Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανάλυση και Πρόβλεψη Δεδομένων Δικτυακής Κίνησης σε Κατανεμημένα Συστήματα Πραγματικού Χρόνου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΥΑΓΓΕΛΟΣ ΚΟΛΥΒΑΣ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανάλυση και Πρόβλεψη Δεδομένων Δικτυακής Κίνησης σε Κατανεμημένα Συστήματα Πραγματικού Χρόνου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΥΑΓΓΕΛΟΣ ΚΟΛΥΒΑΣ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Ιουλίου 2015.

.....	.....	.....
Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Γεώργιος Γκούμας Λέκτορας Ε.Μ.Π.	Δημήτριος Τσουμάκος Επίκουρος Καθηγητής Ιονίου Πανεπιστημίου

Αθήνα, Ιούλιος 2015

Ευάγγελος Κολυβάς

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

Copyright © Ευάγγελος Κολυβάς, 2015.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Αντικείμενο της παρούσας διπλωματικής αποτελεί η Ανάλυση και Πρόβλεψη Δεδομένων Δικτυακής Κίνησης. Το σύνολο των δεδομένων που χρησιμοποιούνται αποτελείται από πραγματικά δείγματα πακέτων τα οποία δρομολογήθηκαν μέσω του κόμβου ουδέτερης διασύνδεσης GR-IX σε διάστημα 29 εβδομάδων. Η καταγραφή αυτών των πακέτων έγινε με το εργαλείο sFlow.

Αρχικά, στόχος μας είναι η υλοποίηση ενός Κατανεμημένου Συστήματος Πραγματικού Χρόνου το οποίο θα είναι σε θέση να λαμβάνει και να επεξεργάζεται τα παραπάνω δεδομένα. Επίσης, θα μπορεί να συνδυάζει αυτές τις αρχικές πληροφορίες με ένα σύνολο δευτερευόντων πληροφοριών, ώστε με την χρήση αυτών και περαιτέρω ανάλυση, να είναι σε θέση να παράγει χρήσιμα στατιστικά στοιχεία όπως τον σημαντικότερο ISP κάθε χώρας, με βάση την εξερχόμενη και εισερχόμενη κίνηση στο δίκτυο του. Για τον σκοπό αυτό χρησιμοποιήθηκαν εργαλεία ανοιχτού κώδικα, κατανεμημένης επεξεργασίας και αποθήκευσης δεδομένων, όπως το Apache ZooKeeper, Apache Hadoop (HDFS), Apache Kafka και Apache Storm. Σε αυτό το πρώτο μέρος θα μας απασχολήσουν ζητήματα όπως η ταχύτητα επεξεργασίας των δεδομένων μας (throughput) και η κλιμακωσιμότητα του συστήματος μας.

Σε δεύτερη φάση ασχοληθήκαμε με την εξόρυξη γνώσης (data mining) από τα δεδομένα μας και την μηχανική μάθηση (machine learning). Συγκεκριμένα, θα προσπαθήσουμε να προβλέψουμε την ωραίας κίνηση (throughput) του δικτύου μας (GR-IX) με βάση τα στοιχεία που διαθέτουμε ως ιστορικό. Για αυτό τον σκοπό θα χρησιμοποιήσουμε το εργαλείο ανοιχτού κώδικα, Weka. Στόχος μας σε αυτό το μέρος είναι η σωστή επιλογή αλγορίθμου κατηγοριοποίησης (classification) για όσο το δυνατόν ακριβέστερη πρόβλεψη.

Λέξεις κλειδιά

sFlow, Apache, ZooKeeper, HDFS, Kafka, Storm, Weka, Network Analytics, Throughput Prediction

Abstract

The object of this thesis is the Analysis and Prediction of Networking Data Traffic. All data we used, come from samples of packets which are routed throughout a large IXP (GR-IX) on a period of 29 weeks. Sample packets were captured by the network monitoring tool, sFlow.

Firstly, our goal is the design and development of a distributed realtime computation system which it will be able to receive and process these data. Moreover, it will be able to combine these data with other, secondary sources of data, in order to produce a batch of useful results, such as the greatest ISP per country, based on the outgoing and incoming traffic at his network. For this goal, open source tools for distributed processing and storage were used, like Apache ZooKeeper, Apache Hadoop (HDFS), Apache Kafka and Apache Storm. In this first part, we care about the speed of processing data (throughput) and scalability of our system.

Secondly, we are concerned with data mining and machine learning. More precisely, we want to predict the hourly traffic (throughput) of our network (GR-IX). For this goal, open source software, Weka, was used. In this part we care most about picking the right algorithm of classification, in order to have the most accurate prediction.

Keywords

sFlow, Apache, ZooKeeper, HDFS, Kafka, Storm, Weka, Network Analytics, Throughput Prediction

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον Καθηγητή Ε.Μ.Π. κύριο Νεκτάριο Κοζύρη για την εμπιστοσύνη που μου έδειξε δίνοντας μου την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον και σύγχρονο θέμα στα πλαίσια της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον μεταδιδακτορικό ερευνητή κύριο Ιωάννη Κωνσταντίνου για την βοήθεια και την καθοδήγηση του κατά την εκπόνηση της παρούσας εργασίας.

Ακόμα, θα ήθελα να ευχαριστήσω όλη την ομάδα του Εργαστηρίου Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, και ιδιαίτερα τον Δημήτριο Σαρλή, η συμβολή του οποίου ήταν καθοριστική, ιδιαίτερα στα πρώτα στάδια αυτής της διπλωματικής.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για την υπομονή και την αμέριστη στήριξη που μου παρείχαν όλα αυτά τα χρόνια, στην επιλογή μου για απόκτηση δεύτερου πτυχίου.

Ευάγγελος Κολυβάς,
Αθήνα, 29 Ιουλίου 2015

Περιεχόμενα

Κεφάλαιο 6: Data Mining & Πρόβλεψη

Κεφάλαιο 7: Επίλογος

Παράρτημα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1
Εισαγωγή

1.1 Κίνητρο

Οι κόμβοι ουδέτερης διασύνδεσης (Internet Exchange Points - IXPs) είναι φυσικός εξοπλισμός (δρομολογητές - routers, μεταγωγείς – switches, κλπ) που επιτρέπει την άμεση διασύνδεση παρόχων υπηρεσιών Internet (Internet Service Providers – ISPs, π.χ. O.T.E., Forthnet, hellas online) με σκοπό την ανταλλαγή κίνησης δεδομένων Internet μεταξύ των δικτύων τους (Autonomous Systems - AS). Στην Ελλάδα, ο κόμβος GR-IX διασυνδέει όλους τους ελληνικούς ISPs: με αυτό τον τρόπο, η επικοινωνία μεταξύ δυο ελληνικών ISPs γίνεται απευθείας μέσω του GR-IX χωρίς να απαιτείται η δρομολόγηση των πακέτων μεταξύ ενός τρίτου δικτύου που βρίσκεται π.χ. στο εξωτερικό.

Προφανώς, ένα IXP δρομολογεί κίνηση τάξης μεγέθους μεγαλύτερη σε σχέση με έναν συγκεκριμένο ISP. Πολλές σύγχρονες μελέτες [13] έχουν δείξει ότι η δειγματοληπτική ανάλυση της κίνησης ενός IXP σε ένα βάθος χρόνου μερικών εβδομάδων μπορεί να εξάγει ενδιαφέροντα συμπεράσματα όχι μόνο για τα συγκεκριμένα AS που διασυνδέει, αλλά και για την κατάσταση ολόκληρου του διαδικτύου. Χρησιμοποιώντας το εργαλείο δειγματοληψίας και καταγραφής πακέτων διαδικτύου sFlow [3] (sampled flow), οι προηγούμενες μελέτες συγκέντρωσαν ένα δείγμα των πακέτων που δρομολογήθηκαν. Η ανάλυση των δειγμάτων απέδειξε ότι ένα IXP έχει τέλεια “ορατότητα” του συνολικού διαδικτύου, καθώς μέσα από αυτό περνάει κίνηση προς όλα σχεδόν τα υπάρχοντα AS και για όλα τα προθέματα δημόσιων δικτύων.

Οι προηγούμενες μελέτες χρησιμοποίησαν παραδοσιακές τεχνικές επεξεργασίας κειμένου δεδομένων σε συγκεντρωμένα αρχεία καταγραφής (log files): Τα δεδομένα αποθηκεύονται σε αρχεία κειμένου στο σύστημα αρχείων (file system) ενός υπολογιστή, και κατόπιν προγράμματα-scripts που εξάγουν την απαραίτητη πληροφορία (π.χ. unique IPs, unique AS, χώρες παραληπτών/αποστολέων, ταξινομημένες λίστες με βάση τον αριθμό εμφανίσεων, κλπ) εκτελούνται στον υπολογιστή αυτό. Είναι προφανές ότι η ισχύς του υπολογιστή, που εκτελεί την ανάλυση, περιορίζει τις δυνατότητες κλιμάκωσης της επεξεργασίας σε μεγαλύτερο αριθμό δεδομένων: σε αυτή την περίπτωση, ο χρόνος της ανάλυσης είναι ανάλογος του μεγέθους των δεδομένων (π.χ. διπλάσια δεδομένα

χρειάζονται διπλάσιο χρόνο για να αναλυθούν). Επίσης, το μέγιστο δυνατό μέγεθος των δεδομένων προς ανάλυση είναι φραγμένο από τις δυνατότητες του συγκεκριμένου υπολογιστή (συνήθως επηρεάζεται από το μέγεθος της συνολικής φυσικής μνήμης του μηχανήματος).

Έχει παρατηρηθεί ότι τα δεδομένα που παράγονται τα τελευταία χρόνια έχουν αυξηθεί δραματικά [1]. Δεδομένα π.χ. από κοινωνικά δίκτυα, χρηματιστηριακές κινήσεις, αναζητήσεις χρηστών στο διαδίκτυο, μετρήσεις επιστημονικών οργάνων όπως μετεωρολογικοί σταθμοί ή ακολουθίες DNA παράγονται με πολύ γρήγορους ρυθμούς. Η γρήγορη και αποδοτική επεξεργασία τέτοιων δεδομένων είναι απαραίτητη για την εξαγωγή χρήσιμων συμπερασμάτων, ανάλογα το εκάστοτε πεδίο: η ανάλυση π.χ. δεδομένων από κοινωνικά δίκτυα ή αναζητήσεις χρηστών μπορεί να εμφανίσει τις προτιμήσεις τους σχετικά με προϊόντα ή υπηρεσίες.

Ως εκ τούτου, οι τεχνικές ανάλυσης μεγάλου όγκου δεδομένων (big data analytics) έχουν γνωρίσει ιδιαίτερη άνθηση [14, 15, 16]. Οι τεχνικές αυτές βασίζονται σε κατανεμημένες προσεγγίσεις, όπου συστοιχίες υπολογιστών χρησιμοποιούνται συνεργατικά για την επεξεργασία του όγκου των δεδομένων, με σκοπό την εύκολη κλιμάκωση της υποδομής καθώς αυξάνονται τα δεδομένα. Τεχνικές όπως το μοντέλο MapReduce της Google χρησιμοποιούνται κατά κόρον για τον σκοπό αυτό. Η υλοποίηση του MapReduce στο πρόγραμμα Apache Hadoop είχε σαν αποτέλεσμα την δημιουργία ενός οικοσυστήματος προγραμμάτων για την διεξαγωγή big-data analytics.

Τα προγράμματα αυτά συνήθως εκτελούνται σε εικονικές συστοιχίες υπολογιστών χρησιμοποιώντας τεχνολογίες cloud computing (π.χ. Amazon Elastic Compute Cloud) που επιτρέπουν το αυτόματο “στήσιμο” της υποδομής για την εκτέλεση των εργασιών και τον αυτόματο τερματισμό της υποδομής μετά το πέρας της επεξεργασίας (όπως η υπηρεσία Amazon Elastic MapReduce).

Μια ακόμα νεότερη εκδοχή των παραπάνω τεχνικών είναι η ανάλυση των δεδομένων εισόδου σε πραγματικό χρόνο και διεξαγωγή έγκαιρων συμπερασμάτων. Αυτή η δυνατότητα μπορεί να φανεί χρήσιμη σε ένα μεγάλο πλήθος εφαρμογών, όπως ο έγκαιρος εντοπισμός και αντιμετώπιση απρόοπτων ή ανεπιθύμητων συμβάντων. Ένα κλασικό παράδειγμα αποτελεί ο εντοπισμός DDoS/DoS attack (Distributed/Denial-of-Service attack, επιθέσεις άρνησης παροχής υπηρεσίας) κατά την οποία ένας ή περισσότεροι υπολογιστές σκοπεύουν εναντίον ενός τρίτου υπολογιστή, ή μιας υπηρεσίας που παρέχεται, με σκοπό να καταστήσουν τον υπολογιστή ή την υπηρεσία ανίκανη να δεχτεί άλλες συνδέσεις και έτσι να μην μπορεί να εξυπηρετήσει άλλους πιθανούς πελάτες.

Η έννοια της εξόρυξης γνώσης από δεδομένα (data mining) αναφέρεται στην εξεύρεση (ενδιαφέρουσας, χρήσιμης, αυτονόητης, ή μη προφανής) πληροφορίας ή προτύπων, από μεγάλες βάσεις δεδομένων, με χρήση αλγορίθμων ομαδοποίησης ή κατηγοριοποίησης

(classification) και των αρχών της στατιστικής, της τεχνητής νοημοσύνης, της μηχανικής μάθησης (machine learning) και των συστημάτων βάσεων δεδομένων. Στόχος της εξόρυξης γνώσης είναι η πληροφορία που θα εξαχθεί και τα πρότυπα που θα προκύψουν να έχουν δομή κατανοητή προς τον άνθρωπο έτσι ώστε να τον βοηθήσουν να πάρει τις κατάλληλες αποφάσεις. Πολλές φορές αυτή η εξαχθείσα πληροφορία χρησιμοποιείται για την πρόβλεψη (prediction), δειγμάτων του ίδιου πληθυσμού που θα ακολουθήσουν ή των χαρακτηριστικών ενός πανομοιότυπου πληθυσμού.

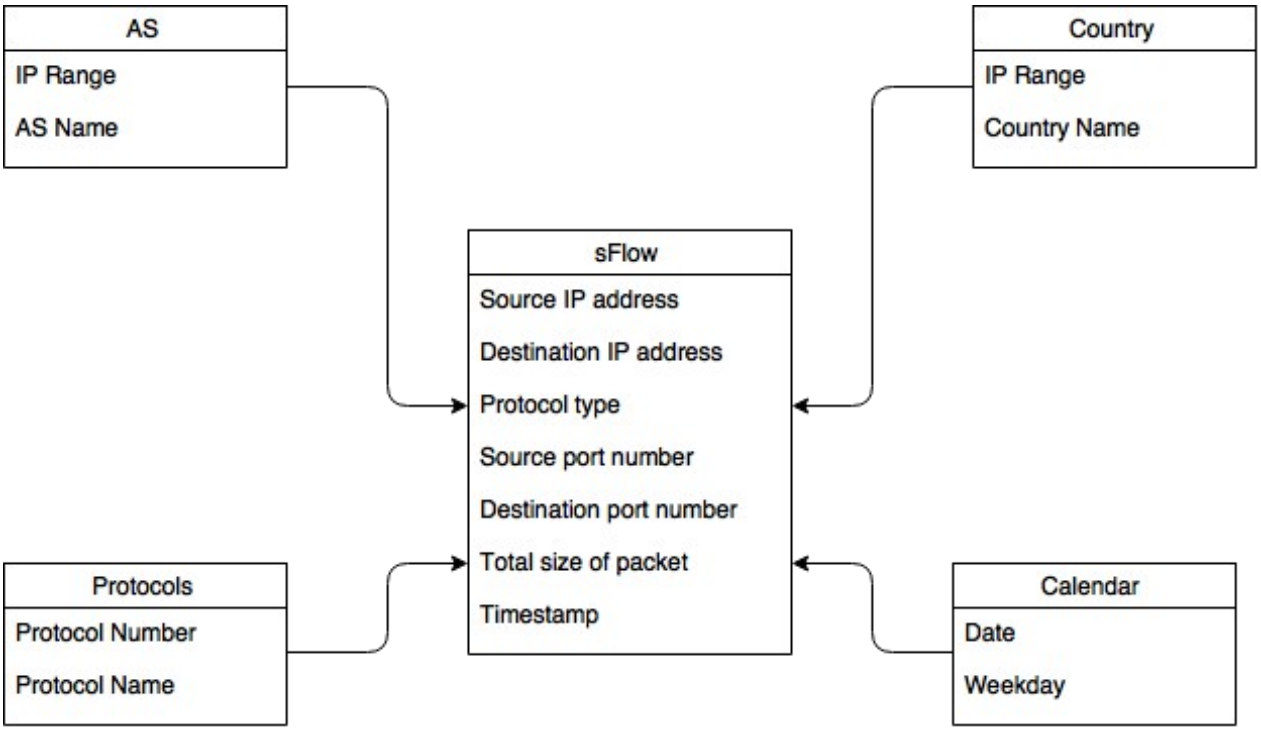
Το πεδίο της εξόρυξης γνώσης από δεδομένα γνωρίζει τρομερή άνθιση στις μέρες μας, τόσο σε ερευνητικό επίπεδο, όσο και σε επίπεδο πρακτικής εφαρμογής. Ο συνδυασμός του με κατανεμημένες τεχνικές επεξεργασίας δεδομένων, μας επιτρέπει την παράλληλη επεξεργασία μεγάλου όγκου δεδομένων (εκατοντάδων εκατομμυρίων δειγμάτων) σε μερικά μόλις λεπτά. Στην περίπτωση δε των κατανεμημένων συστημάτων πραγματικού χρόνου, αυτή πληροφορία μπορεί να ανανεώνεται (και ανάλογα να ισχυροποιείται ή να καταρρίπτεται) με βάση τα δείγματα τα οποία εισέρχονται συνεχώς στο σύστημα μας.

1.2 Σκοπός

Σκοπός της παρούσας διπλωματικής είναι η σχεδίαση και υλοποίηση ενός συστήματος επεξεργασίας μεγάλου όγκου δεδομένων που έχουν τη μορφή ενός κλασικού σχήματος σε διάταξη αστέρα (star schema), όπως για παράδειγμα είναι τα δεδομένα που προέρχονται από log files και δευτερεύουσες πληροφορίες που ενισχύουν αυτά τα δεδομένα με επιπλέον γνωρίσματα. Μία ειδική περίπτωση εφαρμογής αυτού του μοντέλου δεδομένων είναι η ανάλυση πραγματικών δεδομένων κίνησης ενός IXP. Στην περίπτωση αυτή τα ανωνυμοποιημένα δεδομένα που προέρχονται από τον ελληνικό κόμβο GR-IX και καλύπτουν ένα χρονικό διάστημα μερικών εβδομάδων, είναι ο λεγόμενος fact table (πίνακας γεγονότων), ο οποίος βρίσκεται στο κέντρο της διάταξης αστέρα. Όλες οι δευτερεύουσες πληροφορίες, που αφορούν για παράδειγμα τις διευθύνσεις IP των πακέτων που εξετάζουμε, αποτελούν τους λεγόμενους dimension tables, οι οποίοι περιβάλλουν τον fact table και μας παρέχουν επιπλέον πληροφορίες. Στο σχήμα 1.1 φαίνεται εποπτικά το μοντέλο δεδομένων που επιθυμούμε να εξετάσουμε.

Αρχικά, στόχος μας είναι χρησιμοποιώντας τεχνικές κατανεμημένης επεξεργασίας δεδομένων σε πραγματικό χρόνο, με κύριο εργαλείο το ανοιχτό λογισμικό του Apache Storm, να δημιουργήσουμε ένα σύστημα το οποίο θα επεξεργάζεται αυτά τα δεδομένα. Συγκεκριμένα, ενδιαφερόμαστε να εντοπίζουμε τον σημαντικότερο ISP κάθε χώρας, με βάση την εξερχόμενη και εισερχόμενη κίνηση στο δίκτυο του.

Επιπλέον, προσπαθούμε να δώσουμε μια εκτίμηση – πρόβλεψη για το μέγεθος της ωριαίας κίνησης μέσω του δικτύου του GR-IX (throughput), με βάση τα μέχρι στιγμής στοιχεία της καταγραφείσας κίνησης, με χρήση εργαλείων εξόρυξης δεδομένων (data mining), όπως το ανοιχτό λογισμικό του Weka.



Σχήμα 1.1: Μοντέλο Δεδομένων

1.3 Οργάνωση κειμένου

Στο Κεφάλαιο 2 παρουσιάζουμε εργαλεία και τεχνικές κατανεμημένης επεξεργασίας δεδομένων τα οποία επεξεργάζονται δεδομένα με λογική Batch Processing. Τα εργαλεία αυτά είναι χτισμένα πάνω στο Hadoop και είναι μέλη του ευρύτερου οικοσυστήματος στο οποίο εντάσσεται. Σε περίπτωση του Batch Processing όλα μας τα δεδομένα πρέπει να βρίσκονται σε τελική μορφή στο κατανεμημένο σύστημα αρχείο HDFS, και η επεξεργασία έστω κάποιου μικρού μέρους από αυτά, απαιτεί την ανάγνωση ολόκληρου του dataset.

Στο Κεφάλαιο 3 παρουσιάζουμε λεπτομερώς ένα εργαλείο κατανεμημένης επεξεργασίας δεδομένων για συστήματα πραγματικού χρόνου, το Storm. Το Storm σε αντίθεση με τα εργαλεία του Κεφαλαίου 2, δεν πραγματοποιεί διεργασίες MapReduce αλλά κατασκευάζει τοπολογίες (γράφους) των οποίων οι κόμβοι επεξεργάζονται ροές δεδομένων. Οι εργασίες αυτές επιτελούνται in memory (δηλαδή εντός φυσικής μνήμης).

Επικουρικό ρόλο στη παραπάνω διαδικασία έχουν τα εργαλεία ZooKeeper και Kafka, τα οποία βοηθούν στον συγχρονισμό των εργασιών και στην διαχείριση μηνυμάτων αντίστοιχα. Τα εργαλεία αυτά παρουσιάζονται στο Κεφάλαιο 4.

Στο Κεφάλαιο 5 παρουσιάζουμε το cluster που θα εκτελέσουμε το πείραμα μας, την μορφή που έχουν τα δεδομένα μας και η δευτερεύουσα πληροφορία, και πως συσχετίζονται τα δύο τελευταία μεταξύ τους. Στην συνέχεια γίνεται περιγραφή της τοπολογίας που επιθυμούμε να υλοποιήσουμε καθώς και των αποτελεσμάτων τα οποία εκείνη μας προσφέρει.

Στο Κεφάλαιο 6 παρουσιάζουμε το θεωρητικό υπόβαθρο γύρω από την εξόρυξη γνώσης από δεδομένα και τους βασικούς αλγόριθμους κατηγοριοποίησης. Παράλληλα, κάνουμε μία σύντομη αναφορά στο λογισμικό του Weka και εκτελούμε με την βοήθεια του, το πείραμα της πρόβλεψης της δικτυακής κίνησης του δικτύου GR-IX.

Στο Κεφάλαιο 7 γίνεται μία ανασκόπηση του τι είδαμε στα πλαίσια της παρούσας διπλωματικής και συνοψίζονται τα τελικά αποτελέσματα και σχόλια επί των πειραμάτων. Ακόμα, παρουσιάζονται τυχόν επεκτάσεις της παρούσας δουλειάς σε διαφορετικές κατευθύνσεις.

Κεφάλαιο 2
Batch Processing

2.1 MapReduce

Το MapReduce είναι ένα προγραμματιστικό μοντέλο που παρουσιάστηκε από τη Google [11]. Δημιουργήθηκε για την ανάπτυξη εφαρμογών οι οποίες επεξεργάζονται γρήγορα και παράλληλα τεράστιες ποσότητες δεδομένων. Διευκολύνει την κατανεμημένη επεξεργασία τεραστίων ποσοτήτων δεδομένων, χρησιμοποιώντας ένα μεγάλο αριθμό από υπολογιστές. Οι υπολογιστές αυτοί βρίσκονται σε συστοιχίες, είναι συνδεδεμένοι μεταξύ τους μέσω δικτύου (π.χ. Ethernet) και αποτελούν το λεγόμενο cluster υπολογιστών.

Η προγραμματιστική ιδέα που βρίσκεται πίσω από την υλοποίηση του MapReduce βασίζεται στην λογική του διαίρει και βασίλευε (divide and conquer): Το πρόβλημα διαιρείται σε μικρότερα υποπροβλήματα και στη συνέχεια οι λύσεις τους συνδυάζονται για να προκύψει η λύση του αρχικού προβλήματος. Το σχεδιαστικό μοντέλο για την υλοποίηση του MapReduce βασίζεται στην αρχιτεκτονική Master-Slave, όπου ένας κεντρικός κόμβος (Master) είναι αυτός που αναθέτει εργασίες στους κόμβους εργάτες (Slaves).

Η είσοδος ανεβαίνει στο κατανεμημένο σύστημα αρχείων (DFS – Distributed File System) και χωρίζεται σε M κομμάτια, μεγέθους 16 έως 64MB. Κάθε κομμάτι περιέχει ζεύγη εγγραφών <key, value>. Κάθε μηχάνημα που συμμετέχει στον υπολογισμό εκτελεί ένα αντίγραφο του προγράμματος σε ένα κομμάτι δεδομένων. Ένα από όλα τα μηχανήματα αναλαμβάνει το ρόλο του Master. Αυτός αναθέτει εργασίες στα υπόλοιπα (εργάτες). Αυτές μπορεί να είναι map ή reduce εργασίες. Ο master διατηρεί δομές δεδομένων όπως: κατάσταση μίας εργασίας, τοποθεσίες των δεδομένων εισόδου, εξόδου και ενδιάμεσων αποτελεσμάτων. Ο master είναι υπεύθυνος για το χρονοπρογραμματισμό της εκτέλεσης των εργασιών.

Το πρόβλημα χωρίζεται σε δύο φάσεις, την φάση Map και την φάση Reduce. Σε κάθε ένα Slave ανατίθεται ο ρόλος του Mapper ή του Reducer, ανάλογα αν η δουλειά που αναλαμβάνει να φέρει εις πέρας βρίσκεται στην πρώτη ή στην δεύτερη φάση. Ο προγραμματιστής έχει ορίσει εκ των προτέρων τις συναρτήσεις Map και Reduce οι οποίες εκτελούνται σε κάθε φάση και το σύστημα αναλαμβάνει αυτόματα την παραλληλοποίηση και την εκτέλεση της εφαρμογής στο cluster.

- **Φάση Map:** Ένας εργάτης που του έχει ανατεθεί μία map εργασία, διαβάζει από το GFS τα κομμάτια της εισόδου (input split) που του αντιστοιχούν. Τα κομμάτια αυτά είναι μη αλληλο-επικαλυπτόμενα. Τα αναλύει σε ζεύγη <key, value> και τα δίνει σαν είσοδο στη map συνάρτηση. Η map συνάρτηση επεξεργάζεται τα ζεύγη και παράγει ενδιάμεσα ζεύγη και τα συσσωρεύει στη μνήμη. Περιοδικά εκτελείται μία συνάρτηση διαίρεσης (partition function). Αυτή αποθηκεύει τα ενδιάμεσα ζεύγη στον τοπικό δίσκο. Επιπλέον τα χωρίζει σε R ομάδες. Η συνάρτηση αυτή μπορεί να προσδιοριστεί από τον χρήστη. Όταν η συνάρτηση διαίρεσης ολοκληρώσει την αποθήκευση των ζευγών ενημερώνει τον master για το που βρίσκονται τα δεδομένα. Ο master προωθεί αυτή την πληροφορία στους εργάτες που εκτελούν reduce εργασίες.

