Πολυτεχνειο Κρητης
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Εργαστηριο Τεχνολογιας Συστηματων Λογισμικου και
Δικτυακων Εφαρμογων SoftNet

Αποδοτική Πρόβλεψη Εξέλιξης Παράλληλων Καρκινικών Προσομοιώσεων στο Apache Flink

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

της

ΣΩΤΗΡΙΑΣ ΜΑΡΙΑΣ Σ. ΚΑΤΑΡΑ

Εξεταστική Επιτροπή : Καθηγητής Αντώνιος Δεληγιαννάκης (Επιβλέπων)
Καθηγητής Μίνως Γαροφαλάκης
Αναπληρωτής Καθηγητής Βασίλης Σαμολαδάς

Χανιά, Αύγουστος 2021

Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εργαστήριο Τεχνολογίας Συστημάτων Λογισμικού και
Δικτυακών Εφαρμογών SoftNet

Αποδοτική Πρόβλεψη Εξέλιξης Παράλληλων Καρκινικών Προσομοιώσεων στο Apache Flink

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

της

ΣΩΤΗΡΙΑΣ ΜΑΡΙΑΣ Σ. ΚΑΤΑΡΑ

Επιβλέπων: Καθηγητής Αντώνιος Δεληγιαννάκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Αυγούστου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αντώνιος Δεληγιαννάκης	Μίνως Γαροφαλάκης	Βασίλης Σαμολαδάς
Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Χανιά, Αύγουστος 2021

Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εργαστήριο Τεχνολογίας Συστημάτων Λογισμικού και
Δικτυακών Εφαρμογών SoftNet

(Υπογραφή)

. . . . . .
Σωτηρία Μαρία Κατάρα

26η Αυγούστου 2021

Περίληψη

Η αλματώδης ανάπτυξη των υπολογιστικών συστημάτων, τόσο σταθερών όσο και κινητών, σε συνάρτηση με την ολοένα και μεγαλύτερη διείσδυση των ασύρματων και των ενσύρματων δικτύων έχουν ως συνέπεια την δημιουργία πολύ μεγάλων όγκων δεδομένων σε καθημερινή βάση. Η μελέτη των δεδομένων αυτών, επιτρέπει στους επιστήμονες τον εντοπισμό τάσεων και μοτίβων που μπορούν να χρησιμοποιηθούν για μελλοντικό όφελος. Ένας πολύ σημαντικός τομέας εφαρμογής των μελετών αυτών είναι στην Βιοπληροφορική και συγκεκριμένα στην πρόβλεψη της συμπεριφοράς ετερογενών πολυκυτταρικών συστημάτων, παρέχοντας τη δυνατότητα έγκαιρης λήψης αποφάσεων. Στόχος αυτής της διπλωματικής εργασίας είναι ο εντοπισμός των όμοιων χρονικών παραθύρων ενός συνόλου παράλληλων προσομοιώσεων καρκινικών κυττάρων, με σκοπό την εξαγωγή κατάλληλων πληροφοριών που θα χρησιμοποιηθούν στην πρόβλεψη της συμπεριφοράς αυτών. Η εκπλήρωση του στόχου αυτού συναντά δύο πολύ σημαντικές προκλήσεις. Η υψηλή διαστασιμότητα των δεδομένων σε συνδυασμό με την δαπανηρή από άποψη χρόνου και μνήμης σύγκριση όλων των χρονικών στιγμών των χιλίων τετρακοσίων προσομοιώσεων απαιτούν την εφαρμογή ενός αλγόριθμου, του οποίου η λειτουργικότητα θα συνδυάζει την επίλυση των δύο αυτών πολύ σημαντικών προκλήσεων. Ο αλγόριθμος Random Hyperplane Projection του Locality Sensitive Hashing μπορεί να διευθετήσει και τις δύο προκλήσεις εφαρμόζοντας μείωση των διαστάσεων των δεδομένων σε μικρότερες, διατηρώντας παράλληλα την διαφορετικότητα αυτών, ενώ ταυτόχρονα αναλαμβάνει την ομαδοποίηση παρόμοιων αντικειμένων σε ίδιες ομάδες με μεγάλη πιθανότητα, μέσω της χρήσης κατάλληλων συναρτήσεων κατακερματισμού. Ζωτικής σημασίας είναι η κλιμακωσιμότητα της τεχνικής του αλγόριθμου που θα χρησιμοποιήσουμε, ώστε να επιτευχθεί η βέλτιστη χρονική απόδοση ως προς την εξαγωγή αποτελεσμάτων, παρά την αύξηση του όγκου των εισερχόμενων δεδομένων. Το ζητούμενο αυτό σε συνδυασμό με την ανάγκη για μείωση της χωρικής πολυπλοκότητας οδηγεί στην ανάπτυξη του αλγόριθμου σε μία μηχανή διατήρησης συνόψεων δεδομένων (Synopses Data Engine), η οποία είναι χτισμένη στο Apache Flink και έχει ως στόχο την υποστήριξη μεγάλης ποικιλίας συνόψεων και την προσθήκη νέων λειτουργιών κατά τον χρόνο εκτέλεσης παράλληλα και κατανεμημένα, παρέχοντας με αυτό τον τρόπο την λειτουργικότητα synopsis-as-a-service. Της εκτέλεσης του αλγόριθμου έπεται η ανάπτυξη ενός μαθηματικού μοντέλου πρόβλεψης με την μέθοδο της πολλαπλής γραμμικής παλινδρόμησης με σκοπό την πρόβλεψη της συμπεριφοράς στοιχείων του πολυκυτταρικού συστήματος. Η απόδοση του συστήματος ελέγχθηκε τοπικά και απομακρυσμένα - κατανεμημένα, αποδίδοντας θετικά αποτελέσματα.

Λέξεις Κλειδιά

Ροές δεδομένων, σύνοψη, χρονικές στιγμές, καρκινικές προσομοιώσεις, κατακερματισμός, ομαδοποίηση, πρόβλεψη, Apache Flink, κατανεμημένο σύστημα

Abstract

The rapid growth of computer systems, both fixed and mobile, in relation with the growing penetration of wireless and wired networks have resulted in the creation of very large volumes of data on a daily basis. Studying this data allows scientists to identify trends and patterns that can be used for future benefit. A very important field of application of these studies is in Bioinformatics and specifically in the prediction of the behavior of heterogeneous multicellular systems, providing the possibility of timely decision making. The aim of this diploma thesis is to identify the similar time windows of a set of concurrent cancer cell simulations, in order to extract appropriate information that will be used to predict their behavior. Achieving this goal faces two very important challenges. The high dimensionality of the data combined with the time-consuming and memory-costly comparison of all one thousand four hundred simulations of time require the application of an algorithm, the functionality of which will combine the solution of these two very important challenges.The Random Hyperplane Projection form of the Locality Sensitive Hashing algorithm can solve both challenges by reducing the size of the data to smaller ones, while maintaining their diversity, while at the same time undertaking the grouping of similar objects in the same groups with high probability, through the use of appropriate hash functions. Very important is the scalability of the algorithm technique we will use, in order to achieve the optimal time efficiency in terms of exporting results, despite the increase in the volume of incoming data. This, in combination with the need of reduction spatial complexity leads to the development of the algorithm in a Synopses Data Engine, which is built on Apache Flink and aims to support a wide variety of synopses and add new ones, at runtime, in parallel and distributed way, thus providing the synopsis-as-a-service functionality. The execution of the algorithm is followed by the development of a forecasting mathematical model with the method of multiple linear regression in order to predict the behavior of elements of the multicellular system. The performance of the system was tested locally and remotely - distributed, yielding positive results.

Keywords

Data streams, synopsis, time stamps, cancer simulations, hashing, clustering, forecasting, Apache Flink, distributed system

στην αδερφή μου
Ραφαέλα

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον καθηγητή μου κ. Αντώνιο Δεληγιαννάκη για την επίβλεψη της διπλωματικής εργασίας μου και για την πολύτιμη συμβολή του στην εκκίνηση της μετέπειτα καριέρας μου. Επίσης, ευχαριστώ ιδιαίτερα τον συνάδελφό μου και διδακτορικό φοιτητή Αντώνιο Κονταξάκη, για την συνεχή παρουσία του και τη μεγάλη βοήθειά του στην ολοκλήρωση της διπλωματικής εργασίας, προσφέροντάς μου χρήσιμη γνώση και ψυχική υποστήριξη. Εν συνεχεία θα ήθελα να ευχαριστήσω την κ. Αράπη Ξένια, μέλος του Εργαστηριακού και Εκπαιδευτικού Προσωπικού της Σχολής για την επίλυση τεχνικών προβλημάτων που προέκυψαν κατά την πειραματική διαδικασία της εργασίας.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για την υλική και ηθική υποστήριξη κατά τη διάρκεια των ακαδημαϊκών μου σπουδών καθώς και τις συναδέλφους και αδελφικές μου φίλες Μανάρα Χριστίνα και Μαραγκάκη Μαρία για την άψογη συνεργασία, την καθοδήγησή και την ηθική υποστήριξή που μου προσέφεραν όλα αυτά τα χρόνια.

Χανιά, Αύγουστος 2021

Σωτηρία Μαρία Κατάρα

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια των σπουδών για την απόκτηση του Προπτυχιακού Διπλώματος της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης με τη χρήση πόρων του εργαστηρίου Τεχνολογίας Συστημάτων Λογισμικού και Δικτυακών Εφαρμογών SoftNet.

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Οι φρενήρεις ρυθμοί στην ανάπτυξη των υπολογιστικών συστημάτων, τόσο σταθερών όσο και κινητών, σε συνάρτηση με την ολοένα και μεγαλύτερη διείσδυση των ασύρματων και των ενσύρματων δικτύων έχουν ως συνέπεια την δημιουργία πολύ μεγάλων όγκων δεδομένων σε καθημερινή βάση. Η μελέτη των δεδομένων αυτών, επιτρέπει στους επιστήμονες τον εντοπισμό τάσεων και μοτίβων που μπορούν να χρησιμοποιηθούν για μελλοντικό όφελος. Ένας πολύ σημαντικός τομέας εφαρμογής των μελετών αυτών είναι στην Βιοπληροφορική και συγκεκριμένα στην πρόβλεψη της συμπεριφοράς ετερογενών πολυκυτταρικών συστημάτων, παρέχοντας τη δυνατότητα έγκαιρης λήψης αποφάσεων. Η έγκαιρη λήψη αποφάσεων μέσω της παρατήρησης των δεδομένων έχει ως αποτέλεσμα την αύξηση της αποδοτικότητας και της παραγωγικότητας ενός συστήματος καθώς και την αδιάκοπη βελτίωσή του σε συνδυασμό με την οικονομία σε χρόνο και υπολογιστικούς πόρους. Στόχος αυτής της διπλωματικής εργασίας είναι ο εντοπισμός των όμοιων χρονικών παραθύρων ενός συνόλου παράλληλων προσομοιώσεων καρκινικών κυττάρων, με σκοπό την εξαγωγή κατάλληλων πληροφοριών που θα χρησιμοποιηθούν στην πρόβλεψη της συμπεριφοράς αυτών. Οι προσομοιώσεις αυτές έχουν παραχθεί από το εργαλείο PhysiBoss.To PhysiBoss αναπαριστά ένα C++ software framework το οποίο υλοποιήθηκε το 2019. Πρόκειται για ένα εργαλείο μοντελοποίησης βασισμένο σε πράκτορες που χρησιμοποιείται για multiscale προσομοιώσεις ετερογενών πολυκυτταρικών συστημάτων. Κάθε μία από τις προσομοιώσεις αυτές έχει διάρκεια εικοσιτεσσάρων ωρών, κατά την οποία τα κύτταρα της διέρχονται από μια πληθώρα καταστάσεων. Η εργασία αυτή εστιάζει σε 3 διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί ένα κύτταρο, αναφερόμενες ως alive, apoptotic και necrotic.

Η εκπλήρωση του στόχου αυτού συναντά δύο πολύ σημαντικές προκλήσεις. Η υψηλή διαστασιμότητα των δεδομένων σε συνδυασμό με την δαπανηρή από άποψη χρόνου και μνήμης σύγκριση όλων των χρονικών παραθύρων των δεδομένων προσομοιώσεων απαιτούν την εφαρμογή ενός αλγόριθμου, του οποίου η λειτουργικότητα θα συνδυάζει την επίλυση των δύο αυτών πολύ σημαντικών προκλήσεων. Ο αλγόριθμος Random Hyperplane Projection του Locality Sensitive Hashing μπορεί να διευθετήσει και τις δύο προκλήσεις εφαρμόζοντας μείωση των διαστάσεων των δεδομένων σε μικρότερες, διατηρώντας παράλληλα την διαφορετικότητα αυτών, ενώ ταυτόχρονα αναλαμβάνει την ομαδοποίηση παρόμοιων αντικειμένων σε ίδιες ομάδες με μεγάλη πιθανότητα, μέσω της χρήσης κατάλληλων συναρτήσεων κατακερματισμού.

Η καταλληλότητα των συναρτήσεων κατακερματισμού που χρησιμοποιούνται από τον αλγόριθμο εξαρτάται από τον βαθμό μεγιστοποίησης των συγκρούσεων στο εσωτερικό ενός bucket του πίνακα κατακερματισμού. Σε συνέχεια της στοχευμένης ομαδοποίησης των δεδομένων, εφαρμόζεται στα ζητούμενα δεδομένα ένα μοντέλο γραμμικής παλινδρόμησης, κατάλληλα προσαρμοσμένο σε αυτά, με απώτερο σκοπό την πρόβλεψη της μελλοντικής συμπεριφοράς τους.

Μείζον ζήτημα της εργασίας αυτής είναι η κλιμακωσιμότητα της τεχνικής του αλγόριθμου που θα χρησιμοποιήσουμε, ώστε να επιτευχθεί η βέλτιστη χρονική απόδοση ως προς την εξαγωγή αποτελεσμάτων, παρά την αύξηση του όγκου των εισερχόμενων δεδομένων. Το ζητούμενο αυτό σε συνδυασμό με την ανάγκη για μείωση της χωρικής πολυπλοκότητας οδηγεί στην ανάπτυξη του αλγόριθμου σε μία μηχανή διατήρησης συνόψεων δεδομένων (Synopses Data Engine), η οποία είναι χτισμένη στο Apache Flink και έχει ως στόχο την υποστήριξη μεγάλης ποικιλίας συνόψεων και την προσθήκη νέων λειτουργιών κατά τον χρόνο εκτέλεσης παράλληλα και κατανεμημένα, παρέχοντας με αυτό τον τρόπο την λειτουργικότητα synopsis-as-a-service. Για την είσοδο των δεδομένων ως streams στο σύστημα SDE χρησιμοποιείται το κατανεμημένο, εξαιρετικά επεκτάσιμο και ασφαλές σύστημα του Apache Kafka, το οποίο αξιοποιείται ταυτόχρονα για την εγγραφή των αποτελεσμάτων που εξάγονται κατά την εκτέλεση των ερωτημάτων, τα οποία τίθενται στα δεδομένα που διατηρούνται στο εσωτερικό των συνόψεων.

Ακολουθεί η περιγραφή των μερών της διπλωματικής εργασίας, η διάρθρωση και η ανάλυση των οποίων έχουν, ως απώτερο σκοπό, την καλύτερη δυνατή κατανόηση της από τον αναγνώστη.

1.2 Οργάνωση του τόμου

Η διπλωματική εργασία απαρτίζεται από δύο κύρια μέρη, το θεωρητικό και το πρακτικό. Το θεωρητικό μέρος εκκινείται με το Κεφάλαιο 2, οπού πραγματοποιείται αναλυτική περιγραφή των εννοιών και των συστημάτων που κρίνονται απαραίτητα για τη διεξαγωγή της εργασίας. Ακολουθεί το Κεφάλαιο 3, όπου αναφέρονται περιληπτικά έρευνες και εργασίες που πραγματεύονται σημαντικά συνθετικά στοιχεία αυτής της διπλωματικής εργασίας, όπως ο αλγόριθμος Random Hyperplane Projection του Locality Sensitive Hashing και οι παράλληλες καρκινικές προσομοιώσεις. Το Κεφάλαιο 4 σηματοδοτεί την έναρξη του πρακτικού μέρους, οπού αναλύεται διεξοδικά η αρχιτεκτονική του συστήματος και ο αλγόριθμος που χρησιμοποιήθηκε ενώ περιγράφεται και η διαδικασία της πρόβλεψης των χρονοσειρών. Το πρακτικό μέρος ολοκληρώνεται με το Κεφάλαιο 5, όπου πραγματοποιείται η αναλυτική παρουσίαση του ελέγχου ορθότητας του συστήματος. Τέλος, ακολουθεί το Κεφάλαιο 6 ως ο επίλογος της διπλωματικής εργασίας, όπου παρατίθενται τα συμπεράσματα επί της μελέτης που πραγματοποιήθηκε και γίνονται προτάσεις προς μελλοντική επέκτασή της.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό αρχικά γίνεται μια περιγραφή των εννοιών και των συστημάτων που χρησιμοποιούνται στην υλοποίηση της διπλωματικής εργασίας. Στη συνέχεια αναλύεται το σύστημα στο οποίο ενσωματώθηκε η υλοποίηση αυτή. Τέλος περιγράφεται αναλυτικά το θεωρητικό υπόβαθρο του αλγόριθμου που χρησιμοποιήθηκε.

2.1 Ροές Δεδομένων

Μια ασύλληπτη ποσότητα δεδομένων παράγεται κάθε λεπτό της ημέρας από όλο τον κόσμο, η οποία συνεχίζει να πολλαπλασιάζεται με αξιοσημείωτο ρυθμό. Οι εταιρείες αρκετών κλάδων της παγκόσμιας βιομηχανίας, προς εξυπηρέτηση των αναγκών τους, αλλάζουν τον τρόπο επεξεργασίας των δεδομένων τους από batch σε stream processing σε πραγματικό χρόνο προκειμένου να συμβαδίζουν με τις σύγχρονες επιχειρηματικές απαιτήσεις.

Γνωστή και ως event stream processing, streaming data είναι η συνεχής και απεριόριστη ροή δεδομένων που παράγεται από διάφορες πηγές. Χρησιμοποιώντας stream processing τεχνολογία, οι ροές δεδομένων (data streams) μπορούν να υποβληθούν σε επεξεργασία, να αποθηκευτούν, να αναλυθούν και να χρησιμοποιηθούν όπως δημιουργούνται σε πραγματικό χρόνο, με αποτέλεσμα να δίνουν το προνόμιο της γρήγορης ανίχνευσης κρίσιμων γεγονότων στις εταιρίες που τις χρησιμοποιούν. Οι περιπτώσεις χρήσης των ροών δεδομένων είναι ποικίλες και αρκετά διαδεδομένες. Μερικές από αυτές είναι οι ακόλουθες: η επεξεργασία πληρωμών και οικονομικών συναλλαγών σε πραγματικό χρόνο, η συνεχής ανάγνωση και ανάλυση δεδομένων αισθητήρων, η συλλογή και άμεση ανταπόκριση σε αιτήματα πελατών που αφορούν το πλαίσιο εξυπηρέτησης εταιρειών, η διαχείριση ασθενών στα πλαίσια της νοσοκομειακής φροντίδας και η πρόβλεψη αλλαγών στην κατάστασή της υγείας τους με σκοπό την έγκαιρη αντιμετώπιση έκτακτων καταστάσεων, καθώς και πολλές ακόμη περιπτώσεις χρήσης.

2.2 Apache Kafka

Το Apache Kafka [3] συνδυάζει τρεις βασικές δυνατότητες, ώστε ο χρήστης να μπορεί να εφαρμόσει τις περιπτώσεις χρήσεις με τη βοήθεια των data streams:

- write και read των data streams, συμπεριλαμβανομένης της εισαγωγής/εξαγωγής δεδομένων από άλλα συστήματα,

- αποθήκευση των data streams με διάρκεια και αξιοπιστία,
- επεξεργασία των data streams.

Το Apache Kafka είναι ένα κατανεμημένο, εξαιρετικά επεκτάσιμο, ανεκτικό σε σφάλματα και ασφαλές σύστημα το οποίο αποτελείται από servers και clients, που επικοινωνούν μεταξύ τους μέσω ενός TCP πρωτοκόλλου υψηλής απόδοσης. Είναι σημαντικό να αναφερθεί πως το Apache Kafka είναι χτισμένο πάνω στο Apache Zookeeper, με σκοπό την παρακολούθηση της κατάστασης των κόμβων στον Kafka cluster και την διατήρηση των topics και των messages στο εσωτερικό του συστήματος. Ουσιαστικά, το Apache Zookeeper πρόκειται για μια κεντρική υπηρεσία με σκοπό τη διατήρηση πληροφοριών διαμόρφωσης και ονομάτων καθώς και την παροχή κατανεμημένου συγχρονισμού και group services.

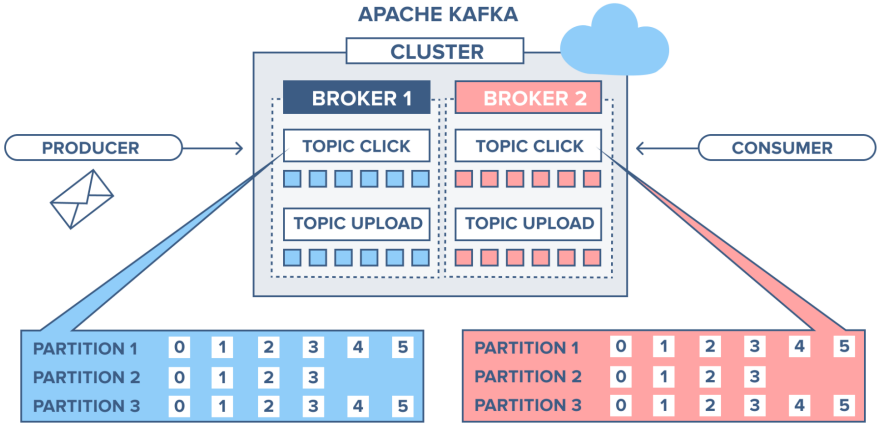
Αναφορικά με τους **servers**, το Kafka τρέχει ως cluster που αποτελείται από έναν ή περισσότερους servers, οι οποίοι βρίσκονται τοποθετημένοι σε data centers ή στο cloud. Ένα υποσύνολο των server αυτών, γνωστοί ως brokers αποτελούν το επίπεδο αποθήκευσης. Άλλοι servers χρησιμοποιούνται στην λειτουργία εργαλείων του Kafka, όπως το Kafka Connect, τα οποία είναι υπεύθυνα για την συνεχή εισαγωγή/εξαγωγή δεδομένων ως streams με σκοπό την ενσωμάτωση του Kafka με τα υπάρχοντα συστήματα, όπως είναι οι σχεσιακές βάσεις και άλλοι Kafka clusters.

