ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εύρεση Συσχετισμένων Γνωρισμάτων σε Σετ Δεδομένων στο Flink.

Finding Correlated Attributes in Datasets at Flink.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΣΤΑΣΙΟΥ ΜΙΧΑΛΗ

Εξεταστική επιτροπή : Αντώνιος Δεληγιαννάκης (επιβλέπων), Καθηγητής

Μίνως Γαροφαλάκης, Καθηγητής

Βασίλειος Σαμολαδάς, Αναπληρωτής Καθηγητή

Περίληψη

Η ταχεία ανάπτυξη της τεχνολογίας επιφέρει τεράστιο όγκο δεδομένων σε καθημερινή βάση. Πρόκειται για δεδομένα, των οποίων ο όγκος είναι δέκα φορές μεγαλύτερος σε σχέση με τον αντίστοιχο πριν από 5 χρόνια. Άρα, δικαίως, η σύγχρονη εποχή χαρακτηρίζεται και ως εποχή των μεγάλων Δεδομένων (Big Data). Η μελέτη αυτών των δεδομένων είναι απαραίτητη τόσο σε ακαδημαϊκό επίπεδο όσο και στις διάφορες βιομηχανίες, αφού μέσω αυτής μπορούν να εξαχθούν συμπεράσματα πολύ πιο εύκολα. Στόχος αυτής της διπλωματικής είναι η εύρεση συσχετισμένων δεδομένων σε πραγματικό χρόνο με σκοπό την εξαγωγή δεδομένων, που μπορούν να χρησιμοποιηθούν για την πρόβλεψη ομοιότητας. Καθώς υπάρχει τεράστιος όγκος δεδομένων, η παρούσα διπλωματική εργασία επεξεργάζεται κατανεμημένα και παράλληλα χιλιάδες ροές δεδομένων με σκοπό την εύρεση των k πιο όμοιων ροών. Ο υπολογισμός ομοιότητας χιλιάδων ροών δεδομένων με μεγάλο μέγεθος θα ήταν πάρα πολύ δαπανηρός, για αυτό έπρεπε να εφαρμοστεί ένας αλγόριθμος για δειγματοληψία των δεδομένων με απώτερο σκοπό την σμίκρυνση τους χωρίς τον κίνδυνο, όμως, απώλειας πληροφορίας. Ο αλγόριθμος αυτός αναπτύχθηκε στην πλατφόρμα διατήρησης συνόψεων δεδομένων (Synopses Data Engine). Η πλατφόρμα αύτη είναι κτισμένη στο framework Apache Flink, και έχει ως κύρια λειτουργία την υποστήριξη διάφορων συνόψεων, οι οποίες λειτουργούν παράλληλα και κατανεμημένα σε πραγματικό χρόνο εκτέλεσης. Έπειτα από την ολοκλήρωση του αλγορίθμου για την σύνοψη, ακολούθησε το μαθηματικό μοντέλο για την εύρεση την ομοιότητας ανάμεσα στις συνόψεις. Το μαθηματικό μοντέλο αποτελείται από το Pearson Correlation συνυπολογίζοντας το τυπικό σφάλμα της δειγματοληψίας χρησιμοποιώντας τον μετασχηματισμό Fisher Z. Για την αποτελεσματικότητα και ορθότητα του συστήματος σχεδιάστηκε, αρχικά, τοπικά όπου έγιναν πειράματα και επαληθεύτηκε η σωστή λειτουργία. Έπειτα, ελέγχθηκε απομακρυσμένα και κατανεμημένα, όπου έγιναν τα τελικά πειράματα, πετυχαίνοντας θετικά και ικανοποιητικά αποτελέσματα.

