνση IP.

4.4 Παρουσίαση πειράματος

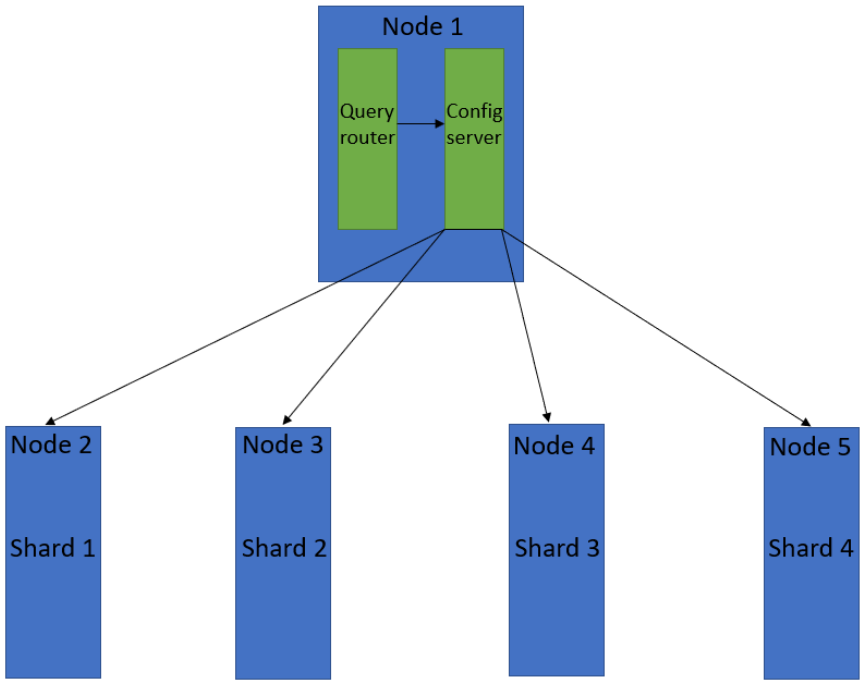
Σε αυτή την ενότητα θα παρουσιάσουμε την τοπολογία των τεχνολογιών στο cluster, την διαδικασία διεξαγωγής των μετρήσεων, τα αποτελέσματα τους καθώς επίσης και όποιες τυχόν παραδοχές χρειάστηκαν να γίνουν. Η ενότητα οργανώνεται σε υποενότητες με κάθε υποενότητα να αφιερώνεται σε μια τεχνολογία.

4.4.1 Μετρήσεις με mongoDb sharded

Όσο αφορά την mongoDb επιλέχθηκε τοπολογία με διαμοιρασμό φορτίου (sharded) σε έκδοση mongo server 4.2.

Στην εικόνα 4.3 φαίνεται η τοπολογία που δημιουργήθηκε για τις μετρήσεις. Στο πρώτο node του cluster είναι εγκατεστημένος ο query router και ο configuration server. Στα υπόλοιπα nodes έχουμε τους shard servers. Σε παραγωγικά περιβάλλοντα συνίσταται τόσο οι configuration server όσο και οι shard servers να έχουν ενεργά αντίγραφα (replicas) ώστε να διασφαλίζεται ότι δεν χάνονται δεδομένα σε περίπτωση σφαλμάτων. Για τις ανάγκες αυτής της εργασίας δεν κρίνεται απαραίτητο να ακολουθηθεί replication στρατηγική.

[22] https://www.digitalocean.com/.



Εικόνα 4.3: Τοπολογία mongoDb sharded σε cluster με 5 nodes

Η πρώτη πρόκληση που αντιμετωπίστηκε ήταν η αποθήκευση των δεδομένων στη βάση. Παρότι δεν αποτέλεσε κομμάτι άμεσης αξιολόγησης της απόδοσης της βάσης, χρήζει ιδιαίτερης αναφοράς. Η mongoDb αποθηκεύει τα δεδομένα της σε μορφή BSON (δυαδικό JSON) και δεν υποστηρίζει την απευθείας αποθήκευση δεδομένων σε μορφή Parquet. Έπρεπε λοιπόν τα δεδομένα να μετατραπούν από Parquet σε JSON με την χρήση κάποιου λογισμικού και μετά να σταλθούν στην βάση για αποθήκευση. Η μορφή Parquet όμως συμπιέζει τα δεδομένα σε αρκετά καλό βαθμό πράγμα που σημαίνει ότι κατά την μετατροπή τους σε JSON αυξάνεται αρκετά ο όγκος τους. Προκειμένου λοιπόν η τοπολογία του cluster και τα μηχανήματα να μείνουν ίδια δεν εισήχθησαν όλα τα δεδομένα στη βάση τους αλλά μέρος τους με μέγεθος 11,2GB (περίπου ίδιο με το μέγεθος του αρχικού Parquet αρχείου). Το ακριβές ποσοστό είναι 24,85% του συνόλου (βαθυμετρία 10/10, μετεωρολογικά 76/330)

Η αποθήκευση στην βάση έγινε μέσω Java εφαρμογής η οποία χρησιμοποίησε το Spark για το διάβασμα των αρχείων και τον MongoSpark Connector για την αποθήκευση. Η εφαρμογή αυτή είχε δυνατότητα επιλογής του επιθυμητού συνόλου από Parquet αρχεία. Με αυτόν τον τρόπο επιλέχθηκε ο κατάλληλος αριθμός αρχείων.

Τέλος, αφού ολοκληρώθηκε η εισαγωγή των αρχείων στην βάση εκτελέστηκαν όλα τα ερωτήματα του πίνακα 4.3. Η εκτέλεση των ερωτημάτων έγινε μέσω του mongo shell και τα στατιστικά στοιχεία ελήφθησαν μέσω της εντολής *explain("executionStats")* μετά από κάθε ερώτημα. Στον πίνακα 4.4 φαίνεται η σύνταξη των ερωτημάτων σε mongoDb ενώ στον πίνακα 4.5 φαίνονται τα στατιστικά στοιχεία της εκτέλεσης των ερωτημάτων.