Οι clients επιτρέπουν στους χρήστες να υλοποιούν κατανεμημένες εφαρμογές και microservices οι οποίες γράφουν, διαβάζουν και επεξεργάζονται τα stream events με fault tolerant και παράλληλο τρόπο σε περίπτωση προβλημάτων δικτύου ή αστοχιών του μηχανήματος.Το σύνολο των client που είναι υπεύθυνο για την εγγραφή των event στο κατανεμημένο σύστημα του Kafka συνθέτει τους producers. Στην αντίθετη πλευρά το σύνολο των client που διαχειρίζεται την ανάγνωση και την επεξεργασία των event αντιστοιχεί στους consumers του συστήματος. Οι producers και οι consumers είναι απόλυτα ανεξάρτητα components.

Τα events οργανώνονται και αποθηκεύονται σε topics. Ένα topic μπορεί να έχει μηδέν, έναν ή πολλούς producers που γράφουν events σε αυτό και αντίστοιχα μηδέν, έναν ή πολλούς consumers που διαβάζουν από αυτό. Ένα πολύ σημαντικό χαρακτηριστικό των topics είναι η διαχωρισιμότητά τους (partitioned topics). Αυτό σημαίνει ότι μπορεί να διαχωριστεί σε τμήματα τα οποία κατανέμονται στους Kafka brokers. Ο κατανεμημένος διαχωρισμός των δεδομένων που επιτυγχάνεται με αυτόν τον τρόπο συμβάλλει σημαντικά στην επεκτασιμότητα του συστήματος καθώς επιτρέπεται στις client εφαρμογές η ταυτόχρονη ανάγνωση και εγγραφή δεδομένων από και προς τους Kafka brokers. Στην Εικόνα 2.1 περιγράφεται η αρχιτεκτονική ενός Kafka Cluster.

2.3 Περιορισμός Δεδομένων

Μία από τις μεγαλύτερες προκλήσεις για ένα Data Stream Management System (DSMS) είναι ο χειρισμός δυνητικά άπειρων ροών δεδομένων χρησιμοποιώντας σταθερή ποσότητα μνήμης και χωρίς τυχαία πρόσβαση στα δεδομένα. Υπάρχουν διαφορετικές προσεγγίσεις για τον περιορισμό της ποσότητας δεδομένων σε ένα πέρασμα, οι οποίες μπορούν να χωριστούν σε δύο κατηγορίες. Αφενός, υπάρχουν τεχνικές συμπίεσης που προσπαθούν να συνοψίσουν



Εικόνα 2.1: *Η Αρχιτεκτονική του Apache Kafka [1]*

τα δεδομένα και από την άλλη υπάρχουν τεχνικές παραθύρων που προσπαθούν να χωρίσουν τα δεδομένα σε (πεπερασμένα) μέρη.

2.3.1 Συνόψεις

Η ιδέα πίσω από τις τεχνικές συμπίεσης είναι η διατήρηση μόνο μιας σύνοψης (περίληψης) των δεδομένων, και όχι όλων των δεδομένων της ροής. Συνεπώς, η σύνοψη αντιστοιχεί σε μία κλάση που απαρτίζεται από διάφορες δομές δεδομένων και διαδικασίες (συναρτήσεις) με απώτερο σκοπό την αναπαράσταση, περίληψη ή ακόμη και συμπίεση δεδομένων πολύ μεγάλης κλίμακας σε λογαριθμική ή σταθερή χωρική πολυπλοκότητα. Παραδειγματικά, το αντικείμενο της παρούσας διπλωματικής απαιτεί την αναπαράσταση της πληροφορίας των εκάστοτε παραθύρων σε bitmaps συγκεκριμένου μεγέθους. Βασικά χαρακτηριστικά μιας σύνοψης [4] είναι τα ακόλουθα:

1. **Single Pass**: οι συνόψεις δημιουργούνται σχετικά εύκολα, κατά τη διάρκεια άφιξης των δεδομένων με ένα μόνο πέρασμα.

2. **Small Update Time / Memory Footprint**: ο χρόνος ενημέρωσης και ερωτήματος προς τη σύνοψη καθώς και το αποτύπωμα της μνήμης είναι αρκετά μικρός ώστε να μπορεί το σύστημα να ανταποκριθεί στον υψηλό ρυθμό άφιξης της εισόδου. Στην βέλτιστη περίπτωση ο χρόνος αυτός αντιστοιχεί σε πολυπλοκότητα O(1) και στην χειρότερη σε πολυλογαριθμική σε N, όπου N ο συνολικός αριθμός των streams .

Αρκετές συνόψεις έχουν δύο πολύ σημαντικές ιδιότητες:

1. **Delete-proof**: οι συνόψεις μπορούν να χειριστούν εισαγωγές και διαγραφές κατά τη διάρκεια μιας ενημέρωσης στο data stream.

2. **Composable**: μπορούν να δημιουργηθούν πολλές και διαφορετικές συνόψεις σε ανεξάρτητα τμήματα του stream και στη συνέχεια να συνενωθούν ώστε να συνθέσουν μια μεγάλη σύνοψη κατά μήκος όλου του stream.

2.3.2 Παράθυρα

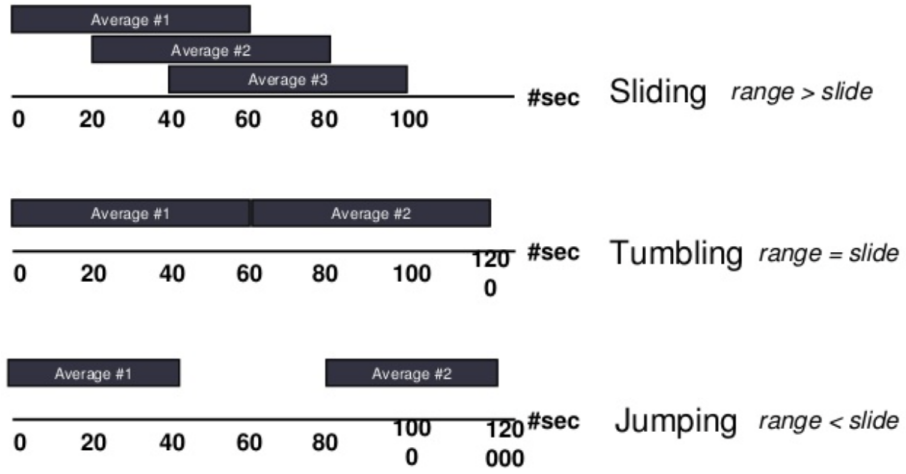
Για την επίτευξη του σκοπού της μείωσης των δεδομένων που εισάγονται σε ένα σύστημα, εκτός των συνόψεων, χρησιμοποιούνται οι τεχνικές παραθύρων. Σε αντίθεση με τις συνόψεις η χρήση παραθύρων αποσκοπεί στην απομόνωση ενός μέρους των δεδομένων και όχι στη συμπίεση αυτών. Συχνό αίτημα αποτελεί η επεξεργασία των πιο πρόσφατα παραγόμενων δεδομένων (fresh data). Δύο πολύ σημαντικά χαρακτηριστικά των παραθύρων, τα οποία ορίζονται από το ζητούμενο του ερωτήματος, είναι το εύρος - μέγεθος και το μέτρο ολίσθησης. Αναλυτικότερα:

- **Εύρος (Range)**: αντιστοιχεί στην χρονική διάρκεια του παραθύρου (1 λεπτό, 1000 tuples)
- **Μέτρο Ολίσθησης (Slide):** αντιστοιχεί στην συχνότητα δημιουργίας ενός νέου παραθύρου.

Στην Εικόνα 2.2 αποδίδεται η σχηματική αναπαράσταση ενός παραθύρου, ενώ αποτυπώνεται η συσχέτιση μεταξύ εύρους (1 λεπτό) και μέτρου ολίσθησης (20 δευτερόλεπτα).

Εικόνα 2.2: *Παράθυρο*

Η συσχέτιση αυτή διαμορφώνει τρεις κατηγορίες παραθύρων (Sliding, Tumbling και Jumping) όπως φαίνεται στην ακόλουθη Εικόνα 2.3.



Εικόνα 2.3: *Κατηγορίες Παραθύρων*

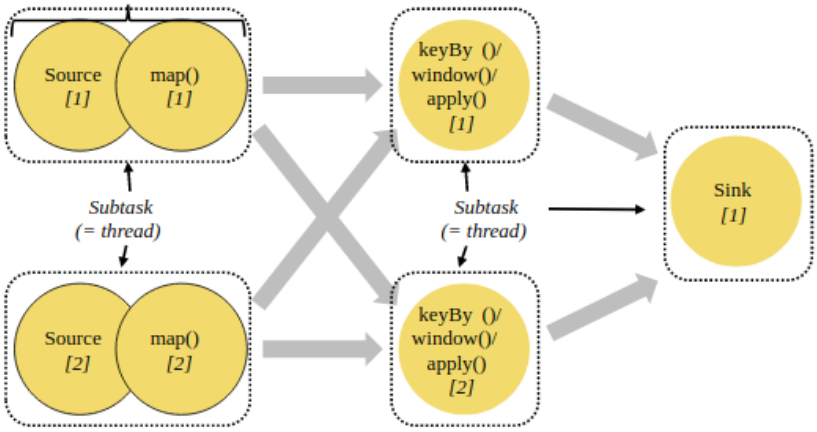
2.4 Apache Flink

Το Apache Flink [2] αναπαριστά μια open-source πλατφόρμα που παρέχει επεκτάσιμες, κατανεμημένες και fault-tolerant δυνατότητες επεξεργασίας ροών δεδομένων. Η σχεδίασή

του επιτρέπει την άρτια λειτουργία του σε κοινά cluster environments και την εκτέλεση υπολογισμών κάθε είδους κλίμακας. Το Apache Flink επιτρέπει την απορρόφηση μαζικών ροών δεδομένων (έως και πολλά terabyte) από διαφορετικές πηγές και την επεξεργασία τους με κατανεμημένο τρόπο σε πολλούς κόμβους. Τα βασικά δομικά στοιχεία ενός Flink pipeline είναι η είσοδος, η επεξεργασία και η έξοδος των δεδομένων. Ο χρόνος εκτέλεσής του υποστηρίζει επεξεργασία χαμηλής καθυστέρησης σε εξαιρετικά υψηλή απόδοση με ανεκτικό σφάλμα. Οι εφαρμογές που έχουν ως βάση τους το Apache Flink επεξεργάζονται τις ροές δεδομένων ως bounded ή unbounded σύνολα δεδομένων. Τα unbounded streams δεν έχουν καθορισμένο τέλος και υποβάλλονται σε συνεχή επεξεργασία, σε αντίθεση με τα bounded streams που χαρακτηρίζονται από οριοθετημένη αρχή και τέλος με άμεση συνέπεια την επεξεργασία τους ως batch data.

Το λογισμικό του Flink περιλαμβάνει τα API DataStream και DataSet για επεξεργασία άπειρων και πεπερασμένων δεδομένων, αντίστοιχα. Το Flink προσφέρει πολλαπλές λειτουργίες σε ροές δεδομένων ή σύνολα, όπως το mapping, το φιλτράρισμα, η ομαδοποίηση δεδομένων, η ενημέρωση κατάστασης, το joining και το aggregating. Οι δύο κύριες μορφές δεδομένων του Flink είναι το DataStream και το DataSet και αντιπροσωπεύουν συλλογές στοιχείων δεδομένων που προορίζονται για ανάγνωση. Η λίστα των στοιχείων είναι περιορισμένη (δηλ. Πεπερασμένη) στο DataSet, ενώ είναι απεριόριστη (δηλαδή, άπειρη) στην περίπτωση του DataStream.

Τα προγράμματα που βασίζονται στο Flink αναπαρίστανται από ένα γράφημα ροής δεδομένων (δηλ. Κατευθυνόμενο ακυκλικό γράφημα-DAG) που εκτελείται στον πυρήνα του Flink, που είναι μια κατανεμημένη μηχανή ροής δεδομένων. Τα γραφήματα ροής δεδομένων αποτελούνται από stateful operators και ενδιάμεσα data stream partitions. Η εκτέλεση κάθε operator πραγματοποιείται από πολλά παράλληλα instances των οποίων ο αριθμός καθορίζεται από το επίπεδο παραλληλισμού. Κάθε παράλληλο operator instance εκτελείται σε ένα ανεξάρτητο task slot σε ένα μηχάνημα που ανήκει σε έναν cluster υπολογιστών. Το παρακάτω σχήμα δείχνει ένα παράδειγμα του γραφήματος ροής δεδομένων για την εφαρμογή του Flink. Η Εικόνα 2.4 δείχνει ένα παράδειγμα του γραφήματος ροής δεδομένων για την εφαρμογή του Flink.



Εικόνα 2.4: *Γράφημα Ροής Δεδομένων στο Apache Flink [2]*

2.5 Synopsis Data Engine (SDE)

Το Synopsis Data Engine (SDE) [5] πρόκειται για ένα DSMS (Data Stream Management System), το οποίο συνδυάζει τα πλεονεκτήματα της παράλληλης επεξεργασίας και των συνόψεων ροών δεδομένων με σκοπό την παροχή διαδραστικών αναλυτικών στοιχειών σε μεγάλη κλίμακα. Η ανάπτυξη του SDE βασίζεται πάνω στην αρχιτεκτονική και στην λειτουργικότητα του Apache Flink, παρέχοντας την δυνατότητα χρήσης των συνόψεων ως υπηρεσίες (synopsis-as-a-service). Μέσω της υλοποίησης αυτής επιτυγχάνεται (α) η ταυτόχρονη διατήρηση χιλιάδων συνόψεων διαφόρων τύπων για χιλιάδες ροές δεδομένων, (β) η επαναχρησιμοποίηση των ήδη διατηρημένων συνόψεων με σκοπό την εξυπηρέτηση διαφόρων ζητούμενων των περιπτώσεων χρήσης, (γ) η παροχή μεθόδων σύνοψης των δεδομένων σε Big Data πλατφόρμες, (δ) η δυνατότητα ενσωμάτωσης νέων συνόψεων on-the-fly καθώς και πολλές ακόμη λειτουργικότητες. Το SDE είναι πολύ χρήσιμο για την εξαγωγή αναλυτικών στοιχείων μεγάλης κλίμακας καθώς επιτρέπει αυξημένη οριζόντια, κάθετη και ενοποιημένη επεκτασιμότητα. Μέσω της οριζόντιας επεκτασιμότητας παραλληλοποιεί τους απαραίτητους υπολογισμούς και τις ζητούμενες διεργασίες σε έναν αριθμό από μηχανές με σκοπό την βέλτιστη διαχείριση του μεγάλου όγκου δεδομένων και του αυξημένου ρυθμού εμφάνισης εισόδου, ενώ στην κλιμάκωση του υπολογισμού σε πολύ μεγάλο αριθμό επεξεργασμένων ροών συμβάλλει η κάθετη επεκτασιμότητα.

2.6 Ο Αλγόριθμος Random Hyperplane Projection (RHP) του Locality Sensitive Hashing (LSH)

Πριν την εκτενής ανάλυση του αλγόριθμου περιγράφονται κάποιες μαθηματικές έννοιες, βασικές για την κατανόηση της λειτουργικότητας αυτού.

2.6.1 Μετρικές Απόστασης

Ο αλγόριθμος που αναπτύσσεται στην συγκεκριμένη εργασία χρησιμοποιεί για την εξυπηρέτηση των υπολογισμών, που γίνονται στο εσωτερικό του, διάφορα είδη μετρικών αποστάσεων με πιο γνωστά την Ευκλείδεια απόσταση μεταξύ δύο διανυσμάτων και την απόσταση Hamming. Η παρούσα εργασία επικεντρώνεται στην δεύτερη (hamming distance) καθώς μέσω αυτής υπολογίζεται το cosine similarity μεταξύ δύο bitmaps (βλ.παρακάτω).

Στην θεωρία πληροφορίας, ως απόσταση Hamming [6] μεταξύ δύο συμβολοσειρών ίσου μήκους ορίζεται ο αριθμός θέσεων στις οποίες τα αντίστοιχα σύμβολα είναι διαφορετικά. Η απόσταση Hamming, μετρά τον ελάχιστο αριθμό αντικαταστάσεων που χρειάζονται ώστε να μετατραπεί η μία συμβολοσειρά στην άλλη, ή αλλιώς, τον αριθμό των λαθών που μετέτρεψαν την μία συμβολοσειρά στην άλλη.

2.6.2 Μετρικές Ομοιότητας

Ο LSH αλγόριθμος χρησιμοποιεί για τον υπολογισμό της ομοιότητας μεταξύ δύο vectors τις παρακάτω μετρικές, κάθε μία σε διαφορετικό μέρος της υλοποίησής του.

Cosine Similarity

Η ομοιότητα των συνημίτονων [7] αντιστοιχεί σε μία μετρική, η οποία έχεις ως στόχο τον προσδιορισμό της ομοιότητας μεταξύ δύο μη μηδενικών διανυσμάτων που προβάλλονται στον πολυδιάστατο χώρο. Από μαθηματικής απόψεως, υπολογίζει το συνημίτονο της γωνίας μεταξύ των διανυσμάτων αυτών. Ο υπολογισμός της ομοιότητας γίνεται μέσω της Εξίσωσης 2.1,

$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.1}$$

όπου τα A_i και B_i αποτελούν στοιχεία των διανυσμάτων A και B αντίστοιχα.

Pearson Correlation Coefficient

Στην στατιστική ο συντελεστής συσχέτισης Pearson [8] αποτελεί ένα μέτρο γραμμικής συσχέτισης μεταξύ δύο συνόλων δεδομένων. Μαθηματικά, ισούται με την συν-διακύμανση δύο μεταβλητών, διαιρεμένη με το γινόμενο των τυπικών αποκλίσεων τους (βλ. Εξίσωση 2.2). Ουσιαστικά, αφορά σε μια κανονικοποιημένη μέτρηση της συν-διακύμανσης, έτσι ώστε το αποτέλεσμα να έχει πάντα μια τιμή μεταξύ -1 και 1. Ο συντελεστής αυτός αντικατοπτρίζει μόνο μια γραμμική συσχέτιση μεταξύ των μεταβλητών, αγνοώντας άλλους τύπους συσχέτισης.

$$\rho_{A,B} = \frac{\sum \left((A_i - \bar{A}) * (B_i - \bar{B})\right)}{\sqrt{\sum \left(A_i - \bar{A}\right)^2} * \sqrt{\sum \left(B_i - \bar{B}\right)^2}} = \frac{\text{cov}(A,B)}{\sigma_A \sigma_B} \tag{2.2}$$

όπου $\text{cov}(A,B)$ είναι η συν-διακύμανση και σ_A, σ_B οι τυπικές αποκλίσεις των και αντίστοιχα. Παρατηρείται ότι για μηδενικές μέσες τιμές ο Pearson Correlation Coefficient ταυτίζεται με το Cosine Similarity.

2.6.3 Αρχή Λειτουργίας Κατακερματισμού

Ο κατακερματισμός αντικατοπτρίζει την διαδικασία που ακολουθείται με σκοπό την αντιστοίχηση μιας δεδομένης εισόδου σε μία μικρότερη και καθορισμένου μεγέθους μοναδική τιμή, κωδικό αριθμό ή κλειδί που την αντιπροσωπεύει κατάλληλα. Με τον τρόπο αυτό τα δεδομένα εισόδου ομαδοποιούνται σε buckets, με βάση αυτή τη μοναδική τιμή. Η δεδομένη είσοδος μπορεί να αναπαριστά μια σειρά χαρακτήρων ή το παράθυρο ενός data stream όπως απαιτεί το αντικείμενο της παρούσας διπλωματικής. Η αντιστοίχηση της εισόδου σε συγκεκριμένες τιμές κατακερματισμού (hash values) πραγματοποιείται μέσω μιας συνάρτησης η οποία ονομάζεται συνάρτηση κατακερματισμού (hash function). Ιδανικά οι τιμές αυτές θα πρέπει να είναι διαφορετικές για διαφορετική είσοδο, καθώς η κύρια χρησιμότητα αυτών των συναρτήσεων είναι η ταυτοποίηση των δεδομένων. Οι συναρτήσεις κατακερματισμού κυρίως χρησιμοποιούνται σε πίνακες κατακερματισμού (hash tables), για γρήγορη εύρεση εγγραφών σε βάσεις δεδομένων. Σε αρκετές περιπτώσεις οι συναρτήσεις κατακερματισμού μπορούν να παράξουν την ίδια μοναδική τιμή για διαφορετικές τιμές εισόδου. Τότε παρατηρείται το φαινόμενο της σύγκρουσης (collision). Για την αποφυγή των συγκρούσεων, στην συγκεκριμένη εργασία, χρησιμοποιείται η τεχνική του αλυσιδωτού κατακερματισμού κατά την οποία σε κάθε θέση του πίνακα

κατακερματισμού δημιουργείται μια συνδεδεμένη λίστα δεδομένων. Συνεπώς δύο διαφορετικές είσοδοι με κοινή τιμή κατακερματισμού αποθηκεύονται στην ίδια λίστα. Στην περίπτωση αυτή ο πίνακάς κατακερματισμού ορίζεται ως ευρετήριο κατακερματισμού (bucket indices). Η χρονική πολυπλοκότητα της αναζήτησης ενός στοιχείου σε μια τέτοια λίστα ισούται με τον μέσο αριθμό των στοιχείων που αποθηκεύονται σε αυτή. Η μέγιστη χρονική πολυπλοκότητα αντιστοιχεί στην χειρότερη περίπτωση κατά την οποία σε ένα bucket αποθηκεύονται όλες οι τιμές εισόδου.

