ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

"ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΡΜΠΙΛΑ ΘΕΟΔΩΡΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΒΑΣΙΛΗΣ ΤΑΜΠΑΚΑΣ

ΠΑΤΡΑ 2019

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή Πάτρα, Ημερομηνία :

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις οποίες πήγες από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Περίληψη

Ο όγκος των δεδομένων που χρησιμοποιείται για αποθήκευση και ανάλυση είναι πλέον τεράστιος στην εποχή μας. Όσο περισσότερα δεδομένα όμως είναι σε διαθεσιμότητα, τόσο και ποιοτικότερος είναι ο έλεγχος για συγκεκριμένα ερωτήματα. Η τάση για επεξεργασία και ανάλυση αυτού του μεγάλου όγκου δεδομένων, οδήγησε στη δημιουργία της παρούσας εργασίας η οποία θα στοχεύει στην υλοποίηση ενός συστήματος που ανταπεξέρχεται καλά σε προβλήματα όγκου δεδομένων και διαχείρισης, με σκοπό αναλύσεις συναισθήματος σε μεγάλα σύνολα ακατέργαστων δεδομένων.

Summary

The amount of data used for storage and analysis is now enormous in our time. But the more data available, the better the control over specific queries. The tendency to process and analyze this large amount of data has led to the creation of the present work, which aims to implement a system that resolves data volumes and management problems well, for the purpose of emotion analysis in large sets of raw data.

Πρόταση Υλοποίησης

Συγκεκριμένα, στα πλαίσια του πρακτικού μέρους της εργασίας, θα υλοποιηθεί ένα κατανεμημένο σύστημα σε περιβάλλον υπολογιστικού νέφους για την υποστήριξη μιας εφαρμογής η οποία θα διαχειρίζεται και θα αναλύει μεγάλο όγκο δεδομένων. Θα δοθεί έμφαση στη δημιουργία και τον ευέλικτο συγχρονισμό των τεχνολογιών μεταξύ τους με σκοπό τη γρήγορη εξυπηρέτηση πολλαπλών αιτημάτων συναισθηματικής ανάλυσης για μεγάλα σύνολα δεδομένων.

Για τον παραλληλισμό των εργασιών του συστήματος θα χρησιμοποιηθεί ένα κατανεμημένο λογισμικό, το Apache Hadoop, το οποίο χρησιμοποιεί το προγραμματιστικό μοντέλο MapReduce. Τα σύνολα δεδομένων για ανάλυση, προτείνουμε να περιέχουν ακατέργαστο κείμενο, με κριτικές χρηστών τα οποία θα βρίσκονται σε μορφή JSON.

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Πινάκων

Κεφάλαιο 1 – Εισαγωγή

1.1 Σημασία του Προβλήματος

Κατά κοινή ομολογία, ο μεγάλος όγκος δεδομένων μας δίνει γνώσεις και ευκαιρίες σε πολλούς κλάδους, από την υγειονομική περίθαλψη μέχρι τα οικονομικά και την πληροφορική. Η μεγάλη εξέλιξη της πληροφορίας οδήγησε στην άνθιση των κοινωνικών μέσων δικτύωσης. Οι εταιρίες στοχεύοντας στην ανάπτυξή τους, θέλησαν να εκμεταλλευτούν την μεγάλη πρόοδο τον κοινωνικών μέσων δικτύωσης χρησιμοποιώντας διαφημίσεις σε αυτά και συλλέγοντας πληροφορίες για τις προτιμήσεις του κόσμου από προσωπικά tweets (σκέψεις του χρήστη σε κείμενο στο Twitter) ή από αξιολογήσεις των χρηστών σε διάφορες δημοσκοπήσεις. Το κύμα ενδιαφέροντος των εταιριών αυξάνεται όσο αυξάνεται η εξόρυξη γνώσης των κοινωνικών δικτύων και γενικότερα του διαδικτύου. Θα μπορούσε να πει κανείς πως οι εταιρίες, πια, βασίζονται από αυτή τη γνώση και έχουν την ανάγκη να γνωρίζουν την ανατροφοδότηση του διαδικτυακού κόσμου.

Από την άλλη μεριά, ένας απλός χρήστης θα μπορούσε με τη σειρά του, να εκμεταλλευτεί τη γνώση του διαδικτύου για το δικό του συμφέρον, ευκολία και ικανοποίηση. Πιο συγκεκριμένα, οι διαφημίσεις που θα ήθελε να δει, θα ήταν κάποιες που θα είχαν άμεση σχέση με αυτόν τη συγκεκριμένη χρονική περίοδο. Παράλληλα, οι αξιολογήσεις των άλλων χρηστών για προϊόντα και υπηρεσίες, θα του έλυναν τα χέρια, δίνοντάς του την κατάλληλη γνώση για αυτά με σκοπό την ακριβέστερη περιγραφή του. Ας σημειωθεί, ότι δίνοντας ο ίδιος μια ανατροφοδότηση για ένα προϊόν ή υπηρεσία, μπορεί να οδηγήσει αισίως σε μια διόρθωση της εταιρίας για αυτό και τελικώς να βελτιωθεί το προϊόν ή η υπηρεσία προς το συμφέρον όλων.

Γίνεται, επομένως, εύκολα αντιληπτό πως διαθέτοντας γνώση μεγάλου όγκου δεδομένων, από τη μία, οι εταιρίες μπορούν να αναλύσουν με κάθε τρόπο ποια είναι τα συμφέροντά τους και να βρουν αποδοτικότερο τρόπο προώθησης αλλά και βελτίωσης των υπηρεσιών και των προϊόντων τους από τις αξιολογήσεις των χρηστών, ενώ από την άλλη, οι χρήστες καλύπτουν ανάγκες και απαιτήσεις που έχουν για το προϊόν/υπηρεσία δίνοντας κριτικές και συμβουλές για διορθώσεις αυτού.

1.2 Στόχοι της Εργασίας

Σε αυτήν την εργασία θα υλοποιηθεί μια φιλική προς το χρήστη, διαδικτυακή εφαρμογή στην οποία ο χρήστης θα μπορεί να μεταφορτώνει ένα σύνολο δεδομένων από κριτικές χρηστών σε κινηματογραφικές ταινίες και στη συνέχεια η εφαρμογή θα αναλύει τα συναισθήματα των χρηστών από τις κριτικές τους. Ακριβέστερα, η εφαρμογή θα πρέπει να διαβάζει, μέσα από ένα σύνολο δεδομένων, κάθε κριτική από κάθε χρήστη και με συγκεκριμένο τρόπο θα αποφασίζει σε ποια από τις τρεις υπάρχουσες κατηγορίες ανήκει το συναίσθημα που εκφράζεται: θετικό, αρνητικό ή ουδέτερο συναίσθημα.

Το σύνολο δεδομένων που θα μεταφορτώνει ο χρήστης θα πρέπει να είναι σε μορφή JSON. Το κύριο πλεονέκτημα της εφαρμογής είναι πως το σύνολο δεδομένων δεν έχει όριο μεγέθους. Θα μπορούσε να είναι από λίγα KiloBytes έως και της τάξης των GigaBytes. Για να επιτευχθεί αυτό, η βάση δεδομένων που θα επιλεχτεί θα πρέπει να είναι ικανή να διαχειρίζεται τέτοια μεγάλα μεγέθη και ταυτόχρονα να το κάνει γρήγορα. Η MongoDB είναι μία από αυτές διότι αν και δεν παρέχει ίσως μεγαλύτερη ασφάλεια από μια συνηθισμένη σχεσιακή βάση, είναι κατάλληλη για αποθήκευση μεγάλου όγκου δεδομένων, ενώ κερδίζει σε επεκτασιμότητα και απόδοση.

Το λογισμικό για τη συναισθηματική ανάλυση του συνόλου των δεδομένων θα πρέπει να συνεργάζεται άψογα με τη βάση δεδομένων. Θα πρέπει μετά από εντολή της εφαρμογής και αφού έχουν ανέβει τα δεδομένα, να τα διαβάζει από τη βάση δεδομένων και να αποφασίζει την κατηγορία συναισθήματός τους δημιουργώντας ένα νέο αρχείο στη βάση για τα αποτελέσματα. Το Apache Hadoop ανταποκρίνεται άψογα με μη σχεσιακές βάσεις δεδομένων διότι δεν είναι πολύπλοκες και επί προσθέτως τα δεδομένα βρίσκονται σε ακατέργαστη μορφή και δομή.

Τέλος, ο server της εφαρμογής θα πρέπει και αυτός με τη σειρά του να συνεργαστεί γρήγορα και αποδοτικά, πρώτον, με τη βάση δεδομένων, και δεύτερον με το Apache Hadoop. Με τη βάση δεδομένων, στην οποία θα ανεβάσει μέσω του προγράμματος περιήγησης ιστού το σύνολο δεδομένων του χρήστη, ενώ, αφού έχει ολοκληρωθεί η διαδικασία της μεταφόρτωσης στη βάση, θα πρέπει χωρίς μεγάλη απώλεια χρόνου να δώσει σήμα στο Hadoop να ξεκινήσει την ανάλυση συναισθήματος. Κατά τη διάρκεια της διαδικασίας αυτής, ο server θα πρέπει να ελέγχει δυναμικά το Hadoop έως ότου να τελειώσει η διεργασία αυτή.

Κατά συνέπεια, αφού επιτευχθεί η μεταφόρτωση των δεδομένων στη βάση αισίως και πραγματοποιηθεί η συναισθηματική ανάλυση, θα πρέπει η εφαρμογή να παρουσιάσει τα αποτελέσματα της στο χρήστη.

Έμφαση και μεγαλύτερη προσοχή στην εργασία δίνεται στη δημιουργία συνδέσεων και συγχρονισμού μεταξύ των τεχνολογιών ώστε οι διαδικασίες αλληλεπίδρασης χρήστη-εφαρμογής να κυλήσουν ομαλά, χωρίς δημιουργία άσκοπων χρονοτριβών. Απώτερος στόχος είναι η υλοποίηση ενός συστήματος, σε ένα περιβάλλον υπολογιστικού νέφους, όπου διαχειρίζεται μεγάλο όγκο δεδομένων από πολλαπλά αιτήματα χρηστών με σκοπό μια ανασκόπηση συναισθηματικής ανάλυσης.

1.3 Συνεισφορά

Στην παρακάτω εργασία παρουσιάζεται, πέραν των τεχνολογιών που καλύπτουν μια συνοπτική ανασκόπησή τους, τα τεχνικά βήματα της δημιουργίας ενός περιβάλλοντος υπολογιστικού νέφους για τη χρήση μιας διαδικτυακής εφαρμογής.

Έγινε έρευνα στις βασικές τεχνολογίες που θα μπορούσαν να αποτελέσουν μέρος ενός υπολογιστικού νέφους και παρέχονται λεπτομέρειες της αρχιτεκτονικής αυτών, ορισμένα ιστορικά γεγονότα αλλά και οι λειτουργίες τους. Παράλληλα παρουσιάζονται τρόποι εγκατάστασης και σύνδεσης μεταξύ των τεχνολογιών υπολογιστικού νέφους και των τεχνολογιών διαδικτύου για λειτουργικά συστήματα τύπου Unix, Ubuntu. Επίσης παράγεται για πρώτη φορά μια διαδικτυακή εφαρμογή που σκοπό έχει την εξυπηρέτηση χρηστών οι οποίοι επιθυμούν την άμεση και γρήγορη ανάλυση συναισθήματος μεγάλου όγκου συνόλων δεδομένων για κριτικές κινηματογραφικών ταινιών.

Κεφάλαιο 2 - Hadoop

2.1 Υπολογιστικό Νέφος

Το cloud computing, ή υπολογιστικό νέφος στα ελληνικά, είναι ένα μοντέλο που εξυπηρετεί την εύκολη πρόσβαση σε κοινόχρηστους και προσαρμόσιμους υπολογιστικούς πόρους (δίκτυα, διακομιστές, εφαρμογές) που μπορούν εύκολα να παραμετροποιηθούν είτε από το χρήστη είτε από τον εκάστοτε πάροχο [1]. Ο όρος συνήθως χρησιμοποιείται για την αναφορά αποθήκευσης δεδομένων σε εξωτερικό σύστημα, δηλαδή αποθήκευση σε μη τοπικό δίκτυο (π.χ. Dropbox). Λειτουργεί με δομές αντίστοιχες του Διαδικτύου, καθώς οι πόροι είναι προσπελάσιμοι σε αντίθεση με την αποθήκευση σε τοπικούς υπολογιστές. Σύμφωνα με την IBM, το νέφος είναι η κατά παραγγελία παροχή υπολογιστικών πόρων από τις εφαρμογές μέχρι τα κέντρα δεδομένων, μέσω του διαδικτύου, επί πληρωμή [2].

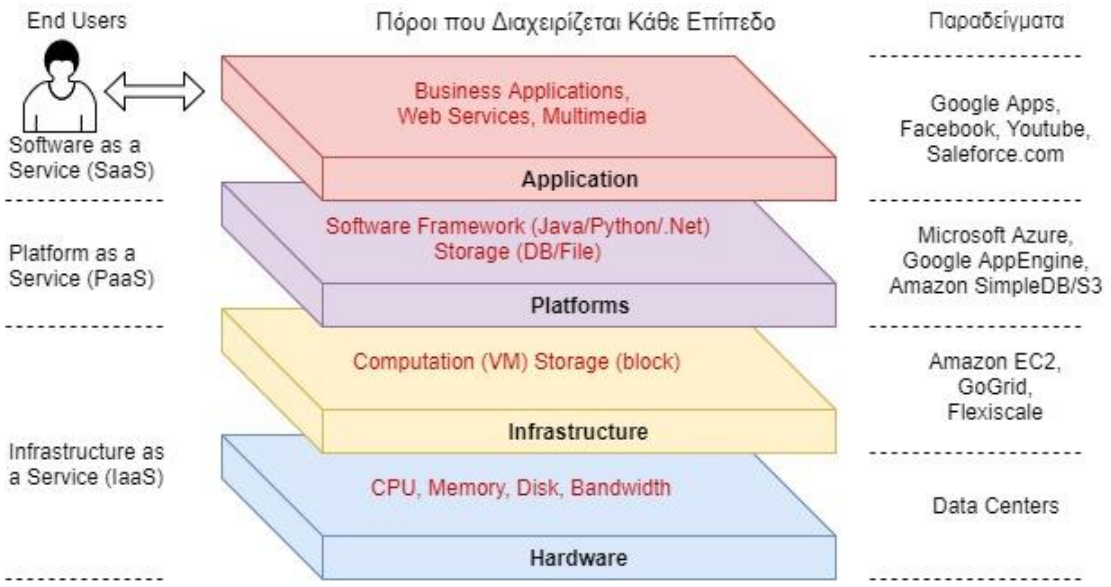
Ένα από τα βασικότερα χαρακτηριστικά του υπολογιστικού νέφους είναι το on-demand self-service, στο οποίο ο χρήστης αποκτά πρόσβαση σε διάφορους υπολογιστικούς πόρους, όπως αποθηκευτικό χώρο ή υπολογιστική ισχύ, όποτε το χρειάζεται και χωρίς την ανάγκη παρέμβασης του παρόχου της υπηρεσίας.

Οι τεχνολογίες και αρχιτεκτονικές που χρησιμοποιούνται από τους παρόχους υπηρεσιών cloud, βασίζονται στη χρήση πολλαπλών συστοιχιών πόρων. Αυτοί, τις περισσότερες φορές, διαμοιράζονται εικονικά και δυναμικά, κάτι που επιτρέπει την εκχώρηση και άλλων πόρων όταν είναι αυτό απαραίτητο. Αυτό γίνεται με αυτόματο τρόπο και μεγάλη ταχύτητα χωρίς να υπάρχουν προβλήματα για τον χρήστη.

Ακόμα ένα χαρακτηριστικό του υπολογιστικού νέφους είναι οι μετρήσεις υπηρεσιών ή Measured Service, όπου δίνεται η δυνατότητα στον πελάτη να πληρώνει ανάλογα με τη χρήση που κάνει, ενώ ο πάροχος μπορεί να δει αν υπάρχουν διαθέσιμοι πόροι σε περίπτωση που χρειαστούν για εκχώρηση σε άλλο χρήστη.

2.1.1 Αρχιτεκτονική και Είδη Υπηρεσιών του Υπολογιστικού Νέφους

Η αρχιτεκτονική ενός περιβάλλοντος υπολογιστικού νέφους αποτελείται από τέσσερα επίπεδα όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 1 - Τα μοντέλα υπηρεσιών του Cloud Computing

Στο πρώτο και πιο χαμηλό επίπεδο έχουμε το Hardware (Υλικό επίπεδο), που αναφέρεται τους φυσικούς πόρους που απαιτεί το cloud (servers, routers, CPU) [3]. Το αμέσως πιο υψηλό επίπεδο, Infrastructure (Υποδομή), είναι γνωστό και ως επίπεδο virtualization. Αυτό το επίπεδο δημιουργεί αποθηκευτικούς χώρους και υπολογιστικούς πόρους χρησιμοποιώντας εικονικές τεχνολογίες (π.χ. VMware) με τη μέθοδο κατακερματισμού (hashing) και με κατανομή των φυσικών πόρων. Στο επίπεδο platform, στόχος είναι να ελαχιστοποιηθεί η επιβάρυνση μέσα από την εγκατάσταση των λειτουργικών συστημάτων και των πλαισίων εφαρμογών απευθείας σε VM containers. Τέλος, στο υψηλότερο επίπεδο της βαθμίδας βρίσκονται οι πραγματικές cloud εφαρμογές.

Όσον αφορά τα είδη των υπηρεσιών υπολογιστικού νέφους, αυτά είναι τρία:

- SaaS (Software as a Service): Σε αυτήν την υπηρεσία, το λογισμικό βρίσκεται σε ένα δίκτυο από διακομιστές και διατίθεται ως υπηρεσία από τον πάροχο μέσω του διαδικτύου στους πελάτες. Οι πελάτες αντί για την αγορά ή την άδεια χρήσης του λογισμικού πληρώνουν ένα ποσό που ορίζεται από τον πάροχο. Αυτό έχει ως αποτέλεσμα να μειώνεται το κόστος για την εκάστοτε επιχείρηση καθώς αποφεύγει να δαπανήσει μεγάλα ποσά με σκοπό να αγοράσει άδειες για εφαρμογές που στο μέλλον μπορεί να μην την ικανοποιούν και να θελήσει να τις αντικαταστήσει με άλλες. Τέλος, ο πάροχος είναι αυτός που αναλαμβάνει την συντήρηση και την αναβάθμιση της εφαρμογής με τον τελικό χρήστη να έχει μόνο τη χρήση της.

- PaaS (Platform as a Service): Πρόκειται για ένα περιβάλλον ανάπτυξης, εγκατάστασης και διανομής λογισμικού που παρέχεται ως υπηρεσία σε προγραμματιστές μέσω του διαδικτύου. Έτσι οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές διαδικτύου χωρίς να χρειάζεται να εγκαταστήσουν κανένα εργαλείο τοπικά στον υπολογιστή τους. Αυτό μειώνει το κόστος και την προσπάθεια που θα γινόταν διαφορετικά για τη διαχείριση της υποδομής [4].

- IaaS (Infrastructure as a Service): Στη συγκεκριμένη υπηρεσία η εταιρία ή ο ιδιώτης έχει τη δυνατότητα να νοικιάσει την υποδομή (υπολογιστική και δικτυακή) από τον πάροχο, ανάλογα τις ανάγκες που έχει τη συγκεκριμένη στιγμή. Έτσι, αντί να αγοράσει τις υποδομές ή να συνάψει κάποιο συμβόλαιο για αυτές, πληρώνει για το χρόνο που τις χρησιμοποιεί.

2.1.2 Μοντέλα Υπολογιστικού Νέφους

Ανάλογα με τον τρόπο που χρησιμοποιείται το υπολογιστικό νέφος, ορίζεται και διαφορετικό μοντέλο.