Πίνακας 4.4: Ερωτήματα σε mongoDb

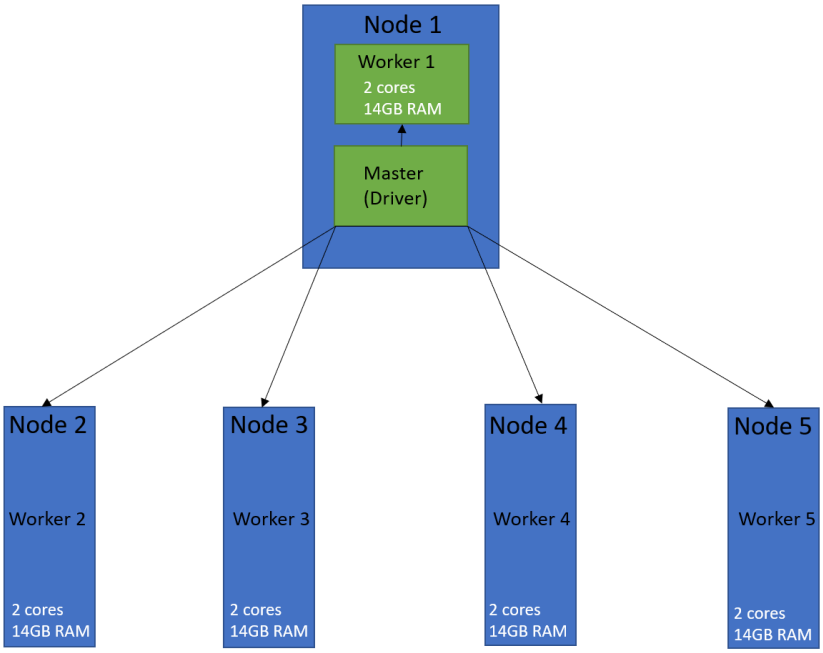
Ερώτημα	***Σύνταξη ερωτήματος για mongoDb***
Q1	*db.vathymetry.find({});*
Q2	*db.forecast.find({});*
Q3	*db.forecast.find({latitude:42});*
Q4	*db.forecast.find({latitude: {$gt: 42, $lt: 42.5}});*
Q5	*db.forecast.find({$and: [{ air_pressure_at_sea_level: { $gt: 1010 } }, { air_pressure_at_sea_level: { $lt: 1010.5 } }]});*
Q6	*db.forecast.aggregate([{$match: {latitude: {$gt: 42, $lt: 42.5} } }, {"$group": { "_id": { time: "$time" }}}]);*
Q7	*db.forecast.aggregate([{$match: {latitude: {$gt: 0, $lt: 1500} } }, {"$group": { "_id": { time: "$time" }}}]);*
Q8	*db.vathymetry.aggregate([{"$group": { "_id": { depth: "$depth" }}}]);*
Q9	*db.forecast.aggregate([{ $lookup: {from: "vathymetry", let: { coll2_latitude: "$latitude", coll2_longitude: "$longitude" }, pipeline: [{ $match: { $expr: { $and: [{ $eq: ["$latitude", "$$coll2_latitude"] },{ $eq: ["$longitude", "$$coll2_longitude"] }]}}}],as: "vathymetryData"}}]);*
Q10	*db.forecast.find({ latitude: {$gt: 42, $lt: 42.5}, air_temperature_2m: { $gt: 29 } }).sort({air_temperature_2m: 1});*
Q11	*db.forecast.aggregate([{$match: {latitude: {$gt: 42, $lt: 42.5} } }, {"$group": { "_id": { time: "$time" }}},{$count: "timestamps"}]);*
Q12	*db.forecast.aggregate([{$match: {latitude: {$gt: 42, $lt: 42.5} } }, { $group: { _id: "$time", avgTemprature: { $avg: "$air_temperature_2m" }}}]);*
Q13	*db.forecast.aggregate([{$match: {latitude: {$gt: 35, $lt: 45} } }, { $group: { _id: "$time", sumTemprature: { $sum: "$air_temperature_2m" }}}]);*
Q14	*db.forecast.aggregate([{$match: {latitude: {$gt: 42, $lt: 42.5} } }, { $group: { _id: "$time", sumTemprature: { $sum: "$air_temperature_2m" }}}]);*

Πίνακας 4.5: Στατιστικά στοιχεία εκτέλεσης ερωτημάτων σε mongoDb

Ερώτημα	Χρόνος (ms)	Εγγραφές
Q1	93	100.804
Q2	88.281	89.754.285
Q3	1.754	256.806
Q4	8.281	2.590.458
Q5	28.767	2.046.929
Q6	23.093	1.297
Q7	809.452	1.297
Q8	181	21.538
Q9	325.236	3.353.474
Q10	6.253	28.019
Q11	22.910	1.297
Q12	23.997	1.297
Q13	536.587	1.297
Q14	24.252	1.297

4.4.2 Μετρήσεις με SparkSQL σε τοπολογία standalone cluster

Για την εκτέλεση των μετρήσεων με SparkSQL χρησιμοποιήθηκε Apache Spark σε έκδοση 2.4.4 και σε τοπολογία standalone cluster. Η τοπολογία αυτή είναι η πιο απλή που προσφέρει το Spark καθώς η διαχείριση των πόρων του cluster αναλαμβάνεται από το ίδιο. Το μόνο που απαιτείται είναι να μπορούν να επικοινωνούν οι κόμβοι μεταξύ τους μέσω του πρωτοκόλλου ssh χωρίς να χρειάζεται να παρέχουν κάποιο κωδικό ασφαλείας.



Εικόνα 4.4: Τοπολογία Spark σε standalone cluster με 5 nodes

Στην εικόνα 4.4 φαίνεται η τοπολογία που χρησιμοποιήθηκε. Στον πρώτο κόμβο υπάρχει η διεργασία οδηγητής (driver) ενώ παράλληλα υπάρχει και ο πρώτος εργάτης (worker). Στους υπόλοιπους τέσσερεις κόμβους υπάρχουν διεργασίες εργάτες. Η πιο κοινή πρόκληση που αντιμετωπίστηκε κατά την διάρκεια των πειραμάτων ήταν η εξάντληση όλης της διαθέσιμης μνήμης σε κάποια από τις διεργασίες εργάτες. Για αυτό τον λόγο δόθηκε αρκετή RAM σε όλες τις διεργασίες εργάτες. Πιο συγκεκριμένα, σε κάθε διεργασία εργάτη δίδονται 14GB RAM και οι δύο πυρήνες που διαθέτει το μηχάνημα, ενώ στην διεργασία οδηγητή δίνουμε 2GB RAM.