2.6.4 Βασική Ιδέα και Αρχή Λειτουργίας RHP του LSH Αλγόριθμου

Η εύρεση της ομοιότητας μεταξύ δεδομένων αποτελεί θεμελιώδη λειτουργία πολλών κατανεμημένων εφαρμογών. Το Locality Sensitive Hashing αποτελεί μια οικογένεια αλγορίθμων, μετρικών αποστάσεων και συναρτήσεων κατακερματισμού, στοιχεία τα οποία συνεργάζονται μεταξύ τους ανάλογα με την περίπτωση χρήσης. Οι LSH συναρτήσεις είναι ειδικά σχεδιασμένες ώστε να υπάρχει μεγαλύτερη πιθανότητα για εμφάνιση collisions στα hash values δύο τιμών εισόδου οι οποίες έχουν έναν βαθμό ομοιότητας παρά για αρκετά διαφορετικές μεταξύ τους εισόδους δεδομένων. Υπάρχουν διάφορες μορφές - υλοποιήσεις του LSH αλγόριθμου, οι οποίες ποικίλλουν ανάλογα με το είδος των δεδομένων εισόδου και τις αποστάσεις μεταξύ αυτών. Στην συγκεκριμένη διπλωματική εργασία, στον υπολογισμό ομοιότητας μεταξύ των δεδομένων εισόδου χρησιμοποιείται μια συγκεκριμένη μορφή LSH, που ονομάζεται Random Hyperplane Projection (RHP). Το RHP αποτελεί μια τεχνική μείωσης των διαστάσεων των δεδομένων εισόδου, η οποία βασίζεται στο LSH και χρησιμοποιείται για την μετατροπή στοιχείων d-διαστάσεων σε πολύ μικρότερα bitmaps των n bits. Μέσω της τεχνικής αυτής επιτυγχάνεται η συμπίεση της πληροφορίας των δεδομένων εισόδου και ο εύκολος υπολογισμός χαρακτηριστικών αυτών με ταυτόχρονη μείωση του κόστους επεξεργασίας τους.

Η τεχνική του LSH είναι μια κατανομή πάνω σε μια οικογένεια συναρτήσεων κατακερματισμού (hash functions). Οι συναρτήσεις αυτές εφαρμόζονται πάνω σε μια συλλογή αντικειμένων, έτσι ώστε για δύο αντικείμενα της συλλογής να ισχύει η ακόλουθη σχέση:

$$\mathbf{Pr}_{h\in\mathcal{F}}[h(x) = h(y)] = \mathrm{sim}(x, y) \tag{2.3}$$

όπου η $sim(x, y)\epsilon[0, 1]$ αντιστοιχεί στην συνάρτηση ομοιότητας, η οποία εφαρμόζεται πάνω στην συλλογή των δεδομένων. Ειδικότερα, ο RHP του LSH αλγόριθμος που υλοποιείται στο εσωτερικό του SDE εφαρμόζεται σε καθορισμένου μεγέθους (W) παράθυρα των data stream εισόδου. Δεδομένου ότι σε ένα stream i αντιστοιχεί ένα w_i παράθυρο, ο αλγόριθμος παράγει bitmaps διάστασης d για το παράθυρο αυτό.

Για την δημιουργία ενός τέτοιου bitmap, ο RHP του LSH αλγόριθμος χρησιμοποιεί έναν πίνακα R. Κάθε στήλη του πίνακα R αντιστοιχεί σε ένα σφαιρικά συμμετρικό τυχαίο διάνυσμα $r_c, c\epsilon\{1, ..., d\}$ μοναδιαίου μήκους και διάστασης $|W|$. Ο R αποτελείται από d στήλες, όπου d είναι το επιθυμητό μέγεθος του bitmap. Για κάθε $r_c, c\epsilon\{1, ..., d\}$ υπολογίζεται το εσωτερικό γινόμενό του με το w_i, $w_i \cdot r_c$. Αν το αποτέλεσμα του πολλαπλασιασμού είναι θετικό, στην αντίστοιχη θέση του bitmap X_i εισάγεται ο αριθμός 1, ενώ στην αντίθετη περίπτωση ο αριθμός 0. Οι αριθμοί που θα εισαχθούν στο bitmap ουσιαστικά ορίζονται από μία συνάρτηση

κατακερματισμού, όπως περιγράφεται στη συνέχεια:

$$h_r(x) = \begin{cases} 1 & ,\text{if } r \cdot x \geq 0 \\ 0 & ,\text{if } r \cdot x < 0 \end{cases} \tag{2.4}$$

Παράγοντας τα bitmap με τον τρόπο που προαναφέρθηκε, στη συνέχεια χρησιμοποιούνται για τον υπολογισμό της ομοιότητας καθώς ισχύουν πιθανοτικά οι ακόλουθες σχέσεις:

$$\frac{D_h\left(X_i, X_j\right)}{d} = \frac{\theta\left(w_i, w_j\right)}{\pi} \Leftrightarrow \theta\left(w_i, w_j\right) = \frac{D_h\left(X_i, X_j\right)}{d}\pi \overset{0 \leq \theta(w_i, w_j) \leq \pi}{\Leftrightarrow} \tag{2.5}$$

$$\cos\left(\theta\left(w_i, w_j\right)\right) = \cos\left(\frac{D_h\left(X_i, X_j\right)}{d}\pi\right) \tag{2.6}$$

όπου η

$$D_h\left(X_i, X_j\right) = \sum_{k=1}^{d} \left|X_{ik} - X_{jk}\right| \tag{2.7}$$

αντιστοιχεί στην απόσταση Hamming των εμπλεκόμενων στην παραπάνω σχέση bitmap. Μέσω της σχέσης αυτής αποδεικνύεται η αναλογία μεταξύ της γωνίας δύο παραθύρων ενός stream και της απόστασης Hamming των RHP bitmap τους. Επιπλέον, παρέχεται η δυνατότητα υπολογισμού της ομοιότητας συνημιτόνου των αρχικών διανυσμάτων με βάση τα bitmap τους, καθώς και του Pearson Correlation Coefficient για μηδενικές μέσες τιμές όπως αναφέρθηκε προηγουμένως.

Συνεπώς, αρχικά, η ιδέα για τη μείωση της χρήσης μνήμης και την απλοποίηση του ελέγχου ομοιότητας με την απλή εκτίμηση απόστασης Hamming είναι η παραγωγή του bitmap κάθε stream. Χρησιμοποιώντας τα bitmap που έχουν προκύψει, ο αλγόριθμος υπολογίζει την απόσταση Hamming κάθε ζεύγους streaming παραθύρων και συγκρίνει την προκύπτουσα απόσταση με ένα όριο Hamming Distance T_h. Για να επιτευχθεί αυτό, ο χρήστης οφείλει να ορίσει ένα όριο ομοιότητας T το οποίο θα μετατραπεί σε γωνία T_{angle} και στη συνέχεια στο όριο Hamming Distance T_h, χρησιμοποιώντας την παρακάτω εξίσωση,

$$T = \cos\left(\frac{T_h}{d}\pi\right) \overset{0 \leq T_{angle} \leq \pi}{\Leftrightarrow} \quad T_{angle} = \arccos(T) = \frac{T_h}{d}\pi \Leftrightarrow T_h = \frac{T_{angle}}{\pi}d \tag{2.8}$$

Αν $D_h(X_i, X_j) > T_h$ τα αντίστοιχα δεδομένα εισόδου κατηγοριοποιούνται ως ανόμοια και την αντίθετη περίπτωση ως όμοια. Η πραγματική ομοιότητα εντοπίζεται μετέπειτα ελέγχοντας την ομοιότητα των πραγματικών διανυσμάτων για τα οποία το hamming weight μετά την εφαρμογή του LSH έχει τιμή κάτω από ένα όριο. Με σκοπό την παράλληλη εξαγωγή όμοιων ζευγαριών από τις διαθέσιμες μονάδες επεξεργασίας ακολουθείται η διαδικασία του κατακερματισμού των bitmap σε buckets ανάλογα με το hamming weight τους. Το hamming weight κάθε bitmap δίνεται από τον ακόλουθο τύπο, ο οποίος υπολογίζει το αθροιστικό σύνολο των bits:

$$W_h\left(X_i\right) = \sum_{k=1}^{d} \left|X_{ik}\right| \tag{2.9}$$

Παρατηρείται ότι $D_h(X_i, X_j) \geq |W_h(X_i) - W_h(X_j)|$ και έτσι αν $T_h \geq |W_h(X_i) - W_h(X_j)|$

(καθώς $D_h(X_i, X_j) > T_h$) , τα bitmaps X_i, X_j είναι πιθανότατα ανόμοια και μπορούν να γίνουν hashed σε διαφορετικά buckets με σκοπό να αποφευχθεί ο έλεγχος για ομοιότητα. Επιθυμώντας μέγιστο βαθμό παραλληλοποίησης των buckets όπου $B = d/T_h$, τα bitmaps γίνονται hash σε αυτούς σύμφωνα με την εξίσωση (2.10). Το κάτω όριο χρησιμοποιείται καθώς η αρχικοποίηση των bucket id εκκινείται από το μηδέν.

$$hash\_index = \left\lfloor \frac{W_h\left(X_i\right)}{\frac{d}{B}} \right\rfloor \tag{2.10}$$

Για να ελεγχθεί η ομοιότητα μεταξύ bitmaps των οποίων το hamming weight βρίσκεται πολύ κοντά στα όρια των hamming weight των buckets και ισχύει η σχέση $T_h \geq |W_h(X_i) - W_h(X_j)|$, το bitmap γίνεται επίσης hash σε γειτονικό bucket σύμφωνα με την ακόλουθη σχέση, με τη προϋπόθεση το αποτέλεσμα της εξίσωσης (2.11) είναι διαφορετικό του αποτελέσματος της σχέσης 2.10 :

$$hash\_index = \left\lfloor \frac{\max\left\{W_h\left(X_i\right) - T_h, 0\right\}}{\frac{d}{B}} \right\rfloor \tag{2.11}$$

Με σκοπό την αποφυγή διπλών υπολογισμών ομοιότητας, για bitmaps τα οποία είχαν αποθηκευτεί εξαρχής στο ίδιο bucket, ο έλεγχος της μεταξύ τους ομοιότητας πραγματοποιείται μόνο στο bucket αυτό.

Τέλος αξίζει να σημειωθεί ότι πρωταρχικός ρόλος της LSH σύνοψης που υλοποιείται στο εσωτερικό του SDE είναι ο υπολογισμός των bitmaps και ο κατακερματισμός αυτών σε bucket, σύμφωνα με παραμέτρους που ορίζονται από τον χρήστη. Ο έλεγχος ομοιότητας και οι συγκρίσεις μεταξύ των λιστών αποτελούν μέρος της διαδικασίας που ακολουθείται με σκοπό την απάντηση στα ερωτήματα (queries) που τίθενται από την πλευρά του χρήστη.

2.7 Πρόβλεψη Χρονοσειρών

Η πρόβλεψη χρονοσειρών [9] είναι ένας σημαντικός τομέας της μηχανικής μάθησης που συχνά παραμελείται. Η πρόβλεψη χρονοσειρών χρησιμοποιεί διαφορετικές τεχνολογίες όπως μηχανική εκμάθηση, τεχνητά νευρικά δίκτυα, support vector machines, ασαφή λογική, διαδικασίες Gauss και κρυφά μοντέλα Markov.

Μια χρονοσειρά είναι μια ακολουθία μετρήσεων που πραγματοποιούνται με την πάροδο του χρόνου, συνήθως λαμβάνονται σε ισόποσα διαστήματα, είτε είναι καθημερινά, μηνιαία, τριμηνιαία ή ετήσια. Η ανάλυση χρονοσειρών περιλαμβάνει μεθόδους για την ανάλυση δεδομένων χρονοσειρών με σκοπό την εξαγωγή σημαντικών στατιστικών και άλλων χαρακτηριστικών των δεδομένων. Η πρόβλεψη χρονοσειρών είναι η χρήση ενός μοντέλου για την πρόβλεψη μελλοντικών τιμών βάσει των τιμών που παρατηρήθηκαν προηγουμένως. Με άλλα λόγια, μια χρονοσειρά είναι μια ακολουθία σημείων δεδομένων που καταγράφονται σε συγκεκριμένους χρόνους.

Η πρόβλεψη είναι η διαδικασία δημιουργίας προβλέψεων για το μέλλον με βάση δεδομένα στο παρελθόν και το παρόν, καθώς και ανάλυση των τάσεων. Η πρόβλεψη περιλαμβάνει τη λήψη μοντέλων που ταιριάζουν σε ιστορικά δεδομένα και τη χρήση τους για την πρόβλεψη

μελλοντικών παρατηρήσεων.

Η χρονοσειρά μπορεί να οριστεί ως μια ταξινομημένη ακολουθία τιμών - παρατηρήσεων ενός μεγέθους με σταθερό δειγματοληπτικό χρόνο [10]. Το κίνητρο για τη μελέτη μοντέλων χρονοσειρών είναι διπλό:

1. Η κατανόηση των υποκείμενων δυνάμεων και της δομής που παρήγαγαν τα παρατηρούμενα δεδομένα.

2. Η διαμόρφωση και η εφαρμογή του μοντέλου με σκοπό την παρακολούθηση και την πρόβλεψη νέων τιμών.

Η ανάλυση χρονοσειρών μπορεί να χωριστεί σε δύο κύριες κατηγορίες ανάλογα με τον τύπο του μοντέλου που μπορεί να τοποθετηθεί. Οι δύο κατηγορίες είναι:

1. Κινητικό μοντέλο: Τα δεδομένα εδώ προσαρμόζονται ως $x_t = f(t)$. Οι μετρήσεις ή οι παρατηρήσεις θεωρούνται ως συνάρτηση του χρόνου.

2. Δυναμικό μοντέλο: Τα δεδομένα εδώ προσαρμόζονται ως $x_t = f(x_t{-}1, x_t{-}2, x_t{-}3 \ldots)$.

Υπάρχουν πολλές στατιστικές τεχνικές διαθέσιμες για την πρόβλεψη χρονοσειρών, με αρκετά αποτελεσματικές τις Simple Moving Average (SMA), Exponential Smoothing (SES), Autoregressive Intergration Moving Average και Γραμμική Παλινδρόμηση (Linear Regression). Στη συγκεκριμένη εργασία, με σκοπό την αποδοτική πρόβλεψη των μελλοντικών τιμών χρησιμοποιείται το μοντέλο Linear Regression.

2.7.1 Μοντέλο Γραμμικής Παλινδρόμησης

Η γραμμική παλινδρόμηση [11] είναι ένα στατιστικό εργαλείο που χρησιμοποιείται για την πρόβλεψη μελλοντικών τιμών από προηγούμενες τιμές. Συνήθως χρησιμοποιείται ως ποσοτικός τρόπος για να προσδιοριστεί η υποκείμενη τάση. Η γραμμική παλινδρόμηση χρησιμοποιεί τη μέθοδο των ελαχίστων τετραγώνων για να σχεδιάσει μια ευθεία γραμμή μέσω των τιμών, έτσι ώστε να ελαχιστοποιηθούν οι αποστάσεις μεταξύ των τιμών και της προκύπτουσας γραμμής τάσης.

Η ανάλυση παλινδρόμησης (regression analysis) είναι ένα σύνολο στατιστικών διαδικασιών που έχουν ως στόχο την εκτίμηση και αξιολόγηση της σχέσης μεταξύ μεταβλητών σε ένα σύνολο δεδομένων. Ειδικότερα, η ανάλυση παλινδρόμησης έχει ως στόχο τον προσδιορισμό της σχέσης μεταξύ μίας μεταβλητής που απαραιτήτως πρέπει να λαμβάνει συνεχείς αριθμητικές τιμές και καλείται εξαρτημένη μεταβλητή μίας ή περισσοτέρων άλλων μεταβλητών, που καλούνται ανεξάρτητες μεταβλητές και οι οποίες μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Η εκτίμηση της σχέσης των μεταβλητών αυτών αποσκοπεί στο να μελετήσει εάν και πόσο οι τιμές της εξαρτημένης μεταβλητής επηρεάζονται από τις τιμές των ανεξάρτητων μεταβλητών με τρόπο που να είναι σύμφωνες με τις παρατηρούμενες - στον πραγματικό κόσμο - τιμές των μεταβλητών αυτών. Για αυτό η ανάλυση παλινδρόμησης απαιτεί να υπάρχουν διαθέσιμες πραγματικές, παρατηρούμενες τιμές των μεταβλητών αυτών, οι οποίες θα αναλυθούν και από όπου θα εξαχθεί η σχέση των υπό εξέταση μεταβλητών. Στα πλαίσια της βιβλιογραφίας οι μεταβλητές μιας παλινδρόμησης μπορεί να εμφανίζονται και με διαφορετική ορολογία: η εξαρτημένη

μεταβλητή μπορεί να βρεθεί επίσης με τους όρους αποκριτική (responsive), παρατηρηθείσα ή μεταβλητή εξόδου, ενώ οι ανεξάρτητες μεταβλητές μπορούν να εμφανιστούν με τους όρους προβλέπουσες, παλινδρομούσες, επεξηγηματικές ή μεταβλητές εισόδου. Ο στόχος της παλινδρόμησης είναι τόσο να περιγράψει και να εξηγήσει τη σχέση των τιμών μεταβλητών αυτών, και δη της εξαρτημένης και των ανεξάρτητων μεταβλητών, όσο και για να προβλέψει τις τιμές της εξαρτημένης μεταβλητής βάσει των τιμών των ανεξαρτήτων μεταβλητών. Η σχέση της εξαρτημένης και των ανεξάρτητων μεταβλητών συλλαμβάνεται με τη μορφή εξίσωσης μεταξύ των μεταβλητών αυτών, που καλείται εξίσωση παλινδρόμησης ή μοντέλο παλινδρόμησης. Η σχέση που αναζητείται μεταξύ των μεταβλητών αυτών, συλλαμβάνεται από αυτήν ακριβώς την εξίσωση και κατά συνέπεια στόχος της παλινδρόμησης είναι η αναζήτηση της εξίσωσης εκείνης που καλύτερα απ'όλες συλλαμβάνει την πραγματική σχέση των μεταβλητών αυτών. Η εξίσωση αυτή μπορεί να λάβει οποιαδήποτε αλγεβρική μορφή ωστόσο, δεν πρέπει να νοηθεί ως ντετερμινιστική συνάρτηση με την αυστηρή μαθηματική έννοια, αλλά ως μία στατιστική σχέση μεταξύ της εξαρτημένης και των ανεξάρτητων μεταβλητών. Αυτό σημαίνει για παράδειγμα ότι η εφαρμογή και η ερμηνεία των αποτελεσμάτων ενός μοντέλου παλινδρόμησης απαιτεί να γίνεται με πολύ συγκεκριμένο τρόπο. Στην εξίσωση παλινδρόμησης, συνηθίζεται η εξαρτημένη μεταβλητή να εμφανίζεται στο αριστερό σκέλος της ενώ οι ανεξάρτητες μεταβλητές εμφανίζονται στο δεξί. Παράδειγμα μιας εξίσωσης γραμμικής παλινδρόμησης είναι το ακόλουθο: $Y = aX + b$, όπου X είναι η ανεξάρτητη μεταβλητή, Υ η εξαρτημένη και τα a (κλίση) και b (τιμή Υ για $a = 0$) υπολογίζονται από τις εξισώσεις [12]

$$a = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

Στα μοντέλα γραμμικής παλινδρόμησης που χρησιμοποιούνται στην πρόβλεψη τιμών οι εξαρτημένες και οι ανεξάρτητες μεταβλητές είναι εξ'αρχής γνωστές ενώ οι τιμές των μεταβλητών είναι οι άγνωστες.

Κεφάλαιο 3

Σχετικές Εργασίες

Στο κεφάλαιο αυτό αναφέρονται περιληπτικά σημαντικές έρευνες και εργασίες που πραγματεύονται κύρια συνθετικά στοιχεία αυτής της διπλωματικής εργασίας, όπως ο αλγόριθμος Random Hyperplane Projection Locality Sensitive Hashing και οι παράλληλες καρκινικές προσομοιώσεις

3.1 Υπολογισμός Ακραίων Τιμών με τη βοήθεια του LSH

Τα ασύρματα δίκτυα αισθητήρων γίνονται ολοένα και πιο δημοφιλή για μια ποικιλία εφαρμογών. Εν τούτις, οι χρήστες εντοπίζουν, κατά την ανάγνωση των δεδομένων τους, ακραίες τιμές, με αποτέλεσμα την μη έγκυρη και αναξιόπιστη εξαγωγή συμπερασμάτων από αυτά. Στο ερευνητικό άρθρο με τίτλο "In-Network Approximate Computation of Outliers with Quality Guarantees" [13] αναπτύσσεται ένα framework εντοπισμού ακραίων τιμών που βασίζεται στον αλγόριθμο Locality Sensitive Hashing. Το σύστημα που αναπτύσσεται θυσιάζει το εύρος ζωής για την ακρίβεια, ενώ ταυτόχρονα υποστηρίζει πληθώρα μετρικών ομοιότητας. Η σχεδίαση του συστήματος βασίζεται σε τρία πολύ σημαντικά βήματα:

1. **Κωδικοποίηση και Μείωση των Δεδομένων**. Οι κόμβοι του αισθητήρα κωδικοποιούν τις τελευταίες W μετρήσεις χρησιμοποιώντας ένα bitmap d ψηφίων με τη μέθοδο Random Hyperplane Projection

2. **Ανίχνευση Ακραίων Τιμών σε επίπεδο Cluster**. Κάθε cluster λαμβάνει στο εσωτερικό του κωδικοποιημένες μετρήσεις των αισθητήρων. Εν συνεχεία, εκτελεί ελέγχους ομοιότητας μεταξύ όλων των ζευγών κόμβων των αισθητήρων, με σκοπό τον εντοπισμό εκείνων που δεν μπορούν να φτάσουν το επιθυμητό επίπεδο support και χαρακτηρίζονται αυτόματα ως outliers

3. **Επικοινωνία μεταξύ των cluster**. Μετά την επεξεργασία των κωδικοποιημένων μετρήσεων εντός του cluster, ο κάθε cluster έχει ορίσει ένα σύνολο πιθανών ακραίων τιμών, μαζί με το support για κάθε ένα από αυτά. Μερικοί από τους outliers είναι πιθανό να λαμβάνουν support από κόμβους αισθητήρων που ανήκουν σε άλλους clusters. Με αυτόν τον τρόπο εκκινείται η επικοινωνία, όπου πιθανοί outliers ενός cluster επικοινωνούν με άλλους clusters προκειμένου να αυξήσουν το support τους.