- Public Cloud: Σε αυτό το μοντέλο, οι υπολογιστικοί πόροι προσφέρονται από τον πάροχο και γίνονται διαθέσιμοι μέσω διαδικτύου στο χρήστη. Ο χρήστης συνήθως πληρώνει ανάλογα με τη χρήση. Το πλεονέκτημα εδώ είναι πως η εξυπηρέτηση είναι άμεση και προσαρμόζεται εύκολα ανάλογα με τις ανάγκες του πελάτη. Ο πάροχος είναι αυτός που αναλαμβάνει τη συντήρηση και την αναβάθμιση.
- Private Cloud: Στο μοντέλο του «ιδιωτικού νέφους» η παροχή του cloud είναι για την αποκλειστική χρήση ενός πελάτη. Μερικές φορές αποτελεί και ιδιοκτησία του πελάτη. Ο πάροχος όμως είναι αυτός που αναλαμβάνει την εγκατάσταση, τη λειτουργία, τη συντήρηση και όποια αναβάθμιση. Οι φυσικοί πόροι είναι δυνατόν να βρίσκονται είτε στις εγκαταστάσεις του πελάτη είτε στου παρόχου. Σε αυτό το μοντέλο υπάρχει και η έννοια του εικονικού ιδιωτικού νέφους όπου δεσμεύονται φυσικές υποδομές από ένα δημόσιο υπολογιστικό νέφος αποκλειστικά για ένα πελάτη [5].
- Community Cloud: Στο συγκεκριμένο μοντέλο έχουμε τους πελάτες που έχουν κάποιες κοινές ανάγκες και αποφασίζουν για διάφορους λόγους να μοιραστούν τις φυσικές υποδομές. Παράλληλα, αναλαμβάνουν και την πλήρη διαχείριση και

παραμετροποίηση είτε από μέλη εντός της κοινότητας είτε από εξουσιοδοτημένα άτομα εκτός κοινότητας.

- Hybrid Cloud: Το μοντέλο αυτό αναφέρεται στον οποιοδήποτε συνδυασμό των παραπάνω μοντέλων υπολογιστικού νέφους.

2.2 Εισαγωγή στο Hadoop

Προκειμένου να διασφαλιστεί υψηλή αξιοπιστία και οικονομία, η τεχνολογία του υπολογιστικού νέφους υιοθετεί κατανεμημένα αποθηκευτικά συστήματα για την αποθήκευση δεδομένων ώστε να εξασφαλίσει την ακεραιότητά τους, παρέχοντας έτσι φθηνή και αξιόπιστη μάζα αποθήκευσης ενός κατανεμημένου και υπολογιστικού συστήματος. Το Apache Hadoop είναι ένα λογισμικό ανοιχτού κώδικα που προσφέρει τεράστια δυνατότητα αποθήκευσης δεδομένων και κατανεμημένη επεξεργασία τεράστιων ποσοτήτων δεδομένων. Το Hadoop καταστεί ιδανικό αποθηκευτικό σύστημα για περιβάλλοντα υπολογιστικού νέφους, ενώ παρέχει εργαλεία που απαιτούνται για την ανάπτυξη και εκτέλεση εφαρμογών λογισμικού. Δημιουργήθηκε για να διαχειριστεί μεγάλες ποσότητες δεδομένων τα οποία, λόγω του μεγάλου όγκου τους, δεν είναι δυνατόν να τοποθετηθούν στον δίσκο ενός υπολογιστή και έτσι τόσο τα δεδομένα όσο και η ανάλυσή τους χρειάζεται να κατανεμηθούν σε μεγάλες συστάδες υπολογιστών.

Η χρήση του Hadoop περιλαμβάνει τα ακόλουθα στάδια:

1. εγκατάσταση του Hadoop σε ένα μεγάλο αριθμό υπολογιστών και χρήση των δίσκων τους για αποθήκευση δεδομένων
2. χρησιμοποίηση των CPU των υπολογιστών για την επεξεργασία των δεδομένων

Το Hadoop είναι γραμμένο σε γλώσσα προγραμματισμού Java και επιτρέπει στους προγραμματιστές να αναπτύξουν προσαρμοσμένα γραπτά προγράμματα σε κώδικα Java ή ακόμα και σε οποιαδήποτε διαφορετική γλώσσα για να επεξεργάζονται δεδομένα παράλληλα σε εκατοντάδες ή χιλιάδες βασικούς διακομιστές (servers).

Τα δεδομένα στο Hadoop χωρίζονται σε μπλοκ και αποθηκεύονται σε διάφορους συνδεδεμένους κόμβους που λειτουργούν μαζί. Κόμβοι μπορεί να είναι ένα σύμπλεγμα από υπολογιστές το οποίο συνήθως ονομάζεται και ως συστάδα (cluster). Μία συστάδα Hadoop μπορεί να εκτείνεται σε χιλιάδες κόμβους. Οι υπολογισμοί εκτελούνται παράλληλα σε όλη τη συστάδα, πράγμα που σημαίνει ότι η εργασία χωρίζεται μεταξύ των κόμβων της συστάδας.

Η πολυπλοκότητα της διαδικασίας και ο όγκος του χρόνου αντίδρασης των δεδομένων μπορεί να κυμαίνεται από λεπτά έως ώρες. Το Hadoop χρησιμοποιεί μια συστάδα από κόμβους για την εκτέλεση των εργασιών MapReduce παράλληλα. Η εργασία MapReduce αποτελείται από δύο βήματα. Αρχικά το βήμα Map επεξεργάζεται δεδομένα εισόδου και στη συνέχεια το βήμα Reduce συγκεντρώνει τα ενδιάμεσα αποτελέσματα σε ένα τελικό αποτέλεσμα. Κάθε κόμβος συστάδας διαθέτει έναν τοπικό επεξεργαστή (CPU) και ένα τοπικό σύστημα αρχείων, στο οποίο θα εκτελεστούν τα προγράμματα MapReduce. Τα δεδομένα αναπαράγονται σε πανομοιότυπα αντίγραφα με σκοπό την ασφαλή ακεραιότητα και αξιοπιστία τους, ενώ χωρίζονται σε μπλοκ δεδομένων και αποθηκεύονται τελικά σε τοπικά αρχεία διαφόρων κόμβων. Τα τοπικά αρχεία αποτελούν το σύστημα αρχείων του συστήματος που ονομάζεται Hadoop Distributed File System (HDFS).

2.3 Ιστορία του Hadoop

Η γένεση του Hadoop ήρθε από μια δημοσίευση τον Οκτώβριο του 2003 που είχε να κάνει με το σύστημα αρχείων της Google, με τίτλο "Google File System" [6]. Η Google είχε δημοσιεύσει εκείνη την εποχή, την αρχιτεκτονική του Google Distributed File System (GFS), η οποία επέλυε αποθηκευτικές απαιτήσεις για πολύ μεγάλο όγκο αρχείων που δημιουργούνταν από λειτουργίες όπως οι ανιχνευτές ιστού (web crawlers), εξόρυξης δεδομένων και άλλων τεχνικών. Με βάση το έργο της Google και για λόγους εξόρυξης δεδομένων και ανακαλύψεις συστημάτων, ξεκίνησε η ανάπτυξή του Apache Nutch project (Open Source Search Engine), το οποίο μεταφέρθηκε στο υποπρόγραμμα Hadoop τον Ιανουάριο του 2006. Ο Doug Cutting, ο οποίος εργαζόταν στην Yahoo την εποχή εκείνη, το ονόμασε έτσι, λόγω του ονόματος που είχε δώσει ο γιος του σε ένα κίτρινο ελεφαντάκι. Ο αρχικός κώδικας που προήλθε από το Apache Nutch αποτελούταν από περίπου 5.000 γραμμές κώδικα για το HDFS και περίπου 6.000 γραμμές κώδικα για το MapReduce.

Το 2004 με βάση την αρχιτεκτονική του GFS, η Nutch υλοποίησε ένα ανοιχτού κώδικα σύστημα αρχείων που ονομάζεται Nutch Distributed File System (NDFS). Το 2004 δημοσιεύτηκε από την Google, το MapReduce, ενώ το 2005 οι προγραμματιστές της Nutch εργάστηκαν για το MapReduce της Nutch Project. Η πλειοψηφία των αλγορίθμων άλλαξαν ώστε να τρέχει χρησιμοποιώντας το MapReduce και το NDFS. Το Φεβρουάριο του 2006 παρουσίασαν τη Nutch για να σχηματίσουν ένα ανεξάρτητο έργο του Apache Lucene, που ονομάζεται Hadoop. Ταυτόχρονα, ο Doug Cutting έγινε μέλος της Yahoo, η οποία παρείχε μια ειδική ομάδα και τους πόρους για να μετατρέψει το Hadoop σε ένα σύστημα το οποίο λειτουργεί σε κλίμακα ιστού. Αυτό παρουσιάστηκε τον Φεβρουάριο του 2008, όταν η Yahoo δήλωσε ότι ο δείκτης αναζήτησης της παραγωγής της παράγεται από μία συστάδα Hadoop με 10.000 πυρήνες.

Τον Ιανουάριο του 2008, η Apache δημιούργησε το δικό της υψηλού επιπέδου Hadoop, επιβεβαιώνοντας την επιτυχία του και την ποικιλόμορφη, αποτελεσματική του κοινότητα. Επιπλέον, το Hadoop χρησιμοποιήθηκε από πολλές άλλες εταιρείες εκτός από τη Yahoo, όπως το Facebook, τη New York Times ενώ χρησιμοποιήθηκε στις υπηρεσίες Amazon Elastic Compute Cloud (EC2) και Amazon Simple Storage Service (S3). Τον Απρίλιο του 2008, το Hadoop έσπασε ένα παγκόσμιο ρεκόρ για να γίνει το σύστημα το οποίο ταξινομεί ένα TeraByte δεδομένων και τρέχει μια συστάδα 910 κόμβων, με ταξινόμηση ενός TeraByte σε μόλις 209 δευτερόλεπτα, κερδίζοντας το νικητή του προηγούμενου έτους που είχε κάνει το ίδιο σε 297 δευτερόλεπτα.

Σήμερα το Hadoop είναι μια συλλογή σχετικών υποέργων, που αφορούν στην υποδομή κατανεμημένων συστημάτων πληροφορικής. Αυτά τα έργα φιλοξενούνται από την Apache Software Foundation, η οποία παρέχει υποστήριξη για έργα ανοιχτού κώδικα.

2.4 Τα Δομικά Στοιχεία του Hadoop

Σε φάση λειτουργίας, το Hadoop, τρέχει ένα σύνολο από διεργασίες, γνωστά και ως Hadoop daemons, στους πολυάριθμους διακομιστές του δικτύου του. Αυτές οι διεργασίες έχουν σημαντικούς ρόλους και είναι απαραίτητες για να λειτουργήσει το Hadoop. Μερικές υπάρχουν μόνο σε ένα διακομιστή, ενώ άλλες υπάρχουν σε πολυάριθμους διακομιστές.

Στη συνέχεια αναφέρονται και αναλύονται η κάθε μία από αυτές και ο ρόλος τους μέσα στο Hadoop.

2.4.1 NameNode

Η διεργασία NameNode είναι το κεντρικό κομμάτι του συστήματος αρχείων HDFS και ο κύριος (master) συντελεστής του HDFS, ο οποίος κατευθύνει τις διεργασίες-σκλάβου, DataNodes, οι οποίες εκτελούν διεργασίες I/O χαμηλού επιπέδου. Το NameNode είναι ο «λογιστής» του HDFS. Είναι υπεύθυνος στο να αποσπαστούν τα δεδομένα σε μπλοκ αρχείων αποδοτικά στους κόμβους του δικτύου, και είναι επίσης υπεύθυνος για τη συλλογική υγεία του κατανεμημένου συστήματος αρχείων. Ο διακομιστής που φιλοξενεί το NameNode δε, συλλέγει κάποια αποτελέσματα ή υπολογισμούς από το πρόγραμμα MapReduce για να μειώσει το φόρτο εργασίας του μηχανήματος. Διατηρεί το δέντρο καταλόγου (tree directory)

όλων των αρχείων στο σύστημα αρχείων και παρακολουθεί το που διατηρούνται τα δεδομένα αρχείων σε όλη τη συστάδα. Δεν αποθηκεύει τα δεδομένα αυτών των αρχείων.

Οι εφαρμογές πελάτη (Client application) «μιλάνε» στο NameNode όποτε επιθυμούν να εντοπίσουν ένα αρχείο ή όταν θέλουν να προσθέσουν, να αντιγράψουν, να μετακινήσουν ή να διαγράψουν ένα αρχείο. Το NameNode ανταποκρίνεται στα επιτυχημένα αιτήματα επιστρέφοντας μια λίστα σχετικών διακομιστών DataNode όπου βρίσκονται τα ζητούμενα δεδομένα.

Το NameNode είναι το σημείο αποτυχίας(point of failure) για το σύστημα αρχείων HDFS. Όταν το NameNode πέσει, το σύστημα αρχείων τερματίζεται. Για αυτές τις περιπτώσεις όπου το NameNode αποτύχει ακαριαία να λειτουργήσει, υπάρχει μια προαιρετική διεργασία, που ονομάζεται SecondaryNameNode, η οποία μπορεί να φιλοξενηθεί σε ένα ξεχωριστό μηχάνημα.

2.4.2 DataNode

Ένα DataNode αποθηκεύει τα δεδομένα στο σύστημα αρχείων του Hadoop. Ένα λειτουργικό σύστημα αρχείων έχει περισσότερους από έναν DataNode, με τα αντίγραφα δεδομένων μοιρασμένα σε αυτούς.

Κατά την εκκίνηση, ένα DataNode προσπαθεί να συνδεθεί με το NameNode, μέχρι να εμφανιστεί η υπηρεσία. Στη συνέχεια, απαντά σε αιτήματα από το NameNode για λειτουργίες συστήματος αρχείων. Όταν θελήσεις να διαβάσεις ή να γράψεις ένα HDFS αρχείο, το αρχείο είναι χωρισμένο σε μπλοκς και ο NameNode θα πει στον πελάτη σε ποιο DataNode βρίσκεται το κάθε μπλοκ.

Οι εφαρμογές πελάτη μπορούν να «μιλήσουν» απευθείας με ένα DataNode, από τη στιγμή εκείνη που το NameNode έχει δώσει τη θέση των δεδομένων. Ομοίως, οι λειτουργίες του MapReduce αναθέτουν στους TaskTracker να δώσουν τις τοποθεσίες των αρχείων ώστε το DataNode να έχει πρόσβαση σε αυτά απευθείας. Οι TaskTracker αναπτύσσονται στους ίδιους διακομιστές που φιλοξενούν παρουσίες DataNode, έτσι ώστε οι λειτουργίες MapReduce να εκτελούνται κοντά στα δεδομένα.

Τέλος, οι DataNode μπορούν να μιλούν μεταξύ τους, κάτι το οποίο κάνουν τη στιγμή που αντιγράφουν τα δεδομένα.

2.4.3 SecondaryNameNode

Οι περισσότεροι γενικά πιστεύουν ότι ο SecondaryNameNode αντικαθιστά το Namenode όταν ο Namenode υποστεί βλάβη, αλλά η αλήθεια είναι ότι δεν είναι έτσι τα πράγματα.

Η λειτουργία του SecondaryNameNode είναι ξεχωριστή και δεν αποτελεί αντικατάσταση του Namenode. Επιπλέον, ο Secondary NameNode (SNN) είναι ένας βοηθητικός δαίμονας/διεργασία για την παρακολούθηση της κατάστασης μιας συστάδας HDFS. Κάθε συστάδα έχει και από έναν SecondaryNameNode και βρίσκεται στο δικό του μηχάνημα. Το SecondaryNameNode διακρίνεται από το NameNode, στο ότι αυτή η διαδικασία δεν λαμβάνει ή δεν καταγράφει καμία μετατροπή πραγματικού χρόνου στο HDFS. Εναλλακτικά, επικοινωνεί με το NameNode για να λαμβάνει στιγμιότυπα των μετα-δεδομένων του HDFS σε διαστήματα που έχουν καθοριστεί από τη διαμόρφωση της συστάδας. Συνεπώς, ένας SecondaryNamenode τοποθετεί ένα σημείο ελέγχου σε ένα σύστημα αρχείων το οποίο είναι ικανό να βοηθήσει το Namenode να λειτουργήσει καλύτερα.

2.4.4 JobTracker

Ο JobTracker είναι η υπηρεσία που είναι υπεύθυνη για τη λήψη αιτημάτων πελάτη, για εργασίες του MapReduce. Τα αιτήματα αυτά, τα κατανέμει στους TaskTrackers στα DataNodes όπου τα απαιτούμενα δεδομένα βρίσκονται εκεί τοπικά και είναι παρόντα. Εάν αυτό δεν είναι δυνατό, ο JobTracker επιχειρεί να κατανείμει τις εργασίες στους TaskTrackers μέσα στο ίδιο πλαίσιο όπου υπάρχουν τα δεδομένα τοπικά. Είτε για κάποιο λόγο αυτό αποτύχει, ο JobTracker αναθέτει ξανά την εργασία σε ένα TaskTracker όπου κατέχει ένα αντίγραφο των δεδομένων. Μόλις υποβληθεί ο κώδικας στη συστάδα, ο JobTracker αποφασίζει το σχέδιο εκτέλεσης καθορίζοντας τα αρχεία που θα επεξεργαστούν και κατανέμοντας τους κόμβους σε ξεχωριστές εργασίες. Στο Hadoop, τα μπλοκ δεδομένων αντιγράφονται σε όλα τα DataNodes ώστε να υπάρξει βεβαιότητα ότι δε θα χαθούν δεδομένα. Επομένως εάν ένας κόμβος στη συστάδα πάθει βλάβη, η εργασία δε θα ακυρωθεί. Υπάρχει μόνο μια διεργασία JobTracker για μία συστάδα Hadoop. Κανονικά, τρέχει σε ένα διακομιστή ως κύριος κόμβος της συστάδας. Τέλος, ο JobTracker είναι το σημείο αποτυχίας (point of failure) της υπηρεσίας MapReduce. Και αυτό γιατί αν ο JobTracker πάθει βλάβη, όλες οι εργασίες του MapReduce σταματούν.

2.4.5 TaskTracker

Ένα TaskTracker είναι ένας κόμβος στη συστάδα που δέχεται εργασίες τύπου Map, Reduce και Shuffle, από ένα JobTracker.

Κάθε TaskTracker έχει διαμορφωθεί με ένα σύνολο υποδοχών, οι οποίες υποδεικνύουν τον αριθμό των εργασιών που μπορεί να δεχτεί. Όταν ο JobTracker προσπαθεί να βρει κάπου να προγραμματίσει μια εργασία μέσα στις λειτουργίες του MapReduce, ψάχνει αρχικά για μια κενή υποδοχή στον ίδιο διακομιστή που φιλοξενεί το DataNode που περιέχει τα δεδομένα και εάν δε βρει, αναζητά μια κενή υποδοχή σε ένα μηχάνημα στο ίδιο πλαίσιο (rack).

Το TaskTracker δημιουργεί ξεχωριστές διαδικασίες JVM για να κάνει την πραγματική δουλειά. Αυτό γίνεται για να διασφαλιστεί ότι η αποτυχία της διαδικασίας δεν δημιουργεί βλάβη στον εντοπιστή εργασιών. Ο TaskTracker παρακολουθεί αυτές τις διεργασίες που δημιουργήθηκαν, καταγράφοντας τους τα αποτελέσματα και τους κώδικες εξόδου. Όταν ολοκληρωθεί η διαδικασία, με επιτυχία ή όχι, ο εντοπιστής εργασιών ειδοποιεί το JobTracker. Οι TaskTrackers στέλνουν επίσης κάποια άλλα μηνύματα στους JobTracker, συνήθως κάθε λίγα λεπτά, ώστε να πληροφορήσουν το JobTracker ότι είναι ακόμα ζωντανοί. Το μήνυμα αυτό ενημερώνει επίσης το JobTracker για τον αριθμό των διαθέσιμων υποδοχέων, έτσι ώστε ο JobTracker να μπορεί να παραμείνει ενημερωμένο με το πού μπορεί να μεταβιβαστεί η εργασία της συστάδας.