Για την εκτέλεση των ερωτημάτων του πίνακα 4.3 δημιουργήθηκε μια εφαρμογή γραμμής εντολών σε Java. Με χρήση του API της Spark διαβάστηκαν τα δεδομένα από το αρχείο Parquet και φορτώθηκαν στην μνήμη ως Datasets. Έπειτα πάνω στα dataset δημιουργήθηκε η εικόνα ενός εικονικού πίνακα ο οποίος και χρησιμοποιήθηκε στα ερωτήματα που εκτελέστηκαν πάνω του. Τα στατιστικά στοιχεία των ερωτημάτων λαμβάνονται προγραμματιστικά μετρώντας το χρόνο πριν και μετά την εκτέλεση του κάθε ερωτήματος. Επίσης το σύνολο των εγγραφών που επιστρέφονται από κάθε ερώτημα λαμβάνεται από το μέγεθος του αντίστοιχου dataset.

Το παραπάνω πρόγραμμα εκτελέστηκε ως Spark job και με την ολοκλήρωση του παρήγαγε μια αναφορά με τα επιθυμητά στατιστικά στοιχεία. Στον πίνακα 4.6 φαίνεται η σύνταξη των ερωτημάτων σε SparkSQL ενώ στον πίνακα 4.7 φαίνονται τα στατιστικά στοιχεία της εκτέλεσης των ερωτημάτων.

Πίνακας 4.6: Ερωτήματα σε SparkSQL

Ερώτημα	*Σύνταξη ερωτήματος για SparkSQL*
Q1	*SELECT * FROM vathymetry;*
Q2	*SELECT * FROM forecast;*
Q3	*SELECT * FROM forecast WHERE latitude=42;*
Q4	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5;*
Q5	*SELECT * FROM forecast WHERE air_pressure_at_sea_level>1010 AND air_pressure_at_sea_level<1010.5;*
Q6	*SELECT first(latitude), first(longitude), time, first(air_temperature_2m), first(x_wind), first(y_wind), first(air_pressure_at_sea_level), first(source), first(observation) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q7	*SELECT first(latitude), first(longitude), time, first(air_temperature_2m), first(x_wind), first(y_wind), first(air_pressure_at_sea_level), first(source), first(observation) FROM forecast WHERE latitude>0 AND latitude<1500 GROUP BY time;*
Q8	*SELECT first(latitude), first(longitude), depth, first(source), first(observation), first(metadata_repository_id) FROM vathymetry GROUP BY depth;*
Q9	*SELECT * FROM forecast f JOIN vathymetry v ON f.latitude = v.latitude AND f.longitude = v.longitude;*
Q10	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5 AND air_temperature_2m>29 ORDER BY air_temperature_2m;*
Q11	*SELECT COUNT(time) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q12	*SELECT AVG(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q13	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>35 AND latitude<45 GROUP BY time;*
Q14	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*

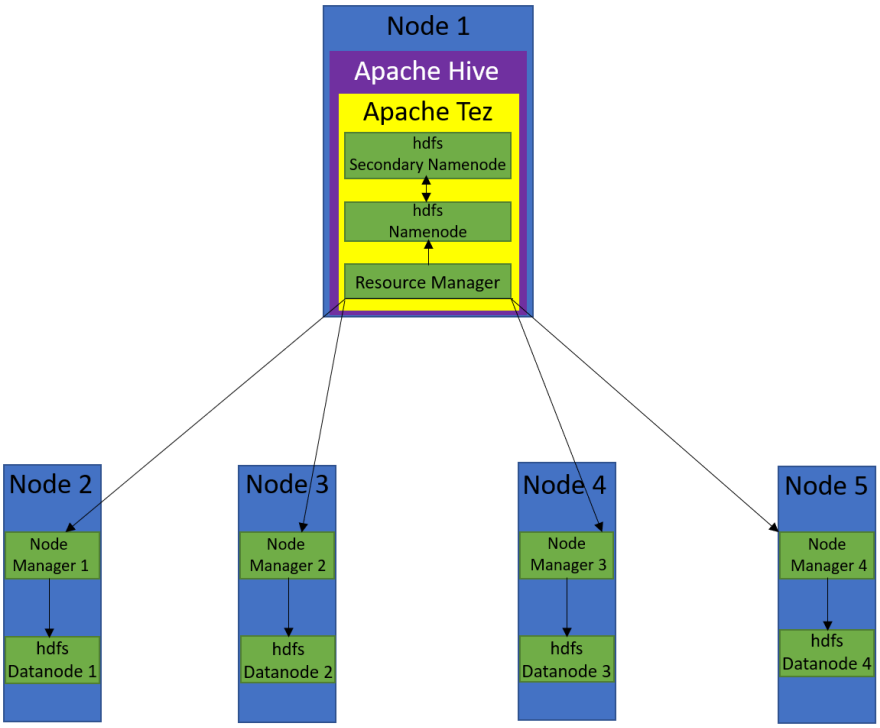
Πίνακας 4.7: Στατιστικά στοιχεία εκτέλεσης ερωτημάτων σε SparkSQL

Ερώτημα	Χρόνος (ms)	Εγγραφές
Q1	2.850	100.804
Q2	73.047	387.609.747
Q3	83.461	1.207.507
Q4	72.142	10.867.563
Q5	73.475	8.856.057
Q6	119.892	1.297
Q7	139.386	1.297
Q8	1.929	21.538
Q9	98.743	14.710.574
Q10	196.940	123.984
Q11	115.173	1.297
Q12	118.165	1.297
Q13	114.371	1.297
Q14	97.279	1.297

4.4.3 Μετρήσεις με HiveQL, hdfs και Apache Tez

Για την εκτέλεση των μετρήσεων με HiveQL χρειάστηκε να ετοιμαστεί ένα κατανεμημένο σύστημα αποθήκευσης αρχείων hdfs στο cluster και μια μηχανή επεξεργασίας των ερωτημάτων όπου επιλέχθηκε το Apache Tez. Συγκεκριμένα, χρησιμοποιήσαμε Apache Hadoop σε έκδοση 2.10.0, Apache Hive σε έκδοση 2.3.6 και Apache Tez σε έκδοση 0.9.1.

Στην εικόνα 4.5 φαίνεται η τοπολογία που χρησιμοποιήθηκε για τις μετρήσεις. Ο πρώτος κόμβος έχει ρόλο master και έχει εγκατεστημένο πάνω του τον Namenode και τον Secondary Namenode του hdfs καθώς επίσης και τον Resource Manager του YARN. Ακόμα, στον πρώτο κόμβο είναι εγκατεστημένα και τα Apache Hive και Apache Tez. Αποτελεί, λοιπόν, τον κόμβο όπου τρέχουν οι εξωτερικές εφαρμογές και συντονίζει την εκτέλεση των εργασιών των υπολοίπων κόμβων. Όλοι οι υπόλοιποι κόμβοι (2-5) είναι ουσιαστικά datanodes και εκεί πραγματοποιούνται όλοι οι υπολογισμοί.