Τέλος, η πειραματική διαδικασία αυτής της ερευνητικής εργασίας αποδεικνύει πως το σύστημα που έχει διαμορφωθεί είναι κατάλληλο για τον αξιόπιστο εντοπισμό των ακραίων τιμών.

3.2 Πρόβλεψη Ελπιδοφόρων Βιολογικών Προσομοιώσεων στο εργαλείο PhysiBoSS

Η ανάγκη για κατανόηση και θεραπεία ασθενειών με μη φυσιολογική και απρόβλεπτη απόκριση, οδήγησε στη δημιουργία διαφορετικών εργαλείων μοντελοποίησης, τα οποία συνυπολογίζουν το ενδο- και εξώ-κυτταρικό περιβάλλον, καθώς και την αλληλεπίδραση μεταξύ των κυττάρων. Ένα τέτοιο εργαλείο είναι το PhysiBoSS, το οποίο προσομοιώνει την εξέλιξη ενός πλήθους κυττάρων στο πέρασμα του χρόνου και κάτω από ορισμένες συνθήκες. Λαμβάνοντας υπόψιν το γεγονός ότι δεν είναι ελπιδοφόρες όλες οι προσομοιώσεις του εργαλείου, προκειμένου να διευκολυνθεί η διαδικασία της διαλογής των καλύτερων προσομοιώσεων και η μελέτη των αποτελεσμάτων, οι κακές προσομοιώσεις πρέπει να εξαιρεθούν. Την επίλυση του προβλήματος αυτού αναλαμβάνει η διπλωματική εργασία με τίτλο 'Forecasting Promising Biological Simulations at PhysiBoSS' [14]. Στην εργασία αυτή σχεδιάζεται και αναπτύσσεται ένα παράλληλο και κατανεμημένο σύστημα, το οποίο εφαρμόζει έναν αλγόριθμο πρόβλεψης σε ένα πλήθος τρεχουσών προσομοιώσεων και αποφασίζει για τη συνέχιση ή όχι της εκτέλεσής της κάθε μίας και τέλος ανιχνεύει και κρατά μόνο τις k πιο ελπιδοφόρες προσομοιώσεις εκ του ομαδοποιημένου συνόλου προσομοιώσεων. Το dataset που παράγεται από το εργαλείο PhysiBoSS και χρησιμοποιείται στην παραπάνω εργασία, αποτελεί είσοδο στο σύστημα που σχεδιάζεται στην τρέχουσα εργασία. Παρ'όλα αυτά για την τρέχουσα εργασία χρήσιμη πληροφορία αποτελεί όλο το σύνολο των προσομοιώσεων και όχι μόνο οι ελπιδοφόρες καθώς σκοπός είναι η πρόβλεψη της συμπεριφοράς τους και όχι η διακοπή εκείνων που έχουν αρνητική έκβαση.

3.3 Αναλυτική Επεξεργασία Χρονοσειρών

Στην μεταπτυχιακή εργασία με τίτλο 'Scaling out streaming time series analytics on Storm' [15] αναπτύσσεται ένα framework ονομαζόμενο ως T-Storm. Η υλοποίηση του συστήματος αυτού δίνει στον χρήστη την δυνατότητα της εύκολης αναλυτικής επεξεργασίας χρονοσειρών πάνω σε ροές δεδομένων. Επιπροσθέτως, μέσω της εργασίας αυτής αντιμετωπίζεται το πρόβλημα της αποτελεσματικής κλιμάκωσης του προβλήματος της παρακολούθησης εξαρτήσεων μεταξύ ζευγών χιλιάδων εισερχόμενων ροών δεδομένων σε πραγματικό χρόνο. Αρκετά components με εγγενώς διαφορετικά χαρακτηριστικά αναπτύσσονται για να παρέχουν ένα ευρύ φάσμα αντισταθμίσεων μεταξύ υπολογισμού σε πραγματικό ή σε σχεδόν πραγματικό χρόνο, ακρίβειας, καθυστέρησης, κ.λ.π.

Πιο συγκεκριμένα και συγκριτικά με την τρέχουσα διπλωματική εργασία, ακολουθείται η τεχνική του sliding window με σκοπό την μείωση της πληροφορίας και τη διατήρηση της χρήσιμης σύμφωνα με τις ανάγκες του χρήστη. Όπως στη τρέχουσα διπλωματική εργασία, έτσι και σε αυτή χρησιμοποιείται ο συντελεστής του Pearson correlation ως μια πολύ σημαντική μετρική που αντικατοπτρίζει την γραμμική εξάρτηση μεταξύ δύο χρονοσειρών. Η συσχέτιση

μεταξύ δύο χρονοσειρών υπολογίζεται και μέσω της Ευκλείδειας απόστασης γεγονός που οδηγεί στην χρήση του Discrete Fourier Transform για την αντιμετώπιση του προβλήματος εύρεσης ομοιότητας μεταξύ των δεδομένων.

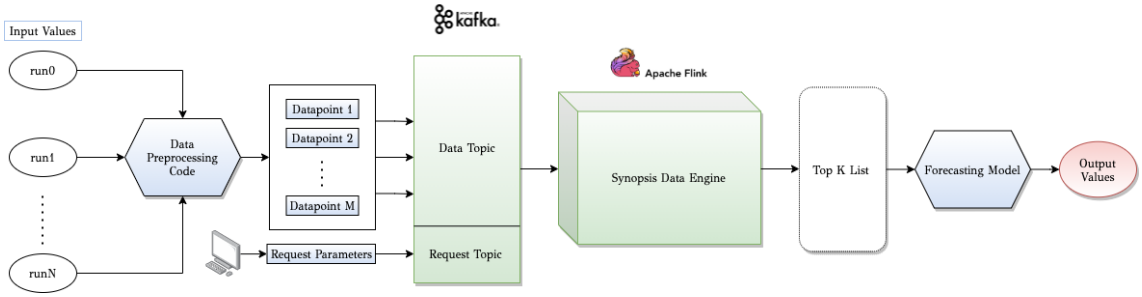
Κεφάλαιο 4

Σχεδίαση και Υλοποίηση Συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη που έγινε για την υλοποίηση του συστήματος. Αρχικά περιγράφεται η αρχιτεκτονική του συστήματος και ακολουθεί η λεπτομερής ανάλυση της σχεδίασης του.

4.1 Περιγραφή της Αρχιτεκτονικής

Στην Εικόνα 4.1 αποτυπώνεται η περιληπτική περιγραφή της σχεδίασης του συστήματος. Ένα σύνολο παράλληλων καρκινικών προσομοιώσεων εισάγεται στο σύστημα με σκοπό την εύρεση όμοιων στοιχείων και την εφαρμογή ενός αλγορίθμου πρόβλεψης πάνω σε αυτά. Τα δεδομένα εισόδου πριν εισαχθούν και εγγραφούν στα partitions του topic εισόδου του Apache Kafka υφίστανται επεξεργασία με σκοπό την απομόνωση των προς μελέτη features. Εν συνεχεία εισάγονται στο κατανεμημένο σύστημα του Apache Kafka απ᾽όπου και αντλούνται από την μηχανή συνόψεων δεδομένων Synopsis Data Engine. Στο εσωτερικό του SDE δημιουργείται η κατάλληλη σύνοψη στην οποία προστίθενται τα δεδομένα και παράγεται η απαιτούμενη έξοδος έπειτα από την εφαρμογή του αλγόριθμου Random Hyperplane Projection Locality Sensitive Hashing. Τελευταίο στάδιο αποτελεί η εφαρμογή ενός αλγορίθμου πρόβλεψης στα δεδομένα εξόδου του SDE με σκοπό την πρόβλεψη της μελλοντικής συμπεριφοράς και πορείας των ζητούμενων καρκινικών προσομοιώσεων.



Εικόνα 4.1: *Η Αρχιτεκτονική του Συστήματος*

4.2 Υλοποίηση - Ανάλυση Σχεδίασης

4.2.1 Δεδομένα

Τα δεδομένα εισόδου του συστήματος αντιστοιχούν σε παράλληλες προσομοιώσεις καρκινικών κυττάρων, οι οποίες με την σειρά του έχουν παραχθεί από το εργαλείο PhysiBoSS [14].

Το PhysiBoSS αναπαριστά ένα C++ software framework το οποίο υλοποιήθηκε το 2019. Πρόκειται για ένα εργαλείο μοντελοποίησης βασισμένο σε πράκτορες που χρησιμοποιείται για multiscale προσομοιώσεις ετερογενών πολυκυτταρικών συστημάτων. Πιο συγκεκριμένα, δημιουργεί μια εικονική αναπαράσταση ενός πολυκυτταρικού συστήματος, δηλαδή, τα κύτταρα, το ενδοκυτταρικό και εξωκυτταρικό περιβάλλον τους, καθώς και την αλληλεπίδραση μεταξύ των κυττάρων και του περιβάλλοντός τους. Αναπαριστά ένα μαθηματικό μοντέλο που βασίζεται σε πράκτορες και κάθε πράκτορας αντιπροσωπεύει ένα ανεξάρτητο κύτταρο.

Για κάθε προσομοίωση που παράγεται από το PhysiBoSS, δημιουργούνται δύο φάκελοι, οι οποίοι περιέχουν δεδομένα της προσομοίωσης αυτής: ο microutput και ο output. Η χρήσιμη πληροφορία για τη λειτουργικότητα του συστήματος βρίσκεται αποθηκευμένη στον φάκελο output. Στο εσωτερικό του φακέλου αυτού βρίσκεται ένα πλήθος .txt αρχείων, κάθε ένα από τα οποία περιέχει πληροφορίες κυττάρων για μια συγκεκριμένη χρονική στιγμή, όπως φαίνεται στην Εικόνα 4.2. Κάθε προσομοίωση έχει διάρκεια ίση με 1 μέρα (1440 λεπτά) και κάθε .txt αρχείο παράγεται κάθε X λεπτά, όπου X είναι τιμή της παραμέτρου της προσομοίωσης. Συνεπώς, έστω ότι η παράμετρος X παίρνει την τιμή 40, κάθε προσομοίωση περιέχει 37 (1440/40+1) τέτοια αρχεία. Η χρονική στιγμή στην οποία έχει παραχθεί το κάθε αρχείο αποτυπώνεται στην παράμετρο Time που βρίσκεται στο εσωτερικό του. Συνεπώς, η παράμετρος Time παίρνει τις τιμές 0, 40, 80,...,1440, μία για κάθε διαφορετικό αρχείο.

```
Time;ID;x;y;z;radius;volume_total;radius_nuclear;contact_ECM;freezer;polarized_fraction;motility;cell_line;Cell_cell;phase;Cycle;NFkB
40;0;-50.0581;-5.24689;-85.4845;8.49311;2566.19;5.10579;0;0;0.1;0.01;0;2.51833;1;0;-1-1;-1
40;1;-36.603;-29.9328;-85.7672;9.92229;4091.9;5.98042;0;0;0.1;0.01;0;3.30799;0;0;-1-1;-1
40;2;-37.8638;-13.7627;-88.6613;8.49322;2566.29;5.1056;0;0;0.1;0.01;0;2.47071;1;0;-1-1;-1
40;3;-36.0509;2.25819;-83.5151;9.05115;3105.99;5.47036;0;0;0.1;0.01;0;4.92856;0;0;-1-1;-1
40;4;-34.396;19.368;-83.3008;10.0693;4276.54;6.06596;0;0;0.1;0.01;0;3.66997;0;0;-1-1;-1
40;5;-21.7338;-38.6656;-87.4388;9.39492;3473.5;5.67391;0;0;0.1;0.01;0;3.29855;0;0;-1-1;-1
40;6;-20.092;-23.1225;-84.2939;8.85119;2904.65;5.36891;0;0;0.1;0.01;0;4.89469;0;0;-1-1;-1
40;7;-19.5509;-7.31756;-83.3195;8.81621;2870.35;5.3602;0;0;0.1;0.01;0;3.88323;0;0;-1-1;-1
40;8;-20.0362;11.233;-84.0795;9.05854;3113.6;5.48156;0;0;0.1;0.01;0;4.50817;0;0;-1-1;-1
40;9;-20.1346;28.3972;-83.3754;9.2113;3273.79;5.56813;0;0;0.1;0.01;0;3.31245;0;0;-1-1;-1
40;10;-3.5376;-48.7779;-80.315;8.24607;2348.71;4.9594;0;0;0.1;0.01;0;3.92956;1;0;-1-1;-1
40;11;-6.70972;-31.5946;-86.1912;9.34368;3416.98;5.6486;0;0;0.1;0.01;0;4.26586;0;0;-1-1;-1
40;12;-6.55822;-15.3245;-85.3221;9.04019;3094.72;5.46941;0;0;0.1;0.01;0;4.92909;0;0;-1-1;-1
40;13;-5.84537;2.15232;-86.2118;10.2139;4463.42;6.14971;0;0;0.1;0.01;0;4.4271;0;0;-1-1;-1
40;14;-5.58409;19.7211;-83.2698;10.2226;4474.81;6.15446;0;0;0.1;0.01;0;5.53403;0;0;-1-1;-1
40;15;-4.64369;37.2046;-84.4707;9.76542;3900.87;5.89179;0;0;0.1;0.01;0;3.669;0;0;-1-1;-1
40;16;8.12299;-39.9385;-85.2696;10.1182;4339.09;6.09463;0;0;0.1;0.01;0;4.46202;0;0;-1-1;-1
40;17;8.34403;-22.3211;-83.787;10.1249;4347.72;6.09655;0;0;0.1;0.01;0;5.96857;0;0;-1-1;-1
40;18;11.9158;-6.34677;-85.8889;8.89288;2945.89;5.39227;0;0;0.1;0.01;0;3.60341;0;0;-1-1;-1
40;19;9.26198;11.9404;-83.6847;8.86617;2919.42;5.37045;0;0;0.1;0.01;0;4.69285;0;0;-1-1;-1
40;20;9.64773;28.3735;-84.5454;8.96193;3015.04;5.42446;0;0;0.1;0.01;0;4.95787;0;0;-1-1;-1
40;21;12.8054;41.9597;-92.7988;8.18248;2294.79;4.92259;0;0;0.1;0.01;0;2.12738;1;0;-1-1;-1
40;22;22.8628;-48.8821;-82.225;9.52215;3616.54;5.74771;0;0;0.1;0.01;0;3.77823;0;0;-1-1;-1
40;23;23.3082;-32.0555;-86.8864;10.1203;4341.75;6.09444;0;0;0.1;0.01;0;4.81641;0;36;-1-1;-1
40;24;23.6903;-16.2135;-83.2093;8.61193;2675.41;5.21539;0;0;0.1;0.01;0;5.28427;0;0;-1-1;-1
40;25;22.9037;4.28461;-85.026;8.72923;2786.23;5.2913;0;0;0.1;0.01;0;4.58364;0;0;-1-1;-1
40;26;23.4095;19.7652;-82.8009;8.99948;3053.09;5.45034;0;0;0.1;0.01;0;5.52736;0;0;-1-1;-1
40;27;23.8878;37.0373;-83.0438;9.89135;4053.73;5.96275;0;0;0.1;0.01;0;4.91256;0;20;-1-1;-1
40;28;38.2075;-40.6001;-83.4443;10.0303;4226.95;6.04419;0;0;0.1;0.01;0;3.60038;0;0;-1-1;-1
40;29;37.0565;-23.5178;-83.289;8.49322;2566.29;5.1055;0;0;0.1;0.01;0;4.34997;1;0;-1-1;-1
40;30;36.6854;-6.33079;-84.4758;10.227;4480.63;6.1565;0;0;0.1;0.01;0;3.3931;0;0;-1-1;-1
40;31;38.1843;10.449;-84.5592;10.2482;4508.51;6.17031;0;0;0.1;0.01;0;3.70718;0;0;-1-1;-1
40;32;38.3295;28.215;-86.1019;10.1158;4336.03;6.09429;0;0;0.1;0.01;0;3.66415;0;0;-1-1;-1
```

Εικόνα 4.2: *Περιεχόμενο αρχείου cell_X*

Χρήσιμη Πληροφορία

Στο εσωτερικό κάθε αρχείου cell_X (Εικόνα 4.2) υπάρχει ένα πλήθος παραμέτρων, κάθε μία από τις οποίες παίρνει διαφορετικές τιμές (νοητές στήλες) ανάλογα με το ID του τρέχοντος κυττάρου. Για το σύστημα, η χρήσιμη πληροφορία αποτυπώνεται στην παράμετρο phase. Η

παράμετρος αυτή περιέχει την πληροφορία σχετικά με την κατάσταση του κάθε πράκτορα/κυττάρου για συγκεκριμένη χρονική στιγμή. Η phase αναπαρίσταται με μια ακέραια κωδική τιμή, η οποία με τη σειρά της αντιστοιχεί σε διαφορετική κατάσταση. Στον Πίνακα που ακολουθεί 4.1, γίνεται η αντιστοίχηση κάθε κωδικού αριθμού στην κατάλληλη κατηγορία.

Πίνακας 4.1: *Κατηγορίες Καρκινικών Κυττάρων*

Category	Alive	Apoptotic	Necrotic
Phase Code	0/1	100	101/102/103

4.2.2 Εισαγωγή Δεδομένων στο Apache Kafka

Τα δεδομένα που πρόκειται να εγγραφούν στο Apache Kafka και αργότερα να αποτελέσουν είσοδο στο SDE διαμορφώνονται κατάλληλα έτσι ώστε η πληροφορία που εμπεριέχεται σε κάθε ένα από αυτά να αποτυπώνει το *σύνολο* των alive, των apoptotic και των necrotic κυττάρων που αντιστοιχούν σε κάθε χρονική στιγμή μια προσομοίωσης. Αναλυτικότερα, διαμορφώνεται ένα JSON String στο οποίο αναγράφονται πληροφορίες που αφορούν το σύνολο των alive, apoptotic και necrotic κυττάρων καθώς και την χρονική στιγμή της εκάστοτε προσομοίωσης μαζί με τον κωδικό αριθμό αυτής για το τρέχον μήνυμα. Εν συνεχεία το Json String εισάγεται ως παράμετρος στο πεδίο values της Java κλάσης Datapoint μαζί με το simulation id και το Dataset key. Επεξηγηματικά, η λειτουργικότητα της κλάσης αυτής αφορά στην μετατροπή των JSON Strings σε streams καθώς κάνει extend το Serializable Interface. Στην ακόλουθη Εικόνα 4.3 αποτυπώνεται η μορφή ενός μηνύματος όπως αυτό εισάγεται στο Kafka Data topic εισόδου:

```
{
  "dataSetkey" : "bio_useCase",
  "streamID" : "run7",
  "values" : {
    "Time" : "1400",
    "data" : "94,11,14",
    "simulationId" : "run7"
  }
}
```

Εικόνα 4.3: *Μορφή Μηνύματος Δεδομένου Εισόδου*

Εκτός από τα δεδομένα των προσομοιώσεων, στο Kafka εισάγονται και οι πληροφορίες-παράμετροι του Request προς την υλοποιημένη σύνοψη. Τα ερωτήματα προς την σύνοψη είναι δύο ειδών: add request (Εικόνα 4.4) και estimate request (Εικόνα 4.5). Το add request ενεργοποιεί την λειτουργικότητα της σύνοψης, η οποία αφορά στην εισαγωγή των δεδομένων από το Kafka Data topic στην σύνοψη και το δεύτερο, estimate request, ενεργοποιεί την διαδικασία κατά την οποία τα δεδομένα κατακερματίζονται στα κατάλληλα buckets ανάλογα με συγκεκριμένα χαρακτηριστικά. Οι λειτουργικότητες που αναφέρθηκαν περιγράφονται λεπτομερώς στην ακόλουθη υποενότητα.

```
{
  "param" : [ "simulationId", "data", "Time", "40", "10" ],
  "uid" : 1110,
  "requestID" : 1,
  "dataSetkey" : "bio_useCase",
  "streamID" : "INTEL",
  "synopsisID" : 28,
  "key" : "bio_useCase",
  "noOfP" : 2
}
```


Εικόνα 4.4: *Μορφή Μηνύματος Add Request*


```
{
  "param" : [ "0.9", "15", "3", "run13_50&run18_75", "50" ],
  "uid" : 1110,
  "requestID" : 3,
  "dataSetkey" : "bio_useCase",
  "streamID" : "INTEL",
  "synopsisID" : 28,
  "key" : "bio_useCase",
  "noOfP" : 2
}
```


Εικόνα 4.5: *Μορφή Μηνύματος Estimate Request*

4.2.3 Λειτουργικότητα Σύνοψης

Έπειτα από την εισαγωγή των δεδομένων, ως streams, στα αντίστοιχα Kafka Topics, το SDE αναλαμβάνει το διάβασμα των δεδομένων αυτών. Πιο συγκεκριμένα, στην περίπτωση του Data Topic ένα ειδικό parser component αναλαμβάνει την εξαγωγή των τιμών κάθε πεδίου σύμφωνα με τις οποίες δημιουργείται ή ενημερώνεται η κατάλληλη σύνοψη. Με την ίδια διαδικασία εξάγονται και οι πληροφορίες του Request Topic.

Add Request

Η διαδικασία με την οποία δημιουργείται μια νέα σύνοψη και ενημερώνεται με νέα δεδομένα, ενεργοποιείται με το διάβασμα του add request ($requestID = 1$) που βρίσκεται στο αντίστοιχο Request Topic. Μέσω του πεδίου *synopsisID*, γνωστοποιείται στο SDE το είδος της σύνοψης που πρέπει να δημιουργηθεί. Σημαντική παράμετρος για την λειτουργία της σύνοψης είναι το μέγεθος W (πεδίο *param*,"40") του παραθύρου δεδομένων που οφείλει να αποθηκεύει, το μέγεθος συμπίεσης d (πεδίο *param*, "10") των δεδομένων καθώς και το μέγεθος του παραλληλισμού του συστήματος (πεδίο *noOfP*).