- **Φάση Reduce:** Ένας εργάτης που του έχει ανατεθεί μία reduce εργασία, διαβάζει τα ζεύγη που του αντιστοιχούν από τις τοποθεσίες που του υποδεικνύει ο master. Όταν όλα τα ενδιάμεσα ζεύγη έχουν ανακτηθεί ταξινομούνται βάσει του key. Όσα values έχουν κοινό key ομαδοποιούνται. Εκτελείται η συνάρτηση reduce με είσοδο τα ζεύγη <key, group_of_values> που προέκυψαν στην προηγούμενη φάση. Η reduce επεξεργάζεται τα δεδομένα εισόδου και παράγει τα τελικά ζεύγη. Τα ζεύγη εξόδου προσαρτώνται σε ένα αρχείο στο τοπικό σύστημα αρχείων. Όταν ολοκληρωθεί η reduce το αρχείο γίνεται διαθέσιμο στο κατανεμημένο σύστημα αρχείων.

Και με σύμβολα:

$$Map(k1, v1) \rightarrow list(k2, v2)$$

$$Reduce(k2, list(v2)) \rightarrow list(v3)$$

Όταν ένας εργάτης ολοκληρώσει την εργασία του ενημερώνει τον master. Όταν όλοι ενημερώσουν τον master τότε αυτός επιστρέφει τη λειτουργία στο αρχικό πρόγραμμα του χρήστη.

Ανοχή στα σφάλματα:
Ο master επικοινωνεί με τους εργάτες περιοδικά. Εάν κάποιος δεν ανταποκριθεί για ένα χρονικό διάστημα τότε αναθέτει την εργασία του σε κάποιον άλλο. Ο master επιπλέον περιοδικά διατηρεί αντίγραφα ελέγχου των δομών του. Σε περίπτωση βλάβης ένας νέος μπορεί να ξεκινήσει άμεσα. Τα ενδιάμεσα αποτελέσματα που παράγονται από τις map και reduce εργασίες διατηρούνται σε προσωρινά αρχεία σε τοπικά συστήματα αρχείων έως όλη η είσοδος να έχει επεξεργαστεί. Στη συνέχεια ενημερώνεται ο master και η πληροφορία γίνεται διαθέσιμη σε όλους.

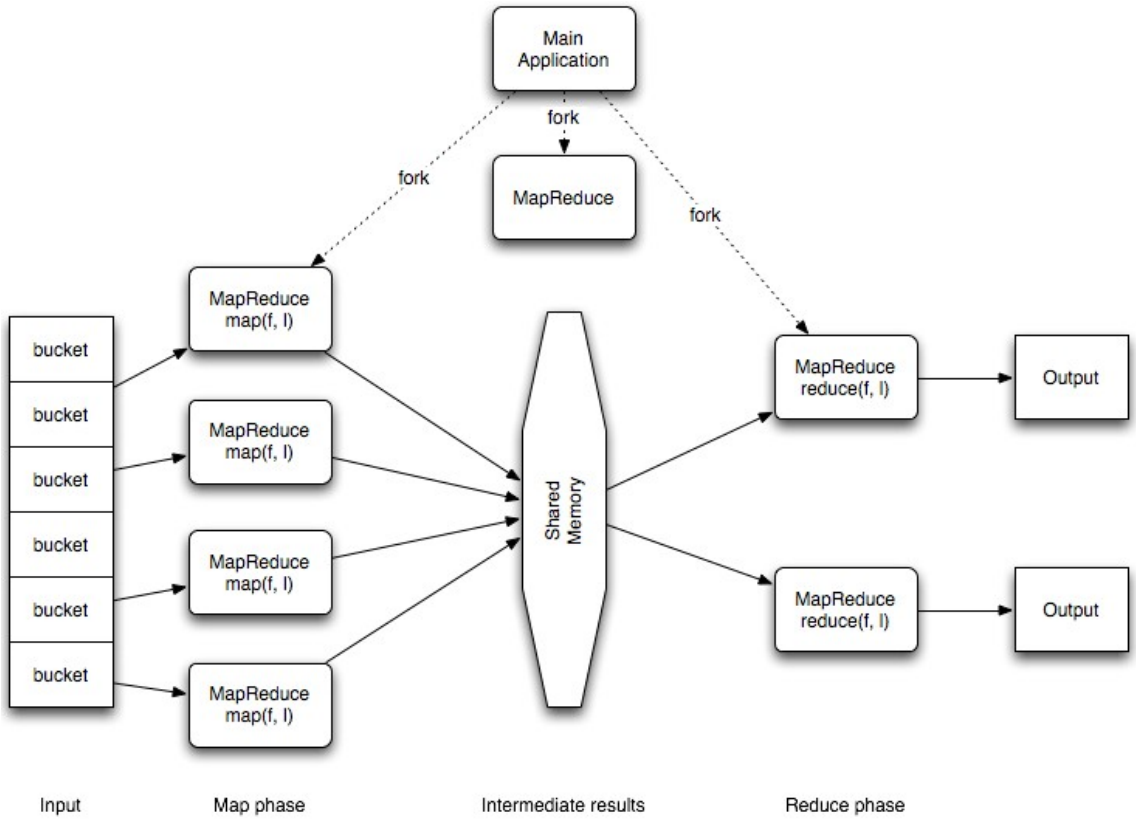
Τοπικότητα :
Τα δεδομένα αποθηκεύονται στους δίσκους των εργατών. Χωρίζονται σε block (64MB συνήθως) με αντίγραφα σε άλλους εργάτες. Move computation near the data: Ο master προσπαθεί να εκτελέσει μία εργασία σε ένα εργάτη “κοντά” στα δεδομένα εισόδου, ώστε να μειωθεί το εύρος δικτύου που θα καταναλωθεί.

Διακριτότητα εργασιών:
Ο αριθμός των προς εκτέλεση εργασιών είναι συνήθως μεγαλύτερος από το πλήθος των διαθέσιμων εργατών. Ένας εργάτης μπορεί να εκτελέσει περισσότερες από μία εργασίες. Έτσι η ισορροπία φόρτου βελτιώνεται και σε περίπτωση που υπάρξει βλάβη σε έναν εργάτη υπάρχει γρηγορότερη ανάρρωση με την επανακατανομή των εργασιών του σε άλλους.

Εφεδρικές εργασίες:
Μερικές εργασίες καθυστερούν την ολοκλήρωση τους και μαζί και την ολοκλήρωση της συνολικής δουλειάς. Η λύση στο πρόβλημα είναι η δημιουργία αντιγράφων της εργασίας (speculative execution). Μία εργασία θεωρείται ολοκληρωμένη όταν ενημερώσει τον master αυτή ή ένα αντίγραφο της.

Partitioning-combining :
Ένας χρήστης μπορεί να ορίσει μία δική του συνάρτηση διαίρεσης κατά το shuffling. Μία συνάρτηση combiner μπορεί να οριστεί για να επεξεργαστεί τα δεδομένα εξόδου μίας εργασίας map πριν αυτά γίνουν διαθέσιμα στους reducers. Εκτελείται από τον ίδιο εργάτη που εκτελεί τη map εργασία και συνήθως είναι παρόμοια με τη συνάρτηση reduce. Ο τύπος των δεδομένων εισόδου και εξόδου μπορεί να καθοριστεί από το χρήστη και δεν έχει περιορισμούς του τι μορφής μπορεί να είναι. Η είσοδος ενός reducer είναι πάντα ταξινομημένη. Υπάρχει δυνατότητα τοπικής εκτέλεσης που εκτελεί όλες τις εργασίες σειριακά. Ο master προσφέρει ένα web interface όπου ο χρήστης μπορεί να παρακολουθεί την εκτέλεση των εργασιών. Παρέχεται η δυνατότητα στον χρήστη να ορίσει καθολικούς μετρητές.

Το πλεονέκτημα του MapReduce είναι το γεγονός ότι οι συναρτήσεις Map και Reduce είναι πλήρως παραλληλοποιήσιμες. Κάθε τέτοια συνάρτηση έχει τα δικά της δεδομένα εισόδου και μπορεί να παράγει τα αποτελέσματα της χωρίς να χρειάζεται να επικοινωνήσει με καμία άλλη. Κατά συνέπεια οι διεργασίες δεν έχουν κάποια εξάρτηση η μία από την άλλη και μπορούν να εκτελεστούν όλες ταυτόχρονα. Αυτό έχει ως αποτέλεσμα να αποφεύγονται τα γνωστά προβλήματα επικοινωνίας που εμφανίζονται σε άλλα είδη παράλληλου προγραμματισμού όταν αυξάνεται ο αριθμός των παράλληλων διεργασιών.



Σχήμα 2.1: MapReduce Framework

2.2 Hadoop & HDFS

Το Hadoop [5] είναι ένα προγραμματιστικό πλαίσιο (framework) ανοιχτού κώδικα που αναπτύσσεται από την εταιρεία Apache Software Foundation [4] και υποστηρίζει κατανεμημένη επεξεργασία μεγάλου όγκου δεδομένων. Η ανάπτυξη του ξεκίνησε το 2005 ως μια εναλλακτική επιλογή ανοιχτού κώδικα του Google MapReduce framework. Ένας από τους σημαντικότερους λόγους επιτυχίας του συστήματος αυτού είναι ότι σχεδιάστηκε για να μπορεί να τρέχει σε συστάδες από απλούς υπολογιστές (clusters) χαμηλού κόστους όπου όμως η συχνότητα βλαβών είναι υψηλή, αναλαμβάνοντας όλες τις απαραίτητες λειτουργίες για την εκτέλεση των εργασιών MapReduce. Τα προγράμματα που γράφουν οι χρήστες μπορούν να τρέξουν σε οποιοδήποτε cluster μηχανημάτων παρέχοντας ανοχή στα σφάλματα υλικού. Στις μέρες μας το Hadoop αποτελεί ένα από τα καλύτερα κατανεμημένα συστήματα για batch processing.

Το Apache Hadoop framework αποτελείται από τα εξής τμήματα:

- Hadoop Common: Περιέχει βιβλιοθήκες και εργαλεία απαραίτητα για τη λειτουργία των υπόλοιπων τμημάτων.

- Hadoop Distributed File System (HDFS): Κατανεμημένο, κλιμακώσιμο (scalable) σύστημα αρχείων.

- Hadoop YARN: Πλατφόρμα υπεύθυνη για τη διαχείριση των υπολογιστικών πόρων και υπεύθυνη για τη χρονοδρομολόγηση των εργασιών του χρήστη.

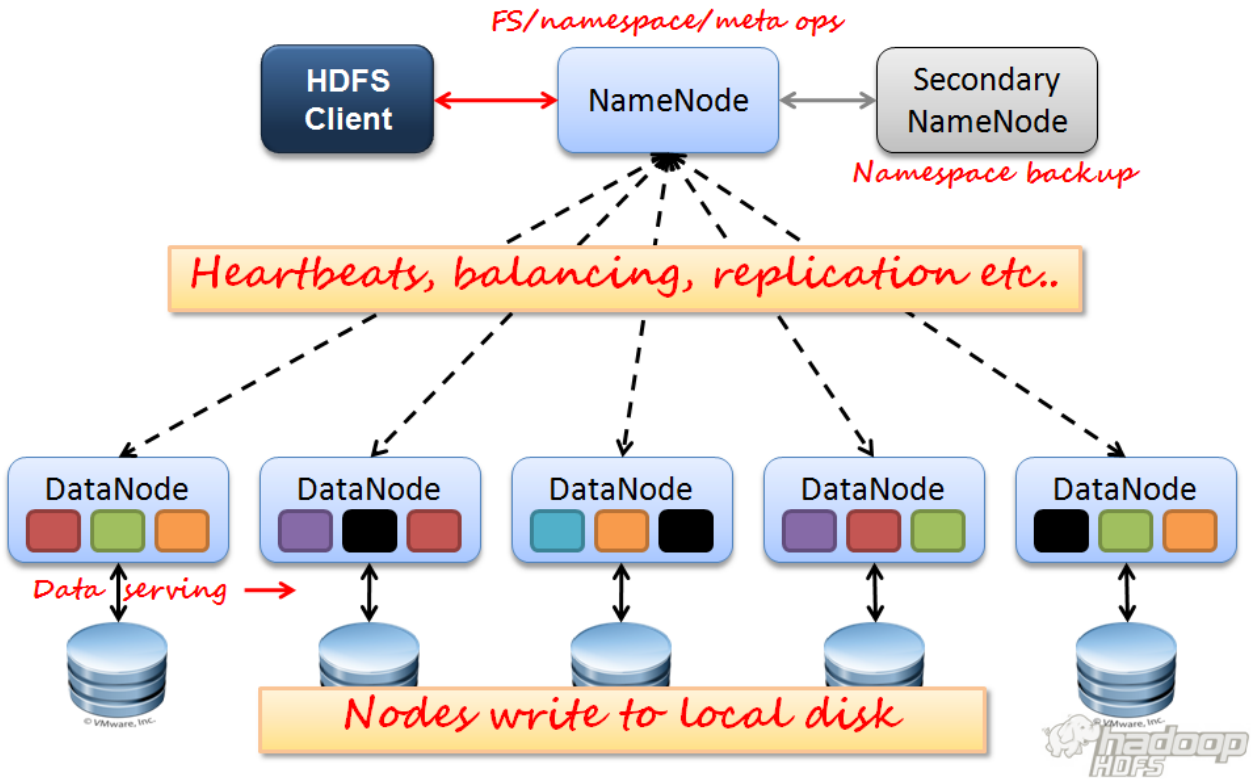
- Hadoop MapReduce: Προγραμματιστικό μοντέλο για την επεξεργασία μεγάλου όγκου δεδομένων.

Το HDFS (Hadoop Distributed File System) είναι μία υλοποίηση ενός κατανεμημένου συστήματος αρχείων το οποίο δημιουργήθηκε ως μία εναλλακτική επιλογή ανοιχτού κώδικα του GFS (Google File System). Το HDFS είναι κατανεμημένο, κλιμακώσιμο και εξασφαλίζει την διαθεσιμότητα (availability) των δεδομένων. Παρέχει ανοχή σε σφάλματα, παρότι εκτελείται σε υπολογιστές με μεσαίας ισχύος υλικό και παράλληλα διαθέτει την ικανότητα να εξυπηρετεί ένα μεγάλο σύνολο από πελάτες. Φροντίζει για τα αντίγραφα και την τοπικότητα των δεδομένων. Χρησιμοποιείται σαν είσοδος δεδομένων και έξοδος αποτελεσμάτων από το MapReduce. Το HDFS ακολουθεί την αρχιτεκτονική Master/Slave. Αποτελείται από έναν Master και πολλούς Chunkservers. Ο χώρος αποθήκευσης μπορεί να βρίσκεται σε ετερογενή λειτουργικά συστήματα και είναι ενιαίος

για όλο το cluster. Οι Clients μπορούν να βρουν την τοποθεσίες των blocks και να προσπελαύνουν τα δεδομένα απευθείας.

Το HDFS επιτυγχάνει να είναι αξιόπιστο, αποθηκεύοντας τα δεδομένα και αντίγραφα αυτών σε περισσότερους από ένα κόμβους. Συγκεκριμένα, ο χώρος των αρχείων είναι ενιαίος για όλο το cluster με τα αρχεία να διασπώνται σε blocks, με κάθε block να αντιγράφεται σε πολλαπλούς κόμβους δεδομένων. Μεταξύ των κόμβων υπάρχει επικοινωνία για εξισορρόπηση των δεδομένων, αντιγραφή ή μετακίνηση τους, έτσι ώστε να παραμείνει ψηλός ο λόγος αντιγραφής (replication). Η επικοινωνία αυτή γίνεται με χρήση του πρωτοκόλλου TCP/IP.

Μοναδικό σημείο αποτυχίας του HDFS (single point of failure) αποτελεί ο κεντρικός κόμβος, ο NameNode, ο οποίος διατηρεί πληροφορίες σχετικά με το που βρίσκονται τα δεδομένα στο HDFS και στη περίπτωση που δεν είναι διαθέσιμος δεν είναι δυνατή η πρόσβαση σε αυτά. Ένας δεύτερος κόμβος, ο SecondaryNameNode, δημιουργεί checkpoint σε τακτά χρονικά διαστήματα. Διατηρεί αντίγραφα των φακέλων του NameNode (snapshots) και μαζί με τα αρχεία χρήσης του NameNode (logs) είναι δυνατή η επαναφορά του συστήματος σε περίπτωση βλάβης. Οι κόμβοι οι οποίοι αποθηκεύονται τα δεδομένα ονομάζονται DataNodes.



Σχήμα 2.2: HDFS Architecture

2.3 Hive

Το Hive [6] είναι μία αποθήκη δεδομένων (data warehouse) που διευκολύνει την επεξεργασία και ανάλυσή τους όταν αυτά βρίσκονται αποθηκευμένα σε κατανεμημένα συστήματα. Είναι φτιαγμένο έτσι ώστε να τρέχει πάνω από το Hadoop και να μπορεί να αποθηκεύσει και να επεξεργαστεί τα δεδομένα που απαιτούνται στο HDFS. Το Hive προσφέρει ένα μηχανισμό κατάλληλο για να μπορεί να επιλέξει ο χρήστης τη δομή των δεδομένων και μία SQL-Like γλώσσα, την HiveQL, για να μπορεί ο χρήστης να επεξεργαστεί τα δεδομένα με τον τρόπο που επιθυμεί. Με αυτή τη γλώσσα ο χρήστης μπορεί να φορτώσει δεδομένα από εξωτερικές πηγές (π.χ. HDFS) και να τα επεξεργαστεί εύκολα. Η HiveQL είναι αρκετά επεκτάσιμη, καθώς ο χρήστης μπορεί να ορίσει δικές του συναρτήσεις column transformation (UDF) υλοποιημένες σε Java.

Τα δεδομένα στο Hive οργανώνονται σε:

- **Tables:** Οι πίνακες αυτοί είναι ανάλογοι με τους πίνακες που γνωρίζουμε από τις σχεσιακές βάσεις δεδομένων. Κάθε πίνακας έχει ένα αντίστοιχο directory στο HDFS όπου είναι αποθηκευμένα τα δεδομένα. Τα δεδομένα σειριοποιούνται και αποθηκεύονται σε διάφορα αρχεία μέσα σε αυτό το directory.

- **Partitions:** Κάθε πίνακας μπορεί να έχει ένα ή περισσότερα partitions που προσδιορίζουν την περαιτέρω κατανομή των δεδομένων σε υπο-φακέλους. Το πλεονέκτημα είναι ότι όλα τα δεδομένα που έχουν την ίδια τιμή στη στήλη που έχει χρησιμοποιηθεί για το partition καταλήγουν στο ίδιο sub-directory και υπάρχει η δυνατότητα να εκτελέσουμε κάποιο ερώτημα σε αυτά τα δεδομένα χωρίς να χρειαστεί να επεξεργαστούμε το σύνολο των δεδομένων.

- **Buckets:** Σε κάθε partition τα δεδομένα μπορούν να διαχωριστούν περαιτέρω σε buckets χρησιμοποιώντας την hash μιας στήλης του πίνακα.

Κεφάλαιο 3
Real-Time Processing

3.1 Συστήματα Επεξεργασίας Πραγματικού Χρόνου

Με τον όρο συστήματα επεξεργασίας πραγματικού χρόνου, αναφερόμαστε στα συστήματα με συνεχή είσοδο, έγκαιρη επεξεργασία και συστηματική έξοδο των δεδομένων. Τέτοιου είδους συστήματα γίνονται όλο και περισσότερο αναγκαία για την κάλυψη συγκεκριμένων αναγκών, όπως επίβλεψη συστήματος, ηλεκτρονικό εμπόριο, ανίχνευση απάτης και λήψη αποφάσεων, τόσο για επιστημονικά όσο και εταιρικά περιβάλλοντα. Είναι απαραίτητο να επεξεργάζονται υψηλό ποσοστό δεδομένων σε μικρό χρονικό διάστημα και κατά συνέπεια πρέπει να εγγυώνται απόκριση μέσα σε αυστηρούς χρονικούς περιορισμούς.

Σύμφωνα με την κρισιμότητα της άμεσης απάντησης, τα συστήματα πραγματικού χρόνου χωρίζονται σε 3 κατηγορίες:

- **Αυστηρά:** Η απώλεια του χρονικού ορίου για την επεξεργασία μιας μονάδας δεδομένων ισοδυναμεί με αποτυχία του συστήματος.

- **Σταθερά:** Η σπάνια απώλεια χρονικού ορίου για την επεξεργασία μας μονάδας δεδομένων είναι ανεκτή αλλά μπορεί να υποβαθμίσει την ποιότητα υπηρεσιών του συστήματος.

- **Χαλαρά:** Η χρησιμότητα ενός αποτελέσματος μειώνεται με το πέρας του χρονικού ορίου, υποβαθμίζοντας την ποιότητα υπηρεσιών του συστήματος.

Ένα επιτυχημένο σύστημα πραγματικού χρόνου πρέπει να είναι ικανό να επεξεργαστεί πολύ μεγάλο αριθμό δεδομένων συνεχόμενα. Επίσης, επιβάλλεται να παρέχει ένα μηχανισμό ερωτήσεων προς τη ροή, για τα τρέχοντα ή τα προγενέστερα δεδομένα. Η ακεραιότητα αυτών των δεδομένων, η αποθήκευση και η ανάκτησή τους πρέπει να

εξασφαλίζεται ανά πάσα στιγμή, ανεξάρτητα από οποιαδήποτε αποτυχία του συστήματος. Συνεπώς το σύστημα είναι απαραίτητο να διαχειρίζεται ατέλειες, όπως καθυστέρηση στην άφιξη, άφιξη εκτός σειράς, ή ακόμη και απώλεια των δεδομένων. Τέλος, η παραγόμενη έξοδος συνηθίζεται να είναι προβλέψιμη και διαθέσιμη για επίβλεψη, έτσι ώστε να επιβεβαιώνεται ο ντετερμινισμός και η επαναληπτικότητα του συστήματος.

3.2 Storm

3.2.1 Εισαγωγή

Το Storm [7] είναι ένα κατανεμημένο υπολογιστικό σύστημα ανοιχτού κώδικα, που επιτρέπει την ανάλυση και επεξεργασία ροών δεδομένων σε πραγματικό χρόνο. Κύρια χαρακτηριστικά του είναι η κλιμάκωση (scalability), η ανεκτικότητα σε λάθη (fault tolerance) και η εγγύηση ότι όλα τα δεδομένα που θα εισαχθούν στο σύστημα θα επεξεργαστούν στο ακέραιο (data processing guarantee). Το Storm δημιουργήθηκε από την εταιρία Backtype, η οποία εξαγοράστηκε το 2011 από το Twitter, ενώ πρόσφατα μετέβη κάτω από την άδεια της Apache. Αποτελεί ένα από τα ισχυρότερα μοντέλα του είδους που έχουν προταθεί μέχρι σήμερα. Χρησιμοποιείται για την αξιόπιστη επεξεργασία απεριόριστου όγκου δεδομένων σε πολλαπλά σενάρια χρήσης, όπως στην ανάλυση συστήματος σε πραγματικό χρόνο, σε online συστήματα μηχανικής μάθησης, στην εξόρυξη δεδομένων από ροές δεδομένων, στο συνεχόμενο υπολογισμό και σε κατανεμημένο RPC. Το Storm έγινε ιδιαίτερα δημοφιλές λόγω της απλότητας χρήσης του, όσο και της μεγάλης ταχύτητας επεξεργασίας δεδομένων: Έχει μετρηθεί ότι μπορεί να επεξεργαστεί πάνω από ένα εκατομμύριο εγγραφές ανά δευτερόλεπτο σε κάθε κόμβο.

Ουσιαστικά το Storm είναι η εκδοχή του Hadoop για συστήματα πραγματικού χρόνου, αφού είναι ικανό να διαχειριστεί τεράστιες ποσότητες δεδομένων σε συστοιχία υπολογιστών και να παρέχει αποτελέσματα. Αυτό που το διαφοροποιεί παρόλα αυτά, από ένα συμβατικό σύστημα μεγάλου όγκου δεδομένων, είναι ο προσανατολισμός στην ανίχνευση γεγονότων. Αυτό σημαίνει ότι μπορεί να αναγνωρίσει τα πιο σημαντικά γεγονότα από μια συνεχή ροή και να ανταποκριθεί άμεσα. Έχει υψηλή ανοχή σε σφάλματα (high fault-tolerance) και είναι κλιμακώσιμο στην ποσότητα δεδομένων εισόδου.

3.2.2 Βασικά Χαρακτηριστικά

Το Storm έχει μια σειρά ιδιοτήτων που το κάνουν μοναδικό στον χώρο της κατανεμημένης επεξεργασίας δεδομένων. Μερικά από αυτά παρουσιάζονται παρακάτω:

- **Γρήγορο:** Είναι ικανό να επεξεργαστεί έως και ένα εκατομμύριο μηνύματα μεγέθους 100 bytes το δευτερόλεπτο ανά κόμβο.

- **Κλιμάκωσιμο (scalable):** Μπορεί να κλιμακώσει την επεξεργασία των δεδομένων του σε πολύ μεγάλο αριθμό μηνυμάτων ανά δευτερόλεπτο. Το μόνο που χρειάζεται για να επιτευχθεί αυτό είναι η προσθήκη περισσότερων κόμβων στο cluster των υπολογιστών που τρέχουν το Storm και η ρύθμιση των παραμέτρων παραλληλοποίησης του συστήματος. Η χρήση του Zookeeper είναι αυτή που παρέχει στο Storm τους απαραίτητους μηχανισμούς για να κλιμακώνεται σε τεράστια μεγέθη cluster υπολογιστών.

- **Ανθεκτικό σε σφάλματα (fault tolerance):** Ένα από τα κύρια χαρακτηριστικά του Storm είναι η ανθεκτικότητα που παρουσιάζει σε σφάλματα που εμφανίζονται κατά τη διάρκεια του κατανεμημένου υπολογισμού δεδομένων. Η ιδιότητα αυτή είναι πολύ σημαντική αφού ένα cluster που μπορεί να αποτελείται από χιλιάδες υπολογιστές και είναι πολύ κρίσιμο να μπορούμε να επεξεργαστούμε τα δεδομένα μας, ακόμα και στην περίπτωση που παρουσιαστεί κάποιο πρόβλημα σε έναν ή περισσότερους κόμβους. Τα είδη των αποτυχιών που μπορούν να προκύψουν είναι πάρα πολλά: Πρόβλημα στον σκληρό δίσκο ενός κόμβου, προβλήματα στο δίκτυο, προβλήματα στο επίπεδο του λογισμικού κ.α. Το Storm παρέχει όλους τους απαραίτητους μηχανισμούς για να αντεπεξέλθει στα λάθη που μπορούν να προκύψουν, αναδιανέμοντας εργασίες που επρόκειτο να επεξεργαστούν, από κόμβους που τελικώς κατέρρευσαν, σε υγιής κόμβους του cluster και παράλληλα παρέχει μηχανισμό αυτόματης επανεκκίνησης των “πεθαμένων” κόμβων - εργατών.

- **Αξιόπιστο:** Παρέχει εγγύηση μη απώλειας δεδομένων με μηχανισμούς εξασφάλισης της επεξεργασίας κάθε μονάδας δεδομένων ακριβώς μια φορά. Ένα σύστημα που πραγματοποιεί επεξεργασία δεδομένων σε πραγματικό χρόνο πρέπει να παρέχει εγγυήσεις ότι όλα τα δεδομένα που δίδονται ως είσοδος θα επεξεργαστούν επιτυχώς από το σύστημα. Το Storm εγγυάται ότι κάθε εισερχόμενη εγγραφή θα επεξεργαστεί, μαρκάροντας μια από αυτές με ετικέτες κατάστασης. Εάν μια εγγραφή έχει χαθεί τότε γίνεται προσπάθεια να επεξεργαστεί εκ νέου.