Εικόνα 4.5: Τοπολογία Hive με hdfs σε cluster με 5 nodes και Tez για execution engine

Για την εκτέλεση των ερωτημάτων του πίνακα 4.3 χρειάστηκαν δύο βήματα. Πρώτον, τα δεδομένα έπρεπε να μεταφερθούν από την φυσική τοποθεσία τους στον πρώτο κόμβο στο hdfs προκειμένου να είναι ορατά από το Hive. Δεύτερο, χρησιμοποιήθηκε το JDBC API που παρέχει το Hive ώστε να φτιαχτεί μια Java εφαρμογή που θα δημιουργεί τους πίνακες στο Hive, θα εκτελεί τα ερωτήματα και θα συλλέγει τα στατιστικά. Εδώ, αξίζει να αναφερθεί ότι οι πίνακες με μορφή Parquet που προσφέρει το Hive αν και βολικοί για την αποθήκευση παρουσίαζαν προβλήματα στην απόδοση, για αυτό τον λόγο αποφασίστηκε τα δεδομένα από τους Parquet πίνακες να μεταφερθούν σε πίνακες τύπου Orc.

Αφού ακολουθήθηκαν τα παραπάνω βήματα έτρεξε η εφαρμογή σε Java ως Hive job και παρήγαγε τις μετρήσεις. Στον πίνακα 4.8 φαίνεται η σύνταξη των ερωτημάτων σε HiveQL ενώ στον πίνακα 4.9 φαίνονται τα στατιστικά στοιχεία της εκτέλεσης των ερωτημάτων.

Πίνακας 4.8: Ερωτήματα σε HiveQL

Ερώτημα	Σύνταξη ερωτήματος για HiveQL
Q1	*SELECT * FROM vathymetry;*
Q2	*SELECT * FROM forecast;*
Q3	*SELECT * FROM forecast WHERE latitude=42;*
Q4	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5;*
Q5	*SELECT * FROM forecast WHERE air_pressure_at_sea_level>1010 AND air_pressure_at_sea_level<1010.5;*
Q6	*SELECT collect_set(latitude), collect_set(longitude), time, collect_set(air_temperature_2m), collect_set(x_wind), collect_set(y_wind), collect_set(air_pressure_at_sea_level), collect_set(source), collect_set(observation) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q7	*SELECT collect_set(latitude), collect_set(longitude), time, collect_set(air_temperature_2m), collect_set(x_wind), collect_set(y_wind), collect_set(air_pressure_at_sea_level), collect_set(source), collect_set(observation) FROM forecast WHERE latitude>0 AND latitude<1500 GROUP BY time;*
Q8	*SELECT collect_set(latitude), collect_set(longitude), depth, collect_set(source), collect_set(observation), collect_set(metadata_repository_id) FROM vathymetry GROUP BY depth;*
Q9	*SELECT * FROM forecast f JOIN vathymetry v ON f.latitude = v.latitude AND f.longitude = v.longitude;*
Q10	*SELECT * FROM forecast WHERE latitude>42 AND latitude<42.5 AND air_temperature_2m>29 ORDER BY air_temperature_2m;*
Q11	*SELECT COUNT(time) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q12	*SELECT AVG(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*
Q13	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>35 AND latitude<45 GROUP BY time;*
Q14	*SELECT SUM(air_temperature_2m) FROM forecast WHERE latitude>42 AND latitude<42.5 GROUP BY time;*

Πίνακας 4.9: Στατιστικά στοιχεία εκτέλεσης ερωτημάτων σε HiveQL

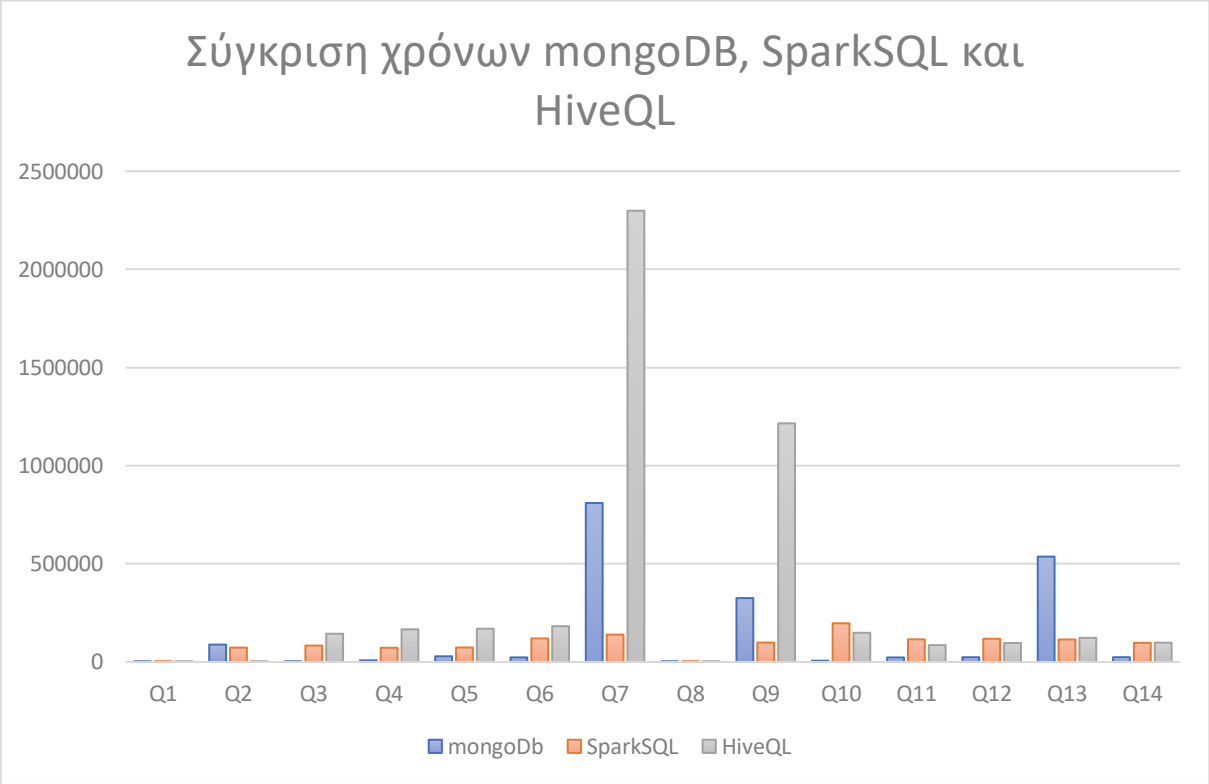
Ερώτημα	*Χρόνος (ms)*	*Εγγραφές*
Q1	*1.003*	*100.804*
Q2	*336*	*387.609.747*
Q3	*143.637*	*1.207.507*
Q4	*166.135*	*10.867.563*
Q5	*169.249*	*8.856.057*
Q6	*182.128*	*1.297*
Q7	*2.298.430*	*1.297*
Q8	*2.930*	*21.538*
Q9	*1.215.380*	*14.710.574*
Q10	*148.450*	*123.984*
Q11	*85.558*	*1.297*
Q12	*96.397*	*1.297*
Q13	*122.929*	*1.297*
Q14	*98.205*	*1.297*