2.4.6 Scheduler

Όταν ο χρονοπρογραμματιστής (Scheduler) ξεκινά, αυτό που κάνει είναι να διατηρεί αυτόματα τις εργασίες, χειριζόμενος έναν αλγόριθμο γλωσσών ελέγχου εργασίας ή μέσω επικοινωνίας με το χρήστη. Ένας χρονοπρογραμματιστής στο Hadoop χρειάζεται την κοινή χρήση της συστάδας μεταξύ των διαφόρων εργασιών και των χρηστών για τη χρήση των πόρων αυτής. Εκτός αυτού, χωρίς έναν χρονοπρογραμματιστή, μια εργασία Hadoop μπορεί να καταναλώσει όλους τους πόρους της συστάδας και άλλες εργασίες πρέπει να περιμένουν να τελειώσει η εν λόγω εργασία.

2.5 Hadoop Distributed File System (HDFS)

Κοινώς γνωστό, είναι το γεγονός ότι κάθε φυσικό σύστημα κατέχει το δικό του όριο αποθήκευσης. Όταν πρόκειται να αποθηκεύσουμε τεράστιο όγκο δεδομένων, τότε αναπόφευκτα θα χρειαστούμε περισσότερα από ένα συστήματα. Με αυτό τον τρόπο, τα δεδομένα θα διαχωριστούν μεταξύ τους, μέσω των συστημάτων, αλλά θα είναι ενωμένα μέσω ενός δικτύου. Το Hadoop, ως ένα φυσικό σύστημα, έχει το δικό του κατανεμημένο σύστημα αρχείων το οποίο είναι γνωστό ως HDFS. Είναι ένα κατανεμημένο σύστημα αρχείων που αποθηκεύει δεδομένα σε μηχανές ενός δικτύου. Θα έλεγε κανείς πως το HDFS είναι ο πυρήνας του Hadoop.

Ένα ενιαίο μεγάλο αρχείο απαιτεί πολύ χώρο αποθήκευσης. Το HDFS χωρίζει ένα μεγάλο αρχείο σε μικρά κομμάτια και στη συνέχεια αποθηκεύει αυτά τα κομμάτια δεδομένων σε πολλαπλούς κόμβους, πράγμα που βοηθά στην παροχή παράλληλης πρόσβασης και υπολογισμού. Το HDFS αντιγράφει τα μπλοκ αρχείων (κομμάτια δεδομένων) σε διαφορετικούς κόμβους για να αποτρέψει την απώλεια δεδομένων, διατηρώντας 3 αντίγραφα των μπλοκ αρχείων το ελάχιστο. Αυτός ο τύπος διαχείρισης για την αποθήκευση τεράστιων όγκων δεδομένων είναι γνωστός ως κατανεμημένο σύστημα αρχείων. Για να διαβαστεί ή να γραφτεί ένα αρχείο στο HDFS, πρέπει να καθοριστεί η μορφή αρχείου εισόδου και η μορφή αρχείου εξόδου αντίστοιχα.

Το Hadoop παρέχει επίσης ένα πολύ υψηλό εύρος ζώνης σε όλη τη συστάδα. Είναι σύστημα κατανομής αρχείων βασισμένο σε Java που μπορεί να αποθηκεύσει όλους τους τύπους δεδομένων χωρίς προηγούμενη οργάνωση. Ένα θέμα για το Hadoop ήταν πως σε όλες τις αναβαθμίσεις του λογισμικού του, η πιθανότητα να καταστραφεί το σύστημα αρχείων λόγω σφαλμάτων λογισμικού ή ανθρώπινης ανακρίβειας ήταν μεγάλη και αυξανόταν. Αυτό διορθώθηκε με τη δημιουργία στιγμιότυπων στο HDFS. Έτσι μειώνονται οι πιθανές ζημιές στα δεδομένα που αποθηκεύονται στο σύστημα κατά τις αναβαθμίσεις.

Ο πυρήνας του HDFS περιλαμβάνει δύο κύριους τύπους συστατικών μερών. Είναι δύο υποσυστήματα τα οποία τρέχουν ως ξεχωριστές διεργασίες.

2.5.1 NameNode

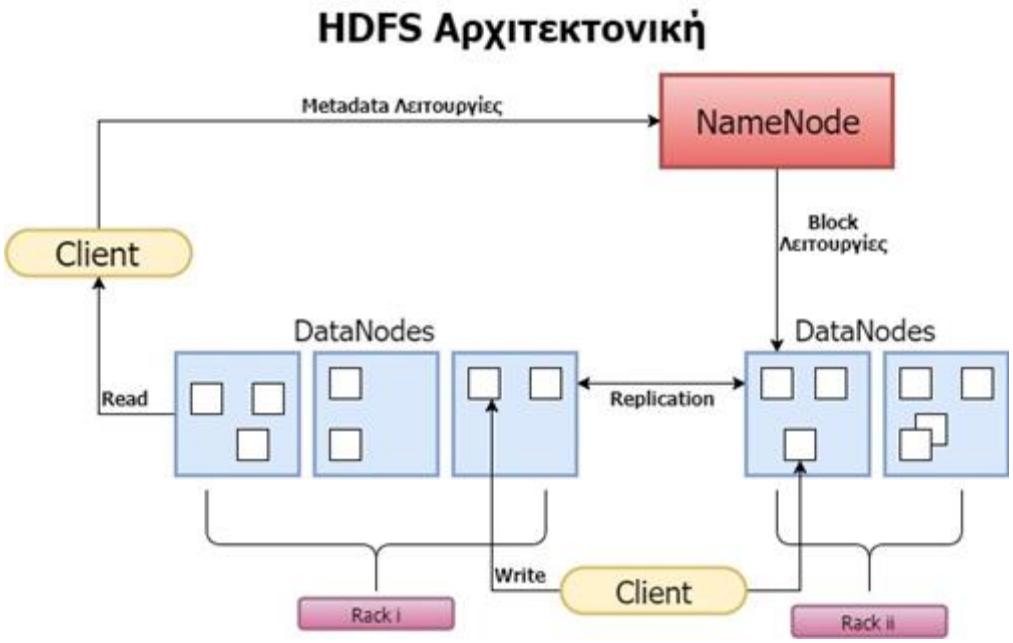
Λειτουργεί στον κύριο κόμβο της συστάδας. Αποθηκεύει τα μετα-δεδομένα για το αρχείο, καταγράφει το όνομά του, την τοποθεσία του στον κατάλογο, συντονίζει τις λειτουργίες και χαρτογράφει τα κομμάτια δεδομένων στον κόμβο δεδομένων. Ο Name Node γνωρίζει πώς

μοιάζει η δομή καταλόγου, πώς διαμορφώνονται τα δικαιώματα πρόσβασης σε κάθε αρχείο και στους καταλόγους, ποιοι χρήστες υπάρχουν και επίσης γνωρίζουν πού φυλάσσεται κάθε μπλοκ κάθε αρχείου. Ολόκληρη αυτή η πληροφορία αναφέρεται ως χώρος ονόματος ή Namespace.

Υπάρχει επίσης ένας δευτερεύων NameNode, ο οποίος λειτουργεί ως αντίγραφο ασφαλείας για τον Name Node.

2.5.2 DataNode

Τα DataNodes βρίσκονται στους δευτερεύον κόμβους του συμπλέγματος και δέχονται εντολές από το Name Node. Τα Data Nodes αποθηκεύουν τα μπλοκ των αρχείων που υπάρχουν στον HDFS. Το τυπικό μέγεθος ενός μπλοκ είναι προκαθορισμένο στα 64 MB αλλά μπορεί να ρυθμιστεί για κάθε αρχείο τη στιγμή που δημιουργείται. Άλλες λειτουργίες των Data Nodes είναι η διαγραφή των μπλοκ αρχείων ή η αντιγραφή αυτών. Τέλος, τα μπλοκ καταλήγουν ως ένα γενικό αρχείο στο τοπικό σύστημα αρχείων σε έναν από τους Data Nodes.



Εικόνα 2 - Αρχιτεκτονική του HDFS

2.6 Yet Another Resource Negotiator (YARN)

Λειτουργεί ως διαχειριστής των πόρων για το Hadoop, προσθέτοντας νήματα μεταξύ του HDFS και των εφαρμογών έτσι ώστε να μπορεί να αλληλεπιδρά με εφαρμογές και να προγραμματίζει πόρους για αυτές.

Η θεμελιώδης ιδέα του YARN είναι να χωρίσει τις δύο μεγάλες ευθύνες του JobTracker, δηλαδή τη διαχείριση των πόρων και τον προγραμματισμό των εργασιών, σε χωριστές διεργασίες-δαίμονες: τον ResourceManager και ανά εφαρμογή τον ApplicationMaster (AM).

Ο ResourceManager και ανά δευτερεύουσα μονάδα, ο NodeManager (NM), αποτελούν το νέο σύστημα διαχείρισης εφαρμογών με κατανεμημένο τρόπο.

Ο ResourceManager είναι ο τελικός παράγοντας που ελέγχει και ρυθμίζει τους πόρους μεταξύ όλων των εφαρμογών του συστήματος. Ο ApplicationMaster είναι μια οντότητα λογισμικού που ελέγχεται και λαμβάνει εντολές από τον ResourceManager, ενώ ταυτόχρονα συνεργάζεται με τους NodeManager με σκοπό την εκτέλεση και την παρακολούθηση των εργασιών.

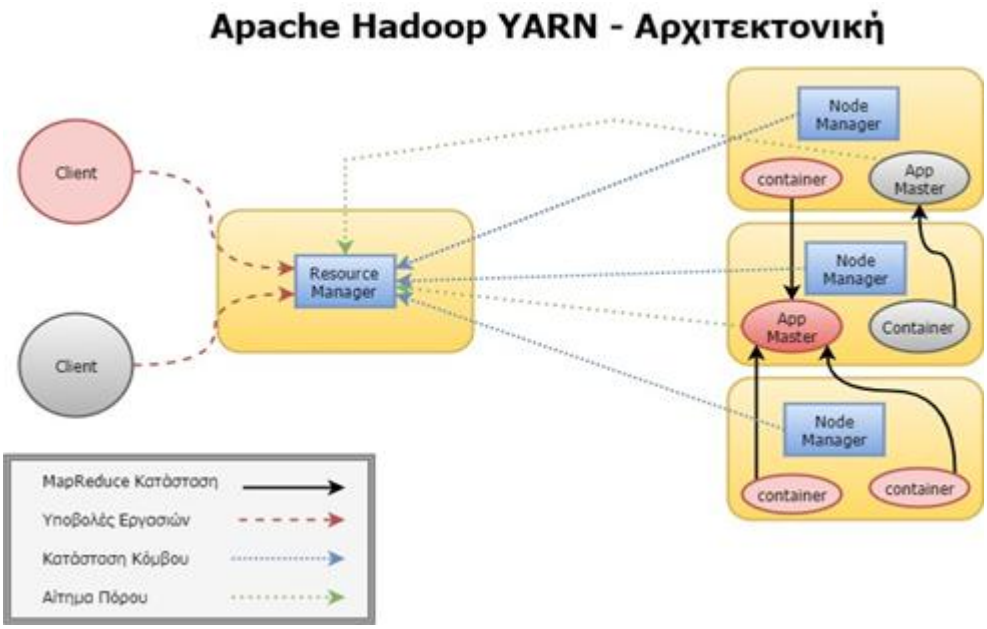
Ο ResourceManager διαθέτει έναν Scheduler (Χρονοπρογραμματιστή), ο οποίος είναι υπεύθυνος για την κατανομή των πόρων στις διάφορες εφαρμογές που εκτελούνται, υπό την προϋπόθεση γνωστών περιορισμών, ουρών κ.λπ. Ο Scheduler είναι ένας απλός χρονοπρογραμματιστής υπό την έννοια ότι δεν εκτελεί καμία παρακολούθηση ή παρακολούθηση της κατάστασης της εφαρμογής. Δεν προσφέρει καμία εγγύηση για την επανεκκίνηση των αποτυχημένων εργασιών είτε λόγω βλάβης εφαρμογής είτε λόγω αποτυχίας υλικού. Ο Scheduler εκτελεί τη λειτουργία προγραμματισμού βάσει των απαιτήσεων των πόρων των εφαρμογών. Αυτό βασίζεται στην αφηρημένη έννοια του Resource Container ο οποίος ενσωματώνει στοιχεία πόρων όπως η μνήμη, μια CPU, ο δίσκος, ένα δίκτυο κλπ.

Ο NodeManager, ένας για κάθε σύστημα, είναι υπεύθυνος για την εκκίνηση των Container των εφαρμογών, την παρακολούθηση της χρήσης πόρων (cpu, μνήμη, δίσκος, δίκτυο) και την αναφορά τους στον ResourceManager. Ένα Container κρατά όλους τους πόρους που έχουν κατανεμηθεί σε έναν κόμβο.

Ο ApplicationMaster, ένας για κάθε εφαρμογή, έχει την ευθύνη της διαπραγμάτευσης των κατάλληλων Container από τον Scheduler, παρακολουθώντας της κατάστασή και την πρόοδό

τους. Από την μεριά του συστήματος, ο ApplicationMaster λειτουργεί ως κανονικό Container.

Ακολουθεί μια αρχιτεκτονική όψη του YARN.



Εικόνα 3 - Αρχιτεκτονική του YARN

2.7 MapReduce

2.7.1 Γενική Ανασκόπηση

Πρόκειται για ένα μοντέλο προγραμματισμού που βοηθά στον παράλληλο υπολογισμό με τρόπο απλό. Επικοινωνεί με το YARN για τον προγραμματισμό των πόρων. Για να γίνει αυτό το μοντέλο προγραμματισμού, ο χρήστης πρέπει να παρέχει δύο λειτουργίες [9]. Απαιτείται μια λειτουργία χαρτογράφησης που ονομάζεται Map(), η οποία θα εφαρμοστεί σε κάθε κόμβο της συστάδας. Η λειτουργία του Map() είναι να διαβάζει τα δεδομένα και να παρέχει το αποτέλεσμα σε μορφή ζεύγους κλειδιών-τιμών (value-key pairs). Η δεύτερη λειτουργία καλείται Reduce() και ο σκοπός της είναι να παίρνει τα ζεύγη κλειδιού-τιμής που παρέχονται από το Map(), να τα συγκεντρώνει και να προσπαθεί να συνοψίσει σε αποτελέσματα με κάποιο συγκεκριμένο τρόπο.

Τα προγράμματα γραμμένα σε αυτό το λειτουργικό στυλ παραλληλίζονται αυτόματα και εκτελούνται σε μία μεγάλη συστάδα μηχανών. Το σύστημα χρόνου εκτέλεσης φροντίζει για τις λεπτομέρειες του διαχωρισμού των δεδομένων εισόδου, τον προγραμματισμό της εκτέλεσης του προγράμματος σε ένα σύνολο μηχανών, τον χειρισμό των βλαβών του

μηχανήματος και τη διαχείριση της απαιτούμενης επικοινωνίας μεταξύ των μηχανών. Αυτό επιτρέπει στους προγραμματιστές χωρίς εμπειρία σε παράλληλα και κατανεμημένα συστήματα να χρησιμοποιούν εύκολα τους πόρους ενός μεγάλου κατανεμημένου συστήματος.

Η επεξεργασία μπορεί να συμβεί σε δεδομένα που είναι αποθηκευμένα είτε σε ένα σύστημα αρχείων (αδόμητο) είτε σε μια βάση δεδομένων (δομημένη). Το MapReduce μπορεί να εκμεταλλευτεί την τοπικότητα των δεδομένων, επεξεργάζοντας αυτά που είναι αποθηκευμένα κοντά μεταξύ τους, προκειμένου να ελαχιστοποιηθεί η γενική επικοινωνία. Ακολουθούν τα τρία βασικά βήματα ενός MapReduce μοντέλου.

Βήμα "Map"

Κάθε ενεργός κόμβος εφαρμόζει τη λειτουργία map() στα τοπικά δεδομένα και γράφει την έξοδο σε μια προσωρινή αποθήκευση. Ένας κεντρικός κόμβος διασφαλίζει ότι γίνεται επεξεργασία μόνο ενός αντιγράφου των δεδομένων εισόδου.

Βήμα "Shuffle"

Κάθε ενεργός κόμβος αναδιανέμει τα δεδομένα με βάση τα κλειδιά εξόδου που παράγονται από τη map() λειτουργία, έτσι ώστε όλα τα δεδομένα που ανήκουν σε ένα κλειδί να βρίσκονται στον ίδιο κόμβο που θα τα επεξεργαστεί.

Βήμα "Reduce"

Στο τελικό βήμα, οι ενεργοί κόμβοι επεξεργάζονται κάθε ομάδα δεδομένων εξόδου, ανά κλειδί, παράλληλα.

Το MapReduce επιτρέπει την κατανεμημένη επεξεργασία του Map και τις λειτουργίες Reduce. Με την προϋπόθεση ότι κάθε λειτουργία χαρτογράφησης (Map) είναι ανεξάρτητη από τις άλλες, όλα τα Map() μπορούν να εκτελούνται παράλληλα, αν και στην πράξη αυτό περιορίζεται από τον αριθμό των ανεξάρτητων πηγών δεδομένων και/ή τον αριθμό CPU κοντά σε κάθε πηγή. Ομοίως, ένα σύνολο από Reduce() μπορεί να εκτελέσει τη φάση Reduce, υπό την προϋπόθεση ότι όλες οι έξοδοι της λειτουργίας χαρτογράφησης που μοιράζονται το ίδιο κλειδί παρουσιάζονται στον ίδιο Reducer ταυτόχρονα ή ότι η λειτουργία Reduce() είναι σχετική. Ενώ αυτή η διαδικασία μπορεί συχνά να φαίνεται ανεπαρκής σε σύγκριση με τους αλγόριθμους που είναι πιο διαδοχικοί, επειδή πρέπει να εκτελούνται περισσότερες από μία Reduce(), το MapReduce μπορεί να εφαρμοστεί σε σημαντικά μεγαλύτερα σύνολα δεδομένων από αυτά που μπορούν να χειριστούν κάποιοι απλοί διακομιστές. Ένα μεγάλο οικοσύστημα διακομιστών μπορεί να χρησιμοποιήσει το MapReduce για να ταξινομήσει ένα petabyte δεδομένων σε λίγες μόνο ώρες. Ο παραλληλισμός προσφέρει επίσης κάποια δυνατότητα αποκατάστασης από μερική αποτυχία διακομιστών είτε και αποθήκευσης κατά τη διάρκεια μιας λειτουργίας όπως: εάν αποτύχει ένας Map() ή ένας Reduce(), η εργασία μπορεί να επανα-προγραμματιστεί υποθέτοντας ότι τα δεδομένα εισόδου είναι ακόμα διαθέσιμα.

Ένας άλλος τρόπος να δούμε το MapReduce είναι ένας παράλληλος και κατανεμημένος υπολογισμός 5 βημάτων:

1. Προετοιμασία των εισερχόμενων δεδομένων του Map()
   Το σύστημα MapReduce προσδιορίζει τους Map επεξεργαστές, εκχωρεί την τιμή κλειδιού εισόδου K1 που θα επεξεργάζεται κάθε επεξεργαστής και παρέχει στον επεξεργαστή όλα τα δεδομένα εισόδου που σχετίζονται με αυτή την τιμή κλειδιού.

2. Εκτέλεση του κώδικα Map()
   Ο Map() εκτελείται ακριβώς μία φορά για κάθε τιμή κλειδιού K1, δημιουργώντας έξοδο που οργανώνεται από τις τιμές κλειδιού K2.

3. Ανακάτεμα της εξόδου του Map προς τους επεξεργαστές Reduce
   Το σύστημα MapReduce προσδιορίζει τους επεξεργαστές Reduce. Στη συνέχεια εκχωρεί την τιμή κλειδιού K2 που πρέπει να επεξεργαστεί κάθε επεξεργαστής και παρέχει στον επεξεργαστή όλα τα δεδομένα που έχουν δημιουργηθεί με τον κώδικα Map που σχετίζονται με αυτή την τιμή κλειδιού.

4. Εκτέλεση του κώδικα Reduce()
   Ο Reduce() εκτελείται ακριβώς μία φορά για κάθε τιμή κλειδιού K2 που παράγεται από την εκτέλεση του κώδικα Map.

5. Δημιουργία της τελικής εξόδου
   Το σύστημα MapReduce συγκεντρώνει όλη την έξοδο του κώδικα Reduce και την ταξινομεί με το K2 για να παράγει το τελικό αποτέλεσμα.