- **Εύκολο στην χρήση:** Σε αντίθεση με συστήματα όπως το Hadoop, που είναι ιδιαίτερα δύσκολος και χρονοβόρος ο χειρισμός τους, το Storm έχει δημιουργηθεί με στόχο ο χρήστης να μπορεί να διαχειρίζεται το cluster των υπολογιστών με όσο το δυνατόν μεγαλύτερη ευκολία.

- **Εύκολο στον προγραμματισμό:** Μειώνει σημαντικά τη περιπλοκότητα της επεξεργασίας σε πραγματικό χρόνο και υποστηρίζει σχεδόν όλες τις γλώσσες εφόσον ο προγραμματιστής κατασκευάσει μια μικρή ενδιάμεση βιβλιοθήκη. Η υλοποίηση ενός προγράμματος στο Storm γίνεται συνήθως στη γλώσσα προγραμματισμού Java, όμως μπορεί να χρησιμοποιηθεί μια πληθώρα άλλων γλωσσών προγραμματισμού, γεγονός που καθιστά το σύστημα ιδιαίτερα ελκυστικό σε όλους τους προγραμματιστές ανεξάρτητα από την γλώσσα την οποία χρησιμοποιούν στις υλοποιήσεις τους.

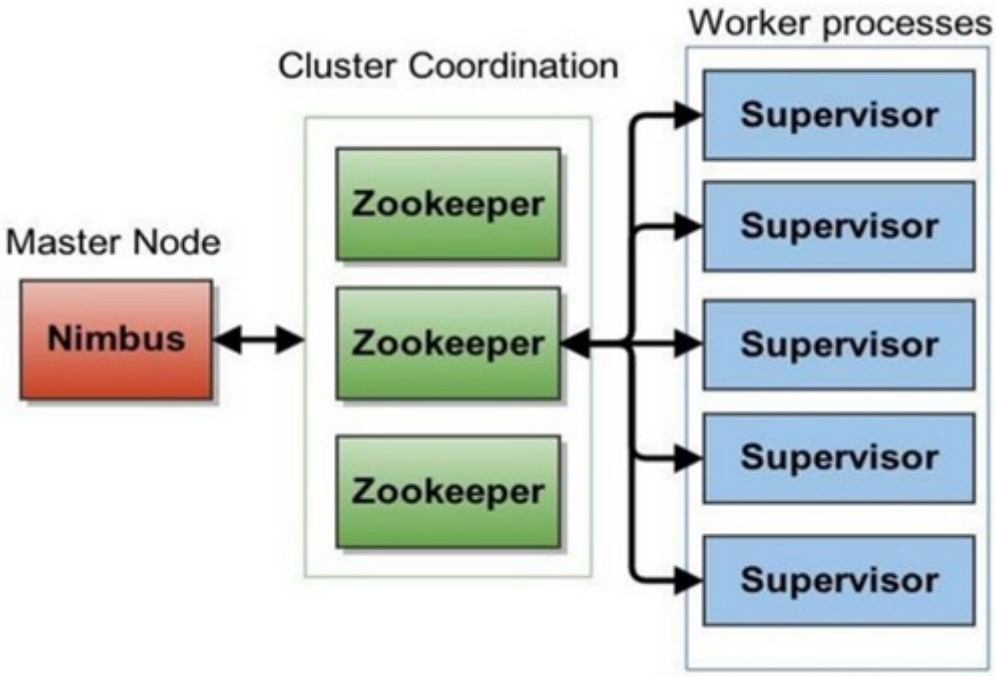
- **Εύκολο στην επίβλεψη:** Παρέχει ένα γραφικό τρόπο αναπαράστασης του παραγόμενου συστήματος, για την ανίχνευση λαθών και την εξαγωγή στατιστικών εκτέλεσης.

3.2.3 Συνιστώσες Συστήματος

Ένα Storm cluster έχει, σε επίπεδο οντοτήτων, πάρα πολλές ομοιότητες με ένα Hadoop cluster. Στο Hadoop τρέχουμε εργασίες MapReduce ενώ στο Storm τοπολογίες (topologies). Η βασική διαφορά τους είναι ότι ενώ μια εργασία MapReduce τερματίζει κάποια στιγμή, ως αποτέλεσμα της ολοκλήρωσης της επεξεργασίας του συνόλου των δεδομένων που μπήκαν στο σύστημα, μια τοπολογία επεξεργάζεται δεδομένα για πάντα (ή μέχρι να σταματήσουμε την τοπολογία) όσο αυτά εισέρχονται στο σύστημα, σε πραγματικό χρόνο.

Υπάρχουν δύο είδη κόμβων σε μια συστοιχία υπολογιστών Storm: Ο κόμβος αφέντης (master node) και οι κόμβοι εργάτες (worker node). Ο κόμβος αφέντης τρέχει έναν δαίμονα (daemon) που ονομάζεται Nimbus (το αντίστοιχο του ResourceManager στο Hadoop), ο οποίος είναι υπεύθυνος για την επίβλεψη της συστοιχίας, τον συντονισμό των κόμβων του cluster, την διανομή των εργασιών στους κόμβους εργάτες και τον έλεγχο για πιθανές αποτυχίες του συστήματος. Κάθε κόμβος εργάτης τρέχει έναν δαίμονα που ονομάζεται Supervisor, ο οποίος αναμένει να του ανατεθούν εργασίες και εκκινεί ή σταματά την εκτέλεση διεργασιών - εργατών (worker) ανάλογα με τις οδηγίες του

Nimbus. Κάθε μηχάνημα μπορεί να τρέχει έναν ή περισσότερους workers, κάθε ένας από τους οποίους επεξεργάζεται ένα υποσύνολο των δεδομένων της τοπολογίας. Η επικοινωνία Nimbus – Supervisor γίνεται με την βοήθεια των μηχανισμών συντονισμού του Zookeeper. Μια τοπολογία αποτελείται από πολλούς workers κατανεμημένους σε πολλά μηχανήματα.



Σχήμα 3.1: Storm Architecture

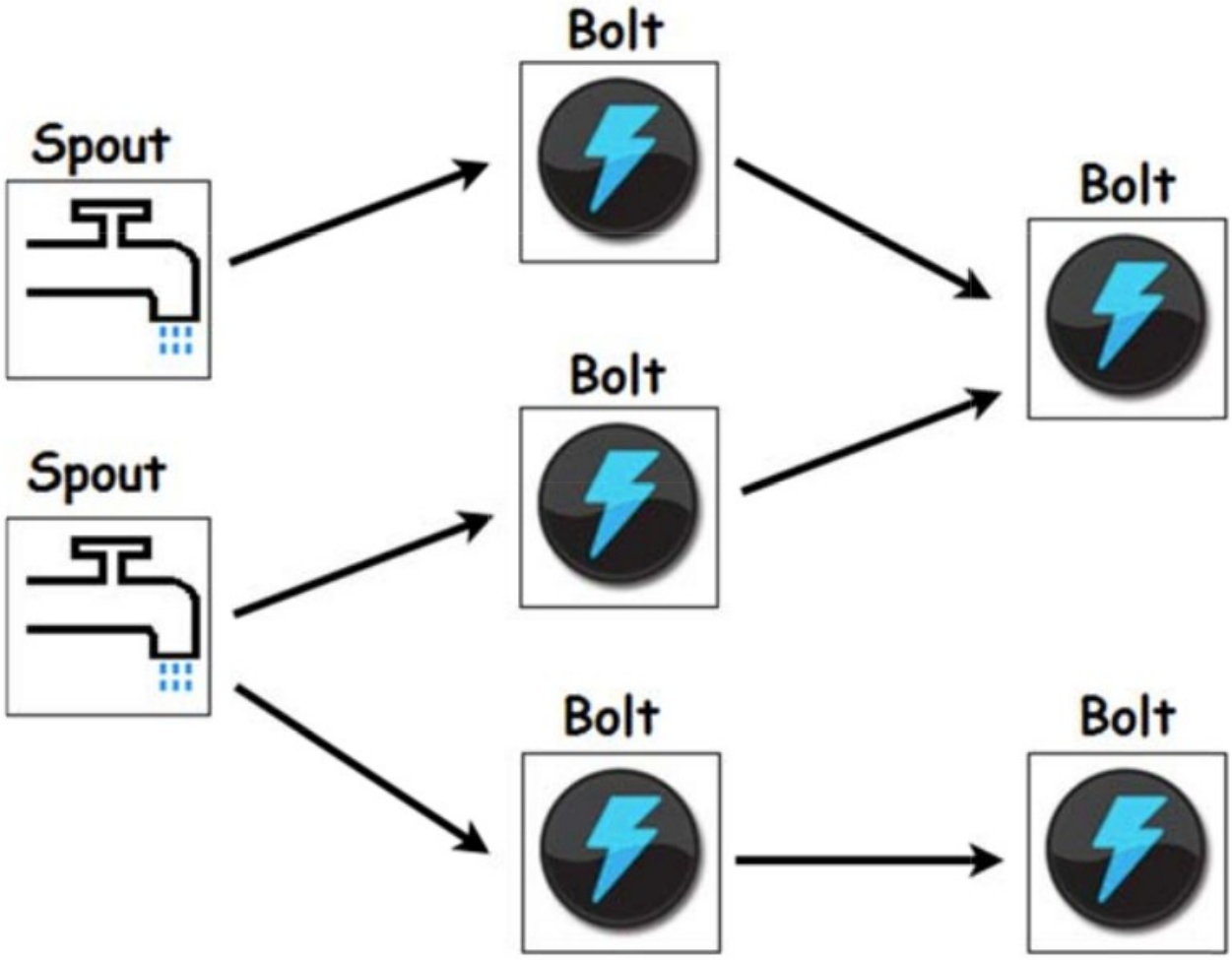
3.2.4 Ροές Δεδομένων & Τοπολογίες

Μια ροή δεδομένων καλείται stream και είναι μια απεριόριστα μεγάλη ακολουθία πλειάδων (tuples). Οι πλειάδες αυτές, στο Storm, επεξεργάζονται ή/και δημιουργούνται κατανεμημένα. Κάθε πλειάδα είναι μία λίστα με προκαθορισμένα πεδία, ο τύπος των οποίων ορίζεται μέσω ενός σχήματος (schema). Τα πεδία μπορεί να περιλαμβάνουν, οποιονδήποτε πρωτόγονο τύπο δεδομένων για μια συνηθισμένη γλώσσα προγραμματισμού (booleans, integers, characters, doubles, κτλ) ή οποιονδήποτε δικό μας κατασκευαζόμενο τύπο δεδομένο (strings, arrays, lists, κλπ). Μια πλειάδα είναι ένα σειριοποιήσιμο αντικείμενο που εκπέμπεται μεταξύ των κόμβων εργατών, για να λάβει μέρος σε μια σειρά από συγκεκριμένους υπολογισμούς. Ο λόγος που οι πλειάδες είναι σειριοποιήσιμα αντικείμενα, είναι η ευελιξία που απαιτείται σε ένα κατανεμημένο σύστημα. Σε κάθε ροή δίδεται ένα μοναδικό αναγνωριστικό (id), το οποίο χρησιμοποιείται από το Storm και τον προγραμματιστή για τη διαχείριση των ροών δεδομένων. Για να γίνει η επεξεργασία των πλειάδων, χρειάζεται να υλοποιήσουμε μία τοπολογία.

Τοπολογία (topology) είναι η κύρια οντότητα που τρέχει μια συστοιχία Storm. Η τοπολογίας είναι ένας γράφος μετασχηματισμού ροών δεδομένων, σε υψηλό επίπεδο αφαίρεσης. Κάθε κόμβος σε μια τοπολογία επεξεργάζεται μερικώς τα δεδομένα, ενώ συσχετίσεις μεταξύ των κόμβων καθορίζουν σε ποιους κόμβους και με ποια σειρά θα μεταβούν τα δεδομένα ώστε να ολοκληρωθεί η επεξεργασία. Πρόκειται δηλαδή για ένα μοντέλο ροής δεδομένων. Τα δεδομένα υφίστανται επεξεργασία σαν ροή, το οποίο σημαίνει ότι ο υπολογισμός είναι άπειρος, εφόσον υπάρχει διαθέσιμη είσοδος. Ένα storm cluster μπορεί να τρέχει μια η περισσότερες τοπολογίες ταυτόχρονα, χωρίς η μία να επηρεάζει την άλλη. Κάθε κόμβος της τοπολογίας μπορεί να αναπαριστά ένα spout ή ένα bolt και κάθε ακμή υποδεικνύει ποιες ροές επεξεργάζονται από ποια bolts.

Το spout αποτελεί το κύριο σημείο εισαγωγής ροών δεδομένων που εισάγονται στο σύστημα προς επεξεργασία. Το spout διαβάζει ταυτόχρονα εγγραφές από διαφορετικές πηγές δεδομένων, όπως ένα σύστημα ουράς (π.χ. RabbitMQ, Kafka, ActiveMQ, ή Kestrel) το οποίο μπορεί να είναι ένας διαμεσολαβητής ή μια διεπαφή, ή ένα απλό αρχείο κειμένου, και το επεξεργάζεται ως ροή. Επίσης μπορεί να εκπέμπει (emit) περισσότερες από μία ροές. Ένα spout μπορεί να είναι αξιόπιστο (reliable) ή αναξιόπιστο (unreliable). Ένα αξιόπιστο spout είναι ικανό να επαναλαμβάνει την αποστολή μίας πλειάδας, όταν γνωρίζει ότι η επεξεργασία της απέτυχε. Αντίθετα, ένα αναξιόπιστο spout δεν ασχολείται με τις πλειάδες από την στιγμή που θα αναχωρήσουν από το spout αυτό. Υπάρχει πληθώρα υλοποιημένων spouts που διασυνδέουν τα συστήματα ουράς που προαναφέρθηκαν με το Storm και οποία μπορούν να ρυθμιστούν και να επεκταθούν προγραμματιστικά, ανάλογα τις ανάγκες του εκάστοτε προγραμματιστή.

Το bolt είναι το στοιχείο που επεξεργάζεται τις ροές δεδομένων που εισάγονται μέσω των spouts στο σύστημα. Τα bolt μπορούν να διαβάσουν πλειάδες από τα spout ή από άλλα bolt. Ένα bolt μπορεί να διαβάσει ένα ή περισσότερα stream, επεξεργάζεται τα δεδομένα και μπορεί είτε να αποθηκεύσει τα αποτελέσματα (την έξοδο του), είτε να τα μετασχηματίσει και να τα αποστείλει σε άλλα bolt για περαιτέρω επεξεργασία, δημιουργώντας με αυτό τον τρόπο ροές δεδομένων στο σύστημα. Τα bolt εφαρμόζουν ένα ενιαίο μετασχηματισμό για κάθε stream σε μία τοπολογία. Μπορούν να εφαρμόσουν λειτουργίες αντίστοιχες του MapReduce ή ακόμα και πιο σύνθετες δράσεις που περιορίζονται σε ένα επίπεδο (single-step functions), όπως φιλτράρισμα και ομαδοποιήσεις ροών, υπολογισμός αθροισμάτων, αποθήκευση αποτελεσμάτων και επικοινωνία με εξωτερικούς φορείς, όπως μια βάση δεδομένων. Για να πραγματοποιηθεί ένας περίπλοκος μετασχηματισμός ροής, χρειάζεται να υλοποιήσουμε περισσότερα του ενός bolt σε μία τοπολογία. Μια τυπική τοπολογία Storm συνηθίζεται να εκτελεί πολλαπλούς μετασχηματισμούς και ως εκ τούτου, απαιτεί πολλαπλά bolts .Ένα bolt, όπως ακριβώς ένα spout, είναι δυνατόν να εκπέμψει (emit) δεδομένα σε πολλαπλά bolt, δηλαδή να δημιουργήσει περισσότερες από μία ροές.



Σχήμα 3.2: Storm Topology

3.2.5 Παραλληλισμός Εργασιών

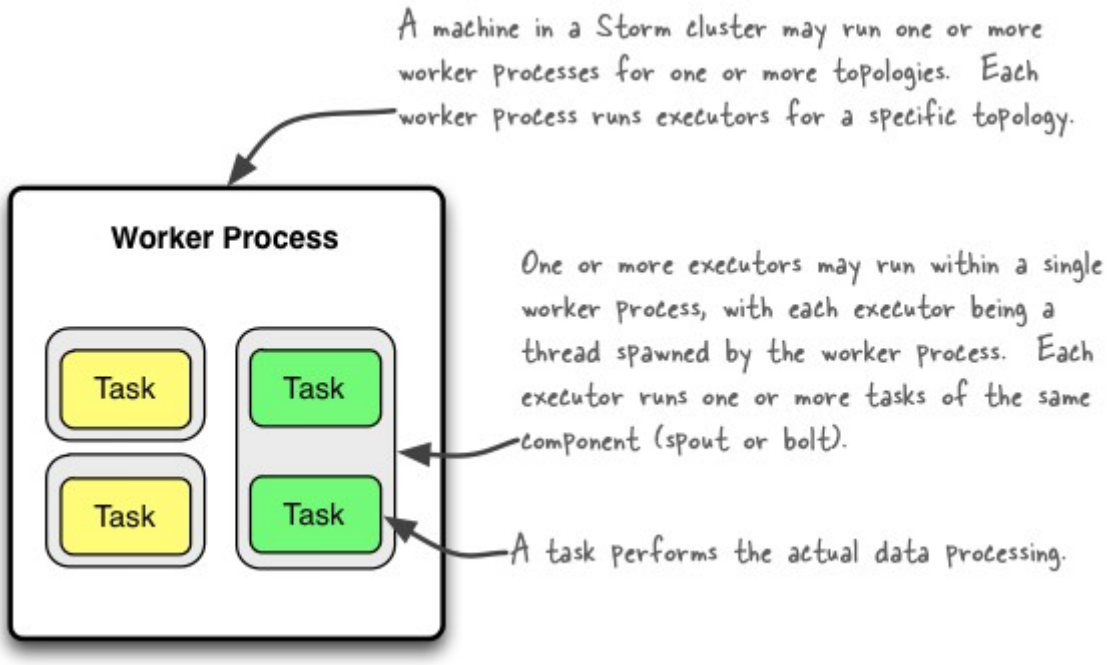
Σε ένα σύστημα επεξεργασίας ροών δεδομένων είναι πάρα πολύ σημαντικό να μπορούμε να επεξεργαστούμε τα δεδομένα μας με ρυθμό μεγαλύτερο από τον ρυθμό με τον οποίο καταφθάνουν ν στο σύστημα. Διαφορετικά οι εγγραφές θα περιμένουν για πολύ μέχρι να επεξεργαστούν ή ακόμα χειρότερα θα λήξει η προθεσμία επεξεργασίας τους και θα χαθούν. Η δυνατότητα που δίνει το Storm για κατανεμημένη επεξεργασία ροών δεδομένων λύνει αυτόματα το παραπάνω πρόβλημα. Σε κάθε τοπολογία μπορούμε να καθορίσουμε το επίπεδο παραλληλισμού που θέλουμε να έχουμε (ως συνάρτηση του αριθμού των Supervisor που έχουμε και του αριθμού των workers που τρέχει κάθε ένας) και το Storm θα δημιουργήσει τον κατάλληλο αριθμό νημάτων στο cluster για να γίνει η επεξεργασία των δεδομένων.

Για να μπορέσει να τρέξει μια τοπολογία, το Storm διακρίνει τις τρεις παρακάτω οντότητες:

- **Worker:** Όπως είπαμε, ένα μηχάνημα στο cluster του Storm μπορεί να τρέχει έναν ή περισσότερους workers για μια ή περισσότερες τοπολογίες. Κάθε worker ανήκει σε μια συγκεκριμένη τοπολογία και εκτελεί ένα υποσύνολο των εργασιών της. Ένας worker περιλαμβάνει έναν αριθμό από εκτελεστές διεργασιών (executors), κάθε ένας από τους οποίους τρέχει ένα η περισσότερα συστατικά (spouts ή bolts) της τοπολογίας στην οποία ανήκει.

- **Executor:** Ένας executor δεν είναι τίποτα περισσότερο από ένα νήμα (thread) που δημιουργείται από έναν worker. Το επίπεδο του παραλληλισμού σε μια τοπολογία αντιστοιχεί ακριβώς στον αριθμό των executors που θα δουλεύουν για αυτή τη τοπολογία, παράλληλα. Κάθε executor μπορεί να τρέχει ένα ή περισσότερα tasks για ένα συγκεκριμένο spout ή bolt. Ο αριθμός των executors μιας τοπολογίας μπορεί να αλλάζει με τον χρόνο.

- **Task:** Το task είναι αυτό που πραγματοποιεί την επεξεργασία των δεδομένων και τρέχει μέσα στο νήμα του executor στον οποίο ανήκει. Ο αριθμός των tasks για ένα spout ή ένα bolt παραμένει σταθερός με τον χρόνο σε αντίθεση με τον αριθμό των executors που είναι μεταβλητός. Το Storm έχει ως προεπιλογή να τρέχει ένα task για κάθε νήμα.



Σχήμα 3.3: Παραλληλισμός Εργασιών στο Storm

3.2.6 Ομαδοποίηση Ροών

Ένας executor όπως είπαμε εργάζεται για ένα συγκεκριμένο συστατικό στοιχείο (spout ή bolt) σε μια τοπολογία. Όταν ένα στοιχείο της τοπολογίας στέλνει ροές δεδομένων σε ένα άλλο, είναι πολύ σημαντικό σε πολλά σενάρια χρήσης να γνωρίζουμε τον τρόπο με τον οποίο διανέμονται τα δεδομένα που στέλνει ένα συστατικό στοιχείο (ως tasks) στους executors του επόμενου στοιχείου. Εάν για παράδειγμα θέλουμε να επιτύχουμε εξισορρόπηση φόρτου μεταξύ των executors ενός bolt, τότε θέλουμε τα δεδομένα που λαμβάνει το bolt να αποστέλλονται με ίση πιθανότητα στους executors του για επεξεργασία. Σε άλλες περιπτώσεις επιθυμούμε κάποια δεδομένα να επεξεργάζονται πάντοτε από τον ίδιο executor μέσα στο bolt, όπως για παράδειγμα στην περίπτωση που θέλουμε να κάνουμε καταμέτρηση λέξεων. Εάν μια λέξη καταμετράται σε περισσότερους του ενός executors μέσα σε ένα bolt, τότε το αποτέλεσμα της καταμέτρησης της θα είναι λανθασμένο στην έξοδο. Σε αυτή τη περίπτωση θα γίνουν emit, για την ίδια λέξη, τόσα αθροίσματα όσοι είναι οι executors που έχουν καταμετρήσει έστω μια φορά την λέξη.

Η ομαδοποίηση ροής καθορίζει τον τρόπο με τον οποίο η τοπολογία στέλνει πλειάδες ανάμεσα σε ένα ζευγάρι συστατικών της (από spout σε bolt ή από bolt σε bolt). Επί της ουσίας καθορίζει τον τρόπο κατάτμησης (partitioning) της εισερχόμενης πληροφορίας μέσα στους executors του bolt. Το Storm παρέχει επτά έτοιμους μηχανισμούς ομαδοποίησης ροής, αλλά δίνει και την δυνατότητα ο χρήστης να υλοποιήσει προγραμματιστικά τον δικό του μηχανισμό ομαδοποίησης.

Οι επτά αυτοί έτοιμοι μηχανισμοί ομαδοποίησης ροής είναι:

- **Shuffle grouping:** Τα tuples διανέμονται σε ένα bolt με τρόπο τέτοιο ώστε κάθε executor να δέχεται τον ίδιο αριθμό tuples για επεξεργασία. Η τεχνική αυτή εγγυάται εξισορρόπηση φόρτου μέσα σε ένα bolt.

- **Fields grouping:** Η ροή τεμαχίζεται και αποστέλλεται στους executors με βάση ένα από τα πεδία του tuple. Για παράδειγμα εάν μια ροή ομαδοποιείται με βάση το πεδίο “λέξη”, τότε τα tuples που περιέχουν την ίδια λέξη θα επεξεργαστούν από τον ίδιο executor.

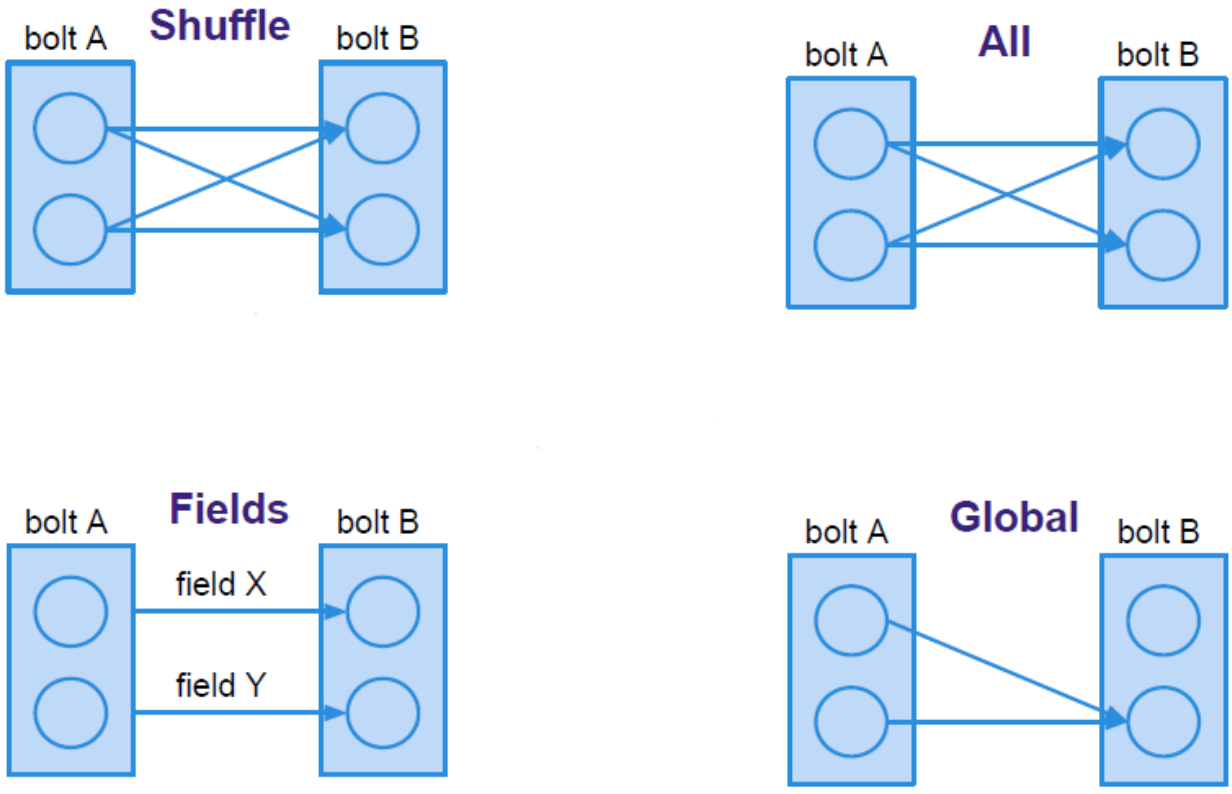
- **All grouping:** Η ροή αντιγράφεται σε όλους τους executors.

- **Global grouping:** Όλη η ροή αποστέλλεται σε έναν μόνο executor.

- **None grouping:** Προς το παρόν λειτουργεί όπως το Suffle grouping.

- **Direct grouping:** Ο παραγωγός του tuple αποφασίζει ποιος executor του bolt θα λάβει το tuple αυτό. Για να γίνει αυτό οι ροές που χρησιμοποιούν αυτόν τον τύπο κατάτμησης πρέπει να δηλωθούν ως direct.