Κεφάλαιο 5. Αξιολόγηση αποτελεσμάτων

Στόχος αυτού του κεφαλαίου είναι να παρουσιαστούν συγκριτικά τα αποτελέσματα των πειραμάτων του κεφαλαίου 4, να σχολιαστούν οι διαφορές που παρουσιάζονται και να προταθούν ιδέες για περεταίρω ανάπτυξη της έρευνας.

5.1 Σύγκριση αποτελεσμάτων και σχολιασμός

Με βάση τα αποτελέσματα από την εκτέλεση των ερωτημάτων του πίνακα 4.3 για mongoDb, SparkSQL και HiveQL όπως αυτά φαίνονται στους πίνακες 4.5, 4.7 και 4.9 προκύπτει ο πίνακας 5.1 και σχεδιάστηκε το σχεδιάγραμμα του σχήματος 5.1.



Σχήμα 5.1: Συγκριτικό διάγραμμα αποτελεσμάτων για mongoDb, SparkSQL και HiveQL

Πίνακας 5.1: Συγκριτικός πίνακας χρόνου απόκρισης για όλα τα ερωτήματα

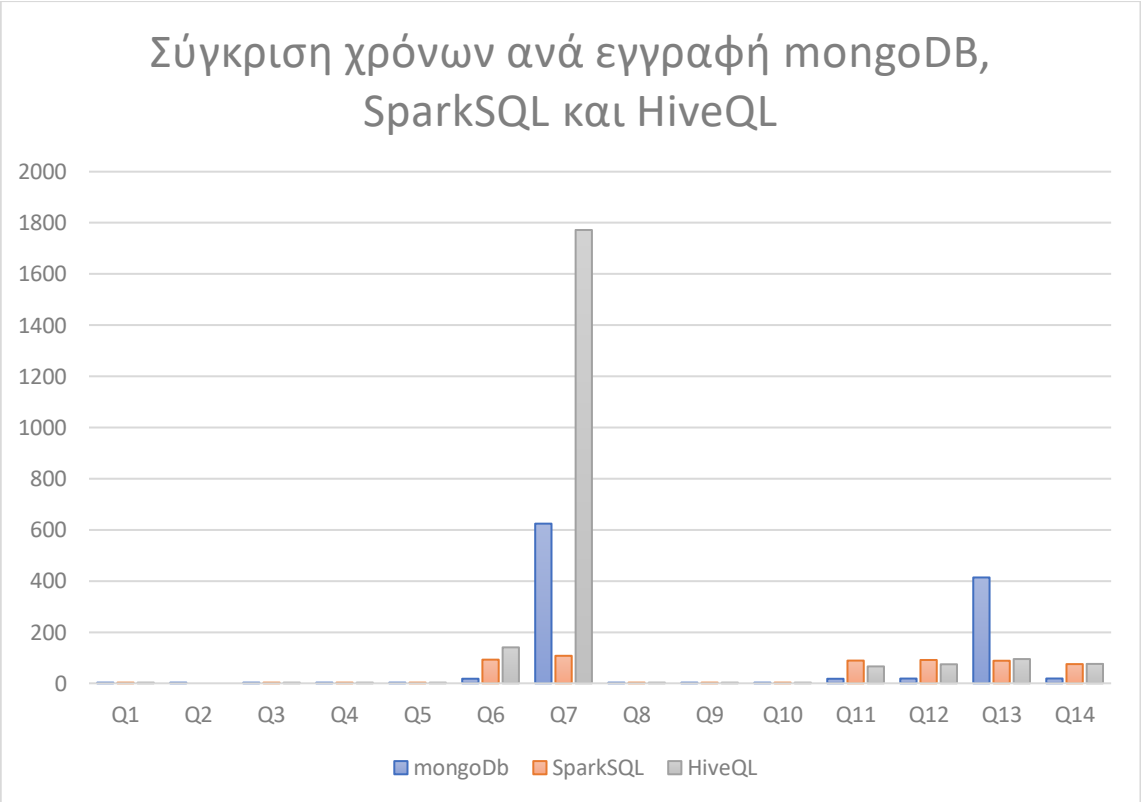
Ερώτημα	*mongoDb (ms)*	*SparkSQL (ms)*	*HiveQL (ms)*
Q1	*93*	*2.850*	*1.003*
Q2	*88.281*	*73.047*	*336*
Q3	*1.754*	*83.461*	*143.637*
Q4	*8.281*	*72.142*	*166.135*
Q5	*28.767*	*73.475*	*169.249*

Q6	*23.093*	*119.892*	*182.128*
Q7	*809.452*	*139.386*	*2.298.430*
Q8	*181*	*1.929*	*2.930*
Q9	*325.236*	*98.743*	*1.215.380*
Q10	*6.253*	*196.940*	*148.450*
Q11	*22.910*	*115.173*	*85.558*
Q12	*23.997*	*118.165*	*96.397*
Q13	*536.587*	*114.371*	*122.929*
Q14	*24.252*	*97.279*	*98.205*