Αυτά τα πέντε βήματα μπορούν λογικά να θεωρηθούν ως διαδοχικά. Κάθε βήμα ξεκινά μόνο μετά την ολοκλήρωση του προηγούμενου βήματος, αν και στην πράξη μπορούν να παρεμβληθούν όσο το τελικό αποτέλεσμα δεν επηρεάζεται.

Σε πολλές περιπτώσεις, τα δεδομένα εισόδου μπορεί ήδη να διανεμηθούν μεταξύ πολλών διαφορετικών εξυπηρετητών, οπότε το βήμα 1 θα μπορούσε μερικές φορές να απλοποιηθεί σε μεγάλο βαθμό με την ανάθεση διεργασιών Map που θα επεξεργάζονταν τα τοπικά υπάρχοντα δεδομένα εισόδου. Ομοίως, το βήμα 3 θα μπορούσε μερικές φορές να επιταχυνθεί με την

ανάθεση επεξεργαστών Reduce οι οποίοι είναι όσο το δυνατόν πιο κοντά στα δεδομένα που δημιουργούνται από την Map φάση.

2.7.2 Λογική προσέγγιση

Οι λειτουργίες Map και Reduce του μοντέλου καθορίζονται σε σχέση με τα εισερχόμενα δεδομένα. Τα δεδομένα είναι δομημένα στο πρότυπο των ζευγών κλειδί-τιμή. Ο Map λαμβάνει ένα ζεύγος δεδομένων με έναν τύπο σε έναν τομέα δεδομένων και επιστρέφει μια λίστα ζευγών σε έναν διαφορετικό τομέα:

Map(k1, v1) → list(k2, v2)

Η λειτουργία Map εφαρμόζεται παράλληλα σε κάθε ζεύγος (με κλειδί το k1) στο σύνολο δεδομένων εισόδου. Αυτό παράγει μια λίστα ζευγών (με κλειδί το k2) για κάθε κλήση της Map. Μετά από αυτό, το MapReduce συλλέγει όλα τα ζεύγη με το ίδιο κλειδί (k2) από όλες τις λίστες και τα ομαδοποιεί δημιουργώντας μία ομάδα για κάθε κλειδί.

Στη συνέχεια, εφαρμόζεται η Reduce παράλληλα σε κάθε ομάδα, η οποία με τη σειρά της παράγει μια συλλογή τιμών στον ίδιο τομέα:

Reduce(k2, list(v2)) → list(v3)

Κάθε κλήση της Reduce συνήθως παράγει είτε μία τιμή v3 είτε μία κενή επιστροφή, αν και επιτρέπεται σε μία κλήση να επιστρέψει περισσότερες από μία τιμές. Οι επιστροφές όλων των κλήσεων συλλέγονται για να διαμορφώσουν την επιθυμητή λίστα αποτελεσμάτων.

Έτσι, το MapReduce μετατρέπει μια λίστα ζευγών κλειδί-τιμή σε μια λίστα τιμών. Αυτή η συμπεριφορά είναι διαφορετική από τον τυπικό λειτουργικό Map και Reduce συνδυασμό, ο οποίος δέχεται μια λίστα τυχαίων τιμών και επιστρέφει μία μόνο τιμή η οποία συνδυάζει όλες τις τιμές που επιστρέφει ο Map.

Οι κατανεμημένες υλοποιήσεις συστημάτων του MapReduce, όπως το Hadoop, απαιτούν ένα μέσο σύνδεσης των διαδικασιών που εκτελούν τις φάσεις Map και Reduce. Αυτό μπορεί να είναι ένα κατανεμημένο σύστημα αρχείων όπως το HDFS ή κάποια κατανεμημένη βάση δεδομένων.

2.7.3 Ροή Δεδομένων

Τα κύρια σημεία που ορίζουν τη ροή δεδομένων του MapReduce είναι τα εξής: ένας αναγνώστη εισόδου, μια λειτουργία Map, μια λειτουργία διαίρεσης, μια λειτουργία σύγκρισης, μια λειτουργία Reduce, ένας γραφέας εξόδου.

Αναγνώστης εισόδου

Ο αναγνώστης εισόδου χωρίζει την είσοδο σε "χωρίσματα" κατάλληλου μεγέθους (τυπικά ανάμεσα σε 64 MB έως 128 MB) και το σύστημα αποδίδει ένα διαχωρισμό σε κάθε Map λειτουργία. Ο αναγνώστης εισόδου διαβάζει δεδομένα από σταθερή αποθήκευση (συνήθως ένα κατανεμημένο σύστημα αρχείων) και δημιουργεί ζεύγη κλειδιών-τιμών.

Ένα κοινό παράδειγμα είναι πως θα διαβάσει έναν κατάλογο γεμάτο αρχεία κειμένου και θα επιστρέψει κάθε γραμμή ως εγγραφή/αρχείο.

Λειτουργία Χαρτογράφησης (Map)

Η λειτουργία Map λαμβάνει μια σειρά από ζεύγη κλειδιών-τιμών, επεξεργάζεται το καθένα και παράγει μηδενικά ή περισσότερα ζεύγη κλειδιών-τιμών εξόδου. Οι τύποι εισόδου και εξόδου του χάρτη μπορούν να είναι (και συχνά είναι) διαφορετικοί ο ένας από τον άλλο.

Εάν η εφαρμογή κάνει το γνωστό πρόβλημα της μέτρησης λέξεων (ποια είναι η συχνότητα των λέξεων σε ένα κείμενο), η Map λειτουργία θα σπάσει τη γραμμή σε λέξεις και θα εξάγει ένα ζεύγος κλειδιού-τιμής για κάθε λέξη. Κάθε ζευγάρι εξόδου θα περιέχει τη λέξη ως κλειδί και τον αριθμό των στιγμιότυπων αυτής της λέξης στη γραμμή ως την τιμή.

Λειτουργία διαίρεσης

Κάθε έξοδος της Map λειτουργίας αντιστοιχεί σε ένα συγκεκριμένο Reducer από τη λειτουργία κατανομής της εφαρμογής. Η λειτουργία διαίρεσης λαμβάνει το κλειδί και τον αριθμό των Reducer και επιστρέφει τον δείκτη του επιθυμητού Reducer.

Μεταξύ του Map και της φάσης του Reducer, τα δεδομένα ανακατεύονται (παράλληλα ταξινομούνται / ανταλλάσσονται μεταξύ των κόμβων) προκειμένου τα δεδομένα να μετακινηθούν από τον κόμβο του Map που τα παρήγαγε στον κόμβο εκείνο που θα γίνει το

Reduce. Το ανακάτεμα μπορεί μερικές φορές να διαρκέσει περισσότερο από τον χρόνο υπολογισμού, ανάλογα με το εύρος ζώνης δικτύου, τις ταχύτητες CPU, τα δεδομένα που παράγονται και το χρόνο που λαμβάνεται από τις λειτουργίες Map και Reduce.

Λειτουργία σύγκρισης

Η είσοδος για κάθε Reduce τραβιέται από το μηχάνημα όπου η λειτουργία Map έτρεξε και ταξινομείται χρησιμοποιώντας τη λειτουργία σύγκρισης της εφαρμογής.

Λειτουργία Μείωσης (Reduce)

Το σύστημα καλεί τη λειτουργία Reduce της εφαρμογής μία φορά για κάθε μοναδικό κλειδί της ταξινομημένης σειράς. Το Reduce μπορεί να επαναλάβει τις τιμές που σχετίζονται με αυτό το κλειδί και να παράγει μηδέν ή περισσότερες εξόδους.

Στο παράδειγμα μέτρησης λέξεων, η λειτουργία Reduce παίρνει τις τιμές εισόδου, τις αθροίζει και παράγει μία μόνο έξοδο της λέξης και το τελικό άθροισμα.

Γραφέας εξόδου

Ο γραφέας εξόδου καταγράφει την έξοδο του Reducer στο σύστημα αποθήκευσης.

Κεφάλαιο 3 - Βάσεις Δεδομένων NoSQL

Ο όρος NoSQL προέρχεται αρχικά από το «non SQL» ή «non relational» και θέλει να δηλώσει πως τα πρότυπά της διαφέρουν από τη γνωστή σχεσιακή βάση δεδομένων SQL. Εξ ορισμού οι βάσεις δεδομένων NoSQL είναι μη σχεσιακές και είναι σχεδιασμένες για την αποθήκευση, επεξεργασία και διαχείριση Μεγάλου Όγκου Δεδομένων. Τέτοιου είδους βάσεις δεδομένων υπήρξαν από την εποχή του 1960, αλλά το όνομα τους δόθηκε στις αρχές του 21ου αιώνα αφού αποτέλεσε έναυσμα από τις ανάγκες που είχαν εταιρίες όπως η Facebook, Google και Amazon για προβλήματα του παγκόσμιου ιστού. Οι βάσεις δεδομένων NoSQL χρησιμοποιούνται όλο και περισσότερο σε Μεγάλο Όγκο Δεδομένων και εφαρμογές παγκόσμιου ιστού σε πραγματικό χρόνο. Τα συστήματα NoSQL ονομάζονται μερικές φορές και "όχι μόνο SQL" για να υπογραμμίσουν ότι μπορούν ακόμα να υποστηρίξουν γλώσσες τύπου SQL.

3.1 Γενικά για NoSQL

Γενικότερα, τα Big Data χαρακτηρίζονται από την θεωρία των 3Vs: Volume, Variety, Velocity. Το Volume αναφέρεται στην ποσότητα των δεδομένων που αποθηκεύονται και επεξεργάζονται. Ο μεγάλος όγκος δεδομένων των NoSQL αυξάνεται συνεχώς και είναι πολύ μεγάλα σε μέγεθος, της τάξης των PetaBytes και του ExaByte. Αυτά τα δεδομένα καθώς εισάγονται στη μη σχεσιακή βάση αντιγράφονται σε διαφορετικούς κόμβους και επεξεργάζονται παράλληλα. Το Variety αναφέρεται στα διάφορα είδη δομών δεδομένων, όπως τα δομημένα, τα ημι-δομημένα και τα αδόμητα δεδομένα. Οι βάσεις δεδομένων NoSQL δεν έχουν δομή σχήματος και μπορούν να χειριστούν αδόμητα δεδομένα. Το τρίτο V, Velocity, αντιπροσωπεύει την ταχύτητα με την οποία τα δεδομένα παράγονται και συλλέγονται ή επεξεργάζεται από τις βάσεις δεδομένων NoSQL.

Με βάση τα χαρακτηριστικά των δεδομένων και των εφαρμογών, έχουν αναπτυχθεί διάφορα είδη βάσεων δεδομένων NoSQL. Οι διαφορετικές βάσεις δεδομένων του NoSQL ακολουθούν διαφορετικά μοντέλα δεδομένων και αρχιτεκτονικές.

3.2 Τύποι Μη Σχεσιακών Βάσεων Δεδομένων

Έχουν υπάρξει διάφορες προσεγγίσεις για την κατηγοριοποίηση των με σχεσιακών βάσεων δεδομένων NoSQL, καθένα με διαφορετικές κατηγορίες και υποκατηγορίες, μερικές από τις οποίες συμπίπτουν. Παρακάτω ακολουθούν οι πιο σημαντικές κατηγορίες βάσεων δεδομένων NoSQL

Key-Value store

Η κατηγορία βάσης δεδομένων Key-Value χρησιμοποιεί πίνακα συσχέτισης (γνωστό και ως χάρτη ή λεξικό) ως βασικό μοντέλο δεδομένων. Είναι η πιο βασική μορφή βάσεων δεδομένων NoSQL, όπου κάθε εγγραφή αποθηκεύεται ως ένα ζεύγος κλειδιού-τιμής με την τιμή να είναι μη ορατή από το σύστημα. Ως εκ τούτου, κάθε εγγραφή μπορεί να ερωτηθεί μόνο από το κλειδί, το οποίο εξασφαλίζει γρήγορη πρόσβαση στα κλειδιά. Επίσης, κάθε πιθανό κλειδί εμφανίζεται το πολύ μία φορά στη συλλογή.

Το μοντέλο Key-Value είναι ένα από τα απλούστερα μοντέλα δεδομένων που δεν είναι τετριμμένα και πολλά από τα πιο μεγάλα μοντέλα δεδομένων εφαρμόζονται συχνά ως επέκταση αυτού. Το μοντέλο Key-Value μπορεί να επεκταθεί σε ένα διακριτικά διατεταγμένο μοντέλο που διατηρεί τα κλειδιά (Key) του σε αλφαβητική σειρά. Αυτή η επέκταση είναι υπολογιστικά ισχυρή, καθώς μπορεί να ανακτήσει αποτελεσματικά επιλεκτικά εύρη κλειδιών.

Οι βάσεις δεδομένων Key-Value μπορούν να χρησιμοποιήσουν μοντέλα συνεκτικότητας που κυμαίνονται από ενδεχόμενη συνέπεια έως και σειριοποίηση. Υπάρχουν διάφορες υλοποιήσεις και ορισμένες από αυτές τις βάσεις δεδομένων υποστηρίζουν και την ταξινόμηση των κλειδιών. Τέλος, κάποιες υλοποιήσεις συγκεντρώνουν τα δεδομένα στη μνήμη (RAM), ενώ άλλες χρησιμοποιούν μονάδες SSD ή περιστρεφόμενους δίσκους.

Παραδείγματα τέτοιου τύπου βάσεων δεδομένων είναι η ArangoDB, η InfinityDB, η βάση δεδομένων Oracle NoSQL, η Redis και η dbm.

Document store

Αυτές οι βάσεις δεδομένων συνήθως αποθηκεύουν τα δεδομένα σε μια δομή τύπου JSON (JavaScript Object Notation) ως έγγραφα. Κάθε έγγραφο περιέχει ένα ή περισσότερα πεδία. Κάθε έγγραφο έχει ένα μοναδικό κλειδί και επιτρέπεται στα έγγραφα να έχουν διαφορετικά πεδία από χαρακτηριστικά.

Ένα από τα άλλα καθοριστικά χαρακτηριστικά μιας βάσης δεδομένων που βασίζεται σε έγγραφα είναι ότι εκτός από την αναζήτηση κλειδιών που εκτελείται από μία βάση δεδομένων Key-Value, η βάση δεδομένων προσφέρει ένα API ή μια γλώσσα ερωτημάτων που ανακτά τα έγγραφα με βάση το περιεχόμενό τους.

Διάφορες υλοποιήσεις προσφέρουν διαφορετικούς τρόπους οργάνωσης και/ή ομαδοποίησης εγγράφων:

- Συλλογές
- Ετικέτες
- Μη ορατά μετα-δεδομένα
- Ιεραρχίες καταλόγων

Σε σύγκριση με τις σχεσιακές βάσεις δεδομένων, για παράδειγμα, οι συλλογές θα μπορούσαν να θεωρηθούν ανάλογες με τους πίνακες και τα έγγραφα ανάλογα με τα αρχεία. Αλλά είναι διαφορετικά, διότι κάθε εγγραφή σε έναν πίνακα έχει την ίδια ακολουθία πεδίων, ενώ στο μοντέλο Document μια συλλογή μπορεί να έχει πεδία που είναι τελείως διαφορετικά.

Κάποια παραδείγματα από key-value βάσεις δεδομένων είναι: Riak, DynamoDB, Voldemort, MongoDB και Redis.

Graph store

Αυτές οι βάσεις δεδομένων NoSQL αντιπροσωπεύουν σχέσεις μεταξύ αντικειμένων σε δομή γραφημάτων. Οι βάσεις δεδομένων γραφημάτων βασίζονται στη θεωρία γραφημάτων και χρησιμοποιούν κόμβους, ακμές και ιδιότητες.

Οι κόμβοι αντιπροσωπεύουν οντότητες όπως άτομα, επιχειρήσεις, λογαριασμούς ή οποιοδήποτε άλλο αντικείμενο που πρέπει να παρακολουθείται. Είναι περίπου το ισοδύναμο του αρχείου, της σχέσης ή της σειράς σε μια σχεσιακή βάση δεδομένων ή του εγγράφου σε μια βάση δεδομένων Document.

Οι ακμές, που ονομάζονται επίσης γραφήματα ή σχέσεις, είναι οι γραμμές που συνδέουν τους κόμβους μεταξύ τους και αντιπροσωπεύει τη σχέση τους. Κατά την εξέταση των συνδέσεων και των διασυνδέσεων κόμβων, ιδιοτήτων και ακμών μπορούμε να συμπεράνουμε σημαντικά μοτίβα. Οι ακμές είναι η βασική ιδέα στις βάσεις δεδομένων των γραφικών παραστάσεων.

Οι ιδιότητες είναι πληροφορίες που σχετίζονται με τους κόμβους.

Οι βάσεις δεδομένων γραφημάτων είναι χρήσιμες όταν οι σχέσεις μεταξύ αντικειμένων και δεδομένων είναι πιθανόν πιο σημαντικές από τα δεδομένα. Παραδείγματα από τέτοιες βάσεις δεδομένων είναι τα Neo4j και Giraph.

Column store

Αυτές οι βάσεις δεδομένων αποθηκεύουν και διαχειρίζονται δομημένα και ημι-δομημένα δεδομένα που μπορούν να κλιμακωθούν σε πολύ μεγάλο αριθμό κόμβων. Σε αντίθεση με τις βάσεις Key-Value, κάθε εγγραφή μπορεί να φιλοξενήσει πολλές στήλες (ιδιότητες) μαζί και οι στήλες μπορούν να ομαδοποιηθούν για να σχηματίσουν οικογένειες στηλών.

Κάποια παραδείγματα από τέτοιες βάσεις δεδομένων είναι τα HBase, BigTable και Cassandra.

3.3 Αρχιτεκτονική Μη Σχεσιακών Βάσεων Δεδομένων

Με σκοπό την κατανόηση της αρχιτεκτονικής των μη σχεσιακών βάσεων δεδομένων, πρέπει να καταλάβουμε το διαχωρισμό ανάμεσα στη διαχείριση δεδομένων και της διαδικασίας αποθήκευσής τους. Παλαιότερα, οι σχεδιαστές των σχεσιακών βάσεων δεδομένων προσπάθησαν να ικανοποιήσουν και τα δύο αυτά θέματα. Αυτό ήταν εξαιρετικά δύσκολο και αναπόφευκτα μέρος του έργου της διαχείρισης δεδομένων το ανέλαβαν εφαρμογές, παρέχοντας ορισμένα καθήκοντα επικύρωσης και προσθέτοντας λογική μοντελοποίησης. Μία από τις βασικές έννοιες των NoSQL ήταν να επικεντρωθούν στις εργασίες κλιμακωτής αποθήκευσης δεδομένων υψηλής απόδοσης και να παρέχει πρόσβαση χαμηλού επιπέδου σε ένα επίπεδο διαχείρισης δεδομένων με τρόπο τέτοιον ώστε να επιτρέπει την εύκολη καταγραφή των εργασιών διαχείρισης δεδομένων σε γλώσσα προγραμματισμού της επιλογής του χρήστη, αντί να υπάρχει η λογική διαχείρισης δεδομένων όπως η SQL ή οι γλώσσες εφαρμογών Turing.

3.4 MongoDB

Η MongoDB, της οποίας το όνομα προήλθε από τη λέξη humongous, που μεταφράζεται ως κάτι πολύ μεγάλο σε όγκο, είναι ένα δωρεάν και ανοιχτού κώδικα cross-platform πρόγραμμα βάσης δεδομένων. Η MongoDB δημιουργήθηκε από τους MongoDB Inc (πρώην 10gen) και κατηγοριοποιείται ως πρόγραμμα βάσης δεδομένων NoSQL και, ειδικότερα, προσανατολισμένη στο Document πρότυπο. Είναι από τις πιο διαδεδομένες βάσεις δεδομένων τύπου NoSQL και χρησιμοποιείται για να στηρίξει πολλές εφαρμογές παγκόσμιου ιστού και κινητής τηλεφωνίας από νεοσύστατες επιχειρήσεις έως και πολύ μεγάλες πολυεθνικές.