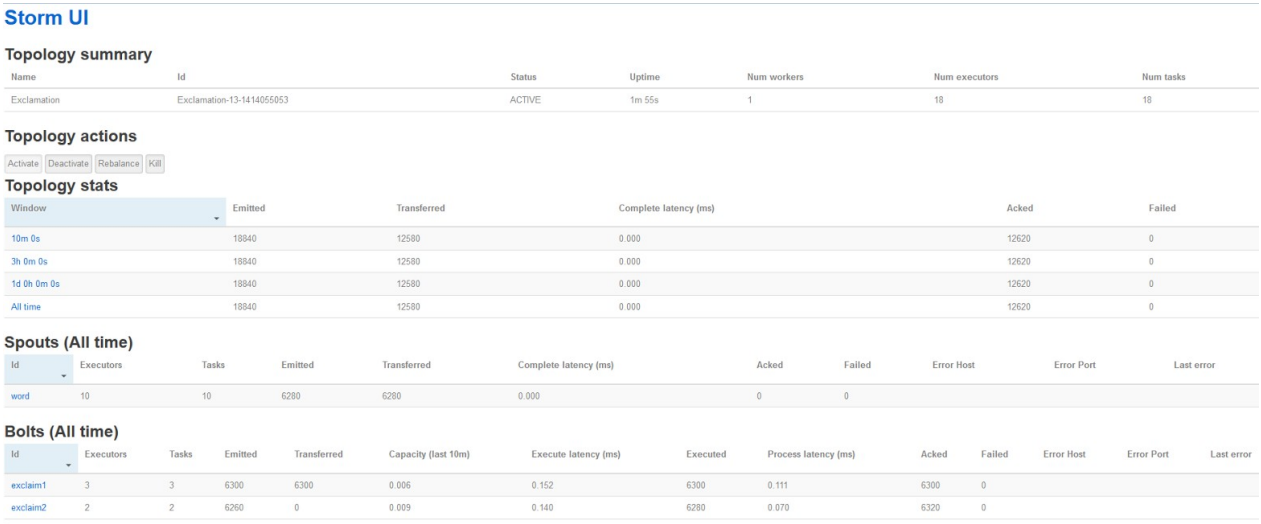
- **Local or shuffle grouping:** Εάν το bolt έχει ένα ή περισσότερα tasks στον ίδιο worker, τα tuples τα εφαρμόσουν την κατάτμηση του Suffle grouping μόνο στα task αυτού του worker.



Σχήμα 3.4: Ομαδοποίηση Ροών στο Storm

3.2.7 Περιβάλλον Χρήσης

Το Storm παρέχει επίσης ένα γραφικό περιβάλλον χρήστη, που μπορεί να χρησιμοποιηθεί για την επίβλεψη των τρεχόντων τοπολογιών, την παρακολούθηση των στατιστικών των εργατών, και την πραγματοποίηση συγκεκριμένων ενεργειών πάνω τους (Ενεργοποίηση, Απενεργοποίηση, Εξισορρόπηση, Σκότωμα, -Activate, Deactivate, Rebalance, Kill-).



Σχήμα 3.5: Storm UI

Οι πληροφορίες που είναι διαθέσιμες στο γραφικό περιβάλλον αφορούν τον αριθμό των πλειάδων που έχουν εκπεμφθεί, εκτελεστεί και επιβεβαιωθεί από όλα τα bolts. Εμφανίζονται τα spout και bolt της τοπολογίας, με τα αναλυτικά στατιστικά τους. Για το κάθε μία από τις συνιστώσες της τοπολογίας μπορούμε να λάβουμε αναλυτικότερες πληροφορίες, για την είσοδο και την έξοδο των πλειάδων και τους executor στους οποίους εκτελείται. Υπάρχει επίσης η δυνατότητα γραφικής αναπαράστασης της τοπολογίας.

3.2.8 Κύκλος Ζωής Τοπολογίας

Μια τοπολογία Storm υποβάλλεται στην συστοιχία από τον Nimbus με την εντολή:

```
storm jar filename.jar mainClass topologyName
```

Αυτή η εντολή θέτει τις μεταβλητές περιβάλλοντος που θα χρησιμοποιηθούν αργότερα από την κλάση που αναλαμβάνει να υποβάλλει την τοπολογία (StormSubmitter).

Ο StormSubmitter ανεβάζει το jar αρχείο στον client του Nimbus, που θα αναλάβει να το υποβάλλει στη συστοιχία. Το αρχείο ανεβαίνει σε κομμάτια των 15KB. Έπειτα, καλεί την συνάρτηση submitTopology, η οποία σειριοποιεί τις ρυθμίσεις της τοπολογίας και εκτελεί την υποβολή της. Μόλις λάβει την επιβεβαίωση ότι η τοπολογία υποβλήθηκε επιτυχώς, επιβεβαιώνει ότι οι ρυθμίσεις είναι ίδιες σε όλους τους κόμβους της τοπολογίας, ώστε να ξεκινήσει να αναθέτει υποεργασίες σε αυτούς. Για να αναθέσεις τις υποεργασίες πρέπει να δημιουργήσει μια αντιστοίχηση μεταξύ των υποεργασιών και των κόμβων που θα τις αναλάβουν. Αυτά, σε συνδυασμό με τις ρυθμίσεις και το αρχείο .jar ορίζουν την στατική κατάσταση της τοπολογίας.

Με την εντολή mk-assignment ο Nimbus αναθέτει στους εργάτες τις υποεργασίες που θα εκτελέσουν. Αρχικά, η τοπολογία είναι σε αδρανή μορφή και περιμένει από τον Nimbus το σήμα για να ξεκινήσει να εκπέμπει πλειάδες από τα spouts. Μόλις ξεκινήσει, δύο διεργασίες Zookeeper του Supervisor, αναλαμβάνουν να κατεβάζουν τον απαιτούμενο κώδικα στους κόμβους εργάτες και να ενημερώνουν τις αναθέσεις των εκτελούμενων υποεργασιών όταν αυτές αλλάζουν. Επίσης, οι εργάτες ξεκινάνε μια διεργασία που είναι υπεύθυνη για τον δικό τους συντονισμό. Αυτή η διεργασία αρχικοποιεί τις συνδέσεις των εργατών, για να ξεκινήσει η επικοινωνία μεταξύ τους, και εκκινεί μια διεργασία- νήμα, η οποία ελέγχει για αλλαγές. Για παράδειγμα, αν κάποιος εργάτης σκοτωθεί και ανατεθεί σε άλλο μηχάνημα, πρέπει να γίνει σύνδεση με την νέα τοποθεσία του μηχανήματος. Παράλληλα, ενημερώνει την κατάσταση της τοπολογίας στη μεταβλητή storm-active-atom, που χρησιμοποιείται από τα spout, ώστε να καθορίσουν αν θα εκπέμψουν επόμενη πλειάδα ή όχι. Τέλος, εκτελεί τις υποεργασίες τρέχοντας τον κώδικα που της αντιστοιχεί.

Κατά τη διάρκεια της ζωής της τοπολογίας, ο Nimbus την επιβλέπει κάνοντας ελέγχους ανά τακτά χρονικά διαστήματα. Σύμφωνα με τη μεταβλητή "nimbus. monitor.freq.secs", που έχει τεθεί κατά την ρύθμιση της τοπολογίας, καλείται η συνάρτηση που ελέγχει αν οι κόμβοι είναι ζωντανοί και επαναλαμβάνει τις αναθέσεις των υποεργασιών, αν είναι απαραίτητο.

Τέλος, με την εντολή kill σκοτώνεται η τοπολογία. Όταν ο Nimbus λάβει αυτή την εντολή, την στέλνει στους κόμβους και δίνει ένα διάστημα αναμονής, ώστε να ολοκληρώσουν τις τρέχουσες διεργασίες. Σε αυτό το διάστημα η κατάσταση της τοπολογίας είναι “σκοτωμένη”, έτσι ώστε να διασφαλιστεί ότι σε περίπτωση αποτυχίας του Nimbus, όταν επανεκκινηθεί η συστοιχία, ο Nimbus θα αναγνωρίσει την σκοτωμένη τοπολογία και θα εκτελέσει τη συνάρτηση αναμονής, πριν τη σκοτώσει οριστικά. Ένα παράλληλο νήμα αναλαμβάνει να καθαρίσει τους φακέλους που χρησιμοποιούνται από τους εργάτες για να δηλώσουν την παρουσία τους καθώς και τους φακέλους στους οποίους αποθηκεύονται τα τμήματα κώδικα και οι ρυθμίσεις.

Κεφάλαιο 4
Συγχρονισμός Εργασιών & Διαχείριση Μηνυμάτων

4.1 ZooKeeper

Το ZooKeeper [8] είναι μία ανοιχτού κώδικα κατανεμημένη υπηρεσία, που παρέχει πληροφορίες, ονοματοδοσία και συγχρονισμό σε κατανεμημένα συστήματα. Όλα τα παραπάνω χαρακτηριστικά αποτελούν βασικότατους παράγοντες επιτυχίας όλων των σύγχρονων κατανεμημένων συστημάτων. Είναι ιδιαίτερα χρήσιμο ως εργαλείο διότι η υλοποίηση και η αποσφαλμάτωση εφαρμογών που εμπλέκουν συγχρονισμό είναι εξαιρετικά επίπονες διαδικασίες για τον προγραμματιστή, λόγω των συνθηκών ανταγωνισμού (race conditions) και των αδιεξόδων (deadlocks) που παρουσιάζονται.

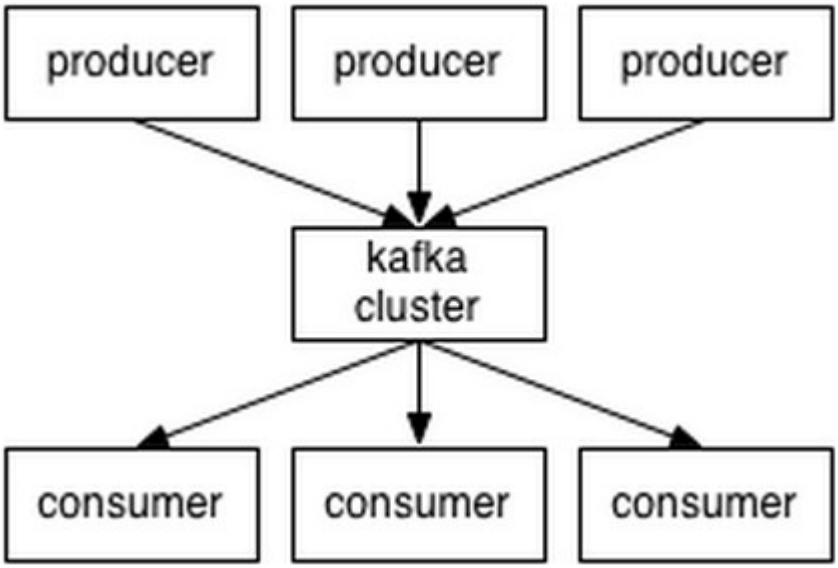
Το ZooKeeper τρέχει σε cluster υπολογιστών και παρέχει πληροφορίες έκδοσης και κατάστασης των δεδομένων (versioning), ταξινόμηση με βάση την ώρα δημιουργίας και τροποποίησης τους, όπως επίσης ένα πολύ εύχρηστο σύστημα ενημερώσεων όταν συμβεί κάποιο μη επιθυμητό γεγονός (για παράδειγμα αν χαθεί η επικοινωνία με κάποιον κόμβο). Κάθε cluster μπορεί να περιλαμβάνει έναν η περισσότερους ZooKeeper εξυπηρετητές, ανάλογα με το μέγεθος του. Όλες οι κατανεμημένες διεργασίες μπορούν και συνεργάζονται μεταξύ τους κάνοντας χρήση των παραπάνω μηχανισμών, αλλά και ενός συστήματος αρχείων δεντρικής μορφής. Όλα τα δεδομένα κρατούνται στην κύρια μνήμη γεγονός που κάνει το ZooKeeper να επιτυγχάνει πολύ υψηλή απόδοση (throughput) στην επικοινωνία. Το ZooKeeper επίσης παρέχει πολύ υψηλή διαθεσιμότητα. Όλοι οι εξυπηρετητές κρατούν στην κύρια μνήμη πληροφορίες κατάστασης όλων των υπόλοιπων κόμβων του συστήματος, μαζί με αρχεία καταγραφής συναλλαγών, με αποτέλεσμα όταν οι περισσότεροι εξυπηρετητές είναι διαθέσιμοι, οι υπηρεσίες του ZooKeeper να είναι διαθέσιμες για τις εφαρμογές που τις χρησιμοποιούν.

4.2 Kafka

Το Kafka [9] είναι ένα ανοιχτού κώδικα κατανεμημένο σύστημα διαχείρισης μηνυμάτων (Messaging System), σχεδιασμένο να μπορεί να επεξεργάζεται πάρα πολύ μεγάλο όγκο δεδομένων, είτε αυτός προέρχεται από ροές (streams) είτε από δέσμες δεδομένων (batches). Αποτελεί ένα message broker. Είναι δηλαδή ένα ενδιάμεσο πρόγραμμα το οποίο μεταφράζει ένα μήνυμα από το επίσημο πρωτόκολλο μηνυμάτων του αποστολέα στο επίσημο πρωτόκολλο μηνυμάτων του δέκτη. Είναι ένα project ανοιχτού κώδικα το οποίο αρχικά σχεδιάστηκε από την LinkIn ως ένα κατανεμημένο σύστημα μηνυμάτων (messaging) για συλλογή και μεταφορά Log αρχείων με ελάχιστη καθυστέρηση (low latency) και high throughput σε πραγματικό χρόνο. Εκτός από την ίδια την εταιρία χρησιμοποιείται ευρέως από πολλές άλλες κορυφαίες του χώρου, όπως το Twitter και το Spotify. Πλέον, την ανάπτυξη του Kafka την έχει αναλάβει η Apache.

Το σύστημα λειτουργεί με βάση το μοντέλο παραγωγού-καταναλωτή (producer-consumer) και συντηρεί μηνύματα ομαδοποιημένα σε κατηγορίες οι οποίες ονομάζονται topics. Κάθε topic είναι στην ουσία μια ουρά μηνυμάτων. Ένας παραγωγός δημοσιεύει μηνύματα σε μια συγκεκριμένη κατηγορία και όσοι καταναλωτές είναι εγγεγραμμένοι στην κατηγορία αυτή λαμβάνουν το δημοσιευμένο μήνυμα. Το Kafka δουλεύει σε cluster υπολογιστών, αποτελούμενο από έναν ή περισσότερους εξυπηρετητές, κάθε ένας από τους οποίους ονομάζεται broker. Ως εκ τούτου σε ένα τέτοιο κατανεμημένο σύστημα οι παραγωγοί στέλνουν μηνύματα στο cluster μέσω δικτύου, αυτά αποθηκεύονται σε ουρές των brokers και στη συνέχεια διατίθενται στους καταναλωτές που ενδιαφέρονται για αυτά.

Το Kafka βρίσκει χρήση στο messaging καθώς χαρακτηριστικά όπως high throughput, τεμαχισμός αρχείων (built-in partitioning), διατήρηση αντιγράφων (replication) και η ανοχή σε σφάλματα (fault-tolerance) το κάνουν ιδανική λύση σε εφαρμογές επεξεργασίας μηνυμάτων (message processing applications) μεγάλης κλίμακας. Συχνά χρησιμοποιείται για παρακολούθηση (monitoring) δεδομένων συγκεντρώνοντας στατιστικά και δεδομένα χρήσης από κατανεμημένες εφαρμογές.



Σχήμα 4.1: Kafka Architecture

Είναι πλήρως αποκεντρωμένο και χρησιμοποιεί το ZooKeeper και τους μηχανισμούς που παρέχονται από αυτό για να εγγυηθεί συγχρονισμό, υψηλή απόδοση και εξισορρόπηση φόρτου στους brokers. Οι παραγωγοί και οι καταναλωτές λαμβάνουν γνώση για τους διαθέσιμους brokers μέσω του ZooKeeper. Στα βασικά χαρακτηριστικά του σχεδιασμού περιλαμβάνεται η προσωρινή ή αποθήκευση μηνυμάτων (caching), η δυνατότητα να ξανά καταναλωθούν μηνύματα (re-consumption), η ομαδοποίηση μηνυμάτων (για μείωση δικτυακού φόρτου), η συμπίεση μηνυμάτων κ.α.

Κεφάλαιο 5
Cluster, Dataset, Τοπολογίες, Αποτελέσματα

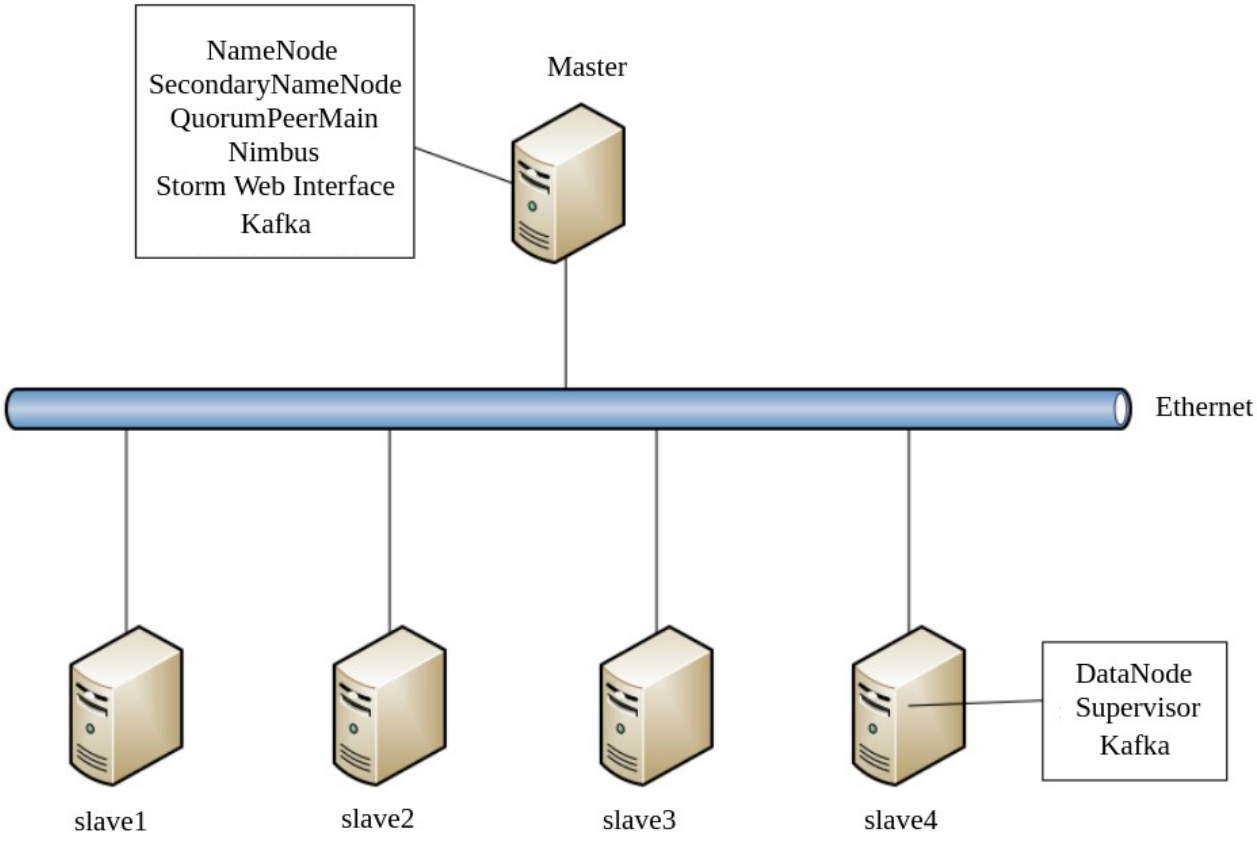
5.1 Cluster

Για την εκτέλεση των πειραμάτων δημιουργήσαμε ένα cluster από υπολογιστές χρησιμοποιώντας την υποδομή του Εργαστηρίου Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, του Εθνικού Μετσόβιου Πολυτεχνείου. Συγκεκριμένα, το cluster μας αποτελείται από 5 εικονικές μηχανές: 1 master και 4 slaves κόμβους. Οι δυνατότητες ανά κόμβο είναι οι εξής:

CPU cores	4
CPU Speed	2400 MHz
RAM	8 GB
Disk	10 GB
Volume	100 GB

Πίνακας 5.1: Χαρακτηριστικά Κόμβων του Cluster

Στον master κόμβο τρέχουν οι master διεργασίες των Hadoop (HDFS), ZooKeeper και Storm. Δηλαδή οι NameNode και SecondaryNameNode του Hadoop (HDFS), ο QuorumPeerMain του ZooKeeper, οι Nimbus και Storm Web Interface για το Storm και ο Kafka για το Kafka (Kafka Producer). Αντίστοιχα, στους slaves τρέχουν ο DataNode για το Hadoop (HDFS), ο Supervisor για το Storm και ο Kafka για το Kafka (Kafka Consumer). Αυτά φαίνονται εποπτικά στο σχήμα 5.1.



Σχήμα 5.1: Διάγραμμα Παράταξης Cluster

5.2 Dataset

Τα δεδομένα που χρησιμοποιήθηκαν σε αυτή τη διπλωματική προέρχονται από τον ελληνικό κόμβο διασύνδεσης GR-IX. Το GR-IX είναι ένα Internet Exchange Point, δηλαδή ένας κόμβος στον οποίο συνδέονται όλοι οι πάροχοι που δρουν στην ελληνική επικράτεια για να μπορούν να ανταλλάσουν μεταξύ τους δικτυακή κίνηση χωρίς να είναι απαραίτητο να την δρομολογούν μέσω τρίτων δικτύων. Ο κόμβος αυτός διαχειρίζεται από το ΕΔΕΤ το οποίο είναι υπεύθυνο για τη συντήρηση και την παροχή των υπηρεσιών του.

Όπως είναι φυσικό από τον κόμβο αυτό περνάει ένα πολύ μεγάλο ποσοστό της δικτυακής κίνησης και τα δεδομένα που προκύπτουν είναι της τάξης των πολλών terra bytes. Κάθε μέρα με τη βοήθεια του εργαλείου sFlow γίνεται δειγματοληψία των πακέτων IP που διέρχονται από αυτό τον κόμβο ανά τακτά χρονικά διαστήματα. Το εργαλείο αυτό αποθηκεύει τα πακέτα που συλλαμβάνει σε μία συγκεκριμένη μορφή, κρατώντας αρκετές πληροφορίες σχετικές με τον αποστολέα και τον παραλήπτη του πακέτου, αλλά και με το ίδιο το πακέτο (μέγεθος πακέτου, timestamp). Τα ανωνυμοποιημένα δεδομένα που χρησιμοποιήσαμε καλύπτουν μία χρονική περίοδο από τα τέλη Ιουλίου του 2013 έως τα μέσα Φεβρουαρίου του 2014. Το συνολικό μέγεθος των δεδομένων που ήταν διαθέσιμα ήταν περίπου 200 GB σε συμπιεσμένη μορφή. Για τους σκοπούς της εργασίας αυτής, επειδή δεν μας ήταν απαραίτητα όλα τα πεδία που προσφέρει το sFlow, κρατήσαμε ορισμένα από αυτά που μας ήταν χρήσιμα. Αυτά τα πεδία είναι:

- Source IP address
- Destination IP address
- Protocol type
- Source port number
- Destination port number
- Total size of packet
- Timestamp

Το μέγεθος των δεδομένων μετά από την επιλογή αυτών των πεδίων είναι περίπου 50 GB σε συμπιεσμένη μορφή, ενώ αυτά απαριθμούνται περίπου ως δύο δισεκατομμύρια εγγραφές.

Σκοπός της ανάλυσής μας είναι να συνδυάσουμε την πληροφορία αυτή με επιπλέον σύνολα δεδομένων που περιέχουν meta-πληροφορίες όπως την κατάταξη της κάθε IP στο αυτόνομο σύστημα (AS) που ανήκει ή την αντιστοίχισή της με τη χώρα προέλευσης. Η πληροφορία αυτή συλλέχτηκε από τη βάση δεδομένων GeoLite [2], μία geolocation βάση δεδομένων που χρησιμεύει για την αντιστοίχιση των διευθύνσεων IP στο αυτόνομο σύστημα που ανήκουν και στη χώρα προέλευσης τους. Τα μεγέθη αυτών των αρχείων είναι 12 MB για το αρχείο που περιέχει τα αυτόνομα συστήματα και 7 MB για το αρχείο με τις χώρες. Τα δεδομένα αυτά ανανεώνονται στις αρχές κάθε μήνα, οπότε μπορούμε να χρησιμοποιούμε διαφορετικά meta-data αναλόγως της ημερομηνίας που προέρχονται τα αρχικά δεδομένα μας. Για το μεν αρχείο με τα αυτόνομα συστήματα τα δεδομένα του αρχείου έχουν την μορφή:

- Lower bound of IP range integer
- Upper bound of IP range integer
- AS Name

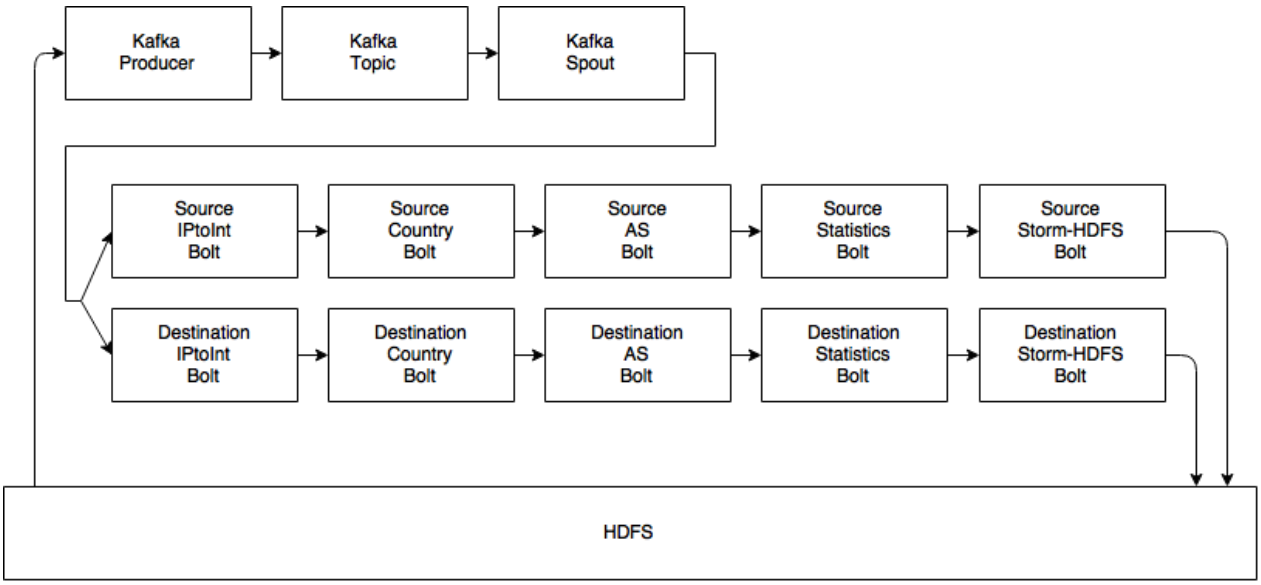
Για το δε αρχείο με τις χώρες τα δεδομένα του αρχείου έχουν την μορφή:

- Lower bound of IP range address
- Upper bound of IP range address
- Lower bound of IP range integer
- Upper bound of IP range integer
- Country Abbreviation Name
- Country Full Name

5.3 Τοπολογίες

5.3.1 Αρχιτεκτονική Συστήματος

Στην ενότητα αυτή παρουσιάζουμε τις τοπολογίες που δημιουργήσαμε για κατανεμημένη επεξεργασία των δεδομένων και όλα τα επιμέρους δομικά τους στοιχεία. Στο σχήμα 5.2 φαίνεται η αρχιτεκτονική του συστήματος μας:



Σχήμα 5.2: Αρχιτεκτονική Κατανεμημένου Συστήματος

Το σύστημα μας αποτελείται από 1 Spout και 10 Bolts. Πάνω στην συγκεκριμένη αρχιτεκτονική θα πειραματιστούμε μεταβάλλοντας τον αριθμό των κόμβων και το επίπεδο παραλληλισμού, ώστε να διαπιστώσουμε αν, και κατά πόσο, επιτυγχάνουμε κλιμακωσιμότητα και αύξηση της ταχύτητας επεξεργασίας των δεδομένων μας (throughput). Πρώτα όμως, θα αναλύσουμε μία προς μία όλες τις οντότητες που αποτελούν τα επιμέρους δομικά στοιχεία της παραπάνω αρχιτεκτονικής.