Αρχικά θα συγκρίνουμε SparkSQL και HiveQL καθώς οι χρόνοι που μετρήθηκαν αφορούν ακριβώς το ίδιο σύνολο δεδομένων (πλήθος εγγραφών και μέγεθος). Στα Q1, Q2 και οι δυο τεχνολογίες παρουσιάζουν παρόμοια συμπεριφορά με το Hive να ξεχωρίζει όμως στο Q2. Το Q2 φέρνει όλα τα δεδομένα του πίνακα με τον μεγαλύτερο όγκο δεδομένων (forecast) εδώ το Hive έχει αποθηκευμένα τα δεδομένα στον δίσκο του κόμβου 1 και τα φέρνει άμεσα όταν ζητώνται, ενώ το Spark τα έχει στη μνήμη RAM μεν αλλά κατανεμημένα σε όλους τους κόμβους οπότε εισέρχεται καθυστέρηση κυρίως από την μεταφορά των δεδομένων μέσω του δικτύου. Για τα Q3-Q5 που αφορούν αναζήτηση με κριτήρια, τα κριτήρια αυτά λαμβάνουν υπόψη τον τρόπο που είναι κατανεμημένα τα δεδομένα στα διάφορα μηχανήματα, παρατηρείται μια σαφής υπεροχή του Spark. Εδώ απαιτούνται να γίνουν κάποιοι υπολογισμοί για να εντοπιστεί το σύνολο των δεδομένων που ικανοποιεί τα κριτήρια, αυτός είναι και ο λόγος που το Hive που στηρίζεται στην ακριβή υπολογιστικά διαδικασία του Map-Reduce έχει χειρότερους χρόνους. Τα ερωτήματα Q6-Q8 αφορούν ταξινόμηση με group by μέθοδο. Εδώ αν αγνοήσουμε το Q8, που αφορά τον μικρό σε μέγεθος πίνακα με τα δεδομένα βαθυμετρίας που και οι δύο τεχνολογίες έχουν παρόμοια αποτελέσματα, παρατηρούμε σαφή υπεροχή του Spark. Ειδικά το Q7 έπαιρνε πολύ χρόνο στο Hive και ανάγκαζε όλη την εφαρμογή να αποτυγχάνει. Για αυτό τον λόγο απομονώθηκε και εκτελέστηκε ξεχωριστά. Να σημειωθεί ότι το Q7 ταξινομεί όλα τα δεδομένα του πίνακα μετεωρολογίας με βάση τον χρόνο. Είναι αναμενόμενη δηλαδή η χειρότερη απόδοση από τα άλλα ερωτήματα, όμως φαίνεται και η μεγάλη διαφορά με το Spark. Για το ερώτημα Q9 που εφαρμόζει ένα join μεταξύ των δύο πινάκων παρατηρούμε και πάλι αρκετά κακή επίδοση του Hive. Για το Q10 που εφαρμόζει order by σε ένα μικρό υποσύνολο των δεδομένων παρατηρούμε καλύτερη απόδοση για το Hive. Το αποτέλεσμα αυτό μπορεί να εξηγηθεί και πάλι όπως και για το Q2 με βάση την τοπικότητα των δεδομένων. Τέλος για τα ερωτήματα Q11-Q14 που αφορούν συναθροιστικές συναρτήσεις σε ένα μικρό υποσύνολο των δεδομένων παρατηρούμε παρόμοιες χρονικές αποκρίσεις, με το Hive όμως να έχει μια ελαφρώς καλύτερη εικόνα.

Η mongoDb αποτελεί ιδιαίτερη περίπτωση καθώς λόγω της μορφής JSON που δέχεται αποκλειστικά για να αποθηκεύσει δεδομένα οδήγησε στην αποθήκευση μικρότερου μέρους του συνόλου των δεδομένων (~25%). Έτσι, παρόλο που ο όγκος των αποθηκευμένων στην βάση δεδομένων είναι ίδιος σε όλες τις τεχνολογίες, στην mongo γίνεται αναζήτηση μεταξύ λιγότερων εγγραφών. Είναι λογικό λοιπόν στα περισσότερα ερωτήματα να παρουσιάζονται καλύτεροι χρόνοι. Αξίζει να αναφερθούν τα ερωτήματα Q2, Q7, Q9 και Q13 που η mongo παρουσιάζει υψηλότερους χρόνους σε σχέση με το Spark. Τα παραπάνω ερωτήματα είναι τα μόνα που περιλαμβάνουν μεγαλύτερο εύρος δεδομένων, μερικά από αυτά αφορούν και το σύνολο των δεδομένων. Εδώ το γεγονός ότι η mongo αποθηκεύει τα δεδομένα της στον δίσκο και όχι στη μνήμη RAM όπως το Spark δικαιολογεί τους αυξημένους χρόνους. Τέλος, φαίνεται ότι τα συστήματα cache που έχει η μηχανή εκτέλεσης της mongo, το WireTiger, δεν επαρκούν για να δώσουν ένα καλύτερο αποτέλεσμα.

Προκειμένου συγκριτικά να πάρουμε μια καλύτερη εικόνα μεταξύ των τεχνολογιών επιλέχθηκε να χρησιμοποιηθεί ως μέτρο σύγκρισης ο λόγος μεταξύ του χρόνου απόκρισης ερωτήματος και του αριθμού των εγγραφών στο αποτέλεσμα. Το διάγραμμα που προκύπτει με αυτούς τους υπολογισμούς φαίνεται στο σχήμα 5.2 και τα αποτελέσματα των υπολογισμών στον πίνακα 5.2.



Σχήμα 5.2: Συγκριτικό διάγραμμα αποτελεσμάτων για mongoDb, SparkSQL και HiveQL (χρόνος/εγγραφή)

Πίνακας 5.2: Συγκριτικός πίνακας χρόνου ανά εγγραφή για όλα τα ερωτήματα

Ερώτημα	*mongoDb (ms/εγγραφή)*	*SparkSQL (ms/εγγραφή)*	*HiveQL (ms/εγγραφή)*
Q1	*0,000922582*	*0,028272688*	*0,009950002*
Q2	*0,000983585*	*0,000188455*	*0,000000867*
Q3	*0,006830058*	*0,06911844*	*0,118953348*
Q4	*0,003196732*	*0,006638287*	*0,015287236*
Q5	*0,014053736*	*0,008296582*	*0,019111101*
Q6	*17,80493446*	*92,43793369*	*140,4225135*
Q7	*624,0956052*	*107,4680031*	*1772,112567*
Q8	*0,008403752*	*0,089562633*	*0,136038629*
Q9	*0,096984798*	*0,006712383*	*0,082619482*
Q10	*0,223169992*	*1,588430765*	*1,197331914*
Q11	*17,66383963*	*88,79953739*	*65,96607556*
Q12	*18,50192753*	*91,10639938*	*74,3230532*
Q13	*413,7139553*	*88,18118736*	*94,77949113*
Q14	*18,69853508*	*75,00308404*	*75,71703932*

Μελετώντας το σχήμα 5.2 και τον πίνακα 5.2, προκύπτει ότι επιβεβαιώνονται οι υποψίες για κακή απόδοση στα Q2, Q7, Q9 και Q13 ενώ αρκετά καλές επιδόσεις παρουσιάζονται στα Q6, Q11, Q12 και Q14. Στην τελευταία τετράδα ερωτημάτων με καλή επίδοση συμβάλλει το γεγονός ότι είναι μικρό το υποσύνολο των δεδομένων που παρουσιάζεται ενώ φαίνεται η mongo να έχει βελτιστοποιημένους αλγόριθμους σε πράξεις που αφορούν το ταίριασμα εγγραφών και την εφαρμογή απλών πράξεων όπως πρόσθεση και μέσο όρο.