Πρόκειται για μια βάση δεδομένων IoT, η οποία αντί των πινάκων (όπως στις σχεσιακές) παρέχει μία ή περισσότερες συλλογές (collections) ως κύρια στοιχεία αποθήκευσης που αποτελούνται από παρόμοια ή διαφορετικά έγγραφα τύπου JSON ή BSON. Τα έγγραφα που τείνουν να έχουν κάποια παρόμοια δομή οργανώνονται σε συλλογές, οι οποίες μπορούν να δημιουργηθούν ανά πάσα στιγμή, χωρίς προκαθορισμένες παραμέτρους. Ένα έγγραφο μπορεί απλώς να θεωρηθεί ως μια σειρά (row) ή μια παρουσία μιας οντότητας σε σχέση με το κλασικό μοντέλο των σχεσιακών βάσεων δεδομένων (RDBMS), αλλά η διαφορά είναι ότι στη MongoDB μπορούμε να έχουμε οντότητες μέσα σε οντότητες ή μπορούμε να έχουμε έγγραφα με έγγραφα, ή ακόμη και λίστες ή πίνακες εγγράφων. Οι τύποι για τα

χαρακτηριστικά ενός εγγράφου μπορούν να είναι οποιουδήποτε βασικού τύπου δεδομένων, όπως αριθμοί, συμβολοσειρές, ημερομηνίες, πίνακες ή ακόμα και ένα υπο-έγγραφο.

3.4.2 Χαρακτηριστικά MongoDB

Ερωτήματα Ad-hoc

Η MongoDB υποστηρίζει ερωτήματα πεδίου, εύρους και αναζητήσεις κανονικής έκφρασης. Ερωτήματα προς τη βάση δεδομένων μπορούν να επιστρέψουν συγκεκριμένα πεδία των εγγράφων και επίσης μπορούν να περιλαμβάνουν συναρτήσεις JavaScript που ορίζονται από το χρήστη. Τα ερωτήματα μπορούν επίσης να ρυθμιστούν ώστε να επιστρέφουν ένα τυχαίο δείγμα αποτελεσμάτων ενός δεδομένου μεγέθους.

Indexing

Η MongoDB παρέχει απόλυτη υποστήριξη για την ευρετηρίαση κάθε πεδίου σε μια συλλογή ή σε έγγραφα. Η τιμή στο δείκτη του ευρετηρίου ενός πεδίου περιγράφει ένα είδος για το συγκεκριμένο πεδίο. Για παράδειγμα, μια τιμή 1 καθορίζει ένα δείκτη που ταξινομεί τα στοιχεία σε αύξουσα σειρά. Μια τιμή -1 καθορίζει ένα δείκτη που ταξινομεί στοιχεία σε φθίνουσα σειρά.

Replication

Η MongoDB παρέχει υψηλή διαθεσιμότητα με σύνολα πανομοιότυπων αντιγράφων. Ένα σύνολο αντιγράφων αποτελείται από δύο ή περισσότερα αντίγραφα των πρωτότυπων δεδομένων. Κάθε μέλος του συνόλου αντιγράφων μπορεί να ενεργεί ως πρωτεύον ή ως δευτερεύον αντιγράφου ανά πάσα στιγμή. Όμως, όλες οι εγγραφές και οι αναγνώσεις γίνονται στο πρωτεύον αντίγραφο, από προεπιλογή. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των δεδομένων του πρωτεύον χρησιμοποιώντας μια ενσωματωμένη αναπαραγωγή. Όταν ένα πρωτεύον αντίγραφο αποτύχει να ανταποκριθεί, το σύνολο αντιγράφου διεξάγει αυτόματα μια εκλογική διαδικασία για να προσδιορίσει ποιο δευτερεύον αντίγραφο πρέπει να

γίνει το πρωτεύον. Τα δευτερεύοντα αντίγραφα μπορούν προαιρετικά να εξυπηρετήσουν και λειτουργίες ανάγνωσης.

Load balancing

Η MongoDB κλιμακώνεται οριζόντια με τη χρήση της μεθόδου διχοτόμησης (sharding). Οριζόντια κλιμάκωση συμβαίνει όταν προσθέτουμε περισσότερους από έναν κόμβους στο σύστημα για λειτουργική υποστήριξη. Στη μέθοδο της διχοτόμησης ο χρήστης επιλέγει ένα κλειδί διχοτόμησης, το οποίο καθορίζει τον τρόπο με τον οποίο θα κατανεμηθούν τα δεδομένα μιας συλλογής. Τα δεδομένα χωρίζονται σε εύρη τιμών (με βάση το κλειδί) και κατανέμονται σε πολλαπλούς τομείς. Ένας τομέας είναι ένας master με έναν ή περισσότερους slaves.

File storage

Η MongoDB μπορεί να χρησιμοποιηθεί ως σύστημα αρχείων με λειτουργίες εξισορρόπησης φορτίου και αναπαραγωγής δεδομένων σε πολλά μηχανήματα για την αποθήκευση αρχείων. Αυτή η διαδικασία ονομάζεται σύστημα αρχείων με πλέγμα. Σε ένα παράδειγμα τέτοιας λειτουργίας το GridFS χωρίζει ένα αρχείο σε τμήματα ή κομμάτια και αποθηκεύει κάθε ένα από αυτά ως ξεχωριστό έγγραφο.

Η MongoDB εκθέτει επίσης συναρτήσεις με σκοπό τη δημιουργία νέων μεθοδων επεξεργασίας αρχείων και περιεχομένου σε προγραμματιστές.

Aggregation

Το μοντέλο MapReduce μπορεί να χρησιμοποιηθεί για την επεξεργασία κατά παρτίδες των δεδομένων και των διεργασιών συγκέντρωσης.

Η διαδικασία του aggregation επιτρέπει στους χρήστες να αποκτήσουν το είδος των αποτελεσμάτων εκείνων που αντίστοιχα στην SQL έχουμε το GROUP BY. Επίσης, χειριστές της διαδικασίας αυτής, μπορούν να συσπειρωθούν μαζί με σκοπό να σχηματίσουν έναν αγωγό (pipeline) ανάλογο με τους αγωγούς Unix. Παράλληλα το aggregation περιλαμβάνει

έναν τελεστή αναζήτησης, ο οποίος μπορεί να συμπτύξει έγγραφα από πολλά έγγραφα, καθώς και στατιστικούς τελεστές, όπως η τυπική απόκλιση.

Server-side JavaScript execution

Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα, συναρτήσεις (όπως το MapReduce) και αποστέλλεται απευθείας στη βάση δεδομένων όπου πρόκειται να εκτελεστεί.

Capped collections

Η MongoDB υποστηρίζει συλλογές σταθερού μεγέθους που ονομάζονται ανώτατες συλλογές (capped collections). Αυτός ο τύπος συλλογής διατηρεί σειρά εισαγωγής των εγγράφων και, μόλις δηλωθεί το καθορισμένο μέγεθος, συμπεριφέρεται σαν μια κυκλική ουρά.

3.5 Λόγοι Επιλογής MongoDB

Αναμφίβολα η MongoDB είναι κατάλληλη για εργασίες που απαιτούν το χειρισμό τεράστιων ποσοτήτων δεδομένων. Παράλληλα, η βάση δεδομένων αυτή ταιριάζει απόλυτα για την ανάπτυξη πλατφορμών παγκοσμίου ιστού, αποθήκευσης δεδομένων και εφαρμογών βασισμένες σε τοποθεσία.

Ένας λόγος επιλογής της MongoDB είναι το δεδομένο πως ως χρήστης δεν είσαι υπεύθυνος να καθορίσεις το σχηματικό της βάσης. Το μόνο που καλείσαι να κάνεις είναι απλά να εισάγεις έγγραφα. Τα έγγραφα μπορεί να έχουν οποιαδήποτε δομή. Δύο έγγραφα μέσα σε μία συλλογή δε χρειάζεται να έχουν τα ίδια πεδία. Αυτό μπορεί να δυσκολεύει την κατάσταση όταν θα κάνεις ερωτήματα στη βάση διότι δεν υπάρχουν αλληλουχίες μεταξύ των εγγράφων, αλλά είναι και ένας λόγος που βάση είναι ταχύτερη.

Κατά κοινή ομολογία, η MongoDB έχει καλύτερη απόδοση σε σχέση με τις μη σχεσιακές βάσεις δεδομένων αλλά και από πολλές άλλες μη σχεσιακές. Ένας από τους λόγους είναι διότι η MongoDB θυσιάζει τη δυνατότητα JOIN και άλλων επιλογών αναζήτησης για να διαθέσει εξαιρετικά εργαλεία ανάλυσης της απόδοσης.

Θα αποτελούσε σοβαρή παράλειψη να μην τονίσουμε το γεγονός ότι η MongoDB χρησιμοποιεί οριζόντια κλιμάκωση και υψηλή διαθεσιμότητα. Όπως ειπώθηκε πιο πάνω η κλιμάκωση της MongoDB γίνεται προσθέτοντας παραπάνω κόμβους στο σύστημα. Η κλιμάκωση της MongoDB γίνεται με τη διχοτόμηση (sharding) το οποίο είναι αρκετά μοναδικό με τον δικό του τρόπο. Μπορείς αν θες να χτίσεις τη δική σου τοπολογία με αντιγραφές και διχοτομήσεις. Για να αυξηθεί η διαθεσιμότητα και η συνέπεια, η βάση αυτή αναπαράγει τα δεδομένα και δημιουργεί πανομοιότυπα αντίγραφα. Αυτό συμβαίνει επειδή η διαδικασία αναπαραγωγής παρέχει συνεπή κλίμακα ανάγνωσης, η διχοτόμηση διευκολύνει τις λειτουργίες ανάγνωσης και εγγραφής, και η βάση δεδομένων χειρίζεται έξυπνα τις λειτουργίες αποτυχίας αντιγραφής.

Η MongoDB διαθέτει, όπως και το Apache Hadoop, τη διαδικασία MapReduce το οποίο επιτρέπει την εύκολη και γρήγορη επεκτασιμότητα. Αυτό σημαίνει ότι μπορείς να λάβεις τη μέγιστη λειτουργικότητα της βάσης δεδομένων MongoDB ακόμα κι αν χρησιμοποιείς κάποιο υλικό χαμηλού κόστους.

Βασικός λόγος παραμένει ο τύπος των δεδομένων. Αφού στην παρούσα διπλωματική χρησιμοποιούνται δεδομένα JSON, η επιλογή μιας NoSQL βάσης δεδομένων ήταν μονόδρομος. Γενικότερα η εξόρυξη δεδομένων από το διαδίκτυο από διάφορα Twitter APIs και web crawlers συνηθίζει να εξάγει αδόμητα δεδομένα. Η MongoDB είναι ιδανική για τη διαχείριση δεδομένων τέτοιου τύπου, διότι διαθέτει εργαλεία αναζήτησης και ερωτημάτων τα οποία είναι κατάλληλα για αδόμητα δεδομένα.

Κεφάλαιο 4 - Συναισθηματική Ανάλυση

Με την πρόοδο της τεχνολογίας του διαδικτύου και την ανάπτυξή της, υπάρχει ένας τεράστιος όγκος δεδομένων σε αυτό, ο οποίος είναι προσπελάσιμο στους χρήστες του, ενώ παράλληλα πολλά δεδομένα δημιουργούνται σε αυτό κάθε δευτερόλεπτο που περνάει. Το Διαδίκτυο έχει γίνει μια πλατφόρμα για την ηλεκτρονική μάθηση, την ανταλλαγή ιδεών και την ανταλλαγή απόψεων. Οι ιστότοποι κοινωνικής δικτύωσης όπως το Twitter, το Facebook, το Google+ γρήγορα κερδίζουν δημοτικότητα, δεδομένου ότι επιτρέπουν στους ανθρώπους να μοιραστούν και να εκφράσουν τις απόψεις τους για τα κοινά και ταυτόχρονα μπορούν να συζητήσουν με διαφορετικές κοινότητες μέσω μηνυμάτων σε όλο τον κόσμο.

4.1 Εισαγωγή στην Ανάλυση Συναισθήματος

Η ανάλυση συναισθήματος (ή στα Αγγλικά Sentiment Analysis), γνωστή και ως "εξόρυξη γνώμης", αναφέρεται στη χρήση της επεξεργασίας φυσικής γλώσσας (Natural Language Processing), της ανάλυσης κειμένου, της υπολογιστικής γλωσσολογίας και των βιομετρικών στοιχείων για τον συστηματικό εντοπισμό, εξαγωγή, ποσοτικοποίηση και μελέτη των συναισθηματικών καταστάσεων και των υποκειμενικών πληροφοριών [12]. Η ανάλυση συναισθήματος εφαρμόζεται μέσω των κοινωνικών μέσων δικτύωσης και ειδικότερα μέσα από κριτικές και απαντήσεις σε έρευνες όπου ο χρήστης πληροφορεί και επηρεάζει άλλους μέσω της άποψής του ή των συναισθημάτων του για κάποιο προϊόν ή υπηρεσία. Ακόμα, συναισθηματική ανάλυση έχουμε σε διαδικτυακά ή κοινωνικά μέσα ενημέρωσης, σε λογισμικά υγειονομικής περίθαλψης, σε εφαρμογές που αφορούν το μάρκετινγκ έως την εξυπηρέτηση πελατών σε κλινικές.

Σε γενικές γραμμές, η ανάλυση συναισθημάτων έχει ως στόχο να καθορίσει τη στάση ενός ομιλητή, συγγραφέα ή άλλου υποκειμένου σε σχέση με κάποιο θέμα ή τη συνολική συμφραζόμενη πολικότητα ή συναισθηματική αντίδραση σε ένα έγγραφο, αλληλεπίδραση ή γεγονός. Η στάση μπορεί να είναι κρίση ή αξιολόγηση, συναισθηματική κατάσταση ή επιδιωκόμενη συναισθηματική επικοινωνία (δηλαδή το συναισθηματικό αποτέλεσμα που επιδιώκει ο συντάκτης ή ο ομιλητής).

Είναι γνωστό πως η περισσότερη δουλειά στον τομέα της ανάλυσης συναισθήματος πραγματοποιείται με δεδομένα του Twitter. Για το λόγο αυτό και για χάριν ευκολίας τα δεδομένα για τα οποία θα δίδονται ως παράδειγμα θα είναι συνήθως δεδομένα Twitter.

4.2 Προ-απαιτούμενες Διαδικασίες

Η ανάλυση συναισθημάτων κατατάσσει την πληροφορία σε κατηγορίες όπως: θετική ή αρνητική ή ουδέτερη. Αναλύει την υποκειμενικότητα, την εξόρυξη γνώμης και την εξόρυξη αξιολόγησης.

Οι λέξεις γνώμη, συναίσθημα, άποψη και πεποίθηση χρησιμοποιούνται εναλλακτικά, αλλά υπάρχουν διαφορές μεταξύ τους [12].

- Γνώμη: Ένα συμπέρασμα ανοικτό σε αλλαγή
- Άποψη: Η υποκειμενική γνώμη
- Πεποίθηση: Η εκούσια αποδοχή και η πνευματική σύμφωνη γνώμη
- Συναίσθημα: Η γνώμη που αντιπροσωπεύει συναισθήματα ενός ατόμου

Μαθηματικά, μπορούμε να παρουσιάσουμε μια γνώμη ως μια πλειάδα από πέντε ορίσματα: (o, f, so, h, t), όπου:

- o = αντικείμενο
- f = χαρακτηριστικό του αντικειμένου o
- so = προσανατολισμός ή πολικότητα της γνώμης για το χαρακτηριστικό f του αντικειμένου o
- h = κάτοχος γνώμης
- t = χρόνος κατά τον οποίο εκφράζεται η γνώμη.

Αντικείμενο: Μια οντότητα που μπορεί να είναι ένα, πρόσωπο, γεγονός, προϊόν, υπηρεσία, οργάνωση ή θέμα

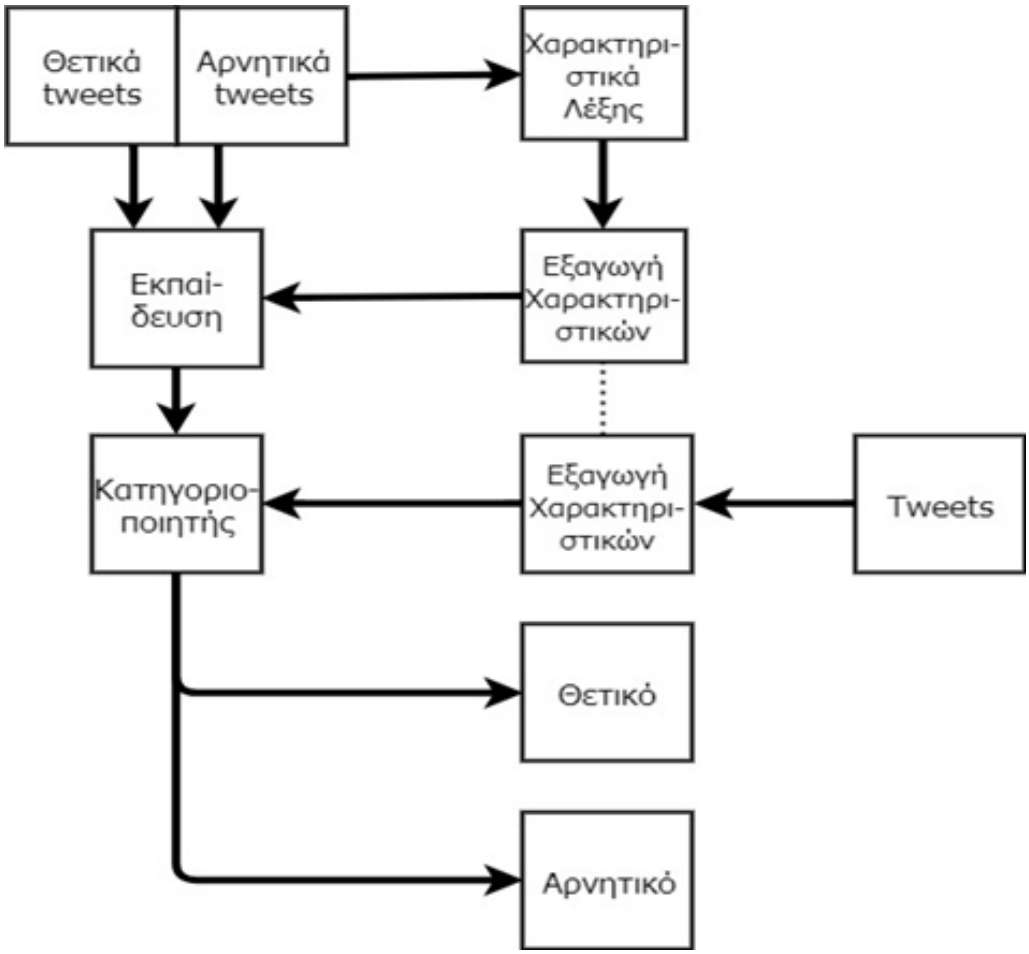
Χαρακτηριστικό: Ένα χαρακτηριστικό (ή μέρος) του αντικειμένου για το οποίο γίνεται η αξιολόγηση.

Ο προσανατολισμός ή η πολικότητα της γνώμης: Ο προσανατολισμός μιας γνώμης σε ένα χαρακτηριστικό f αντιπροσωπεύει εάν η γνώμη είναι θετική, αρνητική ή ουδέτερη.

Κάτοχος γνώμης: Ο κάτοχος μιας γνώμης είναι το πρόσωπο ή ο οργανισμός ή μια οντότητα που εκφράζει τη γνώμη του.

Μια ανάλυση συναισθημάτων περιλαμβάνει πολλές εργασίες όπως: εξόρυξη συναισθήματος, ταξινόμηση αυτού, περιληπτική παρουσίαση των απόψεων ή ανίχνευση λαθεμένων απόψεων [15]. Στόχος είναι να αναλύσει τα συναισθήματα των ανθρώπων, τις στάσεις ή τις απόψεις απέναντι σε στοιχειά όπως προϊόντα, θέματα, οργανώσεις και υπηρεσίες.