Του διαβάσματος αυτού έπεται η δημιουργία ενός νέου αντικειμένου της κλάσης στην οποία συγκεντρώνεται η λειτουργικότητα της σύνοψης (LSHsynopsis class) η οποία με τη σειρά της κάνει extend την abstract κλάση του SDE, Synopsis Εικόνα 4.6. Η κλάση Synopsis περιλαμβάνει μεθόδους για προσθήκη ενός νέου δεδομένου στην σύνοψη (add), για εξαγωγή μιας ζητούμενης εκτίμησης (estimate) και μια μέθοδο συγχώνευσης για τον συνδυασμό συνόψεων. Η συγκεκριμένη διπλωματική εργασία επικεντρώνεται στις δύο πρώτες μεθόδους add και estimate.

```
abstract public class Synopsis {

    protected int SynopsisID;
    protected String keyIndex;
    protected String valueIndex;
    protected String operationMode;

    public abstract void add(Object k);
    public abstract Object estimate(Object k);
    public abstract Estimation estimate(Request rq);
    public abstract Synopsis merge(Synopsis sk);
```

Εικόνα 4.6: *Abstract Synopsis Class*

Η συνάρτηση add της LSHsynopsis δέχεται ως ορίσματα το όνομα της προσομοίωσης, την χρονική στιγμή κατά την οποία εκκινείται το νέο παράθυρο συγκεκριμένου μεγέθους και slide, καθώς και τον αριθμό των alive, apoptotic και necrotic κυττάρων που αντιστοιχούν στη στιγμή αυτή. Κατά την εκτέλεση της συνάρτησης add τα δεδομένα εισόδου μετατρέπονται σε lsh vectors με την διαδικασία (Εικόνα 4.10) που ακολουθεί:

1. Κατά τη δημιουργία ενός νέου αντικειμένου της κλάσης LSHsynopsis (Εικόνα 4.7), αρχικοποιούνται δύο βασικές δομές με τη μορφή Hash Maps (Εικόνα 4.9), ονομαζόμενες ως ExternalHashMap και StoredTimeStampsPerSim. Ρόλος της πρώτης δομής είναι η αποθήκευση των παραγόμενων παραθύρων (LSH objects) για κάθε χρονική στιγμή της κάθε προσομοίωσης, δεδομένου ενός slide s, ενώ η δεύτερη διατηρεί το σύνολο όλων των διαθέσιμων τιμών εισόδου για κάθε χρονική στιγμή της κάθε προσομοίωσης. Κάθε μία από τις παραπάνω δομές διατηρεί στο εσωτερικό της ένα μικρότερο Hash Map ως value, ένα για κάθε key (προσομοίωση) που εισάγεται στις δομές αυτές. Τα μικρότερα Hash Map ονομάζονται InternalHashMap και StoredTimeStamps, αντίστοιχα. Επίσης, στο στάδιο αυτό δημιουργείται και αρχικοποιείται ο πίνακας γκαουσιανής κατανομής R, ο οποίος πρόκειται να χρησιμοποιηθεί σε επόμενο στάδιο με σκοπό την συμπίεση της πληροφορίας των αρχικών παραθύρων. Της δημιουργίας των component αυτών έπεται η χρήση τους στο εσωτερικό της συνάρτησης add.

2. Κατά την έναρξη της εκτέλεσης της συνάρτησης add πραγματοποιείται η διαδικασία κατά την οποία ελέγχεται η ύπαρξη της προσομοίωσης στο εσωτερικό του StoredTimeStampsPerSim Hash Map και του ExternalHashMap. Στην περίπτωση που αυτό δεν συμβαίνει, δημιουργούνται τα μικρότερα Hash Maps στο εσωτερικό των δομών αυτών ώστε να αποθηκεύσουν τις νέες τιμές που αντιστοιχούν στην προσομοίωση αυτή. Στην αντίθετη περίπτωση, η παρουσία της στο εσωτερικό των δομών αυτών, και συγκεκριμένα στο ExternalHashMap υποδηλώνει την ύπαρξη παραθύρων για χρονικές στιγμές που αντιστοιχούν στην προσομοίωση αυτή, και βρίσκονται αποθηκευμένα στο InternalHashMap της.

3. Στην περίπτωση, λοιπόν, που το InternalHashMap είναι κενό, δημιουργείται και αρχικοποιείται με το πρώτο ζευγάρι (key, value) pair δεδομένων. Το key αντιστοιχεί στην

χρονική στιγμή που δέχεται η συνάρτηση add ως input και το value στο αντικείμενο της κλάσης LSH (Εικόνα 4.8), η οποία περιέχει την λίστα των τιμών του παραθύρου μήκους W, στην οποία εισάγεται η πρώτη τριάδα τιμών. Στην αντίθετη περίπτωση η δεδομένη τριάδα τιμών, αφού αποθηκευτεί στην δομή StoredTimeStamps χρησιμοποιείται στην παραγωγή των ενδιάμεσων τριάδων τιμών μεταξύ αυτής και της αμέσως προηγούμενης της, μέσω της εφαρμογής της μεθόδου της γραμμικής παλινδρόμησης. Κατά τη διάρκεια της γραμμικής παλινδρόμησης, κάθε νέα τριάδα τιμών που παράγεται αποθηκεύεται και εκείνη με τη σειρά της στην δομή StoredTimeStamps της τρέχουσας προσομοίωσης.

4. Αφού αποθηκευτούν στην δομή StoredTimeStamps όλες οι τριάδες τιμών που αντιστοιχούν σε κάθε χρονική στιγμή πριν τη χρονική στιγμή των δεδομένων εισόδου, η συνάρτηση add διατρέχει μία προς μία τις χρονικές αυτές στιγμές μαζί με τις τριάδες τιμών που τους αντιστοιχούν. Σε κάθε επανάληψη ελέγχεται το αν η τρέχουσα αρχή του παραθύρου εξυπηρετεί το δεδομένο slide. Στην περίπτωση που το εξυπηρετεί, το αντικείμενο της LSH κλάσης που αντιστοιχεί στην χρονική αυτή στιγμή ανασύρεται από το InternalHashMap. Αν για την συγκεκριμένη χρονική στιγμή δεν έχει ανοίξει κάποιο παράθυρο, τότε η τελευταία αρχικοποιείται και προστίθεται στο παράθυρο της η πρώτη τιμή. Εν συνεχεία το παράθυρο αυτό συμπληρώνεται με τις κατάλληλες τιμές από τη δομή StoredTimeStamps, έτσι ώστε να αποκτήσει το επιθυμητό μήκος W. Όταν το παράθυρο της κλάσης του αντικειμένου που ανασύρεται έχει μήκος W εκκινείται η εκτέλεση της συνάρτησης της LSH κλάσης ,EstimateLSHVectors, η οποία αναλαμβάνει τη μετατροπή των παραθύρων σε lsh vectors.

```
public class LSHsynopsis {
	private final int W;
	private final int d;
	private final int slide;
	double[][] generator;
	private HashMap<String, TreeMap<Integer, WLSH>> externalHashMap;
	private HashMap<String, TreeMap<Integer, BitSet >> externalBitmapHashMap;
	private HashMap<Integer, Bucket> BucketsMap;
	private HashMap<String, NavigableMap<Integer, int[]>> SavedTimesHMPerSim;
}
```

Εικόνα 4.7: *LSHsynopsis Class*

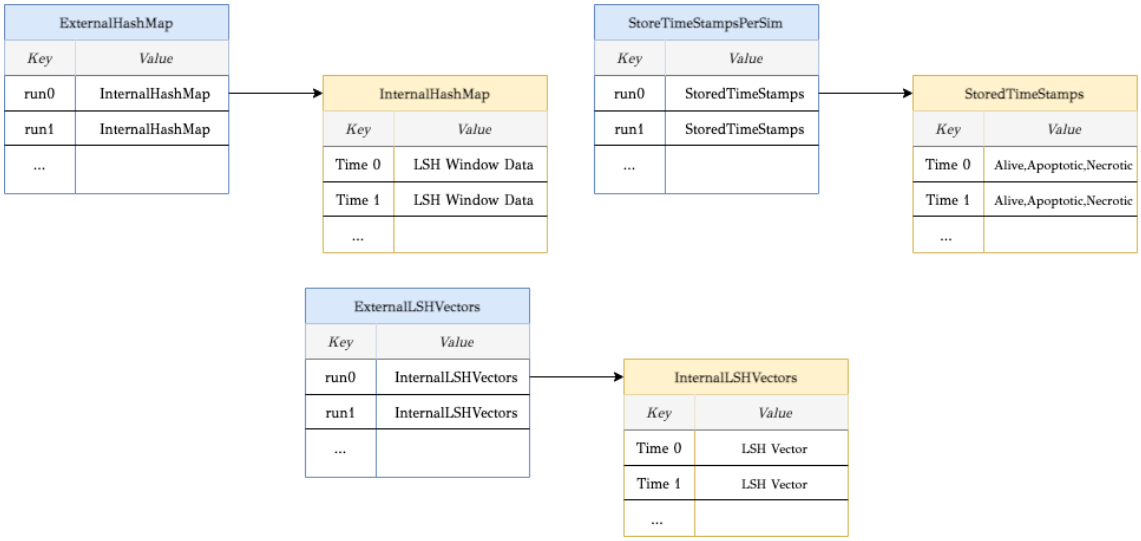
```
public class LSH {
    private final int W;
    private LinkedList<int[]> window_data;
    private int curNumData;

    public WLSH(String key, int w) {
        window_data = new LinkedList<>();
        W = w;
        curNumData = 0;
    }
```

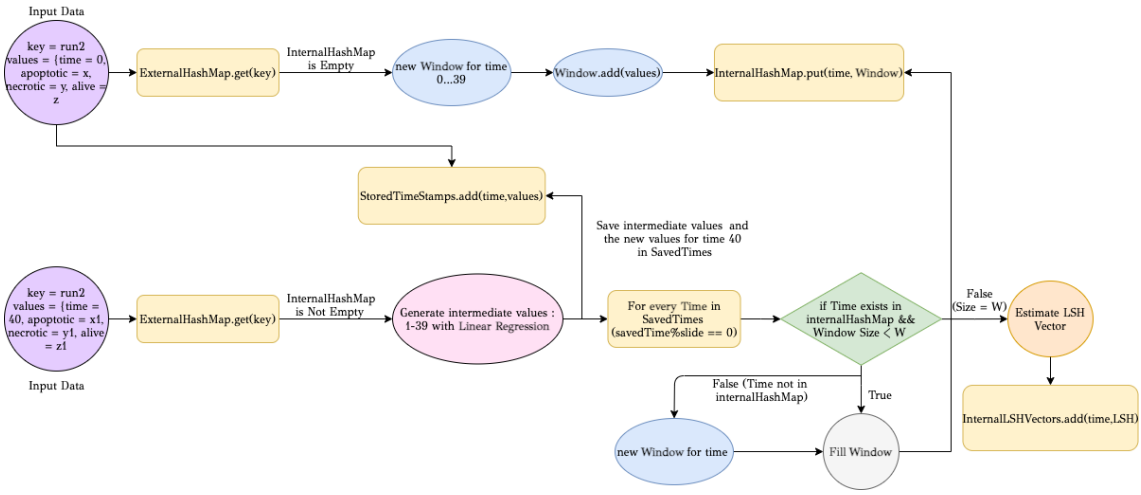
Εικόνα 4.8: *LSH Class*

Αλγοριθμος 4.1: *Add Function*

Input: Simulation, startTime, values
Output: LSH Vectors
if Simulation does not exist in StoredTimeStampsPerSim Hash Map **then**
    Create a StoredTimeStamps Hash Map for this simulation
end if
if Simulation does not exist in ExternalHashMap **then**
    Create an InternalHashMap for this simulation
    Create LSH object for this startTime
    Add values in window data of LSH object
    Put (startTime, LSH_object) in InternalHashMap
else
    Put (startTime, values) in StoredTimeStamps Hash Map
    Fill missing values with Linear Regression and store them in StoredTimeStamps Hash Map
    for every timeStamp in StoredTimeStamps **do**
        if (timeStamp mod(slide)) is equal to 0 **then**
            Get LSH Vector for this timeStamp from InternalHashMap
            if LSH Vector does not exist in InternalHashMap **then**
                Create LSH object for this timeStamp
            else
                if size of window data of LSH is less than W **then**
                    Fill window with values from StoredTimeStamps
                else if size of window data of LSH is equal to W **then**
                    Calculate its LSH Vector
                end if
            end if
        end if
    end for
end if



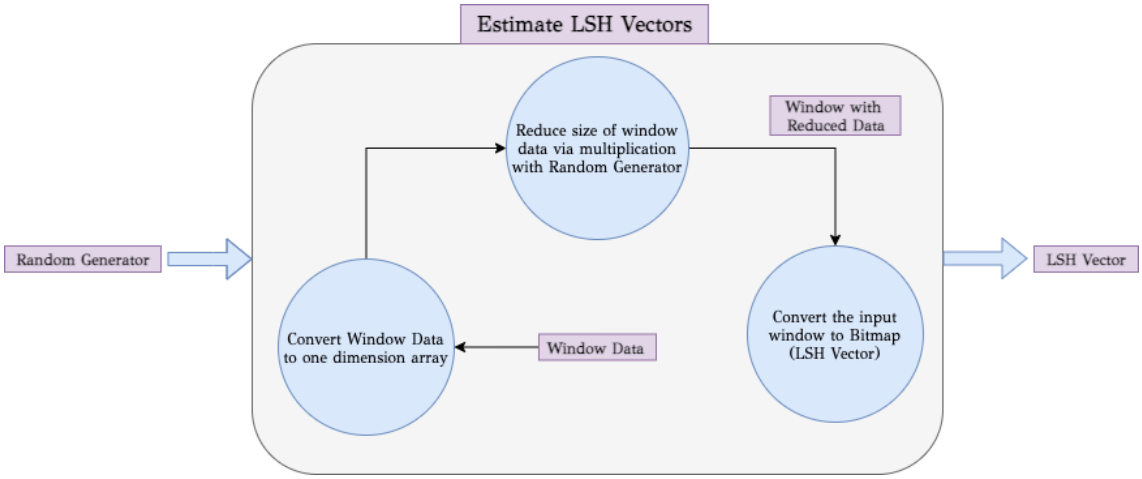
Εικόνα 4.9: *Hash Maps of LSHsynopsis Class*



Εικόνα 4.10: *Διάγραμμα Add Function*

Υλοποίηση Αλγόριθμου (1) - Υπολογισμός LSH Vectors

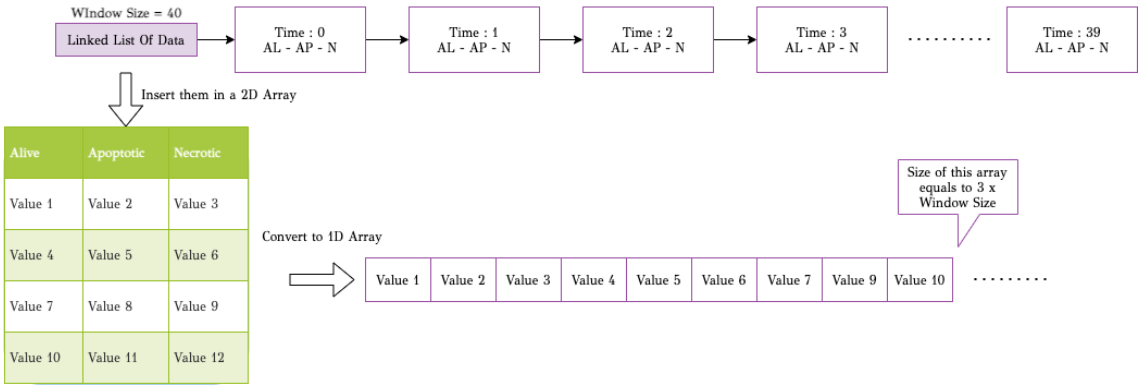
Η abstract σχεδίαση της συνάρτησης EstimateLSHVectors αποτυπώνεται στην Εικόνα 4.11.



Εικόνα 4.11: *Συνάρτηση EstimateLSHVectors Function*

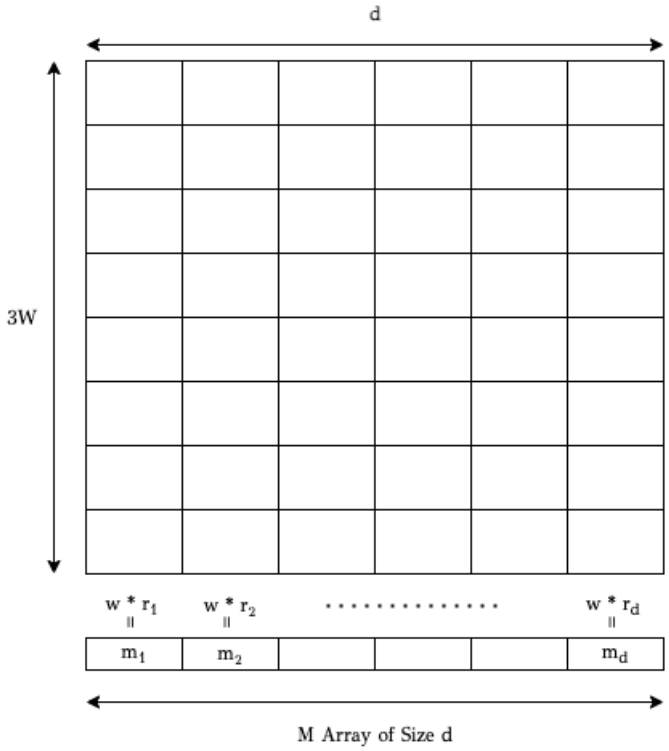
Απαραίτητη προϋπόθεση για τη μετατροπή των δεδομένων σε bitmaps (LSH Vectors) είναι το να βρίσκονται στην κατάλληλη διάσταση. Στην συγκεκριμένη περίπτωση, η διάστασή τους πρέπει να ταυτίζεται με εκείνη των bitmaps, έτσι ώστε να παραχθεί ορθό αποτέλεσμα μέσω της μετατροπής. Προκειμένου να επιτευχθεί μείωση των διαστάσεων των δεδομένων, το παράθυρο των τελευταίων πολλαπλασιάζεται με ένα πίνακα R (Random Generator) διάστασης $3W \times d$, όπου W το μέγεθος του παραθύρου εισόδου. Η διάσταση d αποτυπώνει το επιθυμητό μέγεθος του παραθύρου συμπιεσμένης πληροφορίας το οποίο αργότερα θα μετατραπεί σε LSH Vector. Η διάσταση 3W αντιστοιχεί στο αρχικό παράθυρο δεδομένων, των οποίων οι 2 διαστάσεις $(W \times 3)$ έχουν μετατραπεί σε μία με τον τρόπο που φαίνεται στην εικόνα 4.12.

Μετά την μετατροπή του παραθύρου δεδομένων στην κατάλληλη διάσταση ακολουθεί ο πολλαπλασιασμός του νέου παραθύρου με τη γεννήτρια. Κατά τον πολλαπλασιασμό αυτό, κάθε στήλη της γεννήτριας $r_c, c\epsilon\{1, ..., d\}$ μήκους $3W$ πολλαπλασιάζεται με το νέο παράθυ-



Εικόνα 4.12: *Μετατροπή διαστάσης παραθύρου δεδομένων από 2 σε 1*

ρο στοιχείο προς στοιχείο, όπως φαίνεται στην Εικόνα 4.13. Εν συνεχεία, τα γινόμενα που προκύπτουν από κάθε επιμέρους πολλαπλασιασμό αθροίζονται και το παραγόμενο αποτέλεσμα αποθηκεύεται σε έναν νέο πίνακα M. Η διαδικασία αυτή επαναλαμβάνεται έως ότου εξαντληθούν οι στήλες της γεννήτριας. Για κάθε στήλη προκύπτει ένα νέο άθροισμα που αποθηκεύεται στον πίνακα M. Συνεπώς ο πίνακας αυτός, μετά το πέρας της διαδικασίας, έχει μήκος d. Μέσω της διαδικασίας αυτής επιτυγχάνεται η συμπίεση της πληροφορίας του αρχικού παραθύρου σε αρκετά μικρότερο μέγεθος με σκοπό την αποδοτικότερη επεξεργασία των δεδομένων από τον αλγόριθμο.



Εικόνα 4.13: *Πολλαπλασιασμός Παραθύρου Δεδομένων με Γεννήτρια*

Τέλος, για την παραγωγή του LSH Vector, είναι απαραίτητη η μετατροπή του πίνακα M σε bitmap. Κατά τη διαδικασία αυτή, διατρέχεται κάθε θέση του πίνακα M και επακολουθεί ο έλεγχος της τιμής που βρίσκεται στο εσωτερικό της. Σε περίπτωση που η τιμή είναι θετική

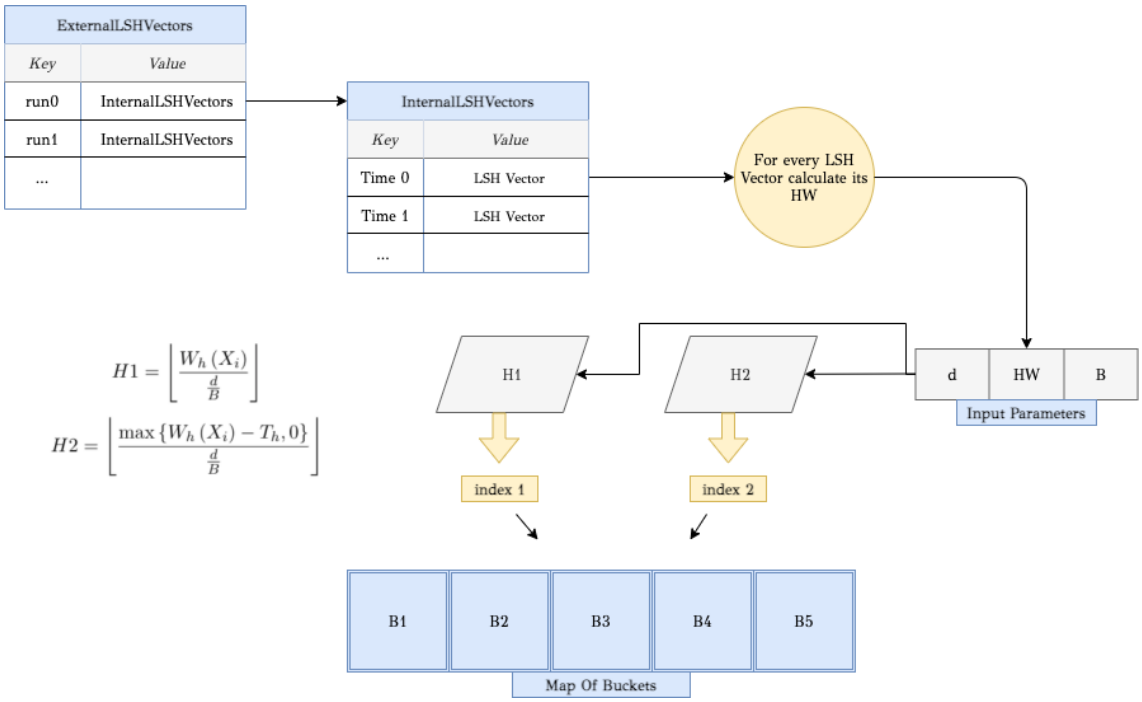
ή μηδενική, στην αντίστοιχη θέση του bitmap εισάγεται ο αριθμός 1, ενώ στην αντίθετη περίπτωση (αρνητική τιμή) εισάγεται ο αριθμός 0. Αφού παραχθεί το bitmap, εισάγεται ως value στο Hash Map (InternalLSHVectors) που έχει διαμορφωθεί με σκοπό τη διατήρηση χρονικών στιγμών - bitmaps, με key την χρονική στιγμή στην οποία αντιστοιχεί. Το Hash Map αυτό εισάγεται με τη σειρά του ως value σε ένα γενικότερο Hash Map (ExternalLSHVectors), με key την προσομοίωση στην οποία αντιστοιχεί. Η σύνδεση και το περιεχόμενο των δύο Hash Map που προαναφέρθηκαν παρουσιάζεται στην Εικόνα 4.9.