5.2 Συμπεράσματα έρευνας

Ανακεφαλαιώνοντας τα σχόλια της προηγούμενης ενότητας προκύπτει ότι όσο αφορά τον χρόνο απόκρισης η SparkSQL αποτελεί την καλύτερη λύση. Αναλυτικότερα, συγκρίνοντας τις SparkSQL και HiveQL η πρώτη έχει καλύτερη επίδοση στα περισσότερα ερωτήματα και ειδικά σε αυτά που απαιτούν υπολογισμούς σε μεγάλο όγκο δεδομένων. Από την άλλη το Hive έχει καλύτερη επίδοση στα ερωτήματα που αφορούν μικρά υποσύνολα, όταν όμως τα υποσύνολα αυτά μεγαλώνουν σε μέγεθος η επίδοση χειροτερεύει εκθετικά. Τέλος, η mongoDb παρουσιάζει και αυτή κακές επιδόσεις όταν πρόκειται να μεταχειριστεί μεγάλο όγκο δεδομένων, ενώ έχει αρκετά

καλούς χρόνους σε ερωτήματα που εκτελούν απλές πράξεις πάνω στα δεδομένα (πρόσθεση, απαρίθμηση, μέσος όρος).

Για κάθε τεχνολογία προκύπτουν τα ακόλουθα συμπεράσματα:

- ✓ Η mongoDb λόγω της μορφής αποθήκευσης των δεδομένων της σε BSON υστερεί σημαντικά συγκριτικά με άλλες τεχνολογίες για μεγάλης κλίμακας δεδομένα. Παρότι πολλές φορές η επίδοση της είναι συγκρίσιμη με το Hive, απαιτεί μηχανήματα με περισσότερη RAM και αποθηκευτικό χώρο γεγονός που την καθιστά μη ανταγωνιστική. Η μόνη περίπτωση που θα αποτελούσε μονόδρομο η mongo είναι όταν το σχήμα των δεδομένων δεν είναι καθορισμένο και απαιτείται ελαστικότητα σχήματος
- ✓ Η Apache Hive επειδή βασίζεται εν πολλοίς στο map-reduce αλγόριθμο για την εκτέλεση των υπολογισμών που απαιτούνται παρουσιάζει γενικά αυξημένους χρόνους απάντησης. Όταν όμως ο χρόνος απάντησης δεν είναι το μόνο κριτήριο, το Hive κερδίζει γιατί υποστηρίζεται από το Hadoop για τους υπολογισμούς του μειώνοντας έτσι τις απαιτήσεις για RAM σε σύγκριση με το Spark πάντα.
- ✓ Η SparkSQL καθώς αποτελεί κυρίως μια υπολογιστική μηχανή και ότι δεδομένα χρειάζεται τα κρατάει στην μνήμη RAM παρουσιάζει τους καλύτερους χρόνους απόκρισης στα περισσότερα των ερωτημάτων. Κερδίζει δε και το στοίχημα όταν πρέπει να κάνει υπολογισμούς σε ένα μεγαλύτερο σύνολο από δεδομένα. Όταν δεν υπάρχει όμως απαίτηση για γρήγορη απόκριση, ίσως αποτελεί μια ακριβή λύση λόγω των απαιτήσεων σε RAM για όποιον την επιλέξει.

Από τα παραπάνω συμπεράσματα γίνεται κατανοητό ότι δεν υπάρχει μια γενικά αποδεκτή καλύτερη λύση αλλά η επιλογή εξαρτάται από το εκάστοτε πρόβλημα και τις απαιτήσεις που υπάρχουν.

5.3 Προτάσεις για μελλοντική έρευνα

Σε γεωχωρικά δεδομένα του είδους που μελετήθηκαν στην παρούσα διπλωματική θα ήταν χρήσιμο τα δεδομένα να διαχωρίζονται με βάση την χρονική στιγμή και όχι με βάση το γεωγραφικό μήκος και πλάτος όπως επιλέχθηκε. Για να γίνει αυτό δυνατό θα έπρεπε στα αρχικά δεδομένα να γίνει κάποια επεξεργασία της χρονικής σφραγίδας που είχαν, ώστε τα δεδομένα να έχουν ακρίβεια μέρας και όχι χιλιοστού του δευτερολέπτου. Με αυτό τον τρόπο θα ήταν πιο εύκολο τα δεδομένα να χωριστούν σε ένα μικρό αριθμό από μέρη-καλάθια, όπου κάθε καλάθι θα είχε ίδια ημερομηνία για παράδειγμα. Και στις τρεις τεχνολογίες που μελετήσαμε υποστηρίζονται τέτοιου είδους partitions τα οποία είναι πολύ πιθανό να φέρναν και καλύτερες αποδόσεις. Με την επιλογή που έχει γίνει τώρα εξαιτίας του μεγάλου πλήθους διαφορετικών συντεταγμένων είναι δύσκολο να αξιοποιηθούν τέτοιου είδους βελτιστοποιήσεις.

Επιπλέον, θα μπορούσε η παρούσα ερεύνα να εμπλουτιστεί περισσότερο ώστε να εξετάζει περισσότερες πτυχές του προβλήματος. Για παράδειγμα, στα υπάρχοντα ερωτήματα θα μπορούσαν προστεθούν και άλλα που να αφορούν ανανέωση τιμών (update), εισαγωγή εγγραφών (insert) και διαγραφή εγγραφών (delete). Ακόμη, σαν μέτρο αξιολόγησης των τεχνολογιών εκτός από τον χρόνο απόκρισης θα μπορούσαν να συνεκτιμηθούν και άλλες σημαντικές παράμετροι όπως η χρήση RAM, CPU, σκληρού δίσκου και δικτύου. Έτσι, ο αναγνώστης θα είχε περισσότερα κριτήρια για να επιλέξει ποια τεχνολογία ταιριάζει καλύτερα στις ανάγκες του.