5.3.2 Οντότητες Τοπολογιών

Το Spout της τοπολογίας ονομάζεται KafkaSpout. Το Spout αυτό αναλαμβάνει τον ρόλο του καταναλωτή του Kafka. Ο Kafka-Producer διαβάζει δεδομένα από το HDFS και τα γράφει στο Topic του Kafka. Τα δεδομένα στην συνέχεια τα παραλαμβάνει το KafkaSpout όπου και τα εκπέμπει στην τοπολογία. Η αποστολή των πλειάδων από το KafkaSpout στο πρώτο Bolt γίνεται με Shuffle grouping ομαδοποίηση ροής, γεγονός που εγγυάται την εξισορρόπηση φόρτου της εισερχόμενης πληροφορίας. Η εξισορρόπηση φόρτου είναι σημαντική, διότι μας ενδιαφέρει όλο οι executors να επεξεργάζονται τον ίδιο όγκο δεδομένων, ανεξάρτητα από το ρυθμό που αποστέλλονται από κάθε παραγωγό του Kafka.

Οι πλειάδες που εκπέμπει το KafkaSpout διαβάζονται από δύο Bolt (SourceIPtoIntBolt και DestinationIPtoIntBolt) τα οποία είναι οι κεφαλίδες δύο παράλληλων υπό-ροών (ροή Source και ροή Destination). Κάθε μία από αυτές τις ροές επεξεργάζεται αποκλειστικά τα πεδία των πακέτων που αφορούν είτε την πηγή, είτε τον προορισμό αντίστοιχα. Επιδιώκουμε όσο το δυνατό μεγαλύτερο παραλληλισμό για το σύστημά μας, γι' αυτό το λόγο εφαρμόζουμε όσο το δυνατόν νωρίτερα διαχωρισμό στα πεδία των δεδομένων μας τα οποία λογίζονται ως ανεξάρτητα μεταξύ τους στην διαμόρφωση του τελικού αποτελέσματος. Στις παραγράφους που ακολουθούν δεν θα γίνει ιδιαίτερη μνεία μεταξύ των Bolt που εκτελούν αντίστοιχες διεργασίες για τα πεδία πηγής και προορισμού. Αντίθετα, παρουσιάζουμε τις λειτουργίες που εκτελεί κάθε Bolt χωρίς να αναφέρουμε το διακριτικό (Source ή Destination) της υπό-ροής στην οποία υπάγεται.

Το πρώτο Bolt ονομάζεται IptoIntBolt. Πραγματοποιεί την αντικατάσταση των IP διευθύνσεων των πλειάδων, με ένα μοναδικό ακέραιο αριθμό μέσω της συνάρτησης μετατροπής. Επίσης, ως πρώτο Bolt της εκάστοτε υπό-ροής, πραγματοποιεί φιλτράρισμα μεταξύ των πεδίων των πλειάδων. Σκοπός είναι να τροφοδοτεί τα επόμενα Bolt της ίδιας υπό-ροής μόνο με τα πεδία που αφορούν την πηγή (ή τον προορισμό, αντίστοιχα) και τα κοινά μεταξύ τους πεδία. Ένα παράδειγμα της συνάρτησης μετατροπής η οποία εκτελείται σε αυτό το Bolt παρουσιάζεται παρακάτω:

```
IP Address:    78.87.23.215

1st Octet:     78
2nd Octet:     87
3rd Octet:     23
4th Octet:     215

  (1st octet * 256³)
+ (2nd octet * 256²)
+ (3rd octet * 256)
+ (4th octet)

= (78 * 16777216)
+ (87 * 65536)
+ (23 * 256)
+ (215)

= 1314330583
```

Οι λόγοι αυτής μετατροπής είναι κυρίως δυο και είναι εξίσου σημαντικοί: Κατ' αρχάς, στην μορφή που βρίσκονται τα δεδομένα μας, δεν θα μπορούσε να γίνει αντιστοίχηση μεταξύ IP διεύθυνσης και αυτόνομου συστήματος. Θυμίζουμε από την προηγούμενη ενότητα, ότι η meta-πληροφορία των αυτόνομων συστημάτων αντιστοιχεί το εύρος των ακεραίων οι οποίοι δημιουργούνται από τις IP διευθύνσεις και την συνάρτηση μετατροπής, με το αυτόνομο σύστημα. Θέλουμε λοιπόν είτε να αλλάξουμε την μορφή των δεδομένων μας, είτε να αλλάξουμε την μορφή της meta-πληροφορίας. Ο λόγος που επιλέγουμε να αλλάξουμε την μορφή των δεδομένων μας είναι γιατί αυτό (όπως θα δούμε παρακάτω) θα μας βοηθήσει σημαντικά στα δύο επόμενα Bolt (CountryBolt και ASBolt), ώστε να μειώσουμε δραματικά την πολυπλοκότητα του αλγορίθμου για την αντιστοίχηση IP διεύθυνσης – αυτόνομου συστήματος και χώρας προέλευσης. Θα είναι πολύ πιο εύκολο να αναζητούμε ουσιαστικά έναν ακέραιο μέσα σε ένα ταξινομημένο εύρος τιμών, παρά να προσπαθούμε να ταιριάξουμε strings. Εκτός των άλλων, η αντικατάσταση IP διεύθυνσης με τον αντίστοιχο ακέραιο, ελαφρώνει σημαντικά τις πλειάδες μας, δημιουργώντας εγγραφές μικρότερου όγκου μνήμης.

Το δεύτερο Bolt ονομάζεται CountryBolt. Το συγκεκριμένο Bolt είναι υπεύθυνο για την αντιστοίχηση IP διεύθυνσης και χώρας προέλευσης. Η αντιστοίχηση αυτή γίνεται ως εξής: Το συγκεκριμένο Bolt έχει στην τοπική του μνήμη έναν πίνακα με όλα τα άνω όρια από τα εύρη των διευθύνσεων των χωρών και έναν δεύτερο πίνακα, ίδιου μεγέθους, με τα ονόματα των χωρών που αντιστοιχούν στον πρώτο πίνακα. Τα άνω όρια είναι ταξινομημένα, μεταξύ τους και εμείς πραγματοποιούμε δυαδική αναζήτηση μεταξύ αυτών των εγγραφών του πίνακα για να εντοπίσουμε την επιθυμητή χώρα προέλευσης. Το παραπάνω φαίνεται εποπτικά στο ψευδοκώδικα 1.

```
for each tuple do
{
     down = 0
     up = CountriesUpperBoundArray.length
     mid = -1

     while ((up – down) != 1)
     {
          mid = (up + down) / 2

          if ( ip > CountriesUpperBoundArray[mid])
               down = mid
          else if ( ip < CountriesUpperBoundArray[mid])
               up = mid
          else
          {
               up = mid
               break
          }
     }

     tuple.emit(ip + “ ” + CountriesNamesArray[mid])
}
```

Ψευδοκώδικας 1: CountryBolt

Το τρίτο Bolt ονομάζεται ASBolt. Το συγκεκριμένο Bolt είναι υπεύθυνο για την αντιστοίχηση IP διεύθυνσης και αυτόνομου συστήματος. Εργάζεται αντίστοιχα όπως το CountryBolt, χρησιμοποιώντας και αυτό δύο πίνακες (τον πίνακα των άνω ορίων από τα εύρη διευθύνσεων και τον πίνακα με τα αυτόνομα συστήματα που τους αντιστοιχούν). Πραγματοποιεί και αυτό δυαδική αναζήτηση μεταξύ αυτών των εγγραφών του πρώτου πίνακα για να εντοπίσει το επιθυμητό αυτόνομο σύστημα. Το παραπάνω φαίνεται εποπτικά στο ψευδοκώδικα 2.

```
for each tuple do
{
     down = 0
     up = ASesUpperBoundArray.length
     mid = -1

     while ((up – down) != 1)
     {
          mid = (up + down) / 2

          if ( ip > ASesUpperBoundArray[mid])
               down = mid
          else if ( ip < ASesUpperBoundArray[mid])
               up = mid
          else
          {
               up = mid
               break
          }
     }

     tuple.emit(ip + “ ” + ASesNamesArray[mid])
}
```

Ψευδοκώδικας 2: ASBolt

Σε αυτό το σημείο φαίνεται το συγκριτικό πλεονέκτημα της χρήσης των ακεραίων (μέσω της συνάρτησης μετατροπής) σε σχέση με τις IP διευθύνσεις. Η αλγοριθμική πολυπλοκότητα των δύο τελευταίων Bolt (CountryBolt και ASBolt), υπεύθυνων για την συνένωση των πληροφοριών είναι αντίστοιχα:

```
Μέγεθος Εισόδου * log (Μέγεθος πίνακα χωρών)
Μέγεθος Εισόδου * log (Μέγεθος πίνακα αυτόνομων συστημάτων)
```

Το τέταρτο Bolt της τοπολογίας μας ονομάζεται StatisticsBolt. Το συγκεκριμένο Bolt αναλαμβάνει να καταγράψει στατιστικά της εξερχόμενης (ή εισερχόμενης, αντίστοιχα) κίνησης των ISP. Το ζευγάρι των δύο αυτών Bolt μετράει ανά χώρα:

- Τον απόλυτο αριθμό πακέτων που είχαν ως προέλευση/προορισμό την συγκεκριμένη χώρα.
- Το δημοφιλέστερο δίκτυο προέλευσης/προορισμού πακέτων της χώρας αυτής.
- Τον απόλυτο αριθμό πακέτων που εξέπεμψε/δέχτηκε το παραπάνω δίκτυο.
- Το ποσοστό των πακέτων που εξέπεμψε/δέχτηκε το παραπάνω δίκτυο σε σχέση με τον συνολικό αριθμό πακέτων που εξέπεμψε/δέχτηκε η συγκεκριμένη χώρα.
- Το πιο συνηθισμένο πρωτόκολλο επικοινωνίας που χρησιμοποίησαν τα πακέτα το οποία εξεπέμφθησαν/απεστάλησαν από την χώρα αυτή.
- Τον απόλυτο αριθμό πακέτων με το παραπάνω πρωτόκολλο, που εξέπεμψε/δέχτηκε η συγκεκριμένη χώρα.
- Το ποσοστό των πακέτων που εξέπεμψε/δέχτηκε με το παραπάνω πρωτόκολλο η συγκεκριμένη χώρα, σε σχέση με τον συνολικό αριθμό πακέτων που εξέπεμψε/δέχτηκε η χώρα αυτή.
- Το μέσο μέγεθος πακέτου από/προς την συγκεκριμένη χώρα.

Το παραπάνω φαίνεται εποπτικά στο ψευδοκώδικα 3:

```
for each tuple do
{
      countryPackets[Country]++
      packetSizeSum[Country] += packetSize

      AScounter= ASHashMap.get(Country+" "+ AS)

      if (AScounter == null)
            AScounter = 0
      AScounter++

      if (AScounter > bestAS[Country])
      {
            bestAS[Country] = AScounter
            bestASName[Country] = AS
      }
      ASHashMap.put(Country+" "+AS, AScounter)

      ProtCounter = ProtHashMap.get(Country+" "+ protocol)

      if (ProtCounter == null)
            ProtCounter = 0
      ProtCounter++

      if (ProtCounter > mCommonProt[Country])
      {
            mCommonProt[Country] = ProtCounter
            mCommonProtName[Country] = protocol
      }
      ProtHashMap.put(Country+" "+protocol, ProtCounter)

      tuple.emit(Country+“ ”+countryPackets[Country]+“ ”+
         bestASName[Country]+“ ”+bestAS[Country]+“ ”+
         (bestAS[Country] / countryPackets[Country])+“ ”+
         mCommonProtName[Country]+“ ”+mCommonProt[Country]+“ ”+
         (mCommonProt[Country] / countryPackets[Country])+“ ”+
         (packetSizeSum[Country] / countryPackets[Country]))

}
```

Ψευδοκώδικας 3: StatisticsBolt

Το τελευταίο Bolt της τοπολογίας ονομάζεται Storm-HDFSBolt. Το συγκεκριμένο Bolt αποθηκεύει τα αποτελέσματα της επεξεργασίας των δεδομένων στο κατανεμημένο σύστημα αρχείων HDFS.

5.3.3 Παραλληλισμός Τοπολογιών

Σε αυτό το σημείο θα πειραματιστούμε με το επίπεδο παραλληλισμού της τοπολογίας. Ο λόγος είναι για να δούμε αν, και κατά πόσο, επιτυγχάνουμε κλιμακωσιμότητα και αύξηση της ταχύτητας επεξεργασίας των δεδομένων μας (throughput).

Έχοντας γνώση του αριθμού των διαθέσιμων executors όλου του cluster καθορίζεται το επίπεδο παραλληλισμού κάθε ενός στοιχείου. Με άλλα λόγια ορίζεται ο αριθμός των executors που θα τρέχουν παράλληλα και θα αναλαμβάνουν tasks για το συγκεκριμένο στοιχείο. Ο αριθμός των διαθέσιμων executors στο cluster ορίζεται ως εξής:

$$Executors = Storm\,Servers \times Workers\,Per\,Server \times Executors\,Per\,Worker$$

Για να υπάρχει ισοκατανομή φόρτου εργασιών μεταξύ των κόμβων σε όλες τις τοπολογίες που θα πραγματοποιήσουμε, θα δημιουργήσουμε ίδιο αριθμό νημάτων (threads) σε όλους τους κόμβους της εκάστοτε τοπολογίας. Θα πειραματιστούμε με τον αριθμό των κόμβων (Storm Servers) και κατά συνέπεια με τον συνολικό αριθμό των workers και την αναλογία executors ανά worker. Θα δοκιμάσουμε τρεις παραλλαγές της ίδιας τοπολογίας, με έναν, δύο και τέσσερις κόμβους αντίστοιχα (Τοπολογία 1, Τοπολογία 2, Τοπολογία 3). Σε κάθε μία από τις παραπάνω τοπολογίες θα έχουμε ενεργοποιημένο μόνο τον αντίστοιχο αριθμό κόμβων-Supervisor. Ο αριθμός των workers για κάθε κόμβο Supervisor θα παραμείνει σταθερός σε όλες τις τοπολογίες και ίσος με 4. Σε αυτό το σημείο λαμβάνουμε υπ' όψη μας τον αριθμό των επεξεργαστών που έχει κάθε Supervisor υπολογιστής (4 CPUs). Θέλουμε δηλαδή ένα worker ανά CPU, από όσες CPUs εργάζονται στην τοπολογία μας. Αυτό συνεπάγεται ότι η Τοπολογία 1 θα έχει αθροιστικά 4 CPUs, η Τοπολογία 2 θα έχει 8 CPUs, ενώ η Τοπολογία 3 θα έχει 16 CPUs. Τέλος, για να μπορέσουμε να έχουμε ίδιο αριθμό νημάτων σε κάθε CPU της ίδιας τοπολογίας, αναζητούμε το ελάχιστο κοινό πολλαπλάσιο μεταξύ του 4, του 8 και του 16, το οποίο φυσικά θα υπερβαίνει το 11 που είναι ο αριθμός των στοιχείων της τοπολογίας (δηλαδή χρειαζόμαστε κατ' ελάχιστον ένα νήμα για κάθε στοιχείο). Ο αριθμός που αναζητάμε είναι φυσικά το 16, οπότε η Τοπολογία 1 θα έχει 4 executors ανά worker, η Τοπολογία 2 θα έχει 2 executors ανά worker και η Τοπολογία 3 θα έχει 1 executor ανά worker.

Από προεπιλογή το Storm ορίζει έναν επιπλέον executor ανά worker με όνομα Acker. Ο Acker πραγματοποιεί επιβεβαίωση των πλειάδων (tuples) που ελήφθησαν επιτυχώς. Μπορούμε να ορίσουμε τον απόλυτο αριθμό των Ackers στην τοπολογίας, ο οποίος μπορεί να είναι και μηδενικός. Στην περίπτωση όμως μηδενικού αριθμού Ackers, οι πλειάδες που εκπέμπει κάποιο Spout, θα σταματήσουν να παρακολουθούνται, αν απέτυχαν σε κάποιο από τα επόμενα στάδια επεξεργασίας τους, γεγονός το οποίο μειώνει αποτελεσματικά την αξιοπιστία του συστήματός μας. Για τον λόγο αυτό θα κρατήσουμε τον αριθμό των Ackers ίσο με αυτόν της προεπιλογής.

	Τοπολογία 1	**Τοπολογία 2**	**Τοπολογία 3**
Storm Servers	1	2	4
Workers / Server	4	4	4
Executors / Worker	5	3	2
Total Threads	**20**	**24**	**32**

Πίνακας 5.2: Παραμετροποίηση Τοπολογιών

# of Threads	**Τοπολογία 1**	**Τοπολογία 2**	**Τοπολογία 3**
KafkaSpout	2	2	2
IPtoIntBolt	6	6	6
CountryBolt	2	2	2
ASBolt	2	2	2
StatisticsBolt	2	2	2
Storm-HDFSBolt	2	2	2
Ackers	4	8	16
Total	**20**	**24**	**32**

Πίνακας 5.3: Thread ανά στοιχείο

5.4 Αποτελέσματα

Στην ενότητα αυτή θα παρουσιάζουμε και θα σχολιάσουμε τα αποτελέσματα των μετρήσεων μας, αλλά και τις επιδόσεις του συστήματος μας βάσει των τριών διαφορετικών παραμετροποιήσεων ως προς το επίπεδο παραλληλισμού της τοπολογίας.

Για αρχή παρατηρούμε το εντυπωσιακό φάσμα διευθύνσεων που έχουμε καταγράψει παγκοσμίως. Συγκεκριμένα, από τις 249 χώρες οι οποίες αυτή την στιγμή υφίστανται, στα δεδομένα μας εμπεριέχουν πακέτα τα οποία απευθύνονται στις 228 από αυτές τις χώρες.

Την μερίδα του λέοντος καταλαμβάνουν τα πακέτα της εσωτερικής κίνησης, δηλαδή με αφετηρία και προορισμό την Ελλάδα. Το ποσοστό των πακέτων που αναφέρεται στην εσωτερική κίνηση είναι της τάξης του 99,27%.

Η κίνηση πακέτων από το εξωτερικό προς την Ελλάδα είναι 1,5 μεγαλύτερη σε σχέση με την κίνηση από την Ελλάδα προς το εξωτερικό. Συγκεκριμένα, η εξερχόμενη κίνηση ανέρχεται περίπου στο 0,44% επί του συνόλου των πακέτων, ενώ η εξερχόμενη κίνηση περίπου στο 0,29%.

Επίσης παρατηρείται ένα μικρό ποσοστό διερχόμενης κίνησης, δηλαδή πακέτα τα οποία δεν έχουν ούτε ως αφετηρία, ούτε ως προορισμό την Ελλάδα. Τα πακέτα συνήθως δρομολογούνται από και προς δίκτυα γειτονικών χώρων και μεγαλυτέρων χώρων.

Γίνεται επίσης αντιληπτό, από την εξερχόμενη, την εισερχόμενη και την διερχόμενη κίνηση η παρουσία άλλων μεγάλων IXP κυρίως προς την κεντρική Ευρώπη.

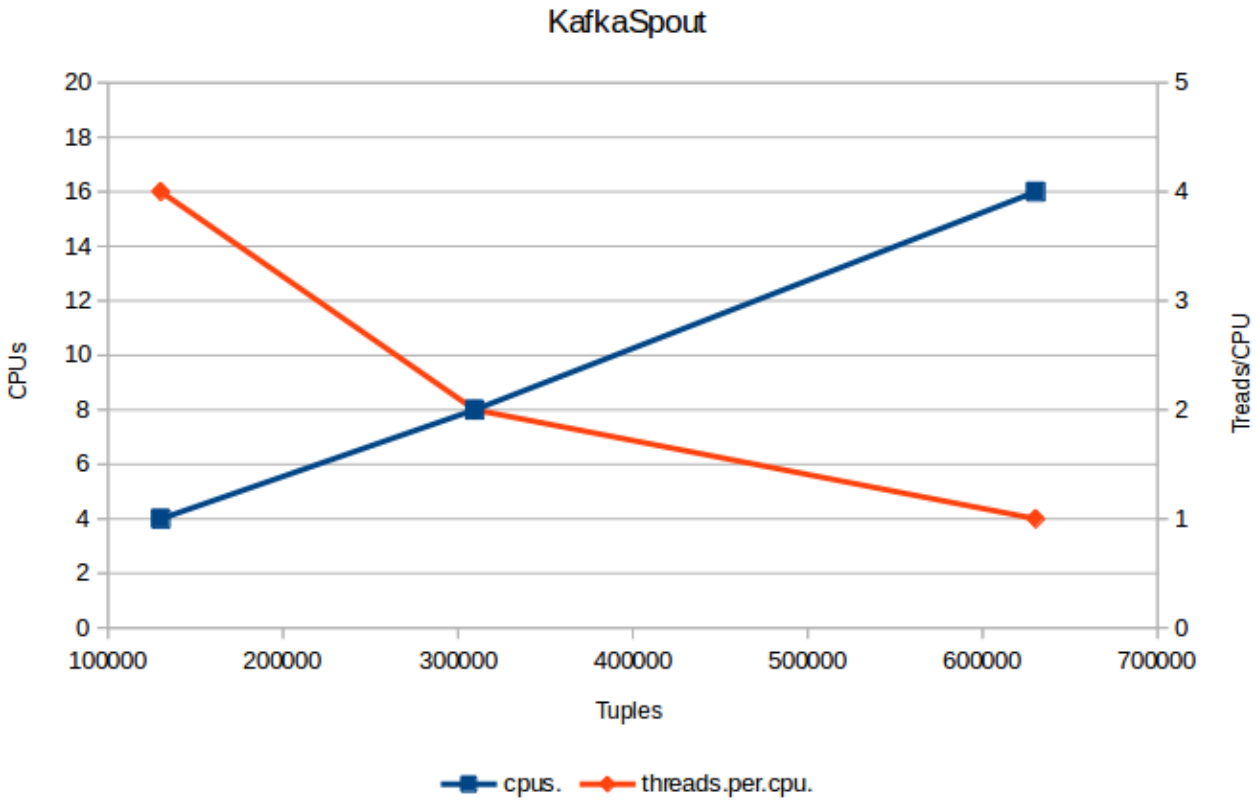
Η πλειοψηφία των πακέτων (πάνω από 70%) είναι πρωτοκόλλου UDP, ενώ τα πακέτα με IPv6 διευθύνσεις κυμαίνονται περίπου στο 0,03% επί του συνόλου.

Συγκεκριμένα στοιχεία ως προς τον κορυφαίο ISP ανά χώρα και το ποσοστό διείσδυσης του στην συγκεκριμένη αγορά, δεν μπορούμε να δώσουμε λόγω δεοντολογίας, αλλά με μια γρήγορη ματιά στα αποτελέσματα μπορούμε να καταλάβουμε ότι τα αποτελέσματα ήταν τα αναμενόμενα για την πλειοψηφία των χωρών.

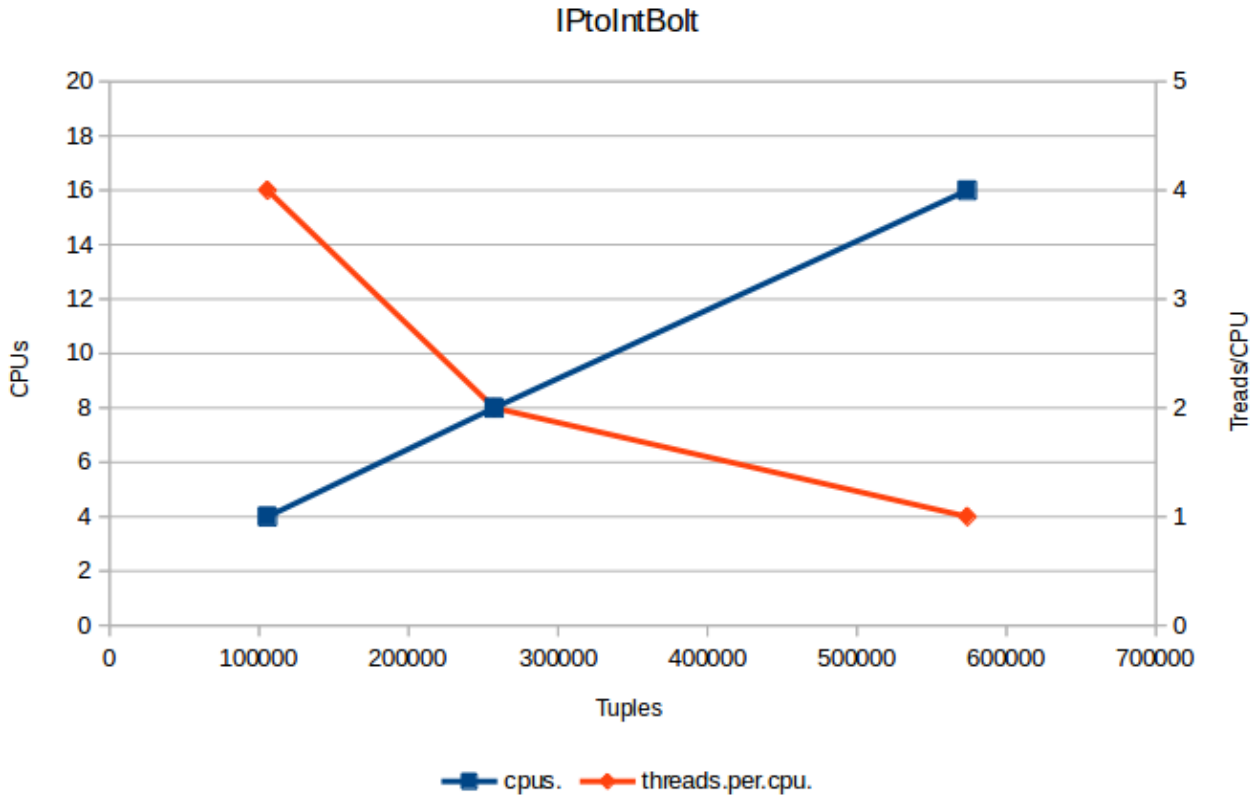
Όσον αφορά την επίδοση του συστήματος μας, παρουσιάζουμε στον πίνακα 5.4 συγκεκριμένα στοιχεία από την εκτέλεση κάθε τοπολογίας. Σε αυτό το σημείο να σημειώσουμε ότι, καθώς το Storm αποτελεί υλοποίηση κατανεμημένου συστήματος το οποίο δέχεται μία αέναη ροή δεδομένων, θα ήταν άστοχο να προσπαθήσουμε να μετρήσουμε απόλυτους χρόνους ως προς την επεξεργασία του συνόλου των δεδομένων. Αυτό που κάνουμε είναι να μετρήσουμε μέσους χρόνους καθυστέρησης ανά βαθμίδα επεξεργασίας και μέσω ρυθμό επεξεργασίας δεδομένων (throughput). Τα παραπάνω μετρούνται αφού το σύστημα μας φτάσει σε “μόνιμη κατάσταση”, δηλαδή σε χρόνο περίπου 10 λεπτών μετά την εκκίνηση της κάθε τοπολογίας, χρόνος ο οποίος επισημαίνεται μέσω του Storm Web Interface το οποίο μας παρέχεται.