Abstract

The rapid development of technology has brought about a huge amount of data on a daily basis. This is data whose volume is ten times greater than it was 5 years ago. So the modern era is rightly described as the era of Big Data. The study of this data is essential both at the academic level and in various industries, since by studying this data, one can draw conclusions much easier. The aim of this thesis is to find correlated data in real-time in order to extract data that can be used to predict similarity. Due to the fact that, as mentioned before, there is a huge amount of data this thesis processes distributed and parallel thousands of data streams in order to find the k most similar streams. Computing the similarity of thousands of data streams with a large size would be too costly to implement an algorithm for sampling the data with the ultimate goal of reducing the data size but without the risk of information loss. This algorithm was developed within the Synopses Data Engine. This platform is built on top of the Apache Flink framework, and its main function is to support several synopses running in parallel and distributed in real time. After completing the algorithm for the synopsis, the mathematical model for finding the similarity between the synopses was followed. The mathematical model consists of Pearson Correlation plus the standard error of sampling using the Fisher Z transformation. For the efficiency and correctness of the system was initially designed locally where experiments were conducted and verified. It was then tested remotely and distributed where final experiments were conducted, achieving positive and satisfactory results.

Ευχαριστίες

Θα ήθελα, αρχικά, να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Αντώνιο Δεληγιαννάκη για όλη την υπομονή και κατανόηση κατά την διάρκεια της συνεργασίας μας και την ευκαιρία που μου έδωσε να συνεργαστούμε. Οι συμβουλές του και οι παροτρύνσεις του με βοήθησαν πάρα πολύ στην εκπόνηση της παρούσας εργασίας και θα συνεχίσουν να με βοηθούν και στην επαγγελματική μου σταδιοδρομία μετά τις σπουδές. Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον διδακτορικό φοιτητή Αντώνιο Κονταξάκη, ο όποιος έχει σχεδιάσει την πλατφόρμα διατήρησης συνόψεων δεδομένων (SDE). Με βοήθησε σε τεράστιο βαθμό και μου παρείχε ιδανικές συμβουλές όπου χρειάστηκα. Η κ. Ξένια Αράπη συνέβαλε σημαντικά στη διεξαγωγή πειραμάτων της εργασίας στον cluster του πολυτεχνείου βοηθώντας με να αντιμετωπίσω τυχόν δυσκολίες.

Καταληκτικά, θα ήθελα να ευχαριστήσω την οικογένεια μου και την κοπέλα μου για την υποστήριξη και καθοδήγηση κατά τη διάρκεια των σπουδών μου που χωρίς αυτούς δεν θα ήταν τίποτα εφικτό.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

Εισαγωγή

Η διάθεση πληροφορίας σε δομημένα σετ δεδομένων άγει ευκαιρίες και δυνατότητες για την μηχανική μάθηση μέσω υπολογισμών συσχετίσεων. Η κατάλληλη απάντηση στην τρέχουσα πρόκληση έγκειται στην εύρεση συσχετισμένων σετ δεδομένων για μία συγκεκριμένη είσοδο σε μία πληθώρα δεδομένων. Στην παρούσα διπλωματική εργασία τα δεδομένα αφορούν οντότητες με αριθμητικές στήλες. Για την καλύτερη κατανόηση της χρησιμότητας της διπλωματικής έχουν παρθεί παραδείγματα από έρευνες που διεξάχθηκαν στην Νέα Υόρκη για τον αριθμό των δυστυχημάτων. Όπως διαπιστώθηκε, ο αριθμός των δυστυχημάτων είχε άμεσο αντίκτυπο με την μείωση του ορίου ταχύτητας στους δρόμους. Παρ' όλο τη μείωση ταχύτητας ο αριθμός των ατυχημάτων παρέμενε πολύ ψηλός. Έτσι, οι αναλυτές θέλοντας να κατανοήσουν τα αίτια του μεγάλου αριθμού είχαν σαν αρχική υπόθεση ότι ο αριθμός των ποδηλάτων και ο καιρός έχουν άμεσο αντίκτυπο στον αριθμό ατυχημάτων. Εξετάζοντας τη συσχέτιση με βάση σετ δεδομένων κατάφεραν να επαληθεύσουν την αρχική τους υπόθεσή, ότι τα δεδομένα είχαν άμεσο αντίκτυπο. Στο παράδειγμα το οποίο έχει προαναφερθεί, οι αναλυτές ήξεραν ακριβώς σε ποια σετ δεδομένων να διεκπεραιώσουν την σχετική έρευνα, έτσι ώστε να έχουν τα επιθυμητά αποτελέσματα. Όπως η κακοκαιρία και ο αριθμός ποδηλάτων μπορούν να έχουν αντίκτυπο στα ατυχήματα, έτσι μπορούν και εκατοντάδες άλλοι παράγοντες μπορούν να επηρεάσουν. Τα αίτια αυτά μπορούν να καλυφθούν σε μεγάλο βαθμό μόνο αν γίνει εκτεταμένη έρευνα σε πολλά σετ δεδομένων πράγμα, που είναι αδύνατο να γίνει από τον άνθρωπο. Η έρευνα συχνά μπορεί να αποδειχθεί χρονοβόρα και δαπανηρή διαδικασία, εάν κάποιος δεν γνωρίζει που να ψάξει. Σύμφωνα με μελέτες, βρέθηκε ότι το ¼ του χρόνου που ξοδεύεται από επιστήμονες διοχετεύεται αποκλειστικά στην εξερεύνηση δεδομένων για την πιστοποίηση καταλληλότητας - χρόνος πολύτιμος, που δεν θα ήταν απαραίτητο να ξοδεύεται, εάν η εξερεύνηση γινόταν αυτοματοποιημένα με χρήση αλγόριθμου.

Στην επίτευξη του στόχου αυτού εμφανίζονται 2 προβλήματα: (α) ο τεράστιος όγκος δεδομένων και (β) η χρονοβόρα επεξεργασία για την πιστοποίηση συσχέτισης. Αρχικά, για τον τεράστιο όγκο δεδομένων εφαρμόζονται KMV συνόψεις [1], οι οποίες δημιουργούνται με τη χρήση random hashing methods και τη χρήση Fibonacci Hashing. Επιτυγχάνεται πρόοδος στη μείωση δεδομένων χωρίς τη μεγάλη απώλεια ωφέλιμης πληροφορίας. Έπειτα, ήταν απαραίτητη η ανάπτυξη ενός αλγορίθμου για τη συσχέτιση. Για τον αλγόριθμο χρησιμοποιήθηκε ο τύπος του Pearson Correlation και ο Fisher's Z correlation transform για τον υπολογισμό λάθους.

1.1 Αντικείμενο της Διπλωματικής

Το αντικείμενο της διπλωματικής επικεντρώνεται στην εύρεση συσχετισμένων δεδομένων σε δομές δεδομένων. Όπως έχει προαναφερθεί, μπορεί να εισάγει στους αναλυτές απρόσμενες συσχετίσεις, αφού αναλώνεται τουλάχιστον το 25% του χρόνου τους στην εύρεση των δεδομένων. Επίσης, θα ήταν πιο εύχρηστο με την υποστήριξη ταξινόμησης, δηλαδή την αναφορά σε ποσοστό συσχέτισης των δεδομένων. Για τον λόγο, όμως, ότι τα δεδομένα στις δομές δεδομένων παρουσιάζονται μη ταξινομημένα με κάποιο τρόπο πρέπει να υπολογιστεί μία συσχέτιση. Για τον λόγο αυτό προτείνονται 2 μέθοδοι κατακερματισμού που εισάγουν την έννοιά της συσχέτισης μεταξύ των δεδομένων. Με την 1η μέθοδο κατακερματισμού προσφέρεται η δυνατότητα στοίχισης των δεδομένων σε διαφορετικά dataset που συσχετίζονται με το ίδιο κλειδί και με την 2η μέθοδο κατακερματισμού προσφέρεται ίδιος τρόπος επιλογής στοιχείων κατά την δειγματοληψία. Με τη διατήρηση της αριθμητικής τιμής των δεδομένων μπορεί επίσης να υπολογιστεί η συσχέτιση μεταξύ των στηλών κατά τον υπολογισμό της συσχέτισης. Προκειμένου να μειωθεί ο τεράστιος όγκος δεδομένων μπορεί να εφαρμοστεί μία μέθοδος δειγματοληψίας που με τη χρήση των παραπάνω μεθόδων κατακερματισμού θα παρέχει ελάχιστη απώλεια πληροφορίας. Όσο αφορά την υλοποίηση της κατανεμημένης λειτουργίας, έγινε χρήση της μηχανής συνόψεων SDE [2] και του Apache Kafka. Επιλέχθηκε η χρήση του SDE για τον λόγο ότι η υλοποίηση του βασίζεται στο framework Apache Flink και έχει επιδείξει κλιμακωσιμότητα σε πολύ μεγάλο αριθμό παραλληλισμού. Με την χρήση του Flink μπορεί να αντιμετωπιστεί το πρόβλημα της χρονικής πολυπλοκότητας, καθώς τα δεδομένα μπορούν να επεξεργάζονται παράλληλα και κατανεμημένα χωρίζοντας τα δεδομένα σε workers για ταυτόχρονη επεξεργασία. Για τον υπολογισμό της συσχέτισης λόγω της κατανεμημένης υλοποίησης δεν επιλέχθηκαν οι πιο ακριβείς (σε μη κατανεμημένη υλοποίηση) μέθοδοι αλλά η μέθοδος με τον πιο μικρό αριθμό ανταλλαγών δεδομένων ανάμεσα σε workers.

1.2 Οργάνωση Τόμου

Στο Κεφάλαιο 2 γίνεται η παρουσίαση και επεξήγηση όλων των εργαλείων που χρησιμοποιήθηκαν, έτσι ώστε να γίνει πιο κατανοητή η λειτουργία του συστήματος. Επίσης αναλύεται το θεωρητικό υπόβαθρο της παρούσας εργασίας. Στο Κεφάλαιο 3 αναφέρονται οι σχετικές εργασίες και το τοπικό πείραμα που προσομοιώθηκε από την ερευνητική μελέτη. Στο Κεφάλαιο 4 αναλύεται η ανάπτυξη του αλγορίθμου και ο τρόπος εφαρμογής του στην μηχανή συνόψεων. Στο Κεφάλαιο 5 παραθέτονται τα αποτελέσματα που έχουν εξαχθεί, στα πειράματα που διεξάχθηκαν στο εργαστήριο της σχολής σε κατανεμημένο περιβάλλον. Τέλος, στο Κεφάλαιο 6 αναφέρονται τα συμπεράσματα και οι μελλοντικές επεκτάσεις που μπορούν να εφαρμοστούν.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Ροές Δεδομένων (Data Streams)

Η ροή δεδομένων χαρακτηρίζεται ως η συνεχής μεταφορά δεδομένων με υψηλό ρυθμό ταχύτητας. Ως επί των πλείστων, οι ροές δεδομένων συλλέγουν δεδομένα σε πραγματικό χρόνο από χιλιάδες πηγές [3]. Μετά την συγκέντρωση των δεδομένων αποστέλνονται, συνήθως, σε μικρότερου μεγέθους συστάδες ταυτόχρονα. Αρχικά, τα δεδομένα υποβάλλονταν σε ομαδική επεξεργασία (batch processing), την οποία υποστηρίζουν εργαλεία σαν το Apache Spark. Στην μέθοδο ομαδικής επεξεργασίας μεγάλοι όγκοι δεδομένων υποβάλλονταν σε επεξεργασία σε τακτικά χρονικά διαστήματα.

Όπως είναι αντιληπτό, όταν η επεξεργασία απαιτεί τεράστιο όγκο δεδομένων, αυτό είναι καλύτερο να γίνεται με δεδομένα σε μικρότερες ομάδες. Ένα άλλο πολύ σημαντικό προτέρημα των ροών δεδομένων είναι ότι λειτουργούν πολύ καλά όταν ο στόχος είναι η ανίχνευση δειγματοληψίας σε πραγματικό χρόνο ή το φιλτράρισμα, και επιτρέπουν στους αναλυτές να λαμβάνουν χρήσιμες πληροφορίες εν κινήσει. Η ανάλυση ροών δεδομένων παρέχει στους οργανισμούς ορατότητα σε ένα ευρύ φάσμα πελατειακών και επιχειρηματικών δραστηριοτήτων. Με αυτήν την ορατότητα οι επιχειρήσεις και οι αναλυτές μπορούν να έχουν άμεση αντίδραση σε αλλαγές, για να μπορούν να έχουν άμεσα αποτελέσματα. Με τη χρήση event stream processing μπορούν επίσης τα δεδομένα να επεξεργάζονται και να αναλύονται σε μικρά αποσπάσματα και μετά να ταξινομούνται σε μία δομή δεδομένων προσαρμοσμένη από τον αναλυτή για καλύτερη χρήση δεδομένων, που στην περίπτωση αυτή είναι η σύνοψη.

2.2 Apache Kafka

Με την ονομασία Apache Kafka είναι γνωστή η πλατφόρμα λογισμικού για επεξεργασία ροών δεδομένων. Αναπτύχθηκε, αρχικά, από την εταιρεία πίσω από την πλατφόρμα LinkedIn και έπειτα δόθηκε σαν δωρεά στο Ίδρυμα Λογισμικού Apache. Είναι μία πλατφόρμα open source με σκοπό την διαχείριση κατανεμημένων event streaming δεδομένων.

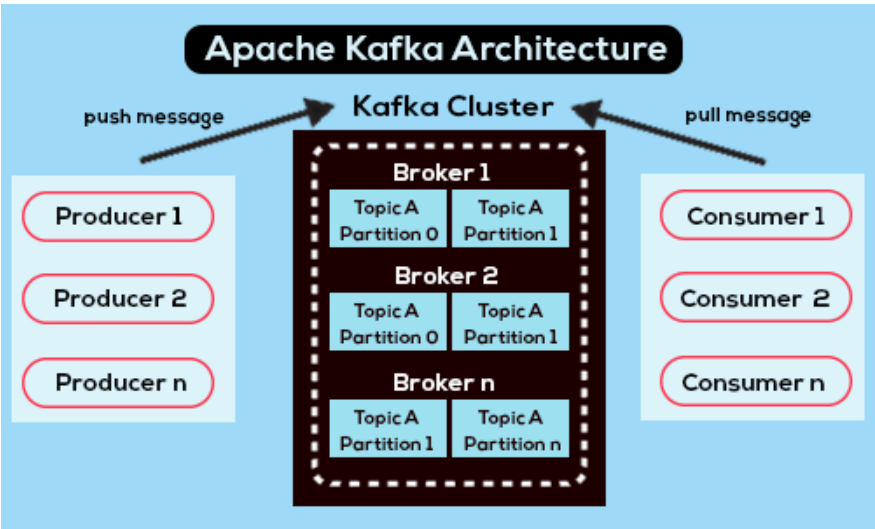
Το Kafka προσφέρει 3 βασικές δυνατότητες (IMB) :

- Προσφέρει τη δυνατότητα σε εφαρμογές να δημοσιεύουν ή να εγγράφονται σε ροές δεδομένων ή συμβάντων.
- Αποθηκεύει τις εγγραφές με ακρίβεια (με τη σειρά την όποια εμφανίστηκαν) με ανεκτικό και ανθεκτικό τρόπο σε σφάλματα, όπως θα αναλυθεί στη συνέχεια.
- Επεξεργάζεται εγγραφές σε πραγματικό χρόνο (data streams).

Αυτό το καθιστά πολύτιμο για εταιρικές λειτουργίες, ώστε να επεξεργάζονται ροές δεδομένων εκμεταλλευόμενες το γρήγορο επεκτάσιμο και ανθεκτικό χαρακτήρα του.

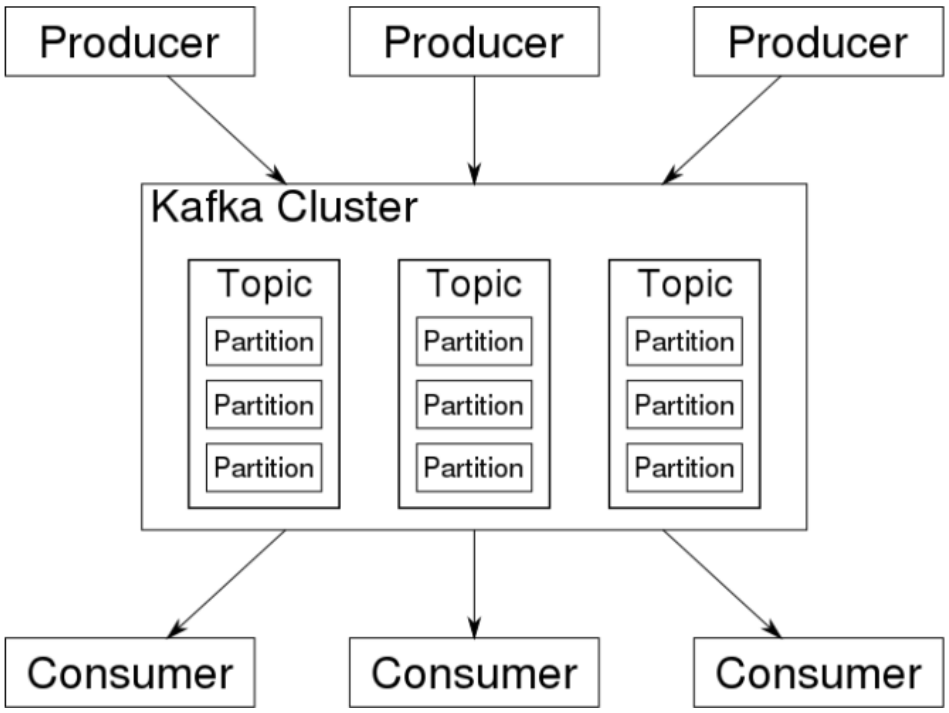
Producer API: Με τη χρήση producer ενεργοποιείται η δυνατότητα μια εφαρμογή να μπορεί να δημοσιεύσει δεδομένα (streams) σε ένα Kafka topic. Topic ονομάζεται ένα αρχείο καταγραφής, το οποίο αποθηκεύει τις εγγραφές με τη σειρά που εμφανίστηκαν. Μετά την εγγραφή δεδομένων σε ένα topic είναι αδύνατη η τροποποίηση ή η διαγραφή. Παραμένει για ένα προκαθορισμένο χρόνο ή μέχρι να τελειώσει ο αποθηκευτικός χώρος.
Consumer API: Ο producer είναι υπεύθυνος για την ενεργοποίηση της λειτουργίας εγγραφής σε ένα ή περισσότερα topic για την εισαγωγή και επεξεργασία δεδομένων (stream) που είναι αποθηκευμένα σε ένα υπάρχον topic. Μπορεί να λειτουργήσει με δεδομένα που εισέρχονται στο topic σε πραγματικό χρόνο ή προϋπάρχοντα δεδομένα.

Είναι αξιοσημείωτο να αναφερθεί, ότι γίνεται χρήση πρωτοκόλλου TCP και για αυτόν τον λόγο παρέχεται υψηλή απόδοση μεταξύ servers και clients. Το Apache Kafka αποτελείται από τον Apache Zookeeper για τη διατήρηση των topics στο σύστημα. Το βασικότερο χαρακτηριστικό του Kafka είναι το Kafka partitioning, κάτι που καθιστά τα topics εξαιρετικά διαχωρίσιμα σε τμήματα ανάμεσα στους Kafka brokers. Οι brokers λαμβάνουν μηνύματα από consumers και τα αποθηκεύουν στον δίσκο με κλειδί και ένα μοναδικό offset. Ένας broker επιτρέπει στους consumers να ανακτούν τα δεδομένα ανά topic, partition και offset. Μπορούν να δημιουργήσουν ένα Kafka cluster μοιράζοντας την πληροφορία μεταξύ τους άμεσα ή έμμεσα χρησιμοποιώντας το Zookeeper.