Θα μπορούσε επίσης να γίνει και μια διερεύνηση το κατά πόσο οι διαφορετικές τεχνολογίες αποθήκευσης αρχείων επηρεάζουν την απόδοση των τεχνολογιών. Έτσι, μια καλή ιδέα θα ήταν μαζί με το Parquet να εξετάζαμε και αποθήκευση σε Avro ή/και orc. Τέλος, θα είχε ενδιαφέρον να χρησιμοποιούνταν υπολογιστές με μεγαλύτερες δυνατότητες σε RAM ώστε να συγκριθεί η mongo σε όλο τον όγκο των δεδομένων.

Βιβλιογραφία

1. Marr, Bernard. *www.forbes.com.* [Ηλεκτρονικό] 21 05 2018. https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#795e07a760ba.

2. *Big Data in Ocean Observation: Opportunities.* Yingjian Liu, Meng Qiu, Chao Liu, and Zhongwen Guo. 2016.

3. *Big Data in the Maritime Industry.* Maris Mirović, Mario Miličević, Ines Obradović. s.l. : Nase More, 2018.

4. *Benchmarking Big Data Systems: A Review.* Rui Han, Member, IEEE, Lizy Kurian John, Fellow, IEEE, Jianfeng Zhan, Member, IEEE. s.l. : IEEE, 2017.

5. https://en.wikipedia.org/. [Ηλεκτρονικό] [Παραπομπή: 25 01 2020.] https://en.wikipedia.org/wiki/Big_data.

6. *Big Data- A Review on Analysing 3Vs.* Abhinandan Banik, Samir Kumar Bandyopadhyay. s.l. : Journal of Scientific and Engineering Research,, 2016.

7. *https://www.gutcheckit.com/.* [Ηλεκτρονικό] [Παραπομπή: 25 01 2020.]

https://www.gutcheckit.com/blog/veracity-big-data-v/.

8. *Efficient Big Data Storage and Retrieval in Multimedia Cloud Computing Systems.* Τέντες, Γεώργιος. Αθήνα : ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ Ε.Μ.Π, 2014.

9. https://nosql-database.org/. [Ηλεκτρονικό] [Παραπομπή: 26 01 2020.]

10. *Benchmarking big data systems: A survey.* Fuad Bajaber, Sherif Sakr, Omar Batarfi, Abdulrahman Altalhi, Ahmed Barnawi. s.l. : Computer Communications, Elsevier, 2020.

11. *Pnuts: Yahoo!'s hosted data serving.* B.F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A.Jacobsen, N. Puz, D. Weaver, R. Yerneni. s.l. : Proc. VLDB Endow. 1 (2), 2008, Τόμ. 1277–1288.

12. *Ycsb++: benchmarking and performance debugging advanced features.* S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. López, G. Gibson, A. Fuchs,. s.l. : Proceedings of the 2nd ACM Symposium on Cloud, ACM, 2011.

13. *YCSB+T: Benchmarking Web-scale Transactional.* Akon Dey, Alan Fekete, Raghunath Nambiar, Uwe Rohm. s.l. : IEEE, 2014.

14. https://github.com/. *HiBench Suite.* [Ηλεκτρονικό] [Παραπομπή: 30 1 2020.] https://github.com/Intel-bigdata/HiBench.

15. *SPARKBENCH: A Comprehensive Benchmarking Suite For In Memory Data Analytic Platform Spark.* Min Li, Jian Tan, Yandong Wang, Li Zhang, Valentina Salapura. s.l. : Cluster Computing, 2017.

16. http://www.tpc.org/. [Ηλεκτρονικό] [Παραπομπή: 30 1 2020.] http://www.tpc.org/tpch/.

17. http://www.tpc.org/. [Ηλεκτρονικό] [Παραπομπή: 30 1 2020.] http://www.tpc.org/tpcds/.

18. http://www.tpc.org/. [Ηλεκτρονικό] [Παραπομπή: 30 1 2020.] http://www.tpc.org/tpcx-bb/.

19. *NoSQL Databases: MongoDB vs Cassandra* . Veronika Abramova, Jorge Bernardino. 2013.

20. *Quantitative Analysis of Scalable NoSQL Databases.* Surya Narayanan Swaminathan, Ramez Elmasri. s.l. : IEEE International Congress on Big Data, 2016.

21. *Big SQL systems: an experimental evaluation.* Victor Aluko, Sherif Sakr. s.l. : Cluster Computing Springer, 2019.

22. Dremel made simple with Parquet. [Ηλεκτρονικό] [Παραπομπή: 1 2 2020.] https://blog.twitter.com/engineering/en_us/a/2013/dremel-made-simple-with-parquet.html.

23. What is MapReduce? How it Works - Hadoop MapReduce Tutorial. [Ηλεκτρονικό] 1 2 2020. https://www.guru99.com/introduction-to-mapreduce.html.

24. Apache Hadoop. [Ηλεκτρονικό] [Παραπομπή: 2 2 2020.] https://en.wikipedia.org/wiki/Apache_Hadoop.

25. Hadoop Components that you Need to know about. [Ηλεκτρονικό] [Παραπομπή: 2 2 2020.] https://www.edureka.co/blog/every-hadoop-component/.

26. Apache Hive (A Complete Journey) Series-1 of 3. [Ηλεκτρονικό] [Παραπομπή: 2 2 2020.] https://medium.com/@zaman.nuces/apache-hive-a-brief-introduction-series-1-of-3-54ec07753dd2.

27. High Level Overview of Apache Spark. [Ηλεκτρονικό] [Παραπομπή: 6 2 2020.] https://medium.com/better-programming/high-level-overview-of-apache-spark-c225a0a162e9.

28. *Spark SQL: Relational Data Processing in Spark.* Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M. Chicago : SIGMOD, 2015.

29. *A Comparative Study: MongoDB vs. MySQL.* Cornelia GYŐRÖDI, Robert GYŐRÖDI, George PECHERLE, Andrada OLAH. Oradea : The 13th International Conference on Engineering of Modern Electric Systems, 2015.

30. Sharding and Replication in MongoDB. [Ηλεκτρονικό] [Παραπομπή: 6 2 2020.] https://medium.com/@anshukumar_14390/sharding-and-replication-in-mongodb-2b0ebc30e467.

31. Introduction to MongoDB! [Ηλεκτρονικό] [Παραπομπή: 6 2 2020.] https://medium.com/@saivittalb/introduction-to-mongodb-859ed4426994.