Estimate Request

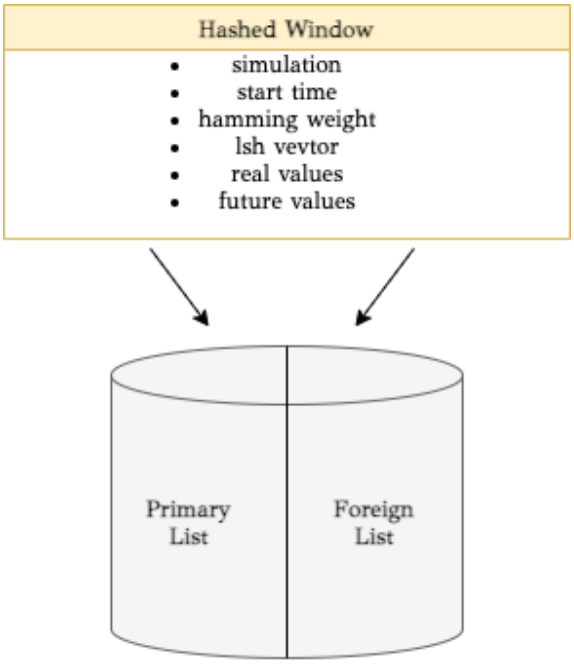
Έπειτα από την εκτέλεση της συνάρτησης add της LSHsynopsis και κατά επέκταση της εξυπηρέτησης του αιτήματος για προσθήκη δεδομένων στην σύνοψη Add Request, ακολουθεί το αίτημα υπολογισμού και διαμόρφωσης των bucket εξόδου, Estimate Request, στα οποία θα πρέπει να έχουν αποθηκευτεί τα κατάλληλα LSH Vectors. Τη διαδικασία αυτή αναλαμβάνει η συνάρτηση estimate η οποία δέχεται ως όρισμα το Threshold του συστήματος. Η επιλογή του Threshold ορίζει τον αριθμό των bucket που πρόκειται να αρχικοποιηθούν καθώς αποτελεί βασική παράμετρο στον τύπο υπολογισμού του αριθμού τους. Συνεπώς, αρχικά υπολογίζεται ο αριθμός των bucket, μέσω του μαθηματικού τύπου $B = \pi/arccos(T)$. Κάθε bucket αποθηκεύεται ως value σε ένα Hash Map, με key το bucket id. Όσο αυξάνεται η τιμή του Threshold, ο αριθμός των bucket πληθαίνει καθώς το σύστημα απαιτεί μεγαλύτερη ομοιότητα μεταξύ των LSH Vectors που αποθηκεύονται στα bucket, ενώ στην αντίθετη περίπτωση μειώνεται. Έπειτα μέσω μιας επαναληπτικής δομής υπολογίζεται, για κάθε χρονική στιγμή των αποθηκευμένων προσομοιώσεων στο Hash Map ExternalLSHVectors, το Hamming Weight του LSH Vector που της αντιστοιχεί. Το Hamming Weight χρησιμοποιείται, ως παράμετρος, από τις Hash Functions του αλγορίθμου με σκοπό τον υπολογισμό του Bucket id στο οποίο πρέπει να αποθηκευτεί (Εικόνα 4.14). Η hash function H1 υποδεικνύει την αποθήκευση του LSH Vector στη Primary List του bucket με το εξαγόμενο bucket id, σε αντίθεση με την hash function H2, η οποία υποδεικνύει την αποθήκευση του διανύσματος στη Foreign List. Πιο συγκεκριμένα μέσα σε κάθε λίστα του bucket δεν αποθηκεύεται το διάνυσμα, αλλά ένα object της κλάσης HashedWindow (Εικόνα 4.15) που το περιέχει. Το αντικείμενο της κλάσης προς αποθήκευση εκτός από το διάνυσμα, εμπεριέχει πληροφορίες για την προσομοίωση και τη χρονική στιγμή στην οποία αντιστοιχεί, τις πραγματικές και τις μελλοντικές τιμές του διανύσματος που πρόκειται να χρησιμοποιηθούν στο στάδιο της πρόβλεψης, καθώς και το Hamming Weight του. Μέρος του Estimate Request αποτελεί και η εκτέλεση των ερωτημάτων, η οποία αναλύεται διεξοδικά στο Κεφάλαιο 4.2.4.

4.2.4 Εκτέλεση Ερωτημάτων

Το σύστημα SDE δίνει την δυνατότητα στον χρήστη να θέσει μια πληθώρα ερωτημάτων στις συνόψεις που έχουν δημιουργηθεί στο εσωτερικό του και να λάβει τις ανάλογες απαντήσεις. Η συγκεκριμένη εργασία πραγματεύεται την εκτέλεση ερωτημάτων σε δεδομένα που εξάγονται από την LSH σύνοψη. Το SDE δουλεύει παράλληλα και κατανεμημένα. Συνεπώς η διαδικασία εκτέλεσης του εκάστοτε ερωτήματος εκκινείται μόνο όταν όλοι οι Workers (Εικόνα 4.16) επιστρέψουν το Hash Map των buckets που έχουν υπολογίσει στο προηγούμενο στάδιο.

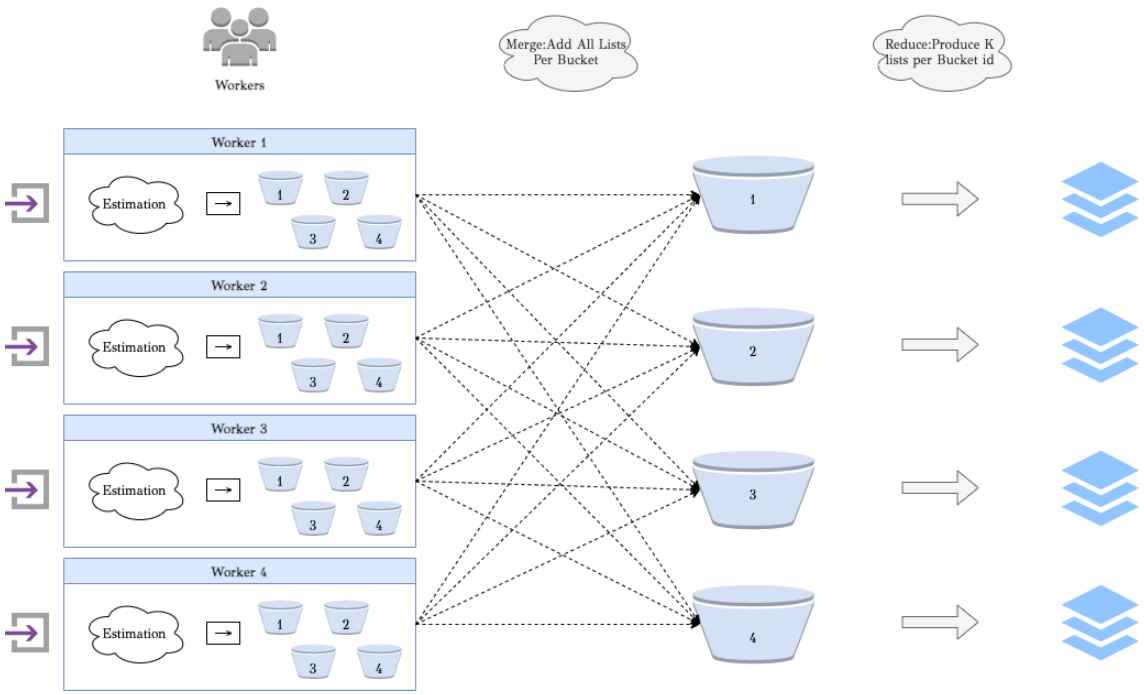


Εικόνα 4.14: *Προσθήκη Δεδομένων στα Buckets*



Εικόνα 4.15: *Αντικείμενο κλάσης Hashed Window*

Το ερώτημα που χρήζει απάντησης γνωστοποιείται στο σύστημα μέσω ενός κωδικού αριθμού (1,2,...) που αντιστοιχεί σε αυτό και εισάγεται με τις αντίστοιχες παραμέτρους του στο SDE κατά το Estimate Request. Εν συνεχεία, παρατίθενται τα δύο διαφορετικά ερωτήματα που έχουν υλοποιηθεί:



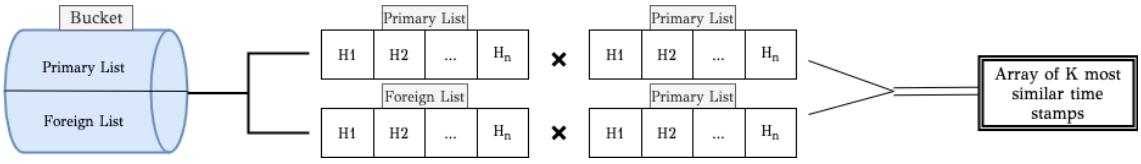
Εικόνα 4.16: *Merge και Reduce για παραλληλισμό 4*

Ερώτημα 1

→ ***Εντόπισε τα K πιο όμοια ζευγάρια χρονικών στιγμών προσομοιώσεων ανάμεσα στα συνολικά δεδομένα εισόδου, δεδομένου ενός Threshold ομοιότητας T_h.***

Με σκοπό την εξαγωγή του επιθυμητού αποτελέσματος, σε πρώτο στάδιο γίνεται αναζήτηση των K πιο όμοιων χρονικών στιγμών προσομοιώσεων στο εσωτερικό ενός bucket. Κατά την αναζήτηση αυτή, κάθε στοιχείο που είναι αποθηκευμένο στην Primary List συγκρίνεται με όλα τα υπόλοιπα που ανήκουν στην ίδια λίστα (Εικόνα 4.18). Σε κάθε σύγκριση γίνεται υπολογισμός της τιμής του correlation μεταξύ των συγκρινόμενων αντικειμένων. Η εξαγωγή της τιμής αυτής πραγματοποιείται μέσω της κλήσης της μεθόδου Pearson Correlation, η οποία δέχεται ως ορίσματα της πραγματικές τιμές των διανυσμάτων. Αναλυτικότερα, της εισαγωγής των πραγματικών τιμών στην μέθοδο που προαναφέρθηκε προηγείται η κατάλληλη διαμόρφωση των τιμών αυτών. Επειδή κάθε διάνυσμα φέρει πληροφορία για τρεις διαφορετικές φάσεις των καρκινικών κυττάρων (alive, apoptotic, necrotic) υπολογίζονται τρεις διαφορετικές τιμές correlation η οποίες αθροίζονται σε μία αφού πολλαπλασιαστούν με τα κατάλληλα βάρη. Τα βάρη αυτά ορίζονται ανάλογα με τη σημαντικότητα των κυττάρων που χαρακτηρίζονται από ένα συγκεκριμένο τύπο φάσης, και αθροίζουν στην μονάδα. Για τα alive κύτταρα ορίζεται βάρος 0.5, ενώ για τα apoptotic και τα necrotic από 0.25. Η τελική τιμή correlation που εξάγεται, σε περίπτωση που είναι ίση ή μεγαλύτερη του απαιτούμενου Threshold, χαρακτηρίζει το τρέχον ζευγάρι χρονικών στιγμών ως όμοιο. Για κάθε ζευγάρι που πληροί τις προϋποθέσεις ομοιότητας, δημιουργείται ένα αντικείμενο της κλάσης SimilarSims. Το αντικείμενο αυτό αποθηκεύει πληροφορίες σχετικές με το τρέχον ζευγάρι. Πιο συγκεκριμένα, διατηρεί τα αντικείμενα Hashed Windows που είναι όμοια, την τιμή του correlation που έχουν καθώς και τη διαφορά των Hamming Weight τους. Συνεπώς, το αντικείμενο της κλάσης SimilarSims

που δημιουργείται αποθηκεύεται σε ένα πίνακα Κ θέσεων που αντιστοιχεί στο τρέχον Bucket. Όταν ο πίνακας γεμίσει με Κ όμοια ζευγάρια, το νέο όμοιο ζευγάρι που θα προκύψει προς εισαγωγή συγκρίνεται με τα ήδη αποθηκευμένα στον πίνακα, και σε περίπτωση που υπερτερεί στο πεδίο της τιμής του correlation του ζευγαριού με το μικρότερο correlation στον πίνακα, το αντικαθιστά. Αφού ολοκληρωθούν οι συγκρίσεις στο εσωτερικό της Primary List, η διαδικασία επαναλαμβάνεται με τη διαφορά ότι οι συγκρίσεις γίνονται μεταξύ των στοιχείων Primary List και της Foreign List. Για κάθε bucket παράγεται ένας νέος πίνακας Κ θέσεων. Οι πίνακες από κάθε bucket συνενώνονται σε έναν ενιαίο και από αυτόν επιλέγονται τα Κ ζευγάρια με το μεγαλύτερο correlation.



Εικόνα 4.17: *Διάγραμμα Ερωτήματος 1*

Ερώτημα 2

→ ***Για κάθε παράθυρο προσομοίωσης με χρονική στιγμή εκκίνησης της λίστας εισόδου, εντόπισε τα Κ πιο όμοια με εκείνο παράθυρα, δεδομένου ενός Threshold ομοιότητας T_h.***

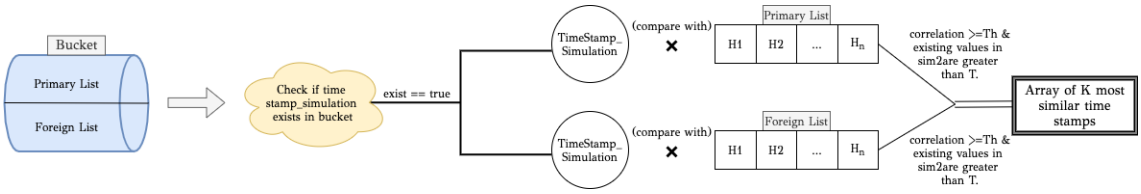
Η συνάρτηση που πρόκειται να εξυπηρετήσει το ερώτημα αυτό δέχεται ως ορίσματα τον επιθυμητό αριθμό όμοιων ζευγαριών Κ, το Threshold, έναν αριθμό Τ που ορίζει το πλήθος των τιμών που πρέπει να διαθέτουν οι όμοιες προσομοιώσεις, προκειμένου να γίνει σωστά η πρόβλεψη στο επόμενο στάδιο. και τη λίστα εισόδου όπου εμπεριέχονται οι χρονικές στιγμές με τις αντίστοιχες προσομοιώσεις τους. Μια μέθοδος καλείται επαναληπτικά για κάθε χρονική στιγμή με σκοπό την εξαγωγή των Κ πιο όμοιων με αυτή χρονικών στιγμών. Είναι προφανές το γεγονός ότι για κάθε χρονική μπορεί να παραχθεί μόνο μία μοναδική Κ-άδα όμοιων χρονικών στιγμών καθώς βρίσκεται αποθηκευμένη μόνο σε ένα από τα υπάρχοντα bucket. Με σκοπό την εξαγωγή του επιθυμητού αποτελέσματος, σε πρώτο στάδιο ελέγχεται η ύπαρξη της τρέχουσας χρονικής στιγμής, με την προσομοίωση στην οποία αντιστοιχεί, στο εσωτερικό της Primary List του bucket. Σε περίπτωση που υπάρχει στο συγκεκριμένο bucket, σε πρώτο στάδιο συγκρίνεται με όλα τα στοιχεία στην ίδια λίστα. Σε κάθε σύγκριση γίνεται υπολογισμός της τιμής του correlation μεταξύ των συγκρινόμενων αντικειμένων, όπως προαναφέρθηκε στο *Ερώτημα 1*. Η δημιουργία του αντικειμένου της κλάσης SimilarSims γίνεται με τον τρόπο που περιγράφηκε στο Ερώτημα 1. Συνεπώς, το αντικείμενο της κλάσης SimilarSims που δημιουργείται αποθηκεύεται σε ένα πίνακα Κ θέσεων που αντιστοιχεί στο τρέχον Bucket, υπό την προϋπόθεση ότι το παράθυρο της χρονικής στιγμής που είναι όμοια με τη τρέχουσα έχει τουλάχιστον Τ χρονικές στιγμές διαθέσιμες μέχρι την λήξη του. Όταν ο πίνακας γεμίσει με Κ όμοια ζευγάρια, το νέο όμοιο ζευγάρι που θα προκύψει προς εισαγωγή συγκρίνεται με τα ήδη αποθηκευμένα στον πίνακα, και σε περίπτωση που υπερτερεί στο πεδίο της τιμής του correlation του ζευγαριού με το μικρότερο correlation στον πίνακα, το αντικαθιστά. Αφού ολοκληρωθούν οι συγκρίσεις με τα στοιχεία της Primary List, η διαδικασία επαναλαμβάνεται

Αλγοριθμος 4.2: *Ερώτημα 1*

```
Input: K, Threshold
Output: Array of K most similar time stamps
New Array[K] of SimilarSims
for every Hashed Window h1 in Primary List do
    for every Hashed Window h2 in Primary List do
        Compute Pearson correlation between h1 and h2
        if correlation is greater or equal to Threshold then
            New Similar Sims object (h1, h2, correlation, HW difference)
            if Size of array not equal to K then
                Array ← newSimilarSimsObject
            else
                Find lowest correlation in array
                if new correlation is greater than lowest correlation then
                    Replace Similar Sims object with lowest correlation with the new one
                end if
            end if
        end if
    end for
end for
for every Hashed Window h1 in Primary List do
    for every Hashed Window h2 in Foreign List do
        Compute Pearson correlation between h1  h2
        if correlation is greater or equal to Threshold then
            New Similar Sims object (h1, h2, correlation, HW difference)
            if Size of array not equal to K then
                Array ← newSimilarSimsObject
            else
                Find lowest correlation in array
                if new correlation is greater than lowest correlation then
                    Replace Similar Sims object with lowest correlation with the new one
                end if
            end if
        end if
    end for
end for
```

με τη διαφορά ότι οι συγκρίσεις γίνονται μεταξύ της τρέχουσας χρονικής στιγμής και των στοιχείων της Foreign List.

Οι μέθοδοι που εξυπηρετούν τα παραπάνω ερωτήματα εφαρμόζονται σε κάθε bucket. Στη συνέχεια τα αποτελέσματα κάθε επανάληψης της μεθόδου συνενώνονται σε περίπτωση που το επιβάλλει το ερώτημα (π.χ. Ερώτημα 1).



Εικόνα 4.18: *Διάγραμμα Ερωτήματος 2*

Αλγοριθμος 4.3: *Ερώτημα 2*

Input: K, Threshold,T, list of time stamps
Output: Array of K most similar time stamps
for every time stamp in list **do**
  New Array[K] of SimilarSims
  if time stamp exists in Primary List **then**
    Get Hashed Window Object h for this time stamp
    for every Hashed Window h1 in Primary List **do**
      Compute Pearson correlation between h and h1
      if correlation is greater or equal to Threshold **then**
        New Similar Sims object (h, h1, correlation, HW difference)
        if Size of array not equal to K **then**
          Array ← *newSimilarSimsObject*
        else
          Find lowest correlation in array
          if new correlation is greater than lowest correlation **then**
            Replace Similar Sims object with lowest correlation with the new one
          end if
        end if
      end if
    end for
    for every Hashed Window h1 in Foreign List **do**
      Compute Pearson correlation between h and h1
      if correlation is greater or equal to Threshold **then**
        New Similar Sims object (h, h1, correlation, HW difference)
        if Size of array not equal to K **then**
          Array ← *newSimilarSimsObject*
        else
          Find lowest correlation in array
          if new correlation is greater than lowest correlation **then**
            Replace Similar Sims object with lowest correlation with the new one
          end if
        end if
      end if
    end for
  end if
end for

4.2.5 Μοντέλο Πρόβλεψης

Τελευταίο αλλά εξίσου σημαντικό ζητούμενο αυτής της διπλωματικής εργασίας είναι η υλοποίηση και η χρήση ενός μοντέλου πρόβλεψης με σκοπό την πρόβλεψη της μελλοντικής συμπεριφοράς μιας προσομοίωσης. Η ανάγκη για την δημιουργία ενός τέτοιου μοντέλου προκύπτει από τον περιορισμό του χρόνου εκτέλεσης πολύπλοκων και χρονοβόρων προσομοιώσεων ενώ παράλληλα συμβάλλει στην έγκαιρη και έγκυρη λήψη αποφάσεων.