Ένα γενικό μοντέλο συναισθηματικής ανάλυσης είναι το παρακάτω:



Εικόνα 4 - Αρχιτεκτονική Ανάλυσης Συναισθήματος σε Tweets

4.2.1 Προ-επεξεργασία

Ένα tweet εμπεριέχει πολλές γνώμες για ένα δεδομένο και εκφράζεται με διαφορετικούς τρόπους από διαφορετικούς χρήστες. Συνήθως τα πρωτογενή δεδομένα που έχουν πολικότητα είναι ιδιαίτερα ευαίσθητα σε ασυνέπεια και πλεονασμό [12]. Μία προ-επεξεργασία ενός tweet θα μπορούσε να περιλαμβάνει τα ακόλουθα σημεία:

- Κατάργηση όλων των διευθύνσεων URL (π.χ. www.xyz.com), ετικετών hashtag (π.χ. #topic), στόχων (@username)
- Διόρθωση ορθογραφίας
- Αντικατάσταση όλων των emoticons με τη συναισθηματική τους έννοια
- Κατάργηση των σημείων στίξης, συμβόλων, αριθμών
- Αφαίρεση συνδέσμων, άρθρων κ.λπ. (stop words)
- Επέκταση των ακρωνύμιων (μπορεί να χρησιμοποιηθεί ένα λεξικό ακρωνύμιων)
- Κατάργηση μη αγγλικών tweets

4.2.2 Εξαγωγή χαρακτηριστικών

Το προ-επεξεργασμένο σύνολο δεδομένων έχει πολλές ξεχωριστές ιδιότητες [12]. Στη μέθοδο εξαγωγής χαρακτηριστικών, εξάγονται απόψεις από την επεξεργασία δεδομένων. Στη συνέχεια, αυτές οι απόψεις χρησιμοποιούνται για τον υπολογισμό της θετικής και αρνητικής πολικότητας σε μια πρόταση, η οποία είναι χρήσιμη για τον καθορισμό του συναισθήματος των ατόμων.

Τεχνικές όπως το Machine Learning (Μηχανική Μάθηση) απαιτούν μια αναπαράσταση χαρακτηριστικού-κλειδιού από το κείμενο ή το έγγραφο ώστε να γίνει η επεξεργασία. Αυτά τα χαρακτηριστικά-κλειδιά θεωρούνται ως ένα διάνυσμα που χρησιμοποιούνται για κατηγοριοποίηση.

Παραδείγματα εξαγωγής χαρακτηριστικού είναι:

1. **Οι λέξεις και οι συχνότητές τους:** Τα unigrams (N-gram μεγέθους «ένα»), τα bigrams (N-gram μεγέθους «δύο») και τα n-gram (συνεχή ακολουθία n στοιχείων από μια δεδομένη ακολουθία κειμένου ή ομιλίας) μοντέλα με τους αριθμούς συχνότητάς τους θεωρούνται ως χαρακτηριστικά. Έχουν γίνει περισσότερες μελέτες σχετικά με τη

χρήση της παρουσίας λέξεων αντί για τις συχνότητες τους για να περιγράφει καλύτερα αυτό το χαρακτηριστικό.

2. **Μέρη των ετικετών ομιλίας:** Τμήματα λόγου όπως επίθετα, επιρρήματα και ορισμένες ομάδες ρημάτων και ουσιαστικών είναι καλοί δείκτες υποκειμενικότητας και συναισθήματος. Μπορούν να δημιουργηθούν συσχετιστικά μοτίβα εξάρτησης από δέντρα ανίχνευσης ή εξάρτησης.

3. **Λέξεις και φράσεις γνώμης:** Συγκεκριμένες λέξεις, ορισμένες φράσεις, μεταφορές και ιδιώματα που μεταφέρουν συναισθήματα μπορούν να χρησιμοποιηθούν ως χαρακτηριστικά. Το επόμενο παράδειγμα αναφέρεται σε συγκεκριμένη μεταφορά λόγου, π.χ. Cost someone an arm and leg.

4. **Θέση των όρων:** Η θέση ενός όρου σε ένα κείμενο μπορεί να επηρεάσει το πόσο ο όρος αυτός διαφέρει στο γενικό συναίσθημα του κειμένου.

5. **Άρνηση:** Η άρνηση είναι ένα σημαντικό αλλά δύσκολο χαρακτηριστικό για την ερμηνεία. Η παρουσία μιας άρνησης συνήθως αλλάζει την πολικότητα της γνώμης.

6. **Σύνταξη:** Συντακτικά μοτίβα, όπως παραθέσεις, χρησιμοποιούνται ως χαρακτηριστικά ώστε οι ερευνητές να μάθουν πρότυπα υποκειμενικότητας από αυτά.

4.2.3 Εκπαίδευση Κατηγοριοποιητή

Η εκπαίδευση ενός κατηγοριοποιητή καθιστά ευκολότερο το έργο για μελλοντικές προβλέψεις για άγνωστα δεδομένα. Είναι ένας τρόπος που χρησιμοποιεί η εποπτευόμενη μάθηση (Supervised Learning), η οποία είναι μια σημαντική τεχνική για την επίλυση προβλημάτων κατηγοριοποίησης [16]. Η εκπαίδευση γίνεται με ένα training set (σύνολο εκπαίδευσης) το οποίο αποτελείται από ένα σύνολο κατηγοριοποιημένων παραδειγμάτων που λειτουργεί ως υπόδειγμα για τον κατηγοριοποιητή.

Αντί των training set, χρησιμοποιούνται επίσης, μαζί με αυτά, τα test set (σύνολο ελέγχου). Τα test set είναι ανεξάρτητα από τα training set αλλά ακολουθούν την ίδια κατανομή πιθανότητας με αυτά. Συνήθως αυτά χρησιμοποιούνται τις περισσότερες φορές για αξιολόγηση της απόδοσης του κατηγοριοποιητή.

4.2.4 Κατηγοριοποίηση

Naïve Bayes

Είναι ένας πιθανολογικός κατηγοριοποιητής και μπορεί να μάθει το μοτίβο της κατηγοριοποίησης εξετάζοντας μια σειρά από έγγραφα που έχουν ήδη κατηγοριοποιηθεί (training set) [12]. Συγκρίνει τα περιεχόμενα με τον κατάλογο των λέξεων για να κατηγοριοποιήσει τα έγγραφα στη σωστή κατηγορία/κλάση. Αν d είναι ένα tweet και C* είναι η κλάση που ανατεθεί το d, τότε:

$$C^* = \arg mac_c P_{NB}(c \mid d)$$

$$P_{NB}(c \mid d) = \frac{(P(c))\sum_{i=1}^{m} p(f \mid c)^{n_{i(d)}}}{P(d)}$$

Από την παραπάνω εξίσωση, το f είναι το χαρακτηριστικό, ενώ ο αριθμός του χαρακτηριστικού f_i συμβολίζεται με το $n_i(d)$ και υπάρχει στο d (το tweet). Το m αντιπροσωπεύει τον αριθμό των χαρακτηριστικών.

Οι παράμετροι P(c) και p(f|c) υπολογίζονται με εκτιμήσεις μέγιστης πιθανότητας και η εξομάλυνση χρησιμοποιείται για αχρησιμοποίητα χαρακτηριστικά. Για την εκπαίδευση και ταξινόμηση χρησιμοποιώντας Μηχανική Μάθηση με Naïve Bayes, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη Python NLTK.

Μέγιστη Εντροπία

Στη Μέγιστη Εντροπία, δεν λαμβάνονται ενέργειες όσον αφορά τις σχέσεις μεταξύ των χαρακτηριστικών που εξάγονται από το σύνολο δεδομένων. Αυτός ο κατηγοριοποιητής προσπαθεί να μεγιστοποιεί την εντροπία υπολογίζοντας την κατάσταση συνεισφοράς από μια πολικότητα.

Η μέγιστη εντροπία χειρίζεται ακόμη τη δυνατότητα επικάλυψης και είναι η ίδια με τη μέθοδο λογιστικής παλινδρόμησης (logistic regression), η οποία βρίσκει την κατανομή σε κλάσεις. Η υπό όρους κατανομή ορίζεται ως MaxEnt και δεν κάνει υποθέσεις ανεξαρτησίας για τα χαρακτηριστικά της, σε αντίθεση με τον Naive Bayes.

Το μοντέλο αντιπροσωπεύεται από τα ακόλουθα:

$$P_{ME}(c \mid d, \lambda) = \frac{\exp[\sum_i \lambda_i f_i(c,d)]}{\sum_c \exp[\sum_i \lambda_i f_i(c,d)]}$$

Όπου c είναι η κλάση, το d είναι το tweet και λ_i είναι το διάνυσμα βάρους. Τα διανύσματα βάρους αποφασίζουν τη σημασία ενός χαρακτηριστικού στην κατηγοριοποίηση.

Support Vector Machine

Ο Support Vector Machine αναλύει τα δεδομένα εντοπίζοντας τους γραμμικούς διαχωριστές (linear separators) σε ένα χώρο αναζήτησης οι οποίοι μπορούν καλύτερα να χωρίσουν τις διαφορετικές κλάσεις. Αρχικά, εισάγονται δεδομένα δύο διανυσμάτων ιδίου μεγέθους. Στη συνέχεια τα δύο διανύσματα κατηγοριοποιούνται σε μια κλάση. Μετά βρίσκεται η διαφορά απόστασης μεταξύ των κλάσεων από κάθε έγγραφο. Η απόσταση καθορίζει τη διαφορά του κατηγοριοποιητή. Μεγιστοποιώντας τη διαφορά, μειώνουμε τα αναποφάσιστα αποτελέσματα. Ο SVM υποστηρίζει επίσης κατηγοριοποίηση και παλινδρόμηση οι οποίες είναι χρήσιμες για τη θεωρία στατιστικής μάθησης και συμβάλλει επίσης στην αναγνώριση των παραγόντων με ακρίβεια, το οποίο είναι αναγκαίο για να κατανοηθεί με επιτυχία.

4.3 Τεχνικές Κατηγοριοποίησης Συναισθήματος

Υπάρχουν δύο κύριες κατηγορίες τεχνικών για ανάλυση συναισθημάτων όπου η καθεμία χρησιμοποιεί και διαφορετικό είδος αλγορίθμων ή συνδυασμό αυτών [12]. Τεχνικές βασισμένες στο Machine Learning και οι Lexicon τεχνικές.

4.3.1 Τεχνικές βασισμένες στη Μηχανική Μάθηση

Οι Machine Learning (Μηχανική Μάθηση) τεχνικές βασίζονται σε προσέγγιση που χρησιμοποιεί τεχνική κατηγοριοποίησης για να κατατάξει ένα κείμενο σε κλάσεις. Υπάρχουν 2 τύποι Machine Learning τεχνικών:

1. **Unsupervised learning/Μάθηση χωρίς επόπτευση:** Δεν υπάρχουν εδώ κατηγορίες/κλάσεις. Βασίζεται στη συσταδοποίηση (clustering).

2. **Supervised learning/Εποπτευόμενη μάθηση:** Βασίζεται σε σύνολα δεδομένων όπου υπάρχουν ήδη οι κλάσεις, οπότε έχουμε και κατηγοριοποίηση (classification). Τα σύνολα δεδομένων εκπαιδεύονται για να γίνει πιο αποδοτική η κατηγοριοποίηση.

Η προσέγγιση Μηχανικής Μάθησης για τη συναισθηματική ανάλυση ανήκει κυρίως στην εποπτευόμενη μάθηση και κατηγοριοποίηση. Στις τεχνικές Μηχανικής Μάθησης απαιτούνται δύο σύνολα δεδομένων:

1. **Training set:** Κείμενα για εκπαίδευση

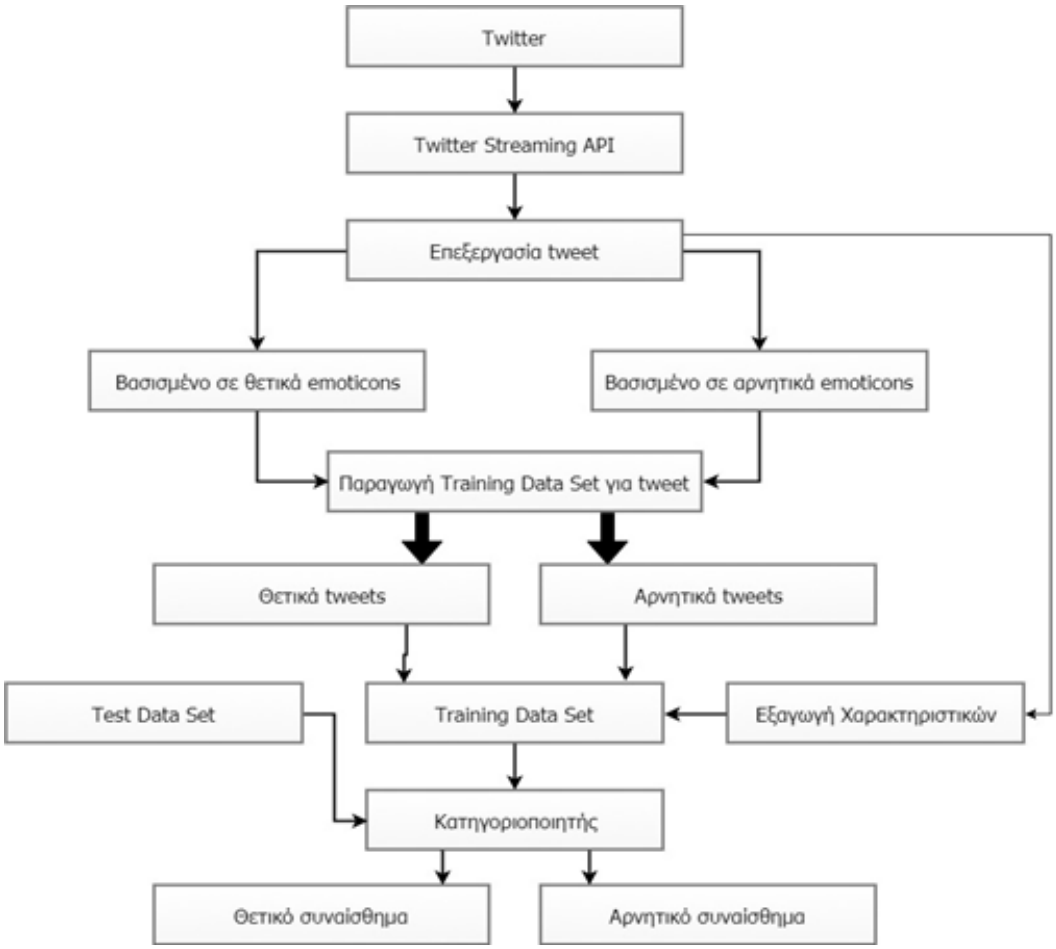
2. **Test set:** Κείμενα όπου θα γίνει ο έλεγχος

Μια σειρά από τεχνικές Μηχανικής Μάθησης έχουν σχεδιαστεί για να κατηγοριοποιήσουν εγγραφές κειμένων (tweets κ.α.) σε κατηγορίες. Τεχνικές Μηχανικής Μάθησης όπως ο Naïve Bayes, ο Support Vector Machine, η μέγιστη εντροπία, έχουν επιτύχει σε αυτό τον τομέα της συναισθηματικής ανάλυσης.

Η διαδικασία Μηχανικής Μάθησης ξεκινά συγκεντρώνοντας τα σύνολα δεδομένων εκπαίδευσης. Ύστερα εκπαιδεύει έναν κατηγοριοποιητή πάνω στα δεδομένα εκπαίδευσης. Από τη στιγμή που επιλέξουμε την τεχνική εποπτευόμενης μάθησης, είναι σημαντικό να επιλέξουμε και χαρακτηριστικό.

Οι πιο συνηθισμένες κατηγορίες χαρακτηριστικών είναι:

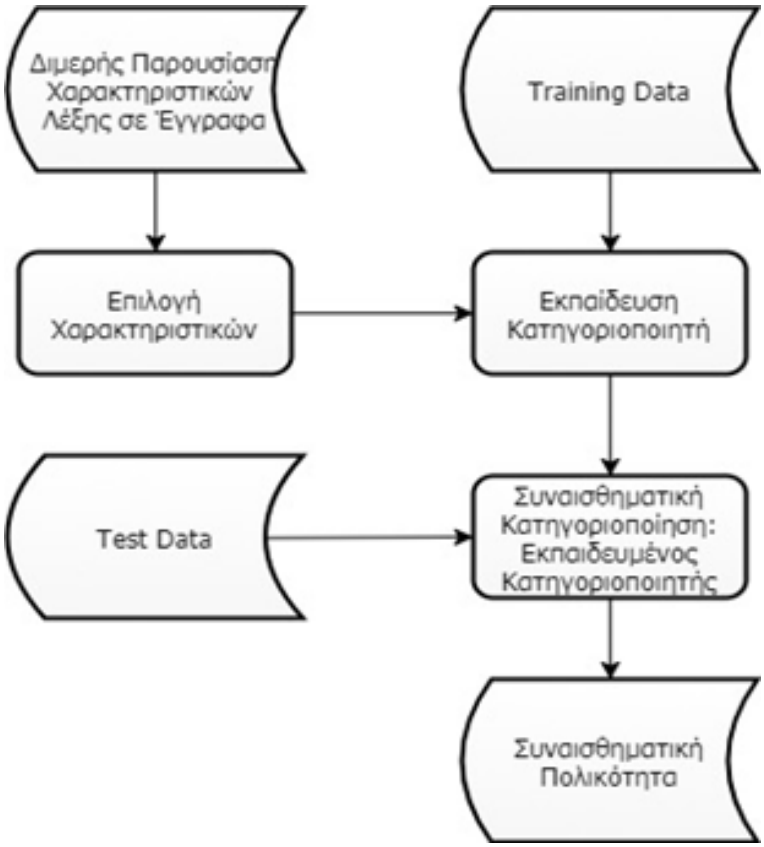
- παρουσία όρων και η συχνότητά τους,
- μέρος των πληροφοριών ομιλίας,
- αρνήσεις,
- λέξεις και φράσεις γνωμών.



Εικόνα 5 - Συναισθηματική Κατηγοριοποίηση Βασισμένη σε Emoticons

4.3.2 Τεχνικές βασισμένες σε Lexicon

Η Lexicon μέθοδος χρησιμοποιεί ένα λεξικό από συναισθήματα, με λέξεις γνωμών. Κατά τη διαδικασία, ταιριάζει τα δεδομένα που εισάγονται με τις λέξεις από το λεξικό για τον προσδιορισμό της πολικότητας. Στη συνέχεια αποδίδει βαθμολογίες συναισθήματος στις λέξεις γνωμών, περιγράφοντας έτσι πόσο θετική, αρνητική ή αντικειμενική είναι η λέξη που συγκρίνει κάθε φορά.



Εικόνα 6 - Διάγραμμα ροής τεχνικής Lexicon

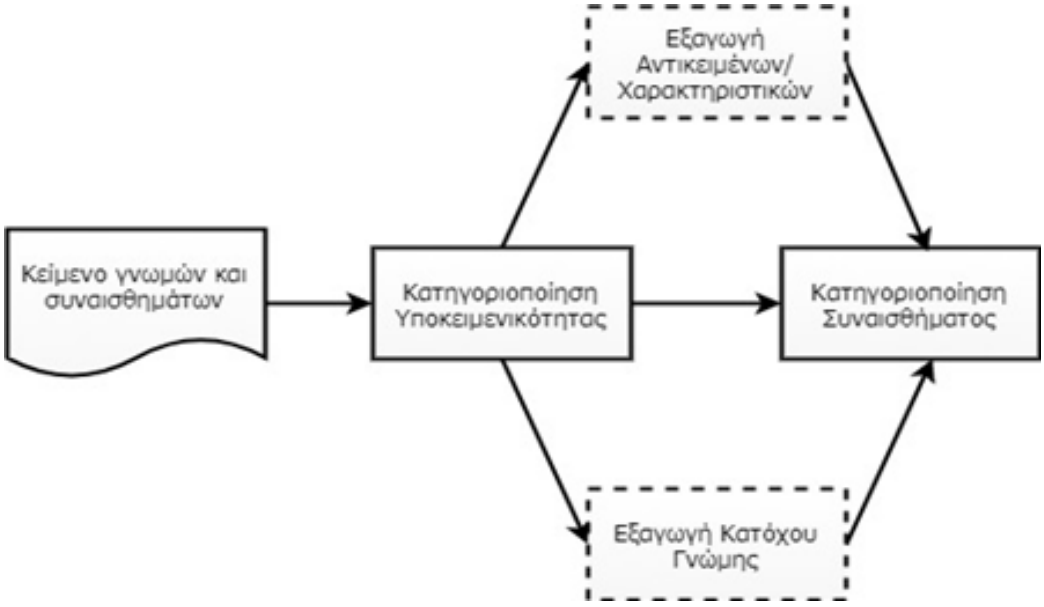
Υπάρχουν δύο υπο-κατηγορίες αυτής της προσέγγισης.