	Τοπολογία 1	**Τοπολογία 2**	**Τοπολογία 3**
Tuples emitted by **KafkaSpout** per min	130210	309614	630400
Tuples emitted by KafkaSpout per thread, min	65105	154807	315200
Tuples executed by **IPtoIntBolt** per min	105678	257288	573883
Tuples executed by IPtoIntBolt per thread, min	35226	85763	191294
IPtoIntBolt process latency (ms)	0,529	0,381	0,188
Tuples executed by **CountryBolt** per min	58213	128796	178428
Tuples executed by CountryBolt per thread, min	58213	128796	178428
CountryBolt process latency (ms)	0,432	0,295	0,203
Tuples executed by **ASBolt** per min	37374	71250	144682
Tuples executed by ASBolt per thread, min	37374	71250	144682
ASBolt process latency (ms)	0,645	0,429	0,253
Tuples executed by **StatisticsBolt** per min	21184	71093	136790
Tuples executed by StatisticsBolt per thread, min	21184	71093	136790
StatisticsBolt process latency (ms)	0,566	0,365	0,261
Tuples executed by **Storm-HDFSBolt** per min	15573	71088	122150
Tuples executed by Storm-HDFSBolt per thread, min	15573	71088	122150
Storm-HDFSBolt process latency (ms)	0,227	0,241	0,287

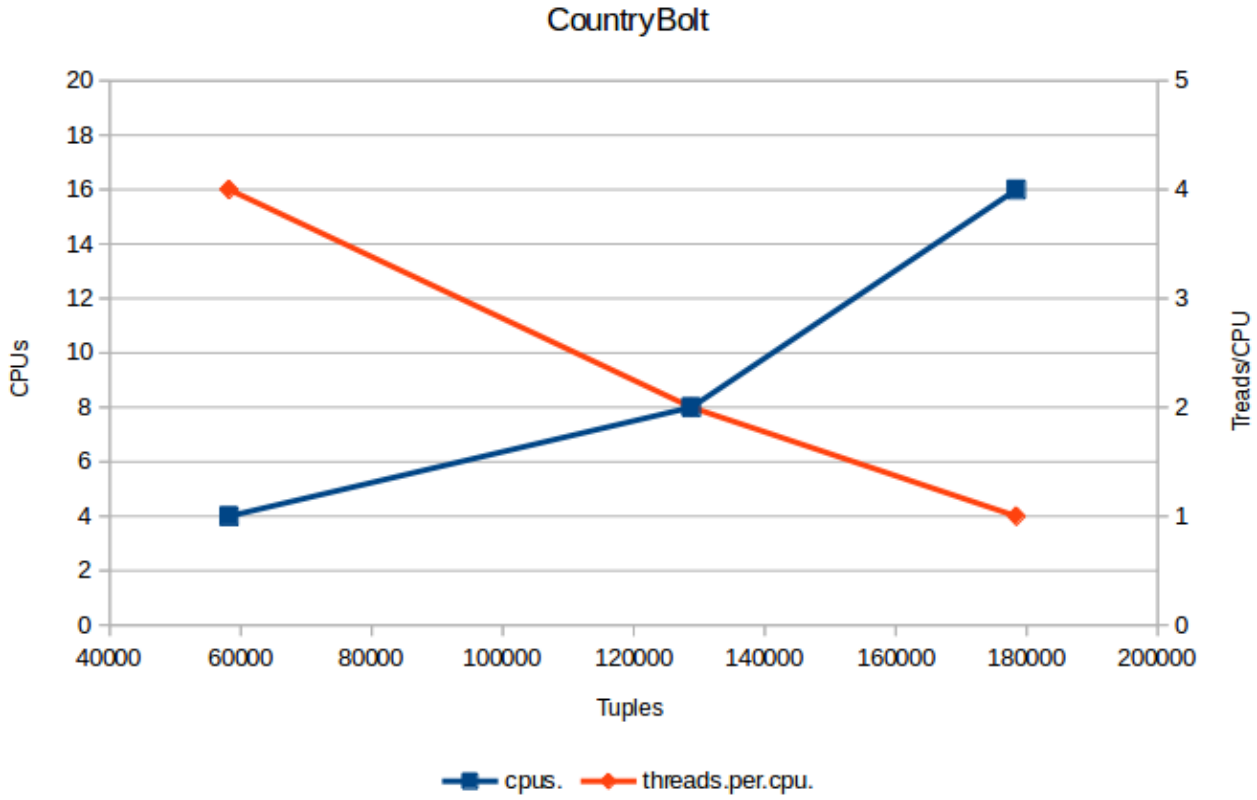
Πίνακας 5.4: Επιδόσεις Τοπολογιών



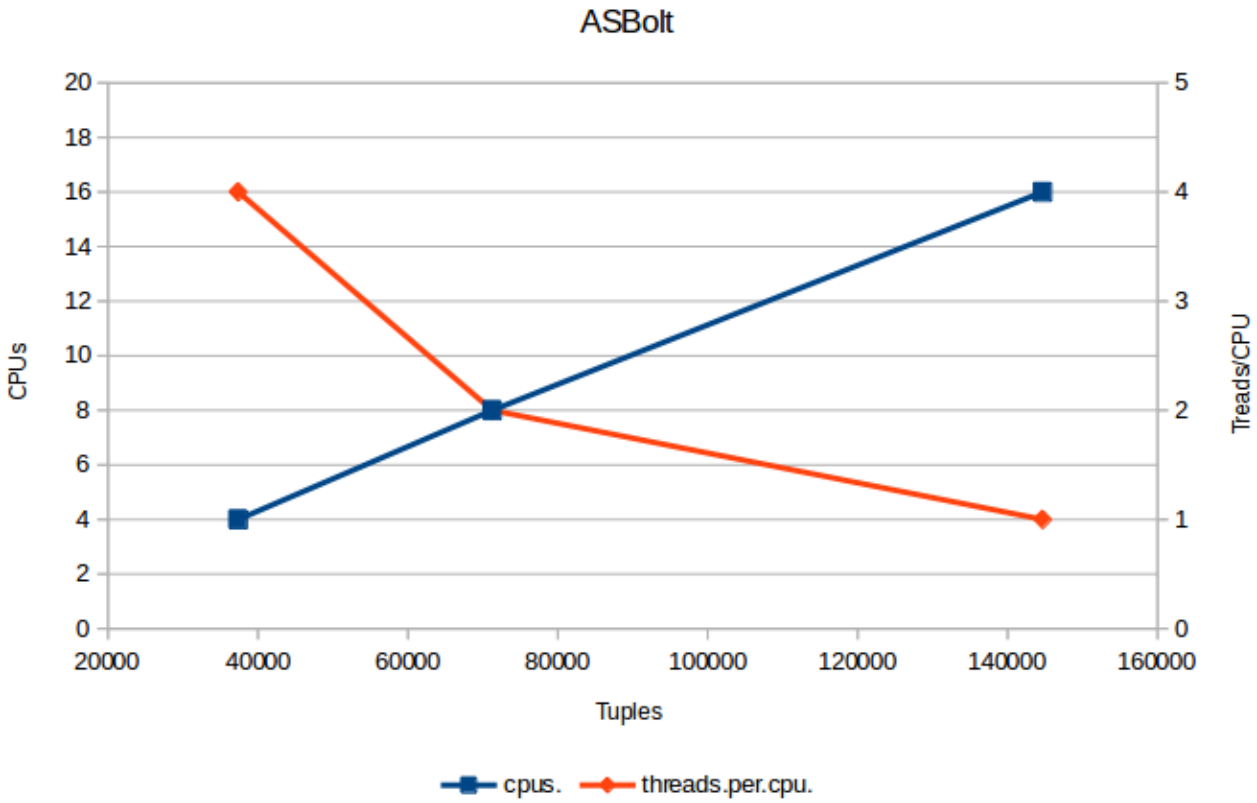
Σχήμα 5.3: KafkaSpout



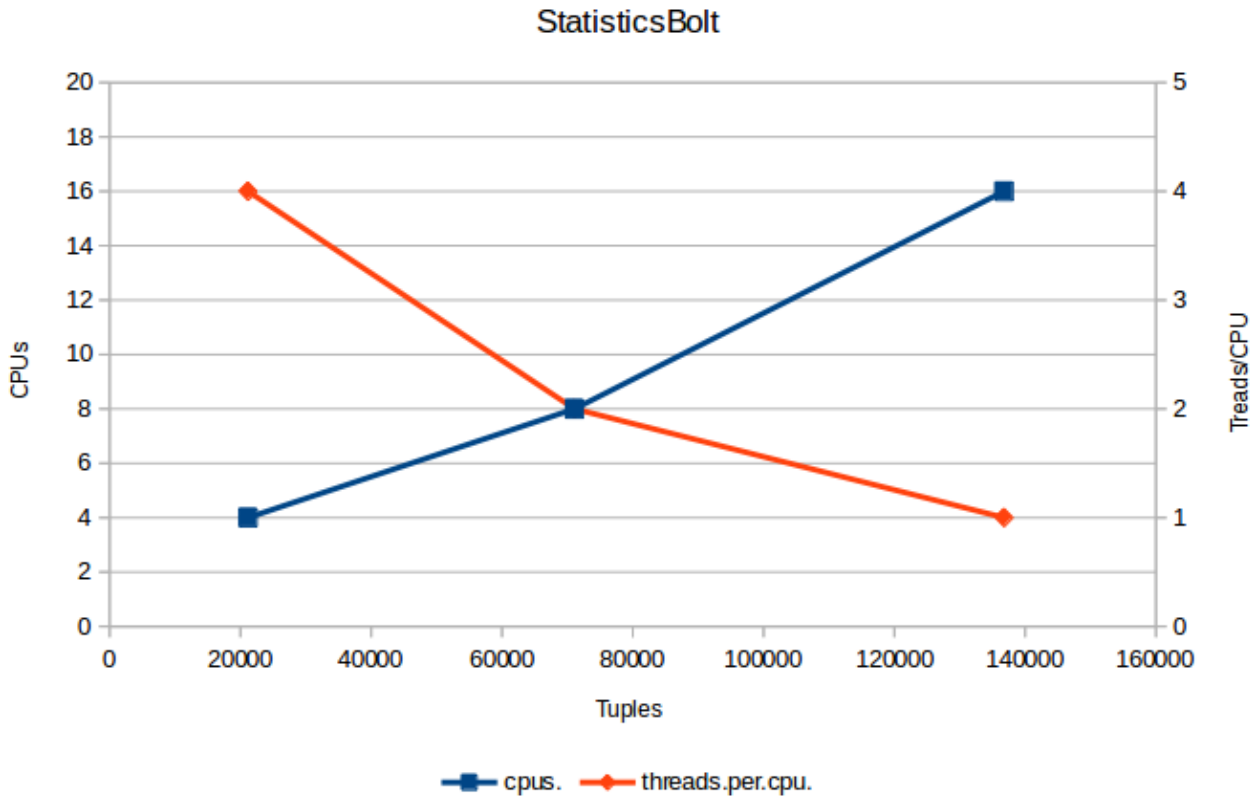
Σχήμα 5.4: IPtoIntBolt



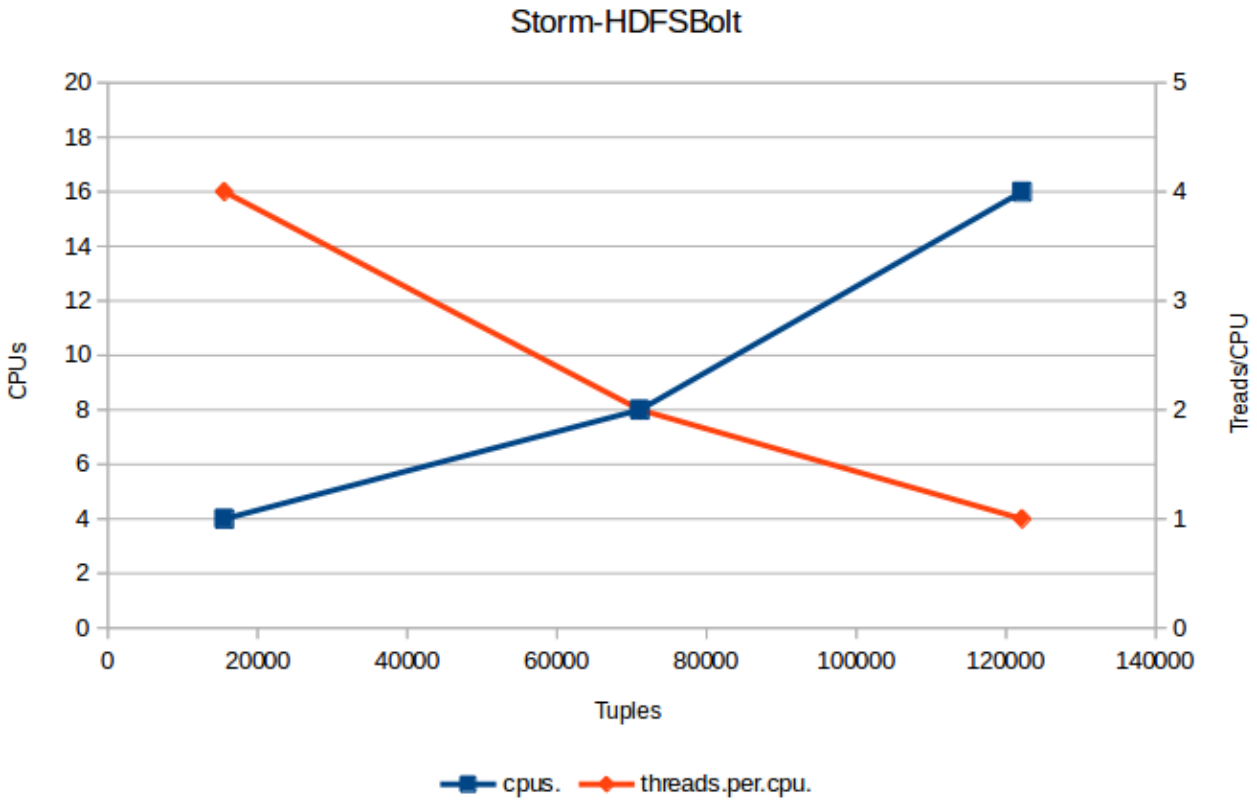
Σχήμα 5.5: CountryBolt



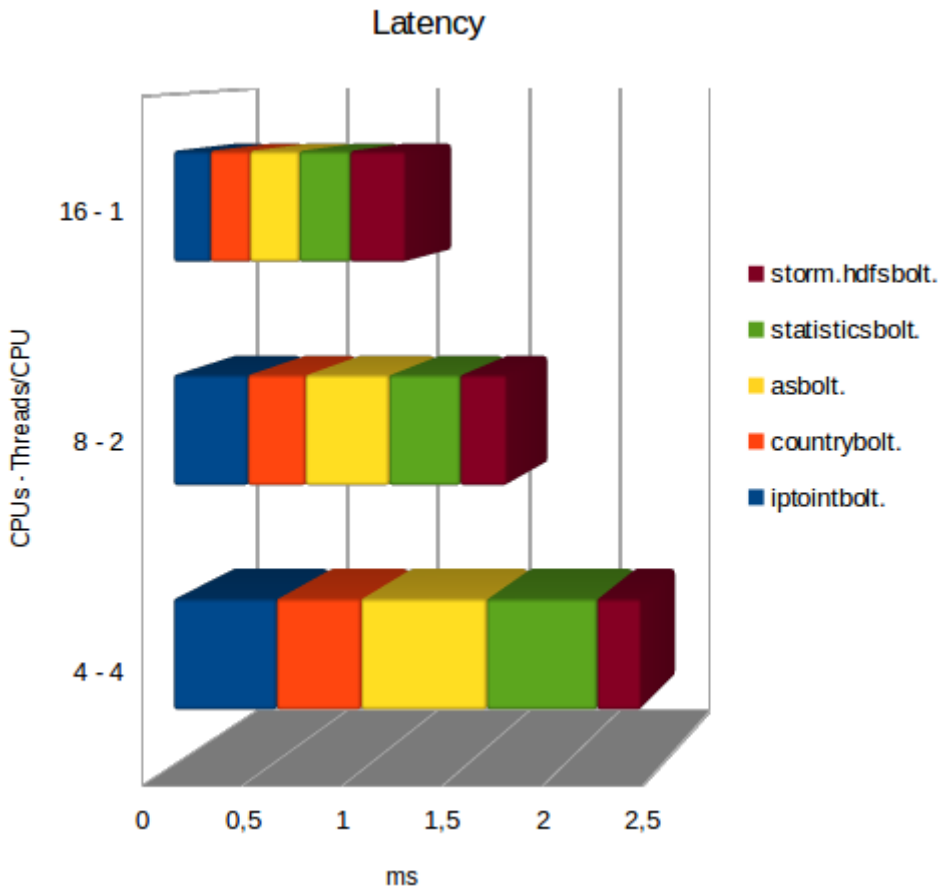
Σχήμα 5.6: ASBolt



Σχήμα 5.7: StatisticsBolt



Σχήμα 5.8: Storm-HDFSBolt



Σχήμα 5.9: Καθυστέρηση Κατανεμημένου Συστήματος

Αυτό που παρατηρούμε από τον πίνακα 5.4 και από τα σχήματα 5.3 έως 5.8 είναι ότι το σύστημα μπορεί να κλιμακώσει. Όσο αυξάνουμε τον αριθμό των CPUs αυξάνεται το throughput του συστήματος συνολικά αλλά και στις επιμέρους βαθμίδες, οδηγώντας μας σε μεγαλύτερες ταχύτητες. Αντίστροφα, επιτυγχάνουμε αύξηση της ταχύτητας όταν σε κάθε μία CPU τρέχουν λιγότερα threads, με ιδανικότερη περίπτωση εκείνη του ενός thread ανά CPU. Όλα μας τα thread τρέχουν ένα task (βάσει της προεπιλογής του Storm) γιατί περισσότερα του ενός task ανά νήμα, δεν θα αντιστοιχούσε σε αύξηση του παραλληλισμού της τοπολογίας. Βλέπουμε επίσης από τον πίνακα 5.4 και το σχήμα 5.9, αύξηση του αριθμού των CPUs να οδηγεί σε μείωση της μέσης καθυστέρησης συνολικά, αλλά και στις επιμέρους βαθμίδες. Τέλος βλέπουμε, επίσης από τα στατιστικά της μέσης καθυστέρησης, και κυρίως από αυτά της Τοπολογίας 3, ότι επιλογή μας ως προς την κατανομή των threads μεταξύ των διάφορων βαθμίδων ήταν σωστή. Η καθυστέρηση που προσθέτει στο σύστημα η κάθε βαθμίδα είναι ελαφρά μεγαλύτερη από εκείνη της προηγούμενης βαθμίδας, μην επιτρέποντας στο σύστημα μας να δημιουργεί μεγάλες ουρές συμφόρησης σε κάποια βαθμίδα ενώ οι επόμενες από αυτήν ουρές θα είναι άδειες.

Κεφάλαιο 6
Data Mining & Πρόβλεψη

6.1 Εισαγωγή

Από τη δεκαετία του 1960 η τεχνολογία των πληροφοριών και των βάσεων δεδομένων εξελίσσεται συστηματικά από τα πρωτόγονα συστήματα επεξεργασίας αρχείων σε περίπλοκα και ισχυρά συστήματα βάσεων δεδομένων. Η έρευνα και η ανάπτυξη στα συστήματα βάσεων δεδομένων από τη δεκαετία του 1970 έχει προοδεύσει από τα πρώτα ιεραρχικά και δικτυακά συστήματα βάσεων δεδομένων στην ανάπτυξη συστημάτων σχεσιακών βάσεων δεδομένων (όπου τα στοιχεία αποθηκεύονται σε δομές σχεσιακών πινάκων), εργαλείων μοντελοποίησης και μεθόδων ευρετηριοποίησης. Επίσης, οι χρήστες πέτυχαν εύκολη και ευέλικτη πρόσβαση στα δεδομένα μέσω της δομημένης γλώσσας ερωτημάτων (SQL), των διεπαφών του χρήστη, της βελτιστοποιημένης επεξεργασίας των ερωτημάτων και της διαχείρισης των συναλλαγών. Αποδοτικές μέθοδοι για την online επεξεργασία των συναλλαγών OLTP έχουν συμβάλει ουσιαστικά στην εξέλιξη και την ευρεία αποδοχή της σχεσιακής τεχνολογίας ως σημαντικό εργαλείο για αποδοτική αποθήκευση, ανάκτηση και διαχείριση μεγάλης ποσότητας δεδομένων.

Η τεχνολογία των βάσεων δεδομένων από τα μέσα της δεκαετίας του 1980 έχει χαρακτηριστεί από εργασίες έρευνας και ανάπτυξης σε νέα και ισχυρά συστήματα βάσεων δεδομένων. Έχουν αναπτυχθεί συστήματα βάσεων δεδομένων πολυμέσων, χωρικών/χρονικών βάσεων, συστήματα βασισμένα στο διαδίκτυο κτλ.

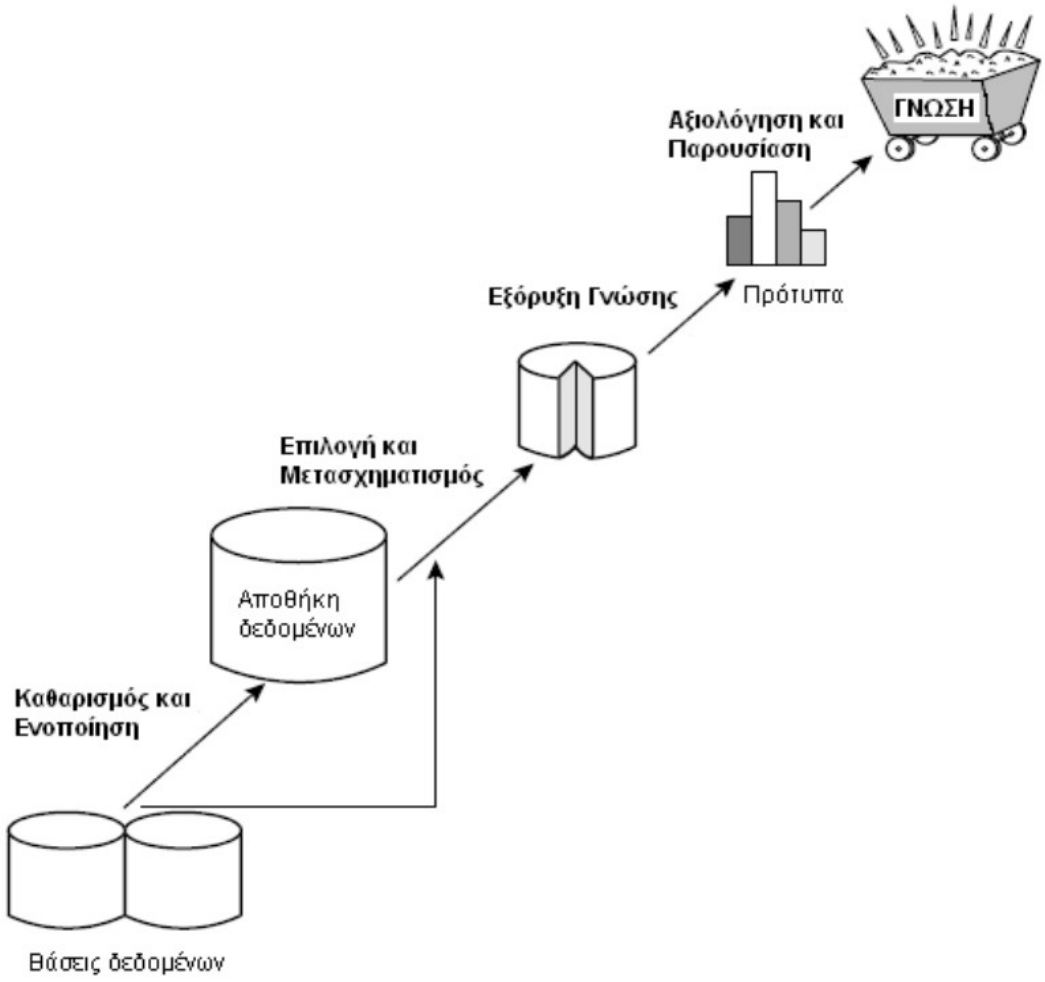
Η πρόοδος της τεχνολογίας υλικού (hardware) τις προηγούμενες τρεις δεκαετίες έχει οδηγήσει στην κατασκευή ισχυρών υπολογιστών, εξοπλισμού συλλογής δεδομένων και μέσων αποθήκευσης. Αυτή η τεχνολογία παρέχει μεγάλη ώθηση στη βιομηχανία των βάσεων δεδομένων και της πληροφορίας και κάνει διαθέσιμο έναν τεράστιο αριθμό από βάσεις δεδομένων και αποθήκες πληροφοριών για διαχείριση συναλλαγών, ανάκτηση πληροφοριών και ανάλυση δεδομένων.

Σε πολλούς κλάδους τα δεδομένα συλλέγονται και συσσωρεύονται με καλπάζοντα ρυθμό. Δημιουργείται έτσι η ανάγκη για εξαγωγή χρήσιμης πληροφορίας (γνώσης) από το ραγδαία αυξανόμενο όγκο δεδομένων. Η παραδοσιακή μέθοδος μετατροπής των

δεδομένων σε γνώση βασίζεται σε χειρωνακτική ανάλυση από ειδικούς. Όμως αυτή η διαδικασία είναι υποκειμενική, χρονοβόρα και δαπανηρή. Τα εργαλεία εξόρυξης γνώσης από δεδομένα πραγματοποιούν ανάλυση δεδομένων και αποκαλύπτουν σημαντικά πρότυπα, συμβάλλοντας σε επιχειρηματικές στρατηγικές και επιστημονική έρευνα.

Ο όρος ανακάλυψη γνώσης σε βάσεις δεδομένων (Knowledge Discovery in Databases - KDD) χρησιμοποιείται για να εκφράσει μια διαδικασία που αποτελείται από πολλά βήματα, ένα από τα οποία είναι η εξόρυξη γνώσης από δεδομένα (Data Mining). Ειδικότερα, η ανακάλυψη γνώσης σε βάσεις δεδομένων ορίζεται ως η διαδικασία εύρεσης έγκυρων, νέων, χρήσιμων και πλήρως κατανοητών προτύπων στα δεδομένα. Η εξόρυξη γνώσης από τα δεδομένα είναι ένα βήμα στη διαδικασία KDD, που συνιστάται στην ανάλυση δεδομένων και χρήση αλγορίθμων που παράγουν πρότυπα στα δεδομένα.

Η ανακάλυψη γνώσης ως διαδικασία αποτελείται από μία επαναλαμβανόμενη αλληλουχία των ακόλουθων βημάτων.



Σχήμα 6.1: Διαδικασία Ανακάλυψης Γνώσης

6.2 Κατηγορίες Μηχανικής Μάθησης

Επιβλεπόμενη Μηχανική Μάθηση (Supervised Machine Learning) ή Μάθηση με παραδείγματα (Learning from examples):
Σ' αυτή την περίπτωση το σύστημα εκπαιδεύεται με τη βοήθεια δεδομένων τα οποία αποτελούνται από εισόδους και τις αντίστοιχες εξόδους. Μέσω της αξιοποίησης αυτών, με τη βοήθεια κάποιου αλγορίθμου, το σύστημα επαγωγικά δημιουργεί γενικούς κανόνες για την παραγωγή εξόδου από οποιαδήποτε είσοδο δεδομένων αντίστοιχου τύπου με αυτόν των δεδομένων εκπαίδευσης.