Προεπεξεργασία Δεδομένων

Σε κάθε χρονικό παράθυρο προσομοίωσης, του οποίου η μελλοντική συμπεριφορά πρέπει να προβλεφθεί, αντιστοιχεί μια λίστα των K πιο όμοιων με εκείνο χρονικών παραθύρων. Η πληροφορία που εμπεριέχεται στα K χρονικά παράθυρα θα αποτελέσει ισχυρό συστατικό της πρόβλεψης. Πριν την επεξεργασία των δεδομένων, αυτά κανονικοποιούνται σύμφωνα με τη σχέση

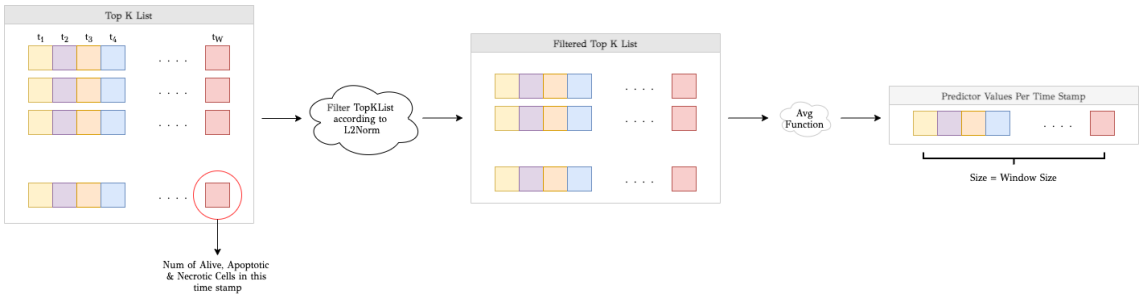
$$\frac{value - min}{\max - \min}$$

έτσι ώστε όλες οι τιμές να κλιμακώνονται στο διάστημα τιμών 0 έως 1. Η κανονικοποίηση των δεδομένων γίνεται γιατί κάθε προσομοίωση εκκινείται με διαφορετικό αριθμό alive κυττάρων.

Με σκοπό την επίτευξη όσο το δυνατόν καλύτερης πρόβλεψης είναι απαραίτητο το φιλτράρισμα των δεδομένων (Εικόνα 4.19). Τα χρονικά παράθυρα που εμπεριέχονται στην Top K λίστα είναι όμοια με το εξεταζόμενο παράθυρο καθώς συνδέονται με εκείνον με ένα μεγάλο συντελεστής ομοιότητας T_h. Ο συντελεστής ομοιότητας έχει προκύψει μέσω της χρήσης της μεθόδου Pearson Correlation. Παρ'όλα αυτά ο υψηλός συντελεστής ομοιότητας δεν συνεπάγεται ισχυρή ταύτιση τιμών. Για να εξασφαλιστεί η μεγάλη ταύτιση τιμών χρησιμοποιείται η μετρική της L2 νόρμας ή Ευκλείδειας απόστασης μεταξύ δύο διανυσμάτων. Η μετρική αυτή χρησιμοποιείται κατά την προ-επεξεργασία των δεδομένων. Κάθε παράθυρο w στην Top K λίστα σε συνδυασμό με το παράθυρο προς πρόβλεψη αποτελούν ένα ζεύγος. Για κάθε τέτοιο ζεύγος υπολογίζεται η Ευκλείδεια απόσταση μεταξύ των διανυσμάτων για κάθε διαφορετική κατάσταση των κυττάρων (Alive, Apoptotic, Necrotic). Σε περίπτωση που μία από τις τρεις αποστάσεις υπερβαίνει το όριο απόστασης που τίθεται κάθε φορά, το παράθυρο w δεν εισέρχεται στην νέα φιλτραρισμένη λίστα. Μετά το φιλτράρισμα των δεδομένων έχει δημιουργηθεί μια νέα σύνθεση δεδομένων που αποτελείται από παράθυρα των οποίων οι τιμές για τις τρεις διαφορετικές καταστάσεις βρίσκονται πολύ κοντά σε εκείνες του προβλεπόμενου παραθύρου. Η νέα λίστα αντικαθίσταται από ένα μεμονωμένο διάνυσμα μεγέθους $|w|$, το οποίο προκύπτει μέσω της εφαρμογής της συνάρτησης του μέσου όρου. Πιο συγκεκριμένα για κάθε χρονική στιγμή των παραθύρων υπολογίζεται ο μέρος όρος του αριθμού των alive, apoptotic και necrotic κυττάρων όλων των παραθύρων για την στιγμή αυτή.

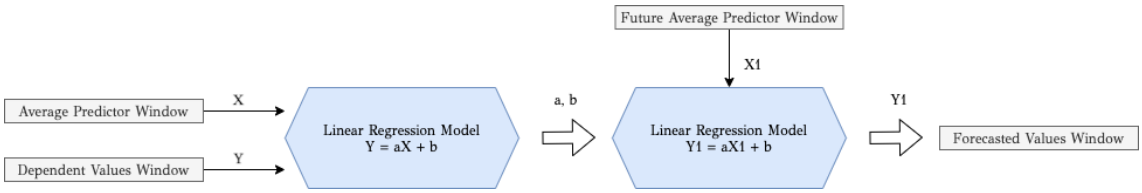
Υλοποίηση Μοντέλου Πρόβλεψης

Μετά την προεπεξεργασία των δεδομένων προκύπτει ένα νέο ζεύγος παραθύρων. Το ζεύγος αυτό αποτελείται από το παράθυρο προς πρόβλεψη και το όμοιό του w_{avg}, το οποίο αντιστοιχεί στο μέσο όρο που περιγράφηκε ως αντικατάσταση της φιλτραρισμένης λίστας. Το προβλεπόμενο παράθυρο έχει τον ρόλο της εξαρτημένης μεταβλητής στο μοντέλο της Γραμμικής



Εικόνα 4.19: *Προεπεξεργασία Δεδομένων Εισόδου*

Παλινδρόμησης, ενώ το παράθυρο w_{avg} έχει τον ρόλο της ανεξάρτητης μεταβλητής. Εφαρμόζοντας τους τύπους που περιγράφηκαν κατά την θεωρητική ανάλυση του μοντέλου πολλαπλής γραμμικής παλινδρόμησης με X και Y τα κατάλληλα διανύσματα, προκύπτουν οι μεταβλητές a και b. Το μοντέλο πρόβλεψης εφαρμόζεται για κάθε διαφορετική κατάσταση των κυττάρων. Συνεπώς προκύπτουν τρία ζεύγη α και β μεταβλητών. Οι μεταβλητές αυτές δείχνουν την γραμμική εξάρτηση μεταξύ των δύο μεταβλητών και χρησιμοποιούνται στην πρόβλεψη της συμπεριφοράς της εξαρτημένης μεταβλητής από τις επόμενες τιμές της ανεξάρτητης μεταβλητής (future average predictor values). Οι μελλοντικές τιμές για κάθε διαφορετική κατάσταση των κυττάρων προκύπτουν από την σχέση $ForecastedValues = a * FuturePredictorValues + b$ με a και b το ζεύγος μεταβλητών της αντίστοιχης κατάστασης.



Εικόνα 4.20: *Μοντέλο Γραμμικής Παλινδρόμησης*

Αλγοριθμος 4.4: *Συνάρτηση αποδοτικής πρόβλεψης μελλοντικών τιμών μιας προσομοίωσης με βάση τις K πιο όμοιες με εκείνη προσομοιώσεις*

Input: TopKList, Current_Values of Simulation For Prediction
Output: Predicted Values Per Dimension
List Total_Avg_Predictor
for every time stamp **do**
    $sum \leftarrow 0$
    filter TopKList with L2Norm
    for every predictor from filtered TopKList **do**
        Calculate a sum per Alive, Apoptotic and Necrotic Value
    end for
    $avg\_values \leftarrow sum/KList.size()$
    $Total\_Avg\_Predictor.add(avg\_values)$
end for
if Total_Avg_Predictor.size() is equal to Current_Values.size() **then**
    $[a0, b0] \leftarrow LR(Current\_Values\_Alive, Total\_Avg\_Predictor\_Alive)$
    $[a1, b1] \leftarrow LR(Current\_Values\_Apoptotic, Total\_Avg\_Predictor\_Apoptotic)$
    $[a2, b2] \leftarrow LR(Current\_Values\_Necrotic, Total\_Avg\_Predictor\_Necrotic)$
end if
$Forecasting\_Alive = predict(a0, b0, Future\_Predictor\_Alive)$
$Forecasting\_Apoptotic = predict(a1, b1, Future\_Predictor\_Apoptotic)$
$Forecasting\_Necrotic = predict(a2, b2, Future\_Predictor\_Necrotic)$

Κεφάλαιο 5

Έλεγχος

Στο κεφάλαιο αυτό γίνεται ο έλεγχος καλής λειτουργίας του συστήματος.

5.1 Μεθοδολογία Ελέγχου

Ο έλεγχος του συστήματος αυτού πραγματοποιήθηκε μέσω της εφαρμογής της τοπολογίας του συστήματος στον cluster του πολυτεχνείου. Ο cluster αποτελείται από 11 μηχανήματα Dell PowerEdge R300 Quad Core Xeon X3323 2.5GHz, 8GB RAM, 500GB HDD, 3 μηχανήματα Dell PowerEdge R310 Quad Core Xeon X3440 2.53GHz, 8GB RAM, 500GB HDD, 3 μηχανήματα Dell PowerEdge R310 Quad Core Xeon X3440 2.53GHz, 16GB RAM, 500GB HDD, 3 μηχανήματα Dell PowerEdge R320 Intel Xeon E5-2430 v2 2.50GHz, 32GB RAM, 1TB HDD και 1 μηχάνημα Dell PowerEdge R320 Intel Xeon E5-2430 v2 2.50GHz, 32GB RAM, 2 x 1TB HDD.

5.2 Αναλυτική παρουσίαση ελέγχου

Η διεξαγωγή των πειραμάτων χωρίζεται σε δύο μέρη. Το πρώτο μέρος αφορά στον έλεγχο της ορθής υλοποίησης του αλγόριθμου και το δεύτερο μέρος, μέσω διαγραμμάτων που αντιστοιχούν στο μέρος της πρόβλεψης, αποδεικνύει κατά πόσο ο αλγόριθμος ανταποκρίνεται στο βασικό ζητούμενο της εργασίας, το οποίο είναι η σωστή ομαδοποίηση των δεδομένων εισόδου ανάλογα με την ομοιότητά τους.

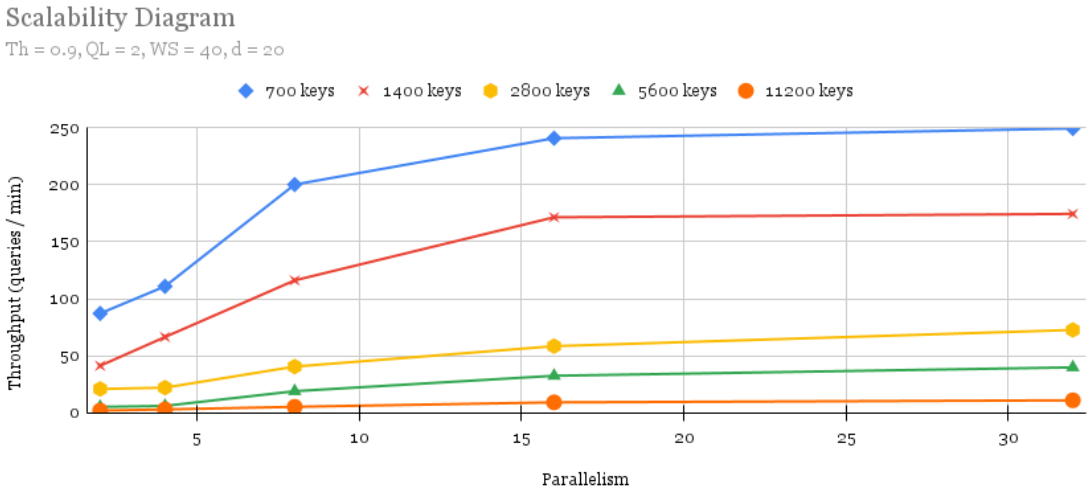
5.2.1 Πρώτο Μέρος Πειραμάτων

Όπως προαναφέρθηκε, εκτελείται ένας αριθμός διαφορετικών πειραμάτων και μετρήσεων με σκοπό τον έλεγχο της ορθής υλοποίησης του αλγόριθμου. Σε κάθε πείραμα μεταβάλλεται και μία διαφορετική παράμετρος του προγράμματος με σκοπό την μέτρηση της ταχύτητας επεξεργασίας των queries από το σύστημα και κατ'επέκταση την απόδοσή του. Αξίζει να σημειωθεί πως το αρχικό διαθέσιμο dataset εμπεριέχει 1400 κλειδιά, τα οποία πολλαπλασιάστηκαν για την εξυπηρέτηση των αναγκών των μετρήσεων. Οι χρόνοι αφορούν στους χρόνους εκτέλεσης των ερωτημάτων και συγκεκριμένα τη διάρκεια μεταξύ της αποστολής του πρώτου ερωτήματος και της απάντησης του τελευταίου, με συνολικό αριθμό 80 ερωτημάτων. Κάτω από τον τίτλο κάθε διαγράμματος παρατίθενται οι παράμετροι που παραμένουν σταθερές και εξηγούνται ως εξής:

- Th = Threshold
- QL = Αριθμός Παραθύρων Ερωτήματος
- WS = Μέγεθος Παραθύρου
- d = Μέγεθος LSH Vector
- P = Αριθμός workers / Μέγεθος Παραλληλισμού

Scalability Συστήματος

Η μεταβολή του μεγέθους του παραλληλισμού επιτυγχάνεται μέσω της μεταβολής της αντίστοιχης παραμέτρου κατά την αρχικοποίηση του συστήματος.Αυξάνοντας την παράμετρο αυτή επιτυγχάνεται αύξηση του αριθμού των μηχανημάτων που αναλαμβάνουν την κατανεμημένη και παράλληλη λειτουργία του συστήματος. Κατά την αύξηση αυτή αναμένεται βελτίωση της απόδοσης του συστήματος και κατ'επέκταση του ρυθμού εκτέλεσης - απάντησης των ερωτημάτων, καθώς ο φόρτος εργασίας των μηχανημάτων μειώνεται όταν αυτά πληθαίνουν. Αυτό αποδεικνύεται και στο διάγραμμα που ακολουθεί (Εικόνα 5.1). Για το πείραμα αυτό έχουν χρησιμοποιηθεί πέντε διαφορετικές τιμές κλειδιών (700, 1400, 2800, 5600, 11200). Για κάθε διαφορετική ποσότητα κλειδιών παρουσιάζεται η μεταβολή της απόδοσης καθώς αλλάζει ο παραλληλισμός. Για τις τιμές του παραλληλισμού από 2 έως 16, η απόδοση του συστήματος αυξάνεται σχεδόν γραμμικά όπου και κορυφώνεται σχεδόν σε όλες τις περιπτώσεις στους 16 workers καθώς ο κατακερματισμός των διεργασιών μεταξύ των workers δεν βελτιώνεται ουσιαστικά για παραλληλισμό ίσο με 32. Τέλος, όπως είναι αναμενόμενο όσο αυξάνεται ο αριθμός των κλειδιών τόσο μειώνεται η απόδοση του συστήματος καθώς το σύστημα υπερφορτώνεται. Οι καλύτερες αποδόσεις αντιστοιχούν στα 700 και στα 1400 κλειδιά με τα δεύτερα να δίνουν μέγιστο ρυθμό απάντησης ερωτημάτων τα 175 ερωτήματα το λεπτό.

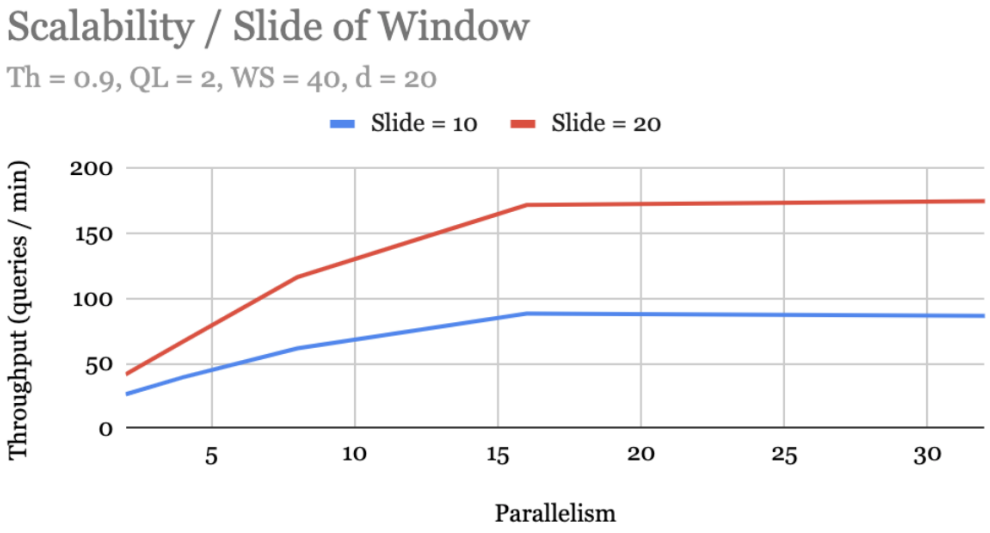


Εικόνα 5.1: *Διάγραμμα Scalability για διαφορετικές τιμές κλειδιών*

Τα πειράματα που ακολουθούν αφορούν στην **επεξεργασία των 1400 κλειδιών** που αντιστοιχούν στο αρχικό dataset.

Συσχέτιση Scalability Συστήματος και Slide Παραθύρου

Το ακόλουθο διάγραμμα 5.2 αποτυπώνει τις ίδιες μετρικές με το προηγούμενο, με τη διαφορά ότι εστιάζει στην απόδοση για 1400 κλειδιά με μεταβολή του slide παραθύρου. Παρατηρείται πως ο υποδιπλασιασμός του slide έχεις ως άμεση συνέπεια τον σχεδόν υποδιπλασιασμό της απόδοσης καθώς τα παράθυρα δεδομένων διπλασιάζονται.



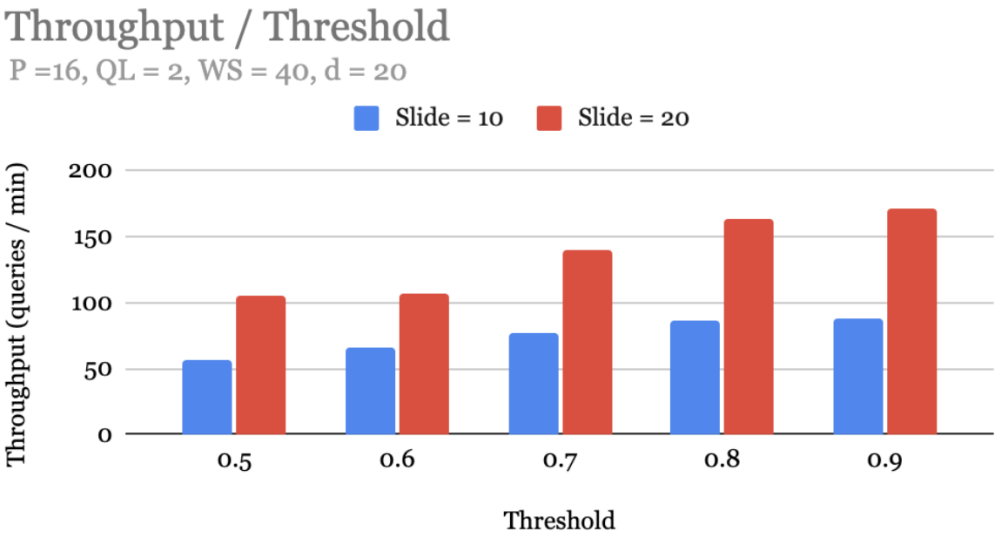
Εικόνα 5.2: *Διάγραμμα Scalability για 1400 κλειδιά και δύο διαφορετικές τιμές slide*

Συσχέτιση Throughput και Threshold Συστήματος

Η μεταβολή του μεγέθους του Threshold επιτυγχάνεται μέσω της μεταβολής της αντίστοιχης παραμέτρου κατά την αρχικοποίηση του συστήματος. Η αύξηση του Threshold έχει ως αποτέλεσμα την αύξηση του αριθμού των buckets στα οποία κατακερματίζονται τα παράθυρα εισόδου. Η ύπαρξη μεγαλύτερου πλήθους buckets συνεπάγεται την αποθήκευση λιγότερων δεδομένων σε κάθε ένα από αυτά καθώς και την αύξηση του βαθμού ομοιότητας των παραθύρων στο εσωτερικό των buckets. Κατά την μεταβολή του Threshold η απόδοση μεταβάλλεται με αργό ρυθμό, όπως φαίνεται στο ακόλουθο διάγραμμα (Εικόνα 5.3), καθώς ο αριθμός των buckets μεταβάλλεται εξίσου αργά (Th = 0.5 με 3 buckets, Th = 0.6 με 3 buckets, Th = 0.7 με 4 buckets, Th = 0.8 με 5 buckets, Th = 0.9 με 7 buckets). Τέλος, παρατηρείται πως ο υποδιπλασιασμός του slide έχεις ως άμεση συνέπεια τον σχεδόν υποδιπλασιασμό της απόδοσης καθώς τα παράθυρα δεδομένων διπλασιάζονται.

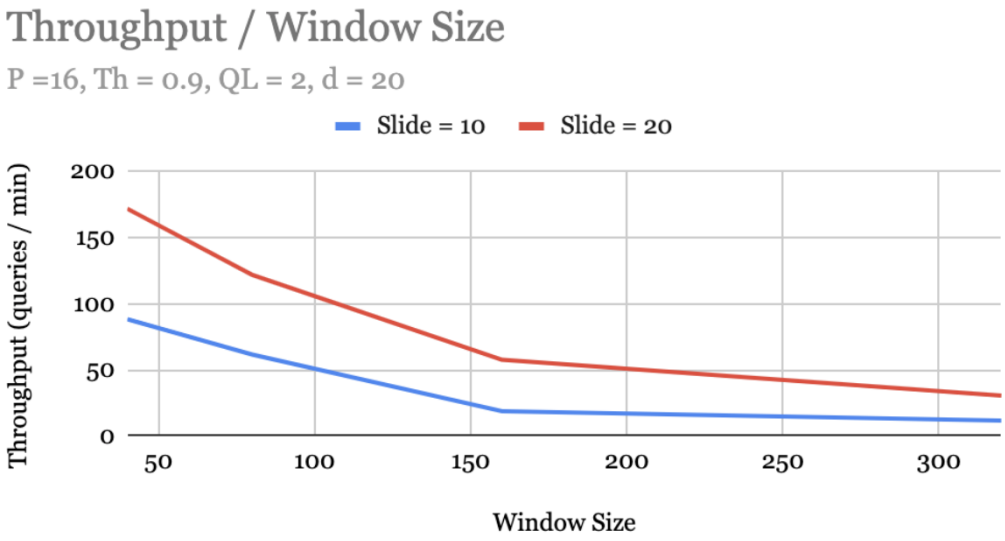
Συσχέτιση Throughput και Μεγέθους Παραθύρου

Η μεταβολή του μεγέθους του παραθύρου επιτυγχάνεται μέσω της μεταβολής της αντίστοιχης παραμέτρου κατά την αρχικοποίηση του συστήματος. Ο διπλασιασμός του μεγέθους των παραθύρων συνεπάγεται υποδιπλασιαμό του μέτρου της απόδοσης του συστήματος καθώς ο αλγόριθμος υπολογισμού των ερωτημάτων ακολουθεί γραμμική πολυπλοκότητα. Η ορθότητα αποδεικνύεται στο ακόλουθο διάγραμμα (Εικόνα 5.4) όπου η απόδοση υποδιπλασιάζεται και τυχόν αποκλίσεις οφείλονται σε επικοινωνιακές καθυστερήσεις μεταξύ των components του



Εικόνα 5.3: *Τιμές Throughput για 5 διαφορετικές τιμές του Threshold*

συστήματος. Τέλος, παρατηρείται πως ο υποδιπλασιασμός του slide έχεις ως άμεση συνέπεια τον σχεδόν υποδιπλασιασμό της απόδοσης καθώς τα παράθυρα δεδομένων διπλασιάζονται.