1. **Dictionary-Based:** Βασίζεται στη χρήση των όρων που συλλέγονται και σχολιάζονται χειροκίνητα. Αυτό το σύνολο μεγαλώνει με την εύρεση των συνωνύμων και αντωνύμων ενός λεξικού. Ένα τέτοιο dictionary-based είναι το WordNet. Ένα μειονέκτημά του είναι πως δεν μπορεί να διαχειριστεί τομείς και κείμενα συγκεκριμένου τομέα. Για αυτό το λόγο υπάρχει το Corpus-Based.
2. **Corpus-Based:** Έχει στόχο την παροχή λεξικών που σχετίζονται σε συγκεκριμένο τομέα. Αυτά τα λεξικά δημιουργούνται από ένα σύνολο όρων γνώμης που μεγαλώνει μέσα από την αναζήτηση των σχετικών λέξεων μέσω της χρήσης στατιστικών ή άλλων τεχνικών, όπως μέθοδοι που βασίζονται σε στατιστικά στοιχεία και μέθοδοι σημασιολογικών τεχνικών.

4.4 Διαδικασίες Ανάλυσης Συναισθήματος

Η συναισθηματική ανάλυση είναι μια διεπιστημονική πρόκληση η οποία περιλαμβάνει επεξεργασία φυσικής γλώσσας, εξόρυξη παγκόσμιου ιστού και Μηχανικής Μάθησης [12]. Είναι ένα πολύπλοκο έργο το οποίο μπορεί να αναλυθεί στις ακόλουθες διαδικασίες, δηλαδή:

- Κατηγοριοποίηση υποκειμενικότητας
- Κατηγοριοποίηση συναισθημάτων



Εικόνα 7 - Διαδικασίες Συναισθηματικής Ανάλυσης

Α. Κατηγοριοποίηση υποκειμενικότητας

Η κατηγοριοποίηση της υποκειμενικότητας είναι η διαδικασία κατηγοριοποίησης των προτάσεων ως γνωμικές ή μη γνωμικές. Δηλαδή εάν ένα κείμενο εκφράζει μια υποκειμενική γνώμη ή μια αντικειμενική.

Έστω $S = \{s_1, \ldots, s_n\}$ ένα σύνολο φράσεων στο έγγραφο D. Το πρόβλημα της κατηγοριοποίησης της υποκειμενικότητας είναι να προσδιοριστούν οι φράσεις που χρησιμοποιούνται για να αντιπροσωπεύσουν απόψεις και άλλες μορφές υποκειμενικότητας (υποκειμενικές φράσεις καθορίζονται με S_s) από τις φράσεις που χρησιμοποιήθηκαν για την αντικειμενική παρουσίαση των πραγματικών πληροφοριών (αντικειμενικές προτάσεις καθορίζονται με S_o), όπου $S_s \cup S_o = S$.

Β. Κατηγοριοποίηση Συναισθήματος

Μόλις τελειώσει η διαδικασία κατηγοριοποίηση υποκειμενικότητας, πρέπει να βρεθεί η πολικότητα της πρότασης, δηλαδή αν εκφράζει θετικό είτε αρνητικό συναίσθημα. Η κατηγοριοποίηση των συναισθημάτων μπορεί να είναι μια δυαδική κατηγοριοποίηση (θετική ή αρνητική), είτε κατηγοριοποίηση πολλαπλών κατηγοριών (εξαιρετικά αρνητική, αρνητική, ουδέτερη, θετική ή εξαιρετικά θετική), είτε παλινδρόμηση, είτε κατάταξη.

Ανάλογα με την εφαρμογή της ανάλυσης του συναισθήματος, τα υποσύνολα της εξαγωγής κατόχου γνώμης και της εξαγωγής χαρακτηριστικών αντικειμένου μπορούν να θεωρηθούν προαιρετικά. Στη συγκεκριμένη διπλωματική η κατηγοριοποίηση συναισθήματος των κειμένων περιείχε τρεις κατηγορίες: αρνητικό, ουδέτερο και θετικό συναίσθημα.

Γ. Άλλες Διαδικασίες

- **Εξαγωγή Κατόχου Γνώμης:** η ανακάλυψη των κατόχων ή των πηγών γνώμης. Η ανίχνευση του κατόχου της γνώμης είναι η αναγνώριση άμεσων ή έμμεσων πηγών γνώμης.

- **Εξαγωγή Αντικειμένων / Χαρακτηριστικών:** η ανακάλυψη μιας επιθυμητής οντότητας.

Κεφάλαιο 5 - Εγκατάσταση και εκτέλεση

5.1 Εγκατάσταση του περιβάλλοντος δοκιμών

5.1.1 Εγκατάσταση του Hadoop

Αρχικά, δημιουργούμε μια εικονική μηχανή (Virtual Machine) , το λειτουργικό σύστημα που επιλέχθηκε για την εγκατάσταση Hadoop σε αυτή τη μεταπτυχιακή εργασία είναι το Ubuntu Linux.

1) Απαιτείται η εγκατάσταση της γλώσσας προγραμματισμού Oracle Java 8, γιατί το Hadoop το χρειάζεται για την λειτουργία του.

```
#sudo apt-get install openjdk-8-jdk
```

2) Στη συνέχεια, κατεβάσαμε το hadoop-3.1.2.tar.gz και το εξάγαμε στο φάκελο "/ usr / local". Μετα μετονομάζουμε το εξαγόμενο φάκελο από hadoop-3.1.2 σε hadoop για πιο εύκολη πρόσβαση.

```
#wget http://apache.cs.utah.edu/hadoop/common/current/hadoop-3.1.2.tar.gz

#tar -xzf Hadoop-3.1.2.tar.gz

#mv Hadoop-3.1.2 hadoop
```

3) Όσον αφορά τις ρυθμίσεις , πρέπει να ενημερώσουμε το αρχείο "~ / .bashrc" με μερικές περιβαλλοντικές μεταβλητές. Έτσι, προσθέτουμε τις επόμενες γραμμές στο αρχείο:

```
export HADOOP_HOME=/ usr / local /hadoop

export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

Επίσης, στο

αρχείο "hadoop-env.sh" πρέπει να γράψουμε το μονοπάτι της java, οπότε προσθέτουμε τη γραμμή:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
```

Μετα μετα φερόμαστε στον φάκελο /usr/local/hadoop/etc/hadoop/
και επεξεργαζόμαστε το αρχείο core-site.xml και του προσθέτουμε τις παρακάτω γραμμές :

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl" href="configuration.xsl"?> <! Put sitespecific property overrides in this file. >

<configuration>

    <property>
     <name>hadoop.tmp.dir</name>
    <value>/usr/local/hadoop/tmp</value>
    <description>Parent directory for other temporary directories.</description>
    </property>
    <property>
    <name>fs.defaultFS </name>
    <value>hdfs://localhost:54310</value>
    <description>The name of the default file system. </description>
    </property>
```

Στην συνέχεια , στον ίδιο φάκελο επεξεργαζόμαστε το αρχείο hdfs.site.xml και

προσθέτουμε τις παρακάτω γραμμές:

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl" href="configuration.xsl"?> <! Put sitespecific property overrides in this file. >

<configuration>

    <property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>Default block replication.</description>
    </property>
    <property>
    <name>dfs.datanode.data.dir</name>
    <value>/usr/local/hadoop/hdfs</value>
    </property>

</configuration>
```

Και τέλος επεξεργαζόμαστε το αρχείο mapred-site.xml

και προσθέτουμε:

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl" href="configuration.xsl"?> <! Put sitespecific property overrides in this file. >

<configuration>

    <property>
    <name>mapreduce.jobtracker.address</name>
    <value>localhost:54311</value>
    <description>MapReduce job tracker runs at this host and port.
    </description>
    </property>

</configuration>
```

4) Όταν τελειώσουμε με τις ρυθμίσεις, διαμορφώνουμε το Name Node και ξεκινάμε όλους τους δαίμονες του hadoop εκτελώντας τις ακόλουθες εντολές:

```
#hdfs namenode –format

#start-dfs.sh

#start-yarn.sh
```

Τρέχοντας την εντολή "jps" στο τερματικό του NameNode, μας εμφανίζεται η λίστα με τις υπηρεσίες που τρέχουν και έτσι μπορούμε να επαληθεύσουμε αν όλα λειτουργούν όπως αναμενόταν.

```
#jps
```

5.1.2 Εγκατάσταση της MongoDB

1. Τα εργαλεία διαχείρισης πακέτων Ubuntu εξασφαλίζουν τη συνοχή και την αυθεντικότητα των πακέτων ελέγχοντας ότι έχουν υπογραφεί με κλειδιά GPG. Η ακόλουθη εντολή θα εισαγάγει το δημόσιο κλειδί GPG του MongoDB.

```
#wget -qO - https://www.mongodb.org/static/pgp/server-4.0.asc | sudo apt-key
```

2. Δημιουργούμε το αρχείο λίστας (source list) /etc/apt/sources.list.d/mongodb-org-4.0.list χρησιμοποιώντας την παρακάτω εντολή:

```
#echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
```

3. ενημερώνουμε την τοπική αποθήκη πακέτων και εγκαθιστούμε την πιο πρόσφατη σταθερή έκδοση του MongoDB

```
#sudo apt-get update

#sudo apt-get install -y mongodb-org
```

4. Το επόμενο βήμα είναι να κατεβάσουμε και να μεταφέρουμε τα αρχεία jar που χρειάζονται για να συνδεθεί το Hadoop με το MongoDB χρησιμοποιώντας τις παρακάτω εντολές:

```
#wget http://repo1.maven.org/maven2/org/mongodb/mongo-java-driver/3.9.1/mongo-java-driver-3.9.1.jar

#wget http://repo1.maven.org/maven2/org/mongodb/mongo-hadoop/mongo-hadoop-core/2.0.2/mongo-hadoop-core-2.0.2.jar

#sudo mv mongo-java-driver-3.9.1.jar /usr/local/hadoop/share/hadoop/mapreduce/

#sudo mv mongo-hadoop-core-2.0.2.jar /usr/local/hadoop/share/hadoop/mapreduce/
```

5. Και τέλος τρέχουμε την παρακάτω εντολή για να ξεκινήσει η MongoDB:

```
# sudo service mongod start
```

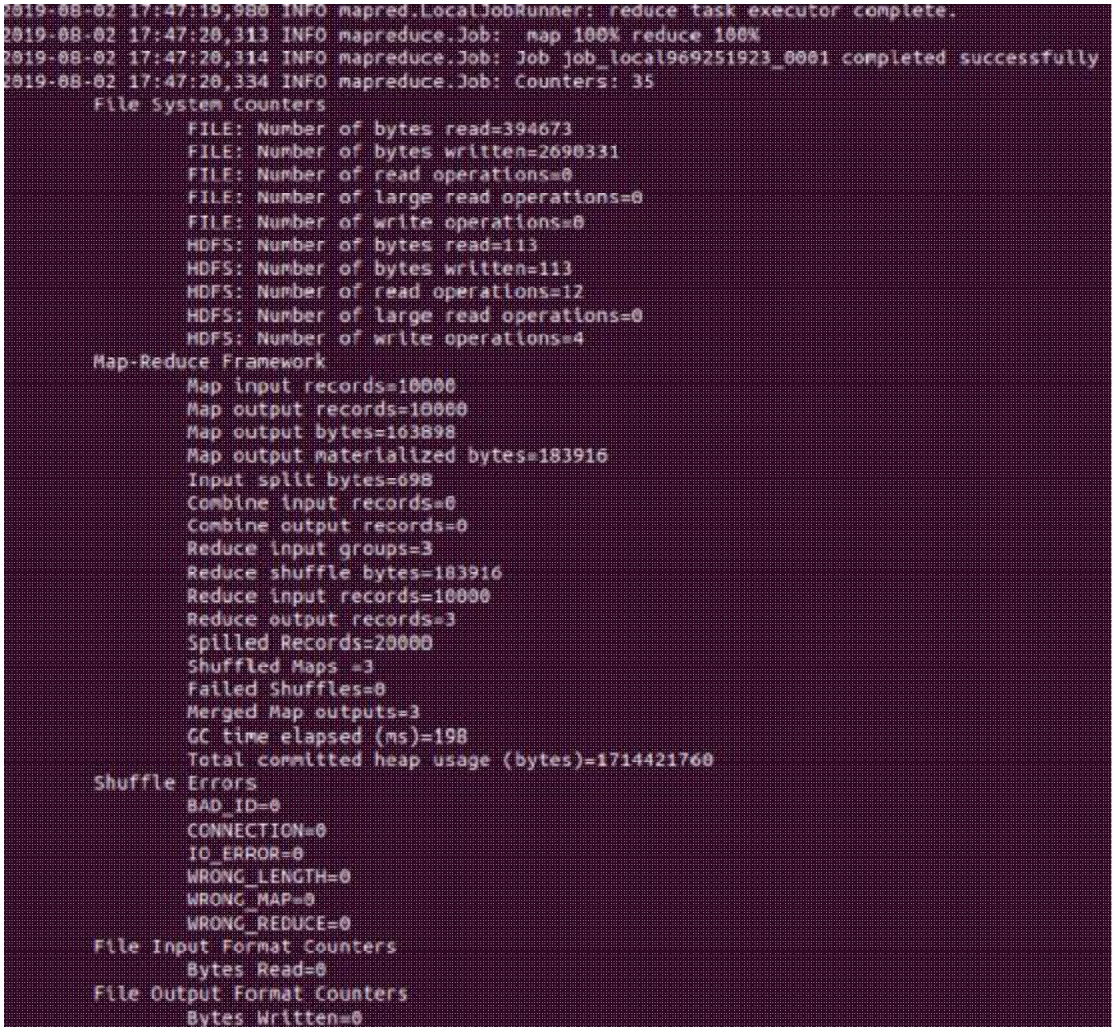
5.2 Εκτέλεση του MapReduce

Αρχικά, θα εισάγουμε στην MongoDB τρία αρχεία Json τα οποία περιέχον 10.000 κριτικές το καθένα για ταινίες. Ο σκοπός μας είναι να βρούμε τον αριθμό θετικών αρνητικών και ουδέτερων κριτικών για κάθε ένα από αυτά .

Εικόνα 8 - Στιγμιότυπο εισαγωγής δεδομένων στην MongoDB

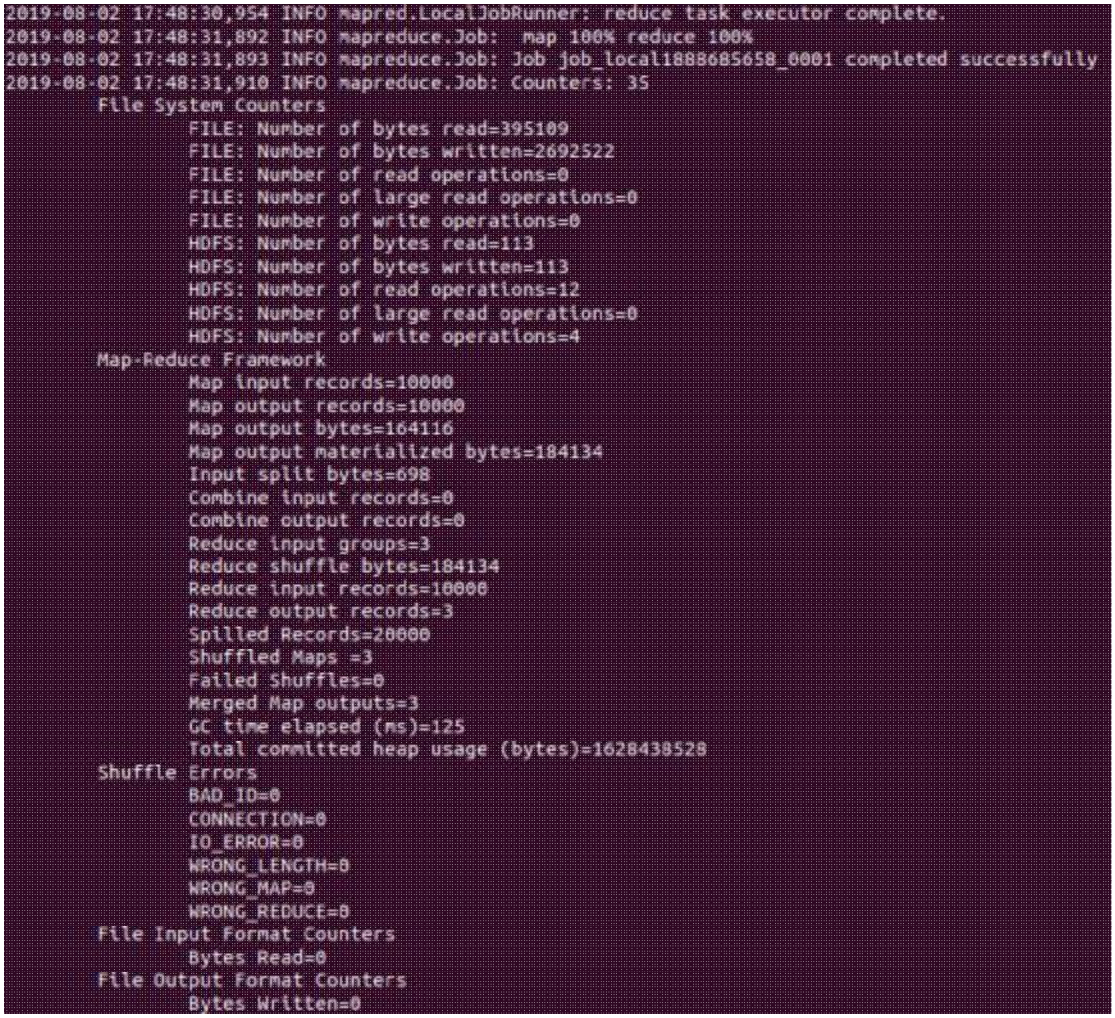
Μετα θα εκτελέσουμε το MapReduce χρησιμοποιώντας τις παρακάτω εντολές για κάθε αρχείο Json που εισάγαγε στην βάση δεδομένων:

```
# hadoop jar MapReduceHadoop.jar -Dmongo.input.split_size=8 -
Dmongo.job.verbose=true -
Dmongo.input.uri=mongodb://localhost:27017/db.mov1 -
Dmongo.output.uri=mongodb://localhost:27017/db.mov1.out
```



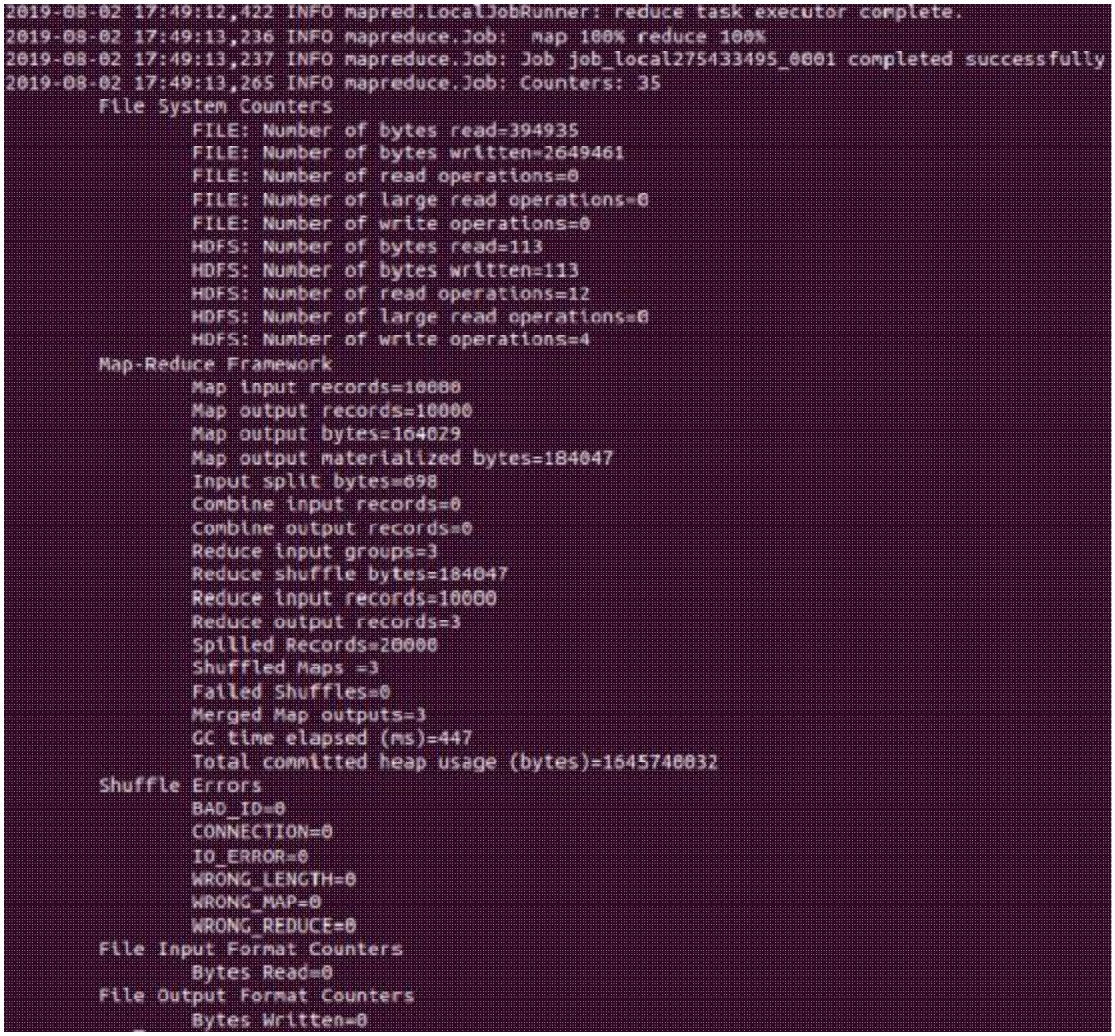
Εικόνα 9 - Στιγμιότυπο Αποτελεσμάτων ταινία 1 στο Hadoop

```
# hadoop jar MapReduceHadoop.jar -Dmongo.input.split_size=8 -
Dmongo.job.verbose=true -
Dmongo.input.uri=mongodb://localhost:27017/db.mov2 -
Dmongo.output.uri=mongodb://localhost:27017/db.mov2.out
```



Εικόνα 10 - Στιγμιότυπο Αποτελεσμάτων ταινία 2 στο Hadoop

```
# hadoop jar MapReduceHadoop.jar -Dmongo.input.split_size=8 -
Dmongo.job.verbose=true -
Dmongo.input.uri=mongodb://localhost:27017/db.mov3 -
Dmongo.output.uri=mongodb://localhost:27017/db.mov3.out
```



Εικόνα 11 - Στιγμιότυπο Αποτελεσμάτων ταινία 3 στο Hadoop

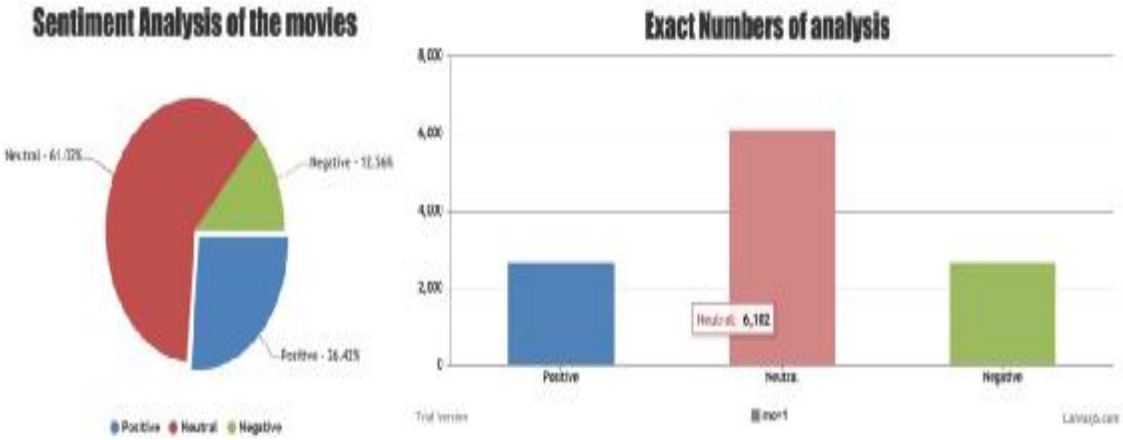
Τα αποτελέσματα από την εφαρμογή φαίνεται στο στιγμιότυπο παρακάτω.

```
> use movies
switched to db movies
> db.mov1.out.find()
{ "_id" : "Negative", "count" : 1256 }
{ "_id" : "Neutral", "count" : 6102 }
{ "_id" : "Positive", "count" : 2642 }
> db.mov2.out.find()
{ "_id" : "Negative", "count" : 1313 }
{ "_id" : "Neutral", "count" : 5884 }
{ "_id" : "Positive", "count" : 2803 }
> db.mov3.out.find()
{ "_id" : "Negative", "count" : 1326 }
{ "_id" : "Neutral", "count" : 5971 }
{ "_id" : "Positive", "count" : 2703 }
>
```

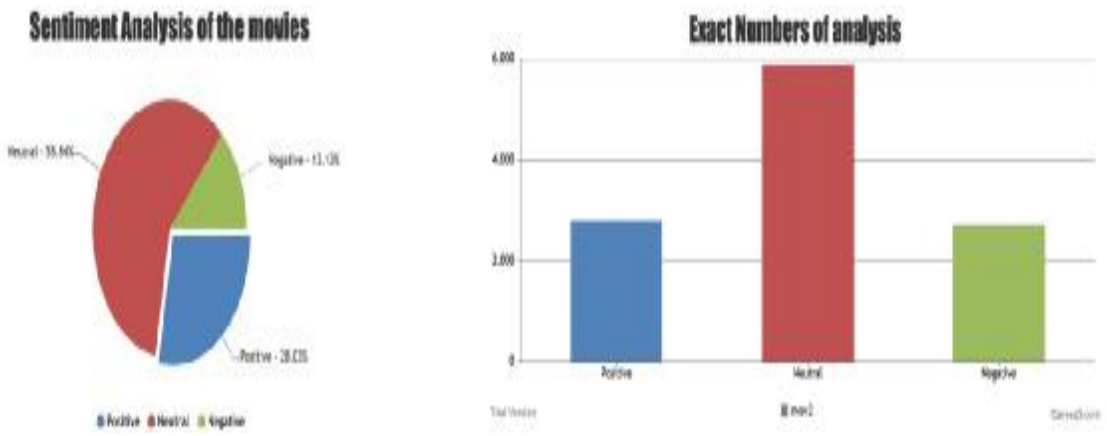
Εικόνα 12 - Στιγμιότυπο αποτελεσμάτων MapReduce στην MongoDB

Κεφάλαιο 6 - Αποτελέσματα και Συμπεράσματα

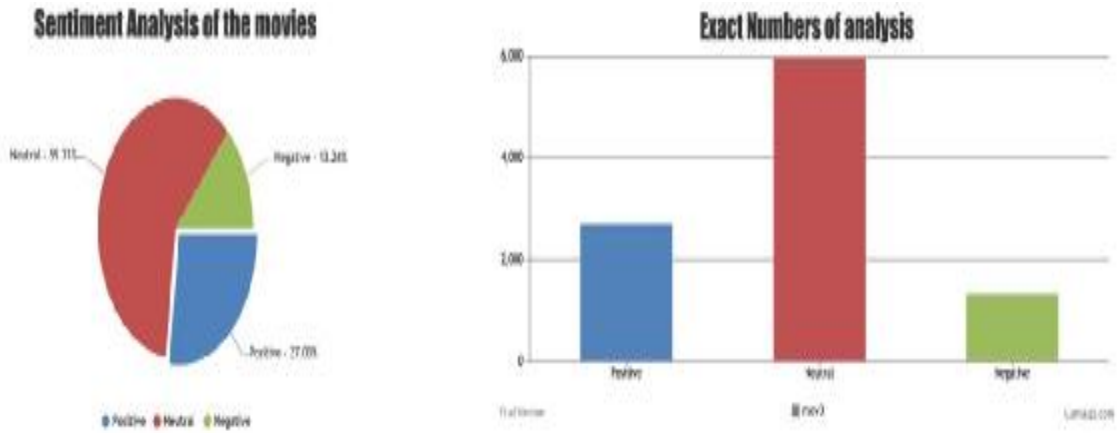
Τα αποτελέσματα από την εφαρμογή της παραπάνω υλοποίησης και ανάλυσης του οικοσυστήματος εμφανίζονται στην οθόνη του χρήστη σε μορφή δύο γραφημάτων όπως φαίνεται στα στιγμιότυπα παρακάτω.



Εικόνα 13 - Στιγμιότυπο Εμφάνισης Αποτελεσμάτων ταινία 1



Εικόνα 14 - Στιγμιότυπο Εμφάνισης Αποτελεσμάτων ταινία 2



Εικόνα 15 - Στιγμιότυπο Εμφάνισης Αποτελεσμάτων ταινία 3

Η αριστερή πλευρά απεικονίζει ένα γράφημα πίτας με ποσοστά της ανάλυσης, ενώ το δεύτερο γράφημα περιλαμβάνει ακριβής αριθμούς, που αναγράφονται όταν μετακινήσεις τον κέρσορα πάνω στην κατηγορία. Ακριβές αριθμοί υπάρχουν μέσα στη βάση δεδομένων, όπου με είσοδο το σύνολο δεδομένων σε κάθε Test Data είχαμε τελικά, για την πρώτη ταινία:

2649 θετικές κριτικές, 1259 αρνητικές και 6092 ουδέτερες.

Κατηγορία Κριτικών	Αριθμός Κριτικών
Θετικές	2.642
Ουδέτερες	6.102
Αρνητικές	1.256
Συνολικός Αριθμός	**10.000**

Πίνακας 1 - Πειραματικά Αποτελέσματα ταινία 1

Για την δεύτερη ταινία 2803 θετικές κριτικές, 2803 αρνητικές και 5884 ουδέτερες.

Κατηγορία Κριτικών	Αριθμός Κριτικών
Θετικές	2.803
Ουδέτερες	5.884
Αρνητικές	2.803
Συνολικός Αριθμός	**10.000**

Πίνακας 2 - Πειραματικά Αποτελέσματα ταινία 2

Και για την τρίτη ταινία 2703 θετικές κριτικές, 1326 αρνητικές και 5971 ουδέτερες.

Κατηγορία Κριτικών	Αριθμός Κριτικών
Θετικές	2.703
Ουδέτερες	5.971
Αρνητικές	1.326
Συνολικός Αριθμός	**10.000**

Πίνακας 3 - Πειραματικά Αποτελέσματα ταινία 3

Τελικά, η διαδικτυακή εφαρμογή κατάφερε να κάνει μια ανάλυση συναισθημάτων με χρήση ενός προτύπου συνόλου δεδομένων δομής JSON και να εξάγει αποτελεσματικά στατιστικά στοιχεία.

Συγκεκριμένα, ο ενδιαφερόμενος χρήστης εισήλθε στο σύστημα αφού πρώτα εγγράφηκε σε αυτό ώστε να ανεβάσει το σύνολο δεδομένων, το οποίο περιλαμβάνει κριτικές και απόψεις που έχουν γίνει σε συγκεκριμένες ταινίες του ιστοτόπου IMDB. Στη συνέχεια, το σύνολο δεδομένων μεταφορτώθηκε στη βάση δεδομένων επιτυχώς, μετά από έγκυρο έλεγχο του τύπου του, και, μέσα σε ορισμένα δευτερόλεπτα, αναλύθηκε με την αρωγή των διαδικασιών των τεχνολογιών και εργαλείων που συγκρότησαν το υπολογιστικό νέφος και εξήχθησαν τα συγκεκριμένα στατιστικά στοιχεία για τις γνώμες και τα συναισθήματα των χρηστών με όψη γραφημάτων.

Ο τρόπος με τον οποίο στήθηκε το οικοσύστημα επιτρέπει την εξυπηρέτηση πολλαπλών χρηστών την ίδια χρονική στιγμή χάρη στις δυνατότητες που προσφέρει το κατανεμημένο περιβάλλον και συγκεκριμένα η χρήση του Hadoop. Οι πολλαπλές εργασίες MapReduce μπορούν να αναλάβουν μεγάλο φόρτο από τις αναλύσεις, οπότε δεν υπάρχουν προβλήματα επίδοσης και λαθών στα αποτελέσματα ωστόσο το γεγονός ότι το περιβάλλον λειτουργεί σε έναν μόνο κόμβο προκύπτει πως δεν εκμεταλλεύονται οι δυνατότητες ταχύτητας που έχουν τα κατανεμημένα συστήματα.

Η συνεισφορά του οικοσυστήματος αποφέρει μια εφαρμογή κατανεμημένου συστήματος με υψηλή προοπτική για γρήγορη και εύκολη ανάλυση μεγάλων συνόλων δεδομένων, ενώ σημαντικές και χρήσιμες αποτελούν οι αναφορές σύνδεσης των τεχνολογιών του περιβάλλοντος καθώς και οι ιδέες υλοποίησης και αρχιτεκτονικής που εφαρμόστηκαν για την ολοκλήρωσή της.

Παράρτημα

Mapper

```
import com.mongodb.hadoop.io.BSONWritable;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapreduce.Mapper;
import org.bson.BSONObject;
import org.bson.types.ObjectId;
import java.io.*;

// Basic Java file IO
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.URI;

// Java classes for working with sets
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;
```

```
// Regular expression utility
import java.util.re

// File I/O
import org.apache
import org.apache
import org.apache
import org.apache

// Mapper parent c
import org.apache
import org.apache

// Wrappers for va
import org.apache
import org.apache
import org.apache
import org.apache

// Configurable co
import org.apache
import org.apache

import java.io.IOE
import java.util.St
```

```
public class Mapper extends Mapper<Object, BSONObject, Text,
DoubleWritable>
    {

    private final Text keyInt;
    private final Text review;
    private final DoubleWritable valueDouble;

    public MapperXaris() {
        super();
        keyInt = new Text();
        review = new Text();
        valueDouble = new DoubleWritable();
    }

    @Override
    @SuppressWarnings("deprecation")
    public void map(final Object pKey, final BSONObject pValue, final Context
pContext) throws IOException, InterruptedException {

        int posCounter = 0;
        int negCounter = 0;

        review.set(((String) pValue.get("review")).toString());
        String[] stringArray = review.toString().split("\\s+|,");
```

```
        for (int i = 0; i < stringArray.length; i++) {
            if (stringArray[i].equals("good")) {
                //System.out.println("WE FIND A GOOD WORD");
                posCounter++;
            }

            if (stringArray[i].equals("bad")) {
                //System.out.println("WE FIND A BAD WORD");
                negCounter++;
            }
        }

        if (posCounter>negCounter) {
            keyInt.set("Positive");
        }else if (posCounter<negCounter) {
            keyInt.set("Negative");
        }else {
            keyInt.set("Neutral");
        }

        valueDouble.set(1);
        pContext.write(keyInt, valueDouble);
    }
}
```

Reducer

```
import com.mongodb.hadoop.io.BSONWritable;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.mapreduce.Reducer;

import org.bson.BasicBSONObject;


import java.io.IOException;

import java.util.Iterator;


public class Reducer extends Reducer<Text, DoubleWritable, Text,
BSONWritable> implements org.apache.hadoop.mapred.Reducer<Text,
DoubleWritable, Text, BSONWritable> {


    private static final Log LOG = LogFactory.getLog(ReducerXaris.class);

    private BSONWritable reduceResult;

    //private final Text category;

    public int countAll;
```

```
    public ReducerXaris() {
        super();
        reduceResult = new BSONWritable();
        countAll = 0;
    }

    @Override
    public void reduce(final Text pKey, final Iterable<DoubleWritable> pValues,
final Context pContext)
        throws IOException, InterruptedException {

        int count = 0;
        double Nsum = 0;
        double Psum = 0;
        double Osum = 0;
        //category.set((pKey).toString());
        //category.set(pKey.toString());
        Text pos = new Text("Positive");
        Text neg = new Text("Negative");

        System.out.println("pKey EINAI:" + pKey);
        for (DoubleWritable val : pValues) {
            if (pos.equals(pKey)) {
                Psum += 1;
                count++;
                countAll++;
```

```
        } else if (neg.equals(pKey)) {
                Nsum += 1;
            count++;
            countAll++;

        } else{
            Osum += 1;
            count++;
            countAll++;
        }
    }

    BasicBSONObject output = new BasicBSONObject();
    output.put("count", count);
    reduceResult.setDoc(output);
    pContext.write(pKey, reduceResult);
}
@Override
public void reduce(final Text key, final Iterator<DoubleWritable> values,
                final OutputCollector<Text, BSONWritable> output,
                final Reporter reporter) throws IOException {
    int count = 0;
    double Psum = 0;
    double Nsum = 0;
```

```
    while (values.hasNext()) {
        if (key.equals("positive")) {
            Psum += 1;
            count++;
        } else{
            Nsum += 1;
            count++;
        }
    }

    final double avg = Psum / count;

    if (LOG.isDebugEnabled()) {
        LOG.debug("Average 10 Year Treasury for " + key + " was " + avg);
    }

    BasicBSONObject bsonObject = new BasicBSONObject();
    bsonObject.put("count", count);
    bsonObject.put("avg of Positives", avg);
    bsonObject.put("Positive sum", Psum);
    bsonObject.put("Negative sum", Nsum);
    reduceResult.setDoc(bsonObject);
    output.collect(key, reduceResult);
}
```

Βιβλιογραφία

[1] Lithicum David S., (2010). Cloud Computing and SOA Convergence in Your Enterprise, A Step-by-Step Guide, Pearson Education, Inc.

[2] AT Velte, TJ Velte, RC Elsenpeter, (2010). Cloud computing: a practical approach, The McGraw-Hill Companies.

[3] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges, 7–18.

[4] S., B., Jain, L., & Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). International Journal of Engineering and Information Technology, 2(1), 60–63.

[5] Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. Future Generation Computer Systems, 28(3), 583–592.

[6] Akhtar, N., Parwej, F., & Perwej, Y. (2017). A Perusal of Big Data Classification and Hadoop Technology A Perusal of Big Data Classification and Hadoop Technology, International Transaction of Electrical and Computer Engineers System, Vol. 4, No. 1, 26-38

[7] Teresa, M., Aparicio, G., Ogunyadeka, A., Younas, M., & Tuya, J. (2017). Transaction processing in consistency - aware user‘s applications deployed on NoSQL databases. Human-Centric Computing and Information Sciences, 7:7.

[8] Mauri R., A new generation of data requires next-generation systems, 2015, https://www.wired.com/insights/2015/01/a-new-generation-of-data-requires-next-generation-systems

[9] Dean, J., & Ghemawat, S. (n.d.), (2004). MapReduce: Simplified Data Processing on Large Clusters, 1–13.

[10] Abhishek, K., Verma, M. K., Shivam, K., Kumar, V., & Mohan, A. (2017). Integrated Hadoop Cloud Framework (IHCF) Integrated Hadoop Cloud Framework, (April), 0–8.

[11] Michael M, Moreira JE, Shiloach D, Wisniewski RW. (2007). Scale-up x scale-out: a case study using nutch/lucene. In: Parallel and distributed processing symposium. IPDPS 2007. New York: IEEE International.

[12] Kharde, V. A., & Sonawane, S. S. (2016). Sentiment Analysis of Twitter Data: A Survey of Techniques. International Journal of Computer Applications, 139(11), 5–15.

[13] Guanhua Wang, (2011). Improving Data Transmission in Web Applications via the Translation between XML and JSON, Communications and Mobile Computing (CMC), Third International Conference on Communications and Mobile Computing, Pages 182-185.

[14] Mapreduce: https://wiki.apache.org/hadoop/MapReduce

[15] Sentiment Analysis: https://en.wikipedia.org/wiki/Sentiment_analysis

[16] Supervised learning: https://en.wikipedia.org/wiki/Supervised_learning