Μη επιβλεπόμενη Μηχανική Μάθηση (Unsupervised learning) ή Μάθηση από παρατήρηση (Learning from observation):
Στην περίπτωση της Μη Επιβλεπόμενης Μάθησης, το σύστημα τροφοδοτείται μόνο με εισόδους και καλείται να ανακαλύψει πιθανές κρυμμένες δομές ανάμεσα τους, ώστε να τις ταξινομήσει σε ομάδες δεδομένων που παρουσιάζουν κάποια ομοιότητα. Η ύπαρξη, ο αριθμός και οι ιδιότητες των ομάδων είναι άγνωστα στο σύστημα αρχικά.

Μηχανική Μάθηση με Μερική Επίβλεψη (Semi-Supervised Machine Learning):
Η Μηχανική Μάθηση με Μερική Επίβλεψη αποτελεί συνδυασμό των δύο προηγούμενων κατηγοριών μάθησης, καθώς σ' αυτή το σύστημα τροφοδοτείται με λίγα ζεύγη εισόδου-εξόδου και αρκετές εισόδους χωρίς τις αντίστοιχες εξόδους.

Ενισχυτική Μηχανική Μάθηση (Reinforcement Learning):
Στην περίπτωση Ενισχυτικής Μηχανικής Μάθησης το σύστημα βελτιώνεται διαρκώς μέσω αλληλεπίδρασης με το περιβάλλον και παρατήρησης των αποτελεσμάτων αυτής της αλληλεπίδρασης.

6.3 Βασικές Κατηγορίες Εξόρυξης Γνώσης από Δεδομένα

Το μοντέλο που δημιουργείται από τη διαδικασία εξόρυξης μπορεί να είναι είτε προβλεπτικό είτε περιγραφικό. Το προβλεπτικό μοντέλο κάνει μία πρόβλεψη για τις τιμές των δεδομένων, χρησιμοποιώντας γνωστά αποτελέσματα που έχει βρει από άλλα δεδομένα. Παραδείγματα προβλεπτικών μοντέλων είναι η κατηγοριοποίηση, η παλινδρόμηση, η ανάλυση χρονοσειρών, η πρόβλεψη. Το περιγραφικό μοντέλο αναγνωρίζει πρότυπα ή συσχετίσεις στα δεδομένα, ενώ επίσης λειτουργεί σαν ένα μέσο που διερευνά τις ιδιότητες των δεδομένων τα οποία εξετάζονται, χωρίς να προβλέπει νέες ιδιότητες. Παραδείγματα περιγραφικών μοντέλων είναι η συσταδοποίηση, η παρουσίαση συνόψεων, οι κανόνες συσχετίσεων και η ανακάλυψη ακολουθιών.

6.3.1 Κατηγοριοποίηση (Classification)

Η κατηγοριοποίηση απεικονίζει τα δεδομένα σε προκαθορισμένες ομάδες ή κατηγορίες-κλάσεις (classes). Αναφέρεται συχνά σαν εποπτευόμενη μάθηση, επειδή οι κατηγορίες-κλάσεις καθορίζονται πριν ακόμη εξεταστούν τα δεδομένα. Οι αλγόριθμοι κατηγοριοποίησης απαιτούν οι κατηγορίες να ορίζονται με βάση τις τιμές των γνωρισμάτων των δεδομένων. Συχνά περιγράφουν αυτές τις κατηγορίες κοιτάζοντας τα χαρακτηριστικά δεδομένων που είναι ήδη γνωστό ότι ανήκουν στις κατηγορίες. Η αναγνώριση προτύπου (pattern recognition) αποτελεί ένα είδος κατηγοριοποίησης, όπου ένα πρότυπο εισόδου κατηγοριοποιείται σε μία από τις διάφορες κλάσεις, με βάση την εγγύτητά του ως προς αυτές τις προκαθορισμένες κλάσεις. Η συγκεκριμένη κατηγορία εξόρυξης γνώσης θα μας απασχολήσει ιδιαίτερα παρακάτω.

6.3.2 Παλινδρόμηση (Regression)

Η παλινδρόμηση χρησιμοποιείται για να απεικονιστεί ένα στοιχειώδες δεδομένο σε μια πραγματική μεταβλητή πρόβλεψης. Στην πραγματικότητα, η παλινδρόμηση περιλαμβάνει την εκμάθηση της συνάρτησης που κάνει αυτή την απεικόνιση. Η παλινδρόμηση προϋποθέτει ότι τα σχετικά δεδομένα ταιριάζουν με μερικά γνωστά είδη συνάρτησης (π.χ. γραμμική, λογαριθμική) και μετά καθορίζει την καλύτερη συνάρτηση αυτού του είδους που μοντελοποιεί τα δεδομένα που έχουν δοθεί. Ένα είδος ανάλυσης σφάλματος χρησιμοποιείται για να καθορίσει ποια συνάρτηση είναι η καλύτερη.

6.3.3 Ανάλυση Χρονοσειρών (Time Series Analysis)

Με την ανάλυση χρονολογικών σειρών ή χρονοσειρών, μελετάται η τιμή ενός γνωρίσματος καθώς μεταβάλλεται στο χρόνο. Οι τιμές συνήθως λαμβάνονται σε ίσα χρονικά διαστήματα (π.χ ημερήσια). Για να παρασταθούν οπτικά οι χρονοσειρές, χρησιμοποιείται ένα διάγραμμα χρονοσειρών. Υπάρχουν τρεις βασικές λειτουργίες που πραγματοποιούνται στην ανάλυση χρονοσειρών. Στη μία περίπτωση, χρησιμοποιούνται μονάδες μέτρησης απόστασης για να καθορίσουν την ομοιότητα ανάμεσα σε διαφορετικές χρονοσειρές. Στη δεύτερη περίπτωση εξετάζεται η δομή της χρονοσειράς για να καθορίσει (και ίσως να κατηγοριοποιήσει) τη συμπεριφορά της. Μια τρίτη εφαρμογή θα μπορούσε να είναι η χρήση διαγραμμάτων χρονοσειρών για την πρόβλεψη μελλοντικών τιμών.

6.3.4 Πρόβλεψη (Prediction)

Πολλές από τις πρακτικές εφαρμογές εξόρυξης γνώσης μπορούν να θεωρηθούν ως πρόβλεψη μελλοντικών καταστάσεων με γνώση των προηγούμενων και των σημερινών δεδομένων. Η πρόβλεψη μπορεί να θεωρηθεί είδος κατηγοριοποίησης. Η διαφορά είναι ότι ως πρόβλεψη θεωρείται περισσότερο το να δίνεται τιμή σε μία μελλοντική κατάσταση παρά σε μία τρέχουσα.

6.3.5 Συσταδοποίηση (Clustering)

Η συσταδοποίηση είναι παρόμοια με την κατηγοριοποίηση, εκτός από το ότι οι συστάδες-ομάδες δεδομένων δεν είναι προκαθορισμένες αλλά ορίζονται κυρίως από τα ίδια τα δεδομένα. Η συσταδοποίηση αναφέρεται εναλλακτικά και ως μη εποπτευόμενη μάθηση ή τμηματοποίηση. Μπορεί να θεωρηθεί σαν μια διαμέριση ή τμηματοποίηση των δεδομένων σε ομάδες που μπορεί να είναι ή να μην είναι διακριτές μεταξύ τους. Η συσταδοποίηση συνήθως επιτυγχάνεται με τον καθορισμό της ομοιότητας, ως προς προκαθορισμένα γνωρίσματα, ανάμεσα στα δεδομένα. Τα πιο σχετικά δεδομένα ομαδοποιούνται στις ίδιες συστάδες.

6.3.6 Παρουσίαση Συνόψεων (Summarization)

Η παρουσίαση συνόψεων απεικονίζει τα δεδομένα σε υποσύνολά τους με συνοδευτικές απλές περιγραφές. Η σύνοψη των δεδομένων ονομάζεται επίσης και χαρακτηρισμός (characterization) ή γενίκευση (generalization). Εξάγει ή παράγει αντιπροσωπευτικές πληροφορίες σχετικά με τις βάσεις δεδομένων. Αυτό γίνεται ανακτώντας στην πραγματικότητα τμήματα από τα δεδομένα. Εναλλακτικά, μπορούν να εξαχθούν από τα δεδομένα συνοπτικές πληροφορίες (όπως ο μέσος όρος κάποιου αριθμητικού γνωρίσματος). Εν ολίγοις, η παρουσίαση συνόψεων χαρακτηρίζει τα περιεχόμενα της βάσης δεδομένων.

6.3.7 Κανόνες Συσχέτισης (Association Rules)

Η ανάλυση συνδέσμων (link analysis), που εναλλακτικά αναφέρεται και ως ανάλυση συγγένειας (affinity analysis) ή συσχέτισης (association) αναφέρεται στη διαδικασία εκείνη της εξόρυξης γνώσης που αποκαλύπτει συσχετίσεις μεταξύ των δεδομένων. Το καλύτερο παράδειγμα αυτού του είδους της εφαρμογής είναι ο προσδιορισμός κανόνων συσχετίσεων. Ένας κανόνας συσχέτισης είναι ένα μοντέλο που αναγνωρίζει ειδικούς τύπους συσχέτισης μεταξύ δεδομένων. Ένα παράδειγμα κανόνα είναι: $X \rightarrow Y$, δηλαδή αν X τότε Y. Υπάρχουν δύο μεγέθη που χρησιμοποιούνται στους κανόνες συσχέτισης: η υποστήριξη (support), η οποία δείχνει το ποσοστό των εγγραφών στη βάση δεδομένων όπου ισχύει ο συνδυασμός X και Y και η εμπιστοσύνη (confidence) που δείχνει το ποσοστό των εγγραφών που όταν ισχύει το X ισχύει και το Y (δεσμευμένη πιθανότητα).

6.3.8 Ανακάλυψη Ακολουθιών (Sequential Discovery)

Η ακολουθιακή ανάλυση (Sequential Discovery) ή αλλιώς ανακάλυψη ακολουθιών (Sequential Discovery) χρησιμοποιείται για να καθοριστούν σειριακά πρότυπα στα δεδομένα. Αυτά τα πρότυπα βασίζονται σε μια χρονική ακολουθία ενεργειών και είναι παρόμοια με τις συσχετίσεις στο ότι συσχετίζονται τα δεδομένα (ή τα γεγονότα) που εξάγονται, με τη διαφορά ότι η συσχέτιση τους αυτή βασίζεται στο χρόνο. Για παράδειγμα, αντίθετα με την ανάλυση καλαθιού αγορών (market basket analysis) που προϋποθέτει να γνωρίζουμε ποια προϊόντα αγοράστηκαν ταυτόχρονα, στην ανακάλυψη ακολουθιών τα προϊόντα αγοράζονται με κάποια σειρά κατά τη διάρκεια μιας περιόδου.

6.4 Βασικοί Αλγόριθμοι Κατηγοριοποίησης

Η κατηγοριοποίηση (classification) αποτελεί μια από τις βασικότερες τεχνικές εξόρυξης γνώσης. Εξετάζοντας τα χαρακτηριστικά ενός αντικειμένου το αντιστοιχεί σε ένα προκαθορισμένο σύνολο κλάσεων χρησιμοποιώντας μεθόδους μάθησης με επίβλεψη. Η εργασία της κατηγοριοποίησης χαρακτηρίζεται από ένα καθορισμένο σύνολο κατηγοριών και ένα σύνολο προκαθορισμένων παραδειγμάτων που χρησιμοποιείται για την εκπαίδευση του μοντέλου. Βασικός στόχος είναι η κατασκευή μοντέλου που θα μπορεί να χρησιμοποιηθεί για την ταξινόμηση μη κατηγοριοποιημένων μελλοντικών δεδομένων. Πιο απλά, έχουμε ένα σύνολο εκπαίδευσης (training set) από το οποίο «μαθαίνει» ο αλγόριθμος ταξινόμησης. Δοθέντος ενός συνόλου δοκιμαστικών δεδομένων (test set) ο αλγόριθμος θα ταξινομήσει τα δεδομένα ανάλογα με την γνώση που απέκτησε. Άρα, μπορούμε να πούμε ότι η κατηγοριοποίηση μαθαίνει σε μία λειτουργία να χαρτογραφεί (ταξινομεί) ένα στοιχείο δεδομένων σε μία από τις διάφορες προκαθορισμένες κατηγορίες.

Όταν θέλουμε να αξιολογήσουμε αλγορίθμους της ταξινόμησης αυτό που μας ενδιαφέρει είναι η ακρίβεια του μοντέλου. Η δυνατότητα, δηλαδή, πρόβλεψης ταξινόμησης των νέων μη κατηγοριοποιημένων δεδομένων. Σε αυτή την μέθοδο έχουμε το σύνολο εκπαίδευσης και το σύνολο δοκιμής. Ένας τρόπος λοιπόν είναι να εφαρμόσουμε ένα αλγόριθμο στα δεδομένα εκπαίδευσης και μετά μέσω του συνόλου δοκιμής να βρούμε την ακρίβεια. Όμως η χρήση ενός τυχαίου δείγματος ως σύνολο δοκιμής δεν κάνει το αποτέλεσμα αξιόπιστο. Για το λόγο αυτό χρησιμοποιούμε την τεχνική k-fold cross-validation (διασταυρωμένη επικύρωση). Χωρίζουμε το σύνολο των δεδομένων σε k υποσύνολα με το καθένα να περιέχει περίπου ίσο αριθμό στοιχείων. Κάθε φορά κρατάμε ένα σύνολο για δοκιμή και τα υπόλοιπα τα εκπαιδεύουμε. Έτσι αφού επαναλάβουμε την διαδικασία k φορές θα έχουμε k ακρίβειες. Η τελική ακρίβεια είναι το άθροισμα των ακριβειών διαιρούμενο με το k.

6.4.1 Δέντρα Αποφάσεων

Τα δέντρα αποφάσεων (decision trees) αποτελούν μια από τις πιο χρησιμοποιημένες τεχνικές στην ταξινόμηση. Είναι ένα δυναμικό και δημοφιλές εργαλείο για ταξινόμηση δεδομένων. Ο λόγος είναι η απλή προσέγγισή τους στην διαδικασία εξόρυξης γνώσης αφού αναπαριστούν γραφικά τα στιγμιότυπα που φέρουν την ίδια τιμή σε κάποιο χαρακτηριστικό. Ένα δέντρο απόφασης ουσιαστικά είναι μια δομή στην οποία κάθε κόμβος αντιπροσωπεύει μια επιλογή μεταξύ διαφόρων εναλλακτικών λύσεων και κάθε κόμβος φύλλων αντιπροσωπεύει μια ταξινόμηση ή μια απόφαση. Στα δέντρα απόφασης, βασική προϋπόθεση αποτελεί το ότι κάθε δείγμα του συνόλου δεδομένων (dataset) πρέπει να μπορεί να εκφράζεται ως μια συλλογή από τα χαρακτηριστικά (attributes) του συνόλου δεδομένων. Ο αλγόριθμος λαμβάνει ως είσοδο κάθε δείγμα ως ένα διάνυσμα εισόδου με τις τιμές των μεταβλητών και την αντίστοιχη κλάση που ανήκει. Αυτά τα διανύσματα εισόδου αποτελούν το σύνολο εκπαίδευσης (training set) του ταξινομητή. Έπειτα, παρατηρώντας και συγκρίνοντας αν τα διανύσματα, των οποίων οι τιμές είναι κοντά η μία με την άλλη, ανήκουν ή όχι στην ίδια κλάση, κατασκευάζει ένα σύνολο από κανόνες αποφάσεων με σκοπό την ταξινόμηση μελλοντικών δειγμάτων στις γνωστές κλάσεις. Αναλυτικότερα για την ταξινόμηση, μία εγγραφή εισέρχεται στο δέντρο από τον κόμβο της κορυφής. Στη ρίζα εφαρμόζεται έλεγχος για να καθοριστεί ποιο κόμβο θα ακολουθήσει στη συνέχεια η εγγραφή. Η επεξεργασία αυτή επαναλαμβάνεται μέχρι η εγγραφή να φτάσει στον κόμβο φύλλο. Όλες οι εγγραφές, οι οποίες καταλήγουν σε ένα συγκεκριμένο φύλλο ταξινομούνται με τον ίδιο τρόπο. Κατά την εκπαίδευση, το δέντρο χτίζεται με την επαναλαμβανόμενη διάσπαση του δοσμένου συνόλου δεδομένων σύμφωνα με τις διάφορες ανεξάρτητες μεταβλητές. Η σειρά με την οποία χρησιμοποιούνται οι ανεξάρτητες μεταβλητές στη δόμηση του δέντρου εξαρτάται από τη δυνατότητα ταξινόμησης της κάθε ανεξάρτητης μεταβλητής. Υπάρχουν διάφοροι αλγόριθμοι για την επιλογή της σειράς, αλλά ο στόχος είναι να επιλέξουμε την μεταβλητή εκείνη που διαχωρίζει καλύτερα τις τελικές κλάσεις. Άρα, στη ρίζα του δέντρου θα βρίσκεται το χαρακτηριστικό εκείνο το οποίο διαχωρίζει καλύτερα τα δεδομένα εκπαίδευσης. Ο αλγόριθμος σταματάει όταν φτάσει στον κόμβο από τον οποίο δεν είναι δυνατό να ξεκινήσει μία νέα διάσπαση.

6.4.2 Αλγόριθμος ID3

Ο ID3 είναι ένας αλγόριθμος για να κατασκευάζουμε δέντρα απόφασης. Η βασική ιδέα αυτού του αλγορίθμου είναι να κατασκευάσει το δέντρο εφαρμόζοντας μια top-down μέθοδο και βρίσκει το ποιο χρήσιμο χαρακτηριστικό, αυτό δηλαδή που μας διαχωρίζει καλύτερα τα δεδομένα, για την κατηγοριοποίηση εφαρμόζοντας την στατιστική ιδιότητα του κέρδους πληροφορίας (information gain), ενώ συγχρόνως προσπαθεί να ελαχιστοποιήσει τον αριθμό συγκρίσεων. Ο στόχος της κατηγοριοποίησης ενός δέντρου απόφασης είναι να διαχωρίσει επαναληπτικά το υπό εξέταση σύνολο δεδομένων σε υποσύνολα που όλα τα στοιχεία σε κάθε τελικό υποσύνολο να ανήκουν στην ίδια κατηγορία. Η έννοια που χρησιμοποιείται για να μετρηθεί η ποσότητα της ομογένειας ή αλλιώς αβεβαιότητας του δείγματος λέγεται εντροπία (entropy). Όταν όλα τα δεδομένα ενός συνόλου ανήκουν σε μία κατηγορία, δεν υπάρχει καθόλου αβεβαιότητα. Σε αυτή την περίπτωση η εντροπία είναι μηδέν και έχουμε ένα πλήρως ομογενές υποσύνολο. Στην περίπτωση που η εντροπία είναι ένα, τότε έχουμε ένα υποσύνολο το οποίο έχει δημιουργηθεί τυχαία και χρειάζεται καλύτερη ταξινόμηση. Η εντροπία χρησιμοποιείται για να καθορίσει ποίος θα είναι ο επόμενος κόμβος που θα διαχωρίσει ο αλγόριθμος ID3. Η βασική στρατηγική που εκτελείται από τον ID3 είναι η επιλογή του χαρακτηριστικού με το υψηλότερο κέρδος πληροφορίας πρώτα. Το κέρδος ορίζεται ως η διαφορά μεταξύ της πληροφορίας που είναι απαραίτητη για να κάνουμε μια σωστή κατηγοριοποίηση πριν από τη διάσπαση και της πληροφορίας που χρειαζόμαστε μετά.

6.4.3 Αλγόριθμος C4.5 (J48)

Ο αλγόριθμος C4.5 αποτελεί μια βελτίωση του αλγορίθμου ID3 και χρησιμοποιεί το λόγο του κέρδους πληροφορίας. Έχει πολύ καλό συνδυασμό ακρίβειας και εκπαίδευσης. Ο αλγόριθμος C4.5 βελτιώνει τον ID3 με τους ακόλουθους τρόπους:

- Ελλιπή δεδομένα: Όταν το δέντρο απόφασης χτίζεται, τα ελλιπή δεδομένα αγνοούνται. Για την κατηγοριοποιήσουμε μία εγγραφή με ελλιπή τιμή για ένα γνώρισμα, η τιμή για αυτό το στοιχείο μπορεί να προβλεφθεί με βάση γνωστών τιμών από άλλες εγγραφές.

- Συνεχή δεδομένα: Η βασική ιδέα είναι να χωρίσουμε τα δεδομένα σε διαστήματα, με βάση τις τιμές των γνωρισμάτων για εκείνα τα στοιχεία τα οποία ανήκουν στο δείγμα της εκπαίδευσης.

- Κανόνες: Ο C4.5 επιτρέπει την κατηγοριοποίηση είτε μέσω δέντρων απόφασης είτε μέσω κανόνων οι οποίοι δημιουργούνται από αυτά. Επιπλέον προτείνονται μερικές τεχνικές για την απλούστευση των πολύπλοκων κανόνων.

Ο αλγόριθμος J48 είναι η ανοιχτού κώδικα υλοποίηση του αλγορίθμου C4.5, γραμμένη σε γλώσσα Java.

6.4.4 Αλγόριθμος Naive Bayes

Ο αλγόριθμος αυτός είναι ένας απλός πιθανοτικός αλγόριθμος που βασίζεται στο θεώρημα Bayes σε συνδυασμό με την υπόθεση ότι οι πιθανότητες των στοιχείων που εμπλέκονται είναι μεταξύ τους ανεξάρτητες,στην οποία υπόθεση οφείλεται και ο χαρακτηρισμός του ως naive (αφελής). Το θεώρημα Bayes έχει στόχο την εύρεση της πιο πιθανής υπόθεσης από ένα σύνολο υποθέσεων ή δεδομένου ενός συνόλου εκπαίδευσης D, αλλά και της γνώσης που πιθανών εκ των προτέρων διαθέτουμε για τις πιθανότητες των διαφόρων υποθέσεων $h \in H$. Το θεώρημα του Bayes είναι το εξής:

$$P(h|D) = \frac{P(h) \cdot P(D|h)}{P(D)}$$

- P(h|D): Η ζητούμενη εκ των υστέρων πιθανότητα που εκφράζει την ισχύ της υπόθεσης h, δεδομένου του συνόλου εκπαίδευσης D.

- P(h): Η εκ των προτέρων γνωστή πιθανότητα που εκφράζει την ισχύ της υπόθεσης h, χωρίς να έχει προηγηθεί παρατήρηση των δεδομένων του συνόλου εκπαίδευσης D.

- P(D|h): Η δεσμευμένη πιθανότητα η οποία εκφράζει το ενδεχόμενο παρατήρησης των δεδομένων εκπαίδευσης D, αποδεχόμενης της ισχύος της υπόθεσης h.

- P(D): Η εκ των προτέρων γνωστή πιθανότητα παρατήρησης των δεδομένων εκπαίδευσης D.

6.4.5 Αλγόριθμος Ibk

Ο αλγόριθμος Ibk είναι υλοποίηση του αλγορίθμου εύρεσης κοντινότερων γειτόνων (k-nearest neighbors) ανήκει στην κατηγορία μάθησης βασισμένης σε στιγμιότυπα. Οι μέθοδοι αυτής της κατηγορίας δεν κατασκευάζουν ένα γενικό μοντέλο απλώς αποθηκεύουν τα δεδομένα και με την εμφάνιση ενός προς ταξινόμηση στοιχείου ανακαλεί ένα σύνολο στιγμιότυπων σχετιζόμενο με αυτό και το ταξινομεί. Η κεντρική ιδέα είναι πως η τιμή της συνάρτησης-στόχου για ένα νέο στιγμιότυπο βασίζεται αποκλειστικά και μόνο στις αντίστοιχες τιμές των k πιο κοντινών του στιγμιότυπων εκπαίδευσης, τα οποία αποτελούν τους γείτονές του. Με άλλα λόγια, κάθε απόφαση για ταξινόμηση ενός δείγματος παίρνεται αποκλειστικά με βάση τις ετικέτες των k πλησιέστερων γειτόνων του. Αφού προσδιορισθούν μέσω κάποιας μετρικής οι k κοντινότεροι γείτονες ενός νέου στιγμιότυπου, οι τιμές της συνάρτησης-στόχου που έχει ο καθένας από αυτούς πρέπει να συνδυαστούν για να δώσουν την εκτιμώμενη τιμή για το νέο στιγμιότυπο. Στην περίπτωση που η συνάρτηση-στόχος παίρνει διακριτές τιμές, η πιο συνηθισμένη τακτική είναι να επιλέγεται η πιο συχνή από τις τιμές των γειτόνων. Σε περίπτωση ισοβαθμιών επιλέγεται εκ των ισοβαθμούντων μια τιμή, είτε τυχαία ή η καθολικά πιο συχνή τιμή.

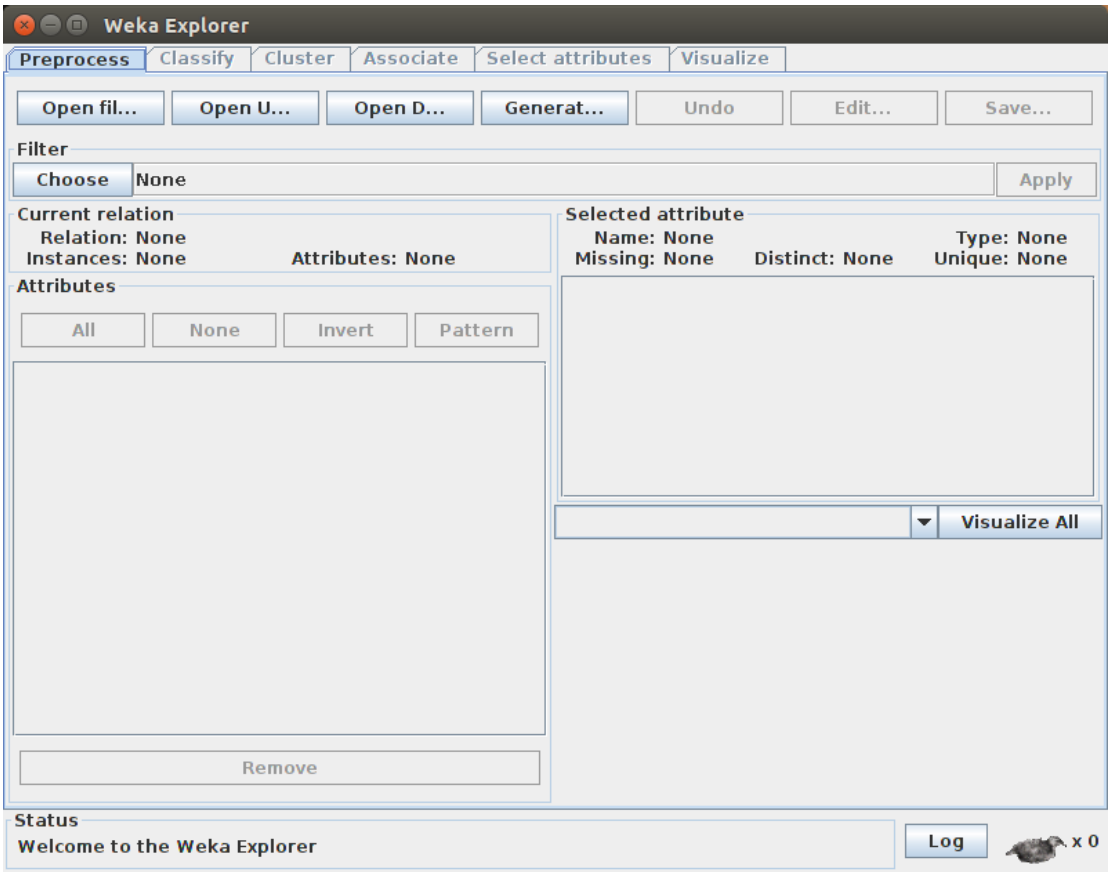
6.5 Λογισμικό Weka

Ως εργαλείο για την υλοποίηση της πρόβλεψης της ωραίας κίνησης του δικτύου GR-IX, επιλέξαμε την πλατφόρμα Weka. Το Weka είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα που περιλαμβάνει μια συλλογή αλγορίθμων Μηχανικής Μάθησης με αντικείμενο εργασίες σχετικές με την εξόρυξη δεδομένων, η οποία δημιουργήθηκε από το Πανεπιστήμιο του Waikato της Νέας Ζηλανδίας. Οι αλγόριθμοι του Weka είναι γραμμένοι σε Java και παρέχουν τις εξής δυνατότητες:

- Προεπεξεργασία δεδομένων
- Ταξινόμηση
- Συσταδοποίηση
- Εύρεση Κανόνων Συσχέτισης

Οι παραπάνω εργασίες μπορούν να γίνουν είτε μέσω του γραφικού περιβάλλοντος της εφαρμογής, είτε μέσω γραμμής εντολών. Για τις ανάγκες τις διπλωματικής έγινε χρήση του γραφικού περιβάλλοντος, και συγκεκριμένα του Explorer.

Σχήμα 6.2: Οθόνη Εκκίνησης Weka



Σχήμα 6.3: Γραφικό Περιβάλλον Explorer του Weka

Τα βασικά αρχεία τα οποία δέχεται το Weka προς επεξεργασία είναι αρχεία τύπου arff (Attribute - Relation File Format). Ένα αρχείο τύπου arff είναι ένα αρχείο κειμένου χαρακτήρων ASCII που περιλαμβάνει μία λίστα παραδειγμάτων (instances) τα οποία περιγράφονται από χαρακτηριστικά (attributes). Αποτελείται από δύο διακριτά τμήματα: το τμήμα Header και το τμήμα Data. Το τμήμα Header περιλαμβάνει το όνομα του συνόλου δεδομένων, τα ονόματα και τους τύπους των χαρακτηριστικών (attributes), ενώ το τμήμα Data περιλαμβάνει τα παραδείγματα και τις τιμές που αντιστοιχούν στο κάθε χαρακτηριστικό τους.

6.6 Πρόβλεψη Ωριαίας Κίνησης

6.6.1 Περιγραφή Dataset

Το Dataset απαρτίζεται από 4700 instances, ένα για κάθε ώρα καταγραφείσας κίνησης. Τα instances αυτά απαρτίζονται από 5 attributes:

Ημερομηνία: Το πεδίο αυτό είναι τύπου numeric. Θα μπορούσαμε να του είχαμε δώσει τύπο string, υπό την συνηθισμένη μορφή γραφής ημερομηνίας “ημέρα/μήνας/χρόνος”, αλλά αυτό θα περιόριζε το πλήθος των δυνατών αλγορίθμων που θα μπορούσαμε να χρησιμοποιήσουμε μέσω του Weka για το πείραμα μας. Εξάλλου, το μόνο που χρειαζόμασταν από αυτό το πεδίο ήταν η ομαδοποίηση των ορών της ίδιας ημερομηνίας. Ένας αύξων αριθμός για κάθε διαφορετική ημερομηνία καταγραφής ήταν αρκετός, εφόσον μας ήταν αδιάφορο αν π.χ. ο αριθμός “5” αντιστοιχεί στην ημερομηνία “10/8/2013” ή στην ημερομηνία “6/2/2014”.

Μέρα μήνα: Το πεδίο αυτό είναι μία λίστα αριθμών: 1, 2, 3 ... 31. Θα μπορούσαμε να είχαμε δώσει στο συγκεκριμένο πεδίο τύπο numeric, αλλά εφόσον οι μέρες του μήνα έχουν συγκεκριμένες τιμές, προτιμούμε την λίστα ώστε να είναι πιο διακριτή και εύκολη η κατηγοριοποίηση με βάση το συγκεκριμένο πεδίο.

Μέρα εβδομάδας: Το πεδίο αυτό είναι λίστα αριθμών: 1 (Κυριακή), 2 (Δευτέρα), 3 (Τρίτη) ... 7 (Σάββατο). Θα μπορούσαμε να είχαμε δώσει στο πεδίο αυτό τύπο string, αλλά όπως είπαμε παραπάνω, αυτό θα μείωνε το πλήθος των αλγορίθμων που θα μπορούσαμε να χρησιμοποιήσουμε.

Ώρα: Το πεδίο αυτό είναι λίστα αριθμών: 0, 1, 2 ... 23. Όπως και το πεδίο “μέρα μήνα”, έτσι και αυτό θα μπορούσε να ήταν τύπου numeric, για τους ίδιους όμως λόγους, όπως και παραπάνω, το αποφεύγουμε.

Πλήθος πακέτων: Το συγκεκριμένο πεδίο είναι ίσως το πιο σημαντικό. Είναι το πεδίο που θέλουμε να προβλέψουμε. Αυτό για το οποίο ζητάμε να εργαστεί ο classifier. Για την συλλογή των στοιχείων του συγκεκριμένου πεδίου, εργάστηκε αρχικά το κατανεμημένο μας σύστημα. Έχουμε αθροίσει όλες τις εγγραφές που υπάρχουν στα δεδομένα και έχουν κοινό timestamp (χρονοσφραγίδα). Χρονοφραγίδες οι οποίες αναλύονται στα 4 παραπάνω πεδία (attributes). Για το συγκεκριμένο πεδίο παρουσιάζουμε 3 διαφορετικές παραλλαγές: 2 με λίστα τιμών και μία με τύπο numeric.

Η παραλλαγή με τύπο numeric αποτελεί μια λίστα με τα ακριβή αθροίσματα ανά χρονοσφραγίδα. Οι δύο παραλλαγές με λίστα αποτελούν κατηγοριοποίηση των αριθμών που έχουμε από τα ακριβή αθροίσματα σε 9 και 18 κλάσεις αντίστοιχα. Οι κλάσεις αυτές φαίνονται στους πίνακες 6.1 και 6.2. Ο λόγος ύπαρξης αυτών των τριών παραλλαγών στο συγκεκριμένο πεδίο μας επιτρέπει την χρήση διαφορετικών αλγορίθμων για την πρόβλεψη. Στην περίπτωση που το πεδίο είναι τύπου numeric επιδιώκουμε πρόβλεψη συγκεκριμένου αριθμού πακέτων, ενώ στην περίπτωση που το πεδίο είναι λίστα επιδιώκουμε πρόβλεψη της αντίστοιχης στάθμης. Φυσικά, αναμένουμε η κατηγοριοποίηση μεταξύ 18 διαφορετικών σταθμών να είναι πιο δύσκολή σε σχέση με εκείνη των 9.

Αύξων αριθμός στάθμης	Πλήθος πακέτων που καταμετρήθηκαν
1	1 – 100.000
2	100.001 – 200.000
3	200.001 – 300.000
4	300.001 – 400.000
5	400.001 – 500.000
6	500.001 – 600.000
7	600.001 – 700.000
8	700.001 – 800.000
9	800.001 – 900.000

Πίνακας 6.1: Κατηγοριοποίηση 9 Σταθμών

Αύξων αριθμός στάθμης	Πλήθος πακέτων που καταμετρήθηκαν
1	1 – 50.000
2	50.001 – 100.000
3	100.001 – 150.000
4	150.001 – 200.000
5	200.001 – 250.000
6	250.001 – 300.000
7	300.001 – 350.000
8	350.001 – 400.000
9	400.001 – 450.000
10	450.001 – 500.000
11	500.001 – 550.000
12	550.001 – 600.000
13	600.001 – 650.000
14	650.001 – 700.000
15	700.001 – 750.000
16	750.001 – 800.000
17	800.001 – 850.000
18	850.001 – 900.000

Πίνακας 6.2: Κατηγοριοποίηση 18 Σταθμών

6.6.2 Αποτελέσματα Πειραμάτων

Στην ενότητα αυτή θα παρουσιάσουμε και θα σχολιάσουμε τα αποτελέσματα της προσπάθειας πρόβλεψης της ωριαίας κίνησης του δικτύου GR-IX.

	9 στάθμες	18 στάθμες
Zero R	19,2766 %	10,4894 %
Naive Bayes	63,9787 %	42,5745 %
J48	**72,2766 %**	46,4043 %
Ibk	31,4894 %	21,6596 %

Πίνακας 6.3: Επιτυχία πρόβλεψης Σταθμών

Όπως ήταν αναμενόμενο, η κατηγοριοποίηση σε 9 στάθμες μας δίνει υψηλότερα ποσοστά σε σχέση με την κατηγοριοποίηση σε 18 στάθμες για όλους τους αλγόριθμους που δοκιμάσαμε. Παρατηρούμε ότι παρόλο που οι στάθμες υπερδιπλασιάζονται, το ποσοστό επιτυχημένης πρόβλεψης αυξάνεται, αλλά δεν διπλασιάζεται. Η περίπτωση Zero R, και στις δύο στήλες, μας δείχνει το ποσοστό της επικρατούσας κλάσης. Σύμφωνα με τον αλγόριθμο Zero R, το αποτέλεσμα της πρόβλεψης είναι πάντα εκείνη η κλάση η οποία έχει τον μεγαλύτερο αριθμό instances σε σχέση με τις υπόλοιπες. Είναι ουσιαστικά ένας δείκτης που περιγράφει κατά πόσο είναι ομοιόμορφα κατανεμημένα τα instance στις διάφορες κλάσεις, και κατά συνέπεια, πόσο δύσκολο είναι για τους υπόλοιπους αλγόριθμους να κάνουν σωστή πρόβλεψη. Το μεγαλύτερο ποσοστό που σημειώνεται στον πίνακα είναι το 72,2766% με κατηγοριοποίηση σε 9 στάθμες και χρήση του αλγορίθμου J48. Το ποσοστό αυτό μεταφράζεται σε ότι: Με 72,2766% πιθανότητα επιτυχίας, μπορούμε να προβλέψουμε την συνολική ωριαίας κίνηση του δικτύου GR-IX μεταξύ των 9 σταθμών που ορίσαμε στον παραπάνω πίνακα. Σε αυτό το σημείο πρέπει να τονίσουμε ιδιαίτερα ότι τα πακέτα που έχουμε καταγράψει και αποτελούν τα δεδομένα μας, αποτελούν προϊόν δειγματοληψίας με αναλογία 1 προς 500.000. Αυτό σημαίνει ότι για να γίνει αναγωγή της πρόβλεψης, από αριθμό δειγμάτων, σε αριθμό πραγματικών πακέτων, θα πρέπει να πολλαπλασιαστεί με το 500.000.

	Linear Regression	M5P
Correlation coefficient Συντελεστής συσχέτισης	0,9242	0,9833
Mean absolute error Μέσο απόλυτο σφάλμα	47879,1336	21696,3082
Root mean squared error Ρίζα μέσου τετραγωνικού σφάλμα	69066,4860	32968,5681
Relative absolute error Σχετικό απόλυτο σφάλμα	30,9204 %	14,0115 %
Root relative squared error Ρίζα σχετικού τετραγωνικού σφάλμα	38,1798 %	18,2250 %

Πίνακας 6.4: Στατιστικά Πρόβλεψης Ακριβούς Αριθμού Πακέτων

Όπως βλέπουμε από τον πίνακα 6.4 υπάρχει ισχυρή συσχέτιση μεταξύ των μεγεθών που εξετάζουμε. Θυμίζουμε ότι όταν η απόλυτη τιμή του συντελεστή συσχέτισης βρίσκεται κοντά στην μονάδα, οι τυχαίες μεταβλητές έχουν ισχυρή συσχέτιση μεταξύ τους. Σε αντίθετη περίπτωση, όταν ο συντελεστής συσχέτισης βρίσκεται κοντά στο μηδέν, δεν υπάρχει συσχέτιση μεταξύ των μεταβλητών. Και σε αυτή την περίπτωση, όπως και στην προηγούμενη με τις στάθμες, βλέπουμε ότι ο αλγόριθμος της κατηγορίας κατασκευής δέντρου αποφάσεων είναι ο πιο εύστοχος (πριν ο J48, τώρα ο M5P). Ο M5P κάνει μη-γραμμικό διαχωρισμό στα δεδομένα και παράγει ένα δέντρο αποφάσεων το οποίου τα φύλλα χρησιμοποιούν τον αλγόριθμο της γραμμικής παλινδρόμησης (linear regression). Τα αποτελέσματα, τόσο στην περίπτωση πρόβλεψης στάθμης, όσο και στην περίπτωση πρόβλεψης συγκεκριμένου αριθμού πακέτων, κρίνονται ικανοποιητικά. Μάλιστα, το σχετικό απόλυτο σφάλμα και η ρίζα του σχετικού τετραγωνικού σφάλματος, είναι μικρότερα στην δεύτερη περίπτωση.

Ακολουθεί στο σχήμα 6.4 το δέντρο απόφασης που σχηματίστηκε για τον αλγόριθμο J48 (περίπτωση με 9 στάθμες) και στο σχήμα 6.5 μία γραφική παράσταση με τον ακριβή αριθμό πακέτων ανά ώρα. Στον άξονα x είναι οι ώρες της ημέρας (24 στήλες), στον άξονα y τα πακέτα που μετρήθηκαν (κόκκινα σημεία) και με μαύρο η παρεμβολή αυτών των σημείων με μία εξίσωση 3ου βαθμού με την βοήθεια του gnuplot. Από το τελευταίο γράφημα μπορούμε να καταλάβουμε την περιοδικότητα στο throughput του δικτύου, καθώς και ποιο από τα πεδία μας (ημερομηνία, μέρα μήνα, μέρα εβδομάδας, ώρα) ήταν πιο χρήσιμο ως προς την μοντελοποίηση που επιχειρήσαμε.

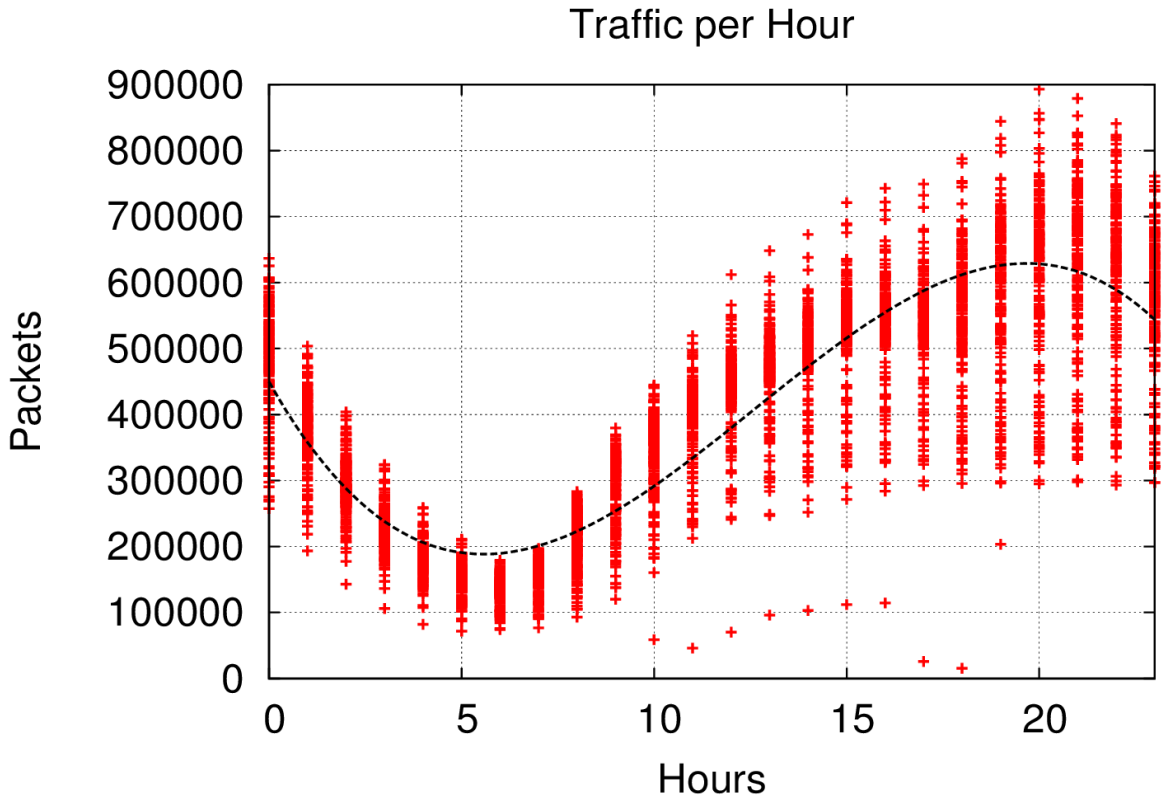
```
J48 pruned tree
------------------

hour = 0
|   date <= 26: 4 (25.0/4.0)
|   date > 26
|   |   date <= 103: 5 (77.0/17.0)
|   |   date > 103: 6 (94.0/28.0)
hour = 1
|   date <= 105
|   |   date <= 19
|   |   |   date <= 3: 4 (2.0)
|   |   |   date > 3: 3 (16.0)
|   |   date > 19: 4 (86.0/3.0)
|   date > 105: 5 (92.0/35.0)
hour = 2
|   date <= 105: 3 (104.0/18.0)
|   date > 105: 4 (92.0/26.0)
hour = 3
|   date <= 21: 2 (20.0/1.0)
|   date > 21: 3 (175.0/14.0)
hour = 4
|   date <= 108: 2 (107.0/2.0)
|   date > 108
|   |   date <= 180
|   |   |   date <= 149: 2 (41.0/9.0)
|   |   |   date > 149: 3 (30.0/8.0)
|   |   date > 180: 2 (18.0/5.0)
hour = 5
|   date <= 194: 2 (192.0/4.0)
|   date > 194: 1 (4.0)
hour = 6
|   date <= 19
|   |   date <= 10: 2 (9.0/2.0)
|   |   date > 10: 1 (9.0)
|   date > 19
|   |   date <= 194: 2 (174.0)
|   |   date > 194: 1 (4.0)
hour = 7: 2 (196.0/10.0)
hour = 8
|   weekday = 1: 2 (28.0/1.0)
|   weekday = 2: 3 (29.0/5.0)
|   weekday = 3: 3 (28.0/3.0)
|   weekday = 4: 3 (27.0/6.0)
|   weekday = 5: 3 (28.0/7.0)
|   weekday = 6: 3 (28.0/6.0)
|   weekday = 7: 2 (28.0/10.0)
hour = 9
|   weekday = 1: 3 (28.0/6.0)
|   weekday = 2: 4 (29.0/8.0)
|   weekday = 3: 4 (28.0/7.0)
|   weekday = 4: 4 (27.0/7.0)
|   weekday = 5: 4 (28.0/10.0)
|   weekday = 6: 4 (28.0/8.0)
|   weekday = 7: 3 (28.0/9.0)
hour = 10
|   date <= 21: 3 (20.0/5.0)
|   date > 21: 4 (176.0/45.0)
hour = 11
|   date <= 65: 4 (65.0/25.0)
|   date > 65: 5 (132.0/28.0)
hour = 12
|   date <= 26: 4 (26.0/8.0)
|   date > 26: 5 (170.0/22.0)
hour = 13
|   date <= 26: 4 (26.0/7.0)
|   date > 26: 5 (170.0/48.0)
hour = 14
|   date <= 33: 4 (33.0/16.0)
|   date > 33: 6 (163.0/71.0)
hour = 15
|   date <= 26: 4 (26.0/10.0)
|   date > 26: 6 (169.0/36.0)
hour = 16
|   date <= 33: 5 (33.0/17.0)
|   date > 33: 6 (162.0/35.0)
hour = 17
|   date <= 33: 5 (33.0/16.0)
|   date > 33: 6 (162.0/36.0)
hour = 18
|   date <= 33: 4 (33.0/17.0)
|   date > 33
|   |   date <= 90: 6 (57.0/8.0)
|   |   date > 90: 7 (106.0/52.0)
hour = 19
|   date <= 65
|   |   date <= 33: 4 (32.0/16.0)
|   |   date > 33: 6 (32.0/1.0)
|   date > 65: 7 (131.0/38.0)
hour = 20
|   date <= 61
|   |   date <= 33: 4 (33.0/17.0)
|   |   date > 33: 6 (28.0/3.0)
|   date > 61
|   |   date <= 107: 7 (46.0/12.0)
|   |   date > 107: 8 (89.0/50.0)
hour = 21
|   date <= 60
|   |   date <= 32
|   |   |   date <= 19: 4 (19.0/4.0)
|   |   |   date > 19: 5 (13.0/1.0)
|   |   date > 32: 6 (28.0/6.0)
|   date > 60: 7 (136.0/72.0)
hour = 22
|   date <= 61
|   |   date <= 26: 4 (26.0/12.0)
|   |   date > 26: 6 (35.0/11.0)
|   date > 61: 7 (135.0/62.0)
hour = 23
|   date <= 33: 5 (33.0/17.0)
|   date > 33
|   |   date <= 90: 6 (57.0/18.0)
|   |   date > 90: 7 (106.0/51.0)

Number of Leaves  :     72

Size of the tree :      111
```

Σχήμα 6.4: Δέντρο Απόφασης J48 με 9 Στάθμες



Σχήμα 6.5: Πακέτα ανά Ώρα

Κεφάλαιο 7
Επίλογος

7.1 Σύνοψη - Συμπεράσματα

Στην εργασία αυτή ασχοληθήκαμε με τη σχεδίαση και υλοποίηση ενός συστήματος για την αποδοτική επεξεργασία μεγάλου όγκου δεδομένων. Τα δεδομένα που χρησιμοποιήθηκαν προέρχονται από τον κόμβο GR-IX, τον ελληνικό κόμβο ουδέτερης διασύνδεσης. Σκοπός της διπλωματικής ήταν ο συνδυασμός αυτής της βασικής πληροφορίας με δευτερεύουσες πληροφορίες που αφορούν το αυτόνομο σύστημα που ανήκουν οι διευθύνσεις IP και η χώρα προέλευσης. Σε σχέση με προηγούμενες διπλωματικές εργασίες [12] αυτό που επιδιώξαμε ήταν να γίνει η ανάλυση της δικτυακής κίνησης από ένα σύστημα πραγματικού χρόνου, και όχι με τις κλασσικές τεχνικές Batch Processing. Για τον λόγο αυτό, με πυρήνα μας το εργαλείο Apache Storm, δημιουργήσαμε ένα τέτοιου είδους σύστημα. Στο παραπάνω σύστημα εφαρμόσαμε τρία διαφορετικά configuration με διαφορετικό επίπεδο παραλληλισμού το καθένα για να δούμε τελικά ότι το σύστημα μας μπορεί να κλιμακώσει, δηλαδή να αυξήσει τον ρυθμό επεξεργασίας των δεδομένων αν σε αυτό προστεθούν περισσότεροι κόμβοι. Τέλος, με χρήση του ανοιχτού λογισμικού Weka, επιχειρήσαμε να δώσουμε μία εκτίμηση (πρόβλεψη) της ωριαίας του δικτύου GR-IX. Στο σημείο αυτό επεξεργαστήκαμε τρία σενάρια: υπολογισμό στάθμης κίνησης μέσω 9 σταθμών, μέσω 18 σταθμών και ακριβούς αριθμού πακέτων. Τα αποτελέσματα κρίνονται ικανοποιητικά.

7.2 Μελλοντικές Εργασίες

Το συγκεκριμένο πεδίο έρευνας κρίνεται εξαιρετικά ενδιαφέρον και θα ήταν ιδιαίτερα χρήσιμο για έναν νέο μηχανικό να μπορέσει να συνδυάσει γνώσεις δικτύων υπολογιστών, πληροφορικής, αρχιτεκτονικής και λειτουργίας υπολογιστικών συστημάτων, οι οποίες αποτελούν μέρος των ακαδημαϊκών σπουδών του, για να φέρει εις πέρας μία τέτοιου είδους έρευνα. Οι επεκτάσεις αυτής της έρευνα μπορεί να είναι πραγματικά πολλές ανάλογα το προσωπικό ενδιαφέρον και τις επιδιώξεις του εκάστοτε ερευνητή. Τα δεδομένα μας για παράδειγμα θα μπορούσαν να εμπλουτιστούν με περισσότερα δημοσίως διαθέσιμα δεδομένα όπως IP διευθύνσεις Google ή Akamai Servers και να παραχθούν στατιστικά για συγκεκριμένα αυτόνομα δίκτυα. Εναλλακτικά, μπορούμε να υλοποιήσουμε real-time machine learning αλγόριθμους οι οποίοι θα τρέχουν σε ένα τέτοιο σύστημα και θα εντοπίζουν έγκαιρα ανωμαλίες στην ροή των δεδομένων. Ένα πιθανό σενάριο στην περίπτωση δεδομένων που προέρχονται από δικτυακή κίνηση είναι ο έγκαιρος εντοπισμός DDoS attacks.

Παράρτημα

Βιβλιογραφία

[1] Data, data everywhere. http://www.economist.com/node/15557443

[2] GeoLite Database. http://dev.maxmind.com/geoip/legacy/geolite/

[3] SFlow Tool. http://www.sflow.org/

[4] Apache Software Foundation. http://www.apache.org/

[5] Apache Hadoop. http://hadoop.apache.org/

[6] Apache Hive. http://hive.apache.org/

[7] Apache Storm. http://storm.apache.org/

[8] Apache ZooKeeper. http://zookeeper.apache.org/

[9] Apache Kafka. http://kafka.apache.org/

[10] Weka. http://www.cs.waikato.ac.nz/ml/weka/

[11] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[12] Σαρλής, Δ. (2014). Κατανεμημένοι Αλγόριθμοι Ερωτημάτων Ένωσης με Εφαρμογές στην Ανάλυση Δεδομένων Δικτυακής Κίνησης.

[13] Chatzis, N., Smaragdakis, G., Böttger, J., Krenc, T., & Feldmann, A. (2013, October). On the benefits of using a large IXP as an Internet vantage point. In Proceedings of the 2013 conference on Internet measurement conference (pp. 333-346). ACM.

[14] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., ... & Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment, 2(2), 1626-1629.

[15] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010, June). Spark: cluster computing with working sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (Vol. 10, p. 10).

[16] Xin, R. S., Rosen, J., Zaharia, M., Franklin, M. J., Shenker, S., & Stoica, I. (2013, June). Shark: SQL and rich analytics at scale. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of data (pp. 13-24). ACM.

[17] Li, B., Gunes, M. H., Bebis, G., & Springer, J. (2013, June). A supervised machine learning approach to classify host roles on line using sFlow. In Proceedings of the first edition workshop on High performance and programmable networking (pp. 53-60). ACM.

[18] Bumgardner, V. K., & Marek, V. W. (2014, March). Scalable hybrid stream and hadoop network analysis system. In Proceedings of the 5th ACM/SPEC international conference on Performance engineering (pp. 219-224). ACM.

[19] Mirza, M., Sommers, J., Barford, P., & Zhu, X. (2007, June). A machine learning approach to TCP throughput prediction. In ACM SIGMETRICS Performance Evaluation Review (Vol. 35, No. 1, pp. 97-108). ACM.

[20] Erman, J., Mahanti, A., & Arlitt, M. (2006, November). Qrp05-4: Internet traffic identification using machine learning. In Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE (pp. 1-6). IEEE.