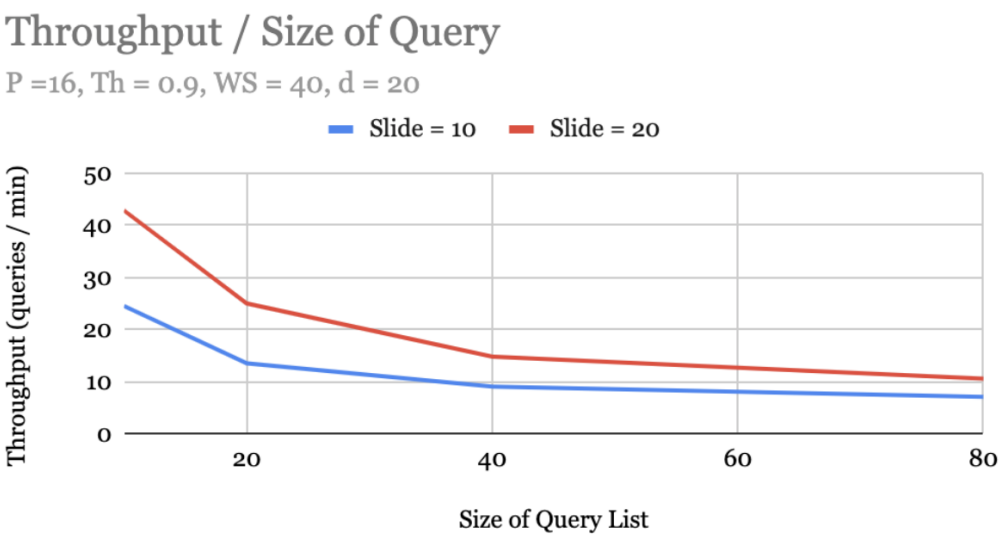


Εικόνα 5.4: *Τιμές Throughput για διαφορετικές τιμές του μεγέθους των παραθύρων των δεδομένων*

Συσχέτιση Throughput και Μεγέθους Ερωτήματος

Η μεταβολή του μεγέθους του ερωτήματος επιτυγχάνεται μέσω της προσθήκης πολλαπλών παραθύρων στην λίστα εισόδου του ερωτήματος. Όπως έχει προαναφερθεί κατά την εκτέλεση ενός ερωτήματος, για κάθε παράθυρο στην είσοδο αναζητούνται τα K πιο όμοια με εκείνο παράθυρο. Συνεπώς μεγαλύτερο μέγεθος ερωτήματος συνεπάγεται μεγαλύτερη χρονική διάρκεια εκτέλεσης του. Διπλασιάζοντας το μέγεθος του ερωτήματος, διπλασιάζεται και ο χρόνος εκτέλεσης του, καθώς ο αλγόριθμος υπολογισμού των ερωτημάτων ακολουθεί γραμμι-

κή πολυπλοκότητα. Η ορθότητα αποδεικνύεται στο ακόλουθο διάγραμμα (Εικόνα 5.5) όπου η απόδοση υποδιπλασιάζεται και τυχόν αποκλίσεις οφείλονται σε επικοινωνιακές καθυστερήσεις μεταξύ των components του συστήματος. Τέλος, παρατηρείται πως ο υποδιπλασιασμός του slide έχεις ως άμεση συνέπεια τον σχεδόν υποδιπλασιασμό της απόδοσης καθώς τα παράθυρα δεδομένων διπλασιάζονται.



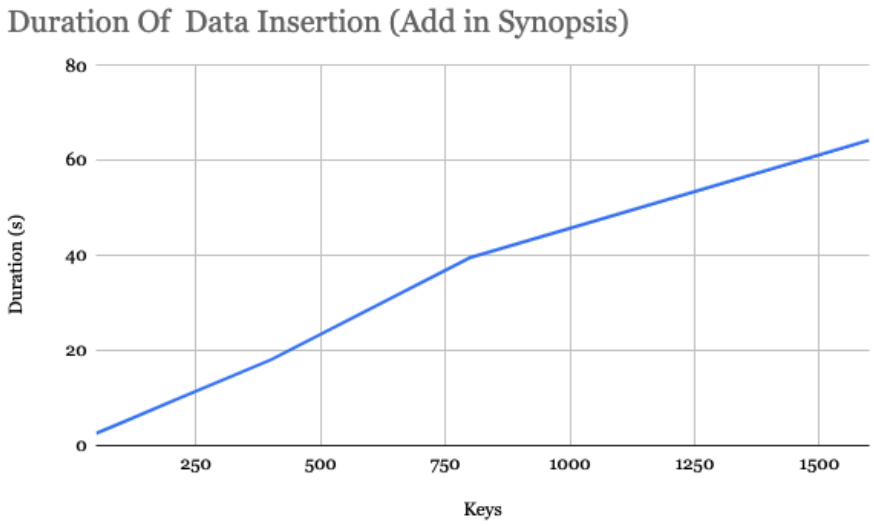
Εικόνα 5.5: *Τιμές Throughput για διαφορετικές τιμές του μεγέθους του ερωτήματος*

Διάρκεια Εισαγωγής Δεδομένων στην Σύνοψη

Η μεταβολή του μεγέθους της εισόδου επιτυγχάνεται μέσω της μεταβολής του συνόλου των κλειδιών του συστήματος στην είσοδο αυτού. Καθώς ο αριθμός των κλειδιών αυξάνεται, ο χρόνος που απαιτείται για την ενημέρωση της σύνοψης μεταβάλλεται με την ίδια τάση και τον ίδιο ρυθμό όπως φαίνεται στην Εικόνα 5.6. Το γεγονός αυτό επιβεβαιώνει την γραμμική πολυπλοκότητα της συνάρτησης που αναλαμβάνει την ενημέρωση της σύνοψης με νέα δεδομένα.

5.2.2 Δεύτερο Μέρος Πειραμάτων

Κατά τη διάρκεια των πειραμάτων αυτών εξετάζεται η τάση της μελλοντικής συμπεριφοράς τον παραθύρων εισόδου του ερωτήματος. Στα πρώτα πειράματα αποτυπώνεται η μελλοντική συμπεριφορά της κατηγορίας των alive κυττάρων. Δεν έχουν επιλεγεί τα apoptotic και τα necrotic κύτταρα καθώς οι τιμές τους είναι πάρα πολύ μικρές με αποτέλεσμα την αδυναμία παρακολούθησης της ξεκάθαρης μελλοντικής τους συμπεριφοράς. Τα διαγράμματα της αριστερής στήλης που ακολουθούν στην Εικόνα 5.7 αποτυπώνουν την μελλοντική συμπεριφορά (κόκκινη γραμμή) για τέσσερα διαφορετικά παράθυρα σε σύγκριση με την τάση των αντίστοιχων predictor παραθύρων τους (πράσινη γραμμή). Κάθε χρονικό παράθυρο αποτελείται από σαράντα συνεχόμενες χρονικές στιγμές και γίνεται πρόβλεψη για την συμπεριφορά των επόμενων σαράντα μετά το τέλος των πρώτων. Και στις τέσσερις περιπτώσεις η τάση των προβλεπόμενων τιμών ταυτίζεται με εκείνη των αντίστοιχων predictor τους γεγονός που αποδεικνύει την



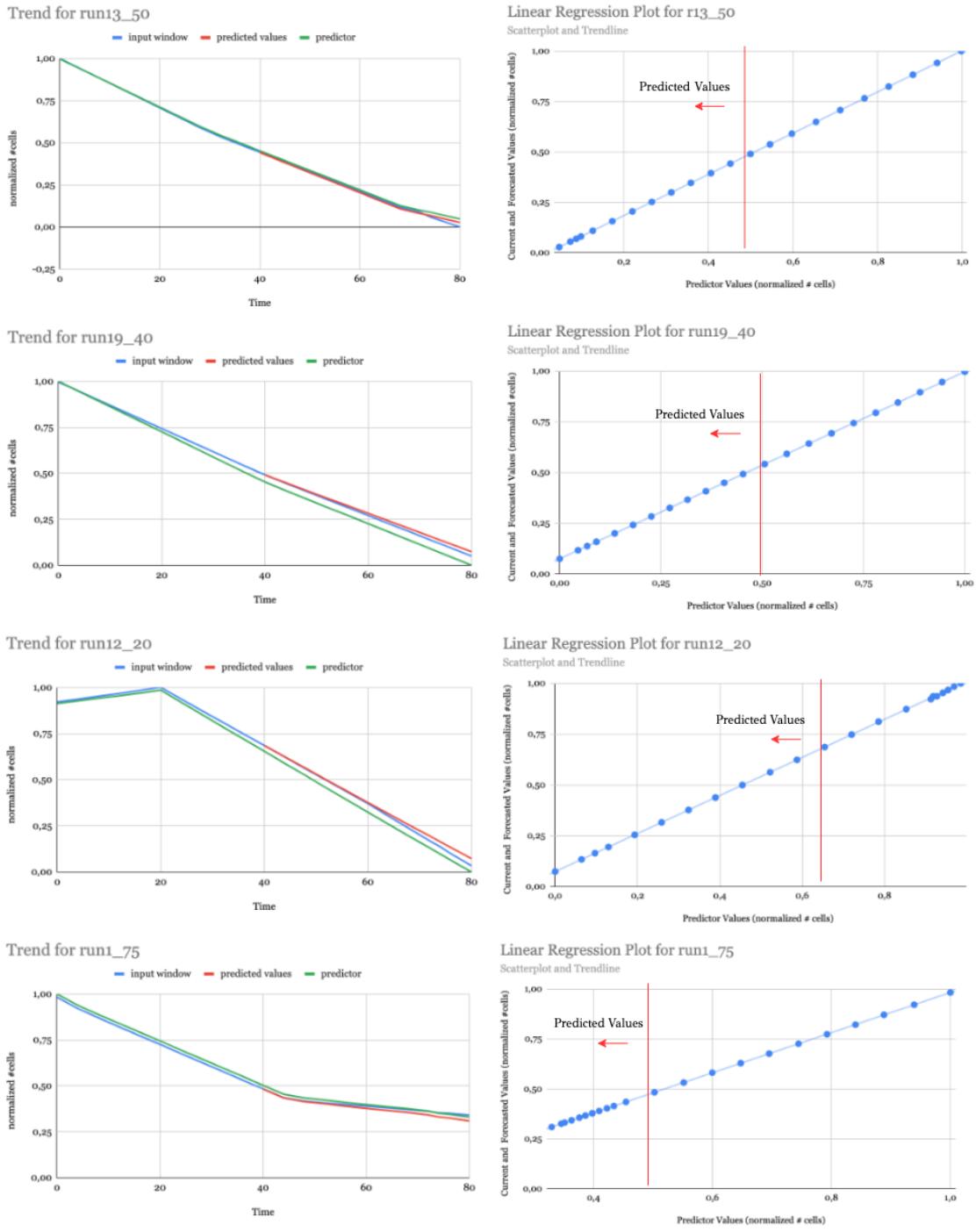
Εικόνα 5.6: *Διάρκεια ενημέρωσης σύνοψης συναρτήσει του μεγέθους της εισόδου*

αποδοτική πρόβλεψη του μοντέλου γραμμικής παλινδρόμησης.

Στη δεξιά στήλη της Εικόνας 5.7 αποτυπώνονται τέσσερα διαγράμματα (ένα για κάθε παράθυρο της αριστερής στήλης) που δείχνουν την γραμμική συσχέτιση μεταξύ των εξαρτημένων Current Values και των ανεξάρτητων τιμών (Predictor Values) με τη μορφή ευθείας γραμμής, όπως αυτή προκύπτει από το μοντέλο γραμμικής παλινδρόμησης. Στην αριστερή πλευρά κάθε διαγράμματος (σήμανση με κόκκινη γραμμή και βέλος) αποτυπώνεται η συσχέτιση μεταξύ των τιμών που προβλέφθηκαν και των μελλοντικών τιμών της εξαρτημένης μεταβλητής. Και στις τέσσερις περιπτώσεις, τα σημεία αυτά πέφτουν πολύ κοντά στην ευθεία γραμμή του μοντέλου γεγονός που αποδεικνύει την ορθή υλοποίησή του.

Η ποσοτική προσέγγιση της απόκλισης μεταξύ των προβλεπόμενων μελλοντικών τιμών και των πραγματικών μελλοντικών γίνεται με τη χρήση του μέσου τετραγωνικού σφάλματος για είκοσι διαφορετικά παράθυρα. Το μέσο τετραγωνικό σφάλμα (Mean Squared Error) [16] υπολογίζεται με την εξίσωση: $\frac{1}{n}\sum(actual - forecast)^2$ για κάθε διαφορετικό παράθυρο. Η μέση τιμή του MSE για τα είκοσι παράθυρα ισούται με 0.4462. Όσο πιο μικρή είναι η μέση τιμή του MSE τόσο πιο ιδανική είναι η ευθεία γραμμικής παλινδρόμησης που έχει χρησιμοποιηθεί στην πρόβλεψη. Συνεπώς το συγκεκριμένο αποτέλεσμα είναι αρκετά θετικό.

Η αποδοτικότητα του μοντέλου γραμμικής παλινδρόμησης στην ακρίβεια της πρόβλεψης, δεν οφείλεται μόνο στην αλγοριθμική διαμόρφωση αυτού αλλά και στον αλγόριθμο RHP του LSH, ο οποίος ανταποκρίνεται βέλτιστα στο κύριο ζητούμενο της εργασίας, το οποίο αποσκοπεί στην ορθή και όσο το δυνατόν καλύτερη ομαδοποίηση των παραθύρων ανάλογα με τον βαθμό ομοιότητας τους.



Εικόνα 5.7: *Διαγράμματα Πρόβλεψης για διαφορετικά χρονικά παράθυρα εισόδου*

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα

Συμπερασματικά, αυτή η διπλωματική εργασία είχε ως αποτέλεσμα, σε πρώτο στάδιο, την δημιουργία μιας σύνοψης δεδομένων, στο εσωτερικό του Synopsis Data Engine (SDE) η οποία κατηγοριοποιεί τα δεδομένα εισόδου ανάλογα με την ομοιότητα τους, εφαρμόζοντας μια συγκεκριμένη μορφή του Locality Sensitive Hashing αλγόριθμου, που ονομάζεται Random Hyperplane Projection. Η αρχιτεκτονική του SDE παρέχει τη δυνατότητα παράλληλης επεξεργασίας των προσομοιώσεων καρκινικών κυττάρων που παράγονται από το PhysiBoSS framework. Το μέγεθος παραλληλισμού εξαρτάται από τον αριθμό των workers που δέχεται ως παράμετρο το σύστημα. Κάθε worker λαμβάνει ένα μέρος των προσομοιώσεων και τις κατακερματίζει στα κατάλληλα buckets που έχουν διαμορφωθεί. Σε δεύτερο στάδιο, ακολουθεί η διαδικασία εντοπισμού των top K όμοιων χρονικών παραθύρων με εκείνα που εισάγονται ως παράμετροι του ερωτήματος. Η διαδικασία αυτή πραγματοποιείται με σκοπό την εξαγωγή κατάλληλων πληροφοριών που θα χρησιμοποιηθούν στην πρόβλεψη της μελλοντικής συμπεριφοράς των ζητούμενων παραθύρων. Η πρόβλεψη πραγματοποιείται μέσω του μαθηματικού μοντέλου γραμμικής παλινδρόμησης το οποίο βρίσκοντας την γραμμική συσχέτιση μεταξύ δύο παραθύρων σε παροντικό χρόνο, μπορεί να υπολογίσει τις μελλοντικές τιμές της εξαρτημένης μεταβλητής (παράθυρο ερωτήματος) από τις μελλοντικές τιμές της ανεξάρτητης.

Η αποδοτικότητα του αλγόριθμου ελέγχθηκε τοπικά και απομακρυσμένα μέσω της χρήσης των μηχανημάτων του cluster του Πολυτεχνείου για διαφορετικούς συνδυασμούς των παραμέτρων του. Κατά τον έλεγχο της κλιμακωσιμότητας του αλγόριθμου προέκυψε η ανάγκη για πολλαπλασιασμό των δεδομένων σε βαθμό τέτοιο ώστε να είναι εμφανείς οι διαφορές στην απόδοση για διαφορετικό μέγεθος εισόδου. Όσον αφορά την αποδοτικότητα του μοντέλου γραμμικής παλινδρόμησης, αυτό κρίνεται αρκετά ακριβές καθώς οι μελλοντικές τιμές της εξαρτημένης μεταβλητής έχουν την επιθυμητή συμπεριφορά στις περισσότερες περιπτώσεις και το μέσο τετραγωνικό σφάλμα βρίσκεται πολύ κοντά στο μηδέν.

6.2 Μελλοντικές Επεκτάσεις

Βελτιώσεις θα μπορούσαν να πραγματοποιηθούν κατά τον κατακερματισμό των δεδομένων στα buckets, μέσω της χρήσης των (ε,δ) guarantees για τον υπολογισμό του αριθμού των τελευταίων. Μέσω της χρήσης της τεχνικής αυτής θα μπορούσε να επιτευχθεί καλύτερη

κατανομή των δεδομένων στο εσωτερικό των buckets.

Στην πρόβλεψη της μελλοντικής συμπεριφοράς των χρονικών παραθύρων υπάρχει η δυνατότητα εφαρμογής μοντέλων διαφορετικών του μαθηματικού μοντέλου γραμμικής παλινδρόμησης. Παρόμοιο μοντέλο εξίσου αποδοτικό είναι το ARIMA (Autoregressive Intergrated Moving Average). Το μοντέλο ARIMA [17] είναι ένα μοντέλο στατιστικής ανάλυσης που χρησιμοποιεί χρονοσειρές δεδομένων με σκοπό την καλύτερη κατανόηση αυτών ή την πρόβλεψη της μελλοντικής τους συμπεριφοράς. Σε πιο εκσυγχρονισμένες εφαρμογές για την πρόβλεψη χρονοσειρών χρησιμοποιούνται υπολογιστικές τεχνολογίες που περιλαμβάνουν την χρήση νευρωνικών δικτύων. Ένα νευρωνικό δίκτυο μπορεί να θεωρηθεί ως ένα δίκτυο «νευρώνων» που είναι οργανωμένα σε επίπεδα. Οι predictors (ή οι είσοδοι) σχηματίζουν το κάτω επίπεδο και οι προβλέψεις (ή οι έξοδοι) το επάνω επίπεδο. Μπορεί επίσης να υπάρχουν ενδιάμεσα στρώματα που περιέχουν «κρυμμένους νευρώνες».

Βιβλιογραφία

[1] https://www.cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html.

[2] *Apache Flink*. https://towardsdatascience.com/an-introduction-to-stream-processing-with-apache-flink-b4acfa58f14d.

[3] *Apache Kafka*. https://kafka.apache.org/intro.

[4] Minos Garofalakis, Johannes Gehrke και Rajeev Rastogi. *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2016.

[5] A. Kontaxakis. *Design and Implementation Of A Distributed Synopsis Data Engine on Apache Flink*. Μεταπτυχιακή διπλωματική εργασία, Technical University Of Crete, 2020.

[6] *Hamming Distance*. https://en.wikipedia.org/wiki/Hamming_distance.

[7] *Cosine Similarity*. https://en.wikipedia.org/wiki/Cosine_similarity.

[8] *Pearson Correlation Coefficient*. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[9] *Time Series Forecasting*. https://dzone.com/articles/time-series-forecasting.

[10] *Timeseries*. http://users.auth.gr/dkugiu/Teach/DataAnalysis/Chp6.pdf.

[11] https://eclass.upatras.gr/modules/document/file.php/ECON1332/Lectures/Lecture%203/06Regression.pdf.

[12] *Linear Regression*. https://www.andrews.edu/~calkins/math/edrm611/edrm06.htm.

[13] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos και Y. Theodoridis. *In-network approximate computation of outliers with quality guarantees*. *Innformation Systems*, 38(8):1285–1308, 2013.

[14] Effrosyni Anesti. *Forecasting Promising Biological Simulations at PhysiBoSS*. Διπλωματική εργασία, Technical University Of Crete, 2020.

[15] Nikolaos Pavlakis. *Scaling out streaming time series analytics on Storm*. Μεταπτυχιακή διπλωματική εργασία, Technical University Of Crete, 2017.

[16] *Mean Squared Error*. https://en.wikipedia.org/wiki/Mean_squared_error.

[17] *ARIMA Statistical Model*. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
ΤΕΙ	Τεχνολογικό Εκπαιδευτικό Ίδρυμα
TCP	Transmission Control Protocol
DSMS	Data Stream Management System

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
ροές δεδομένων	data streams
επεξεργασία συμβάντων ροών	event stream processing
δέσμη/τεμάχιο	batch
διακομιστής	server
εξυπηρετητής	client
νέφος	cloud
σύνοψη	synopsis
παράθυρο	window
πλαίσιο	framework
Απόσταση Χάμινγκ	Hamming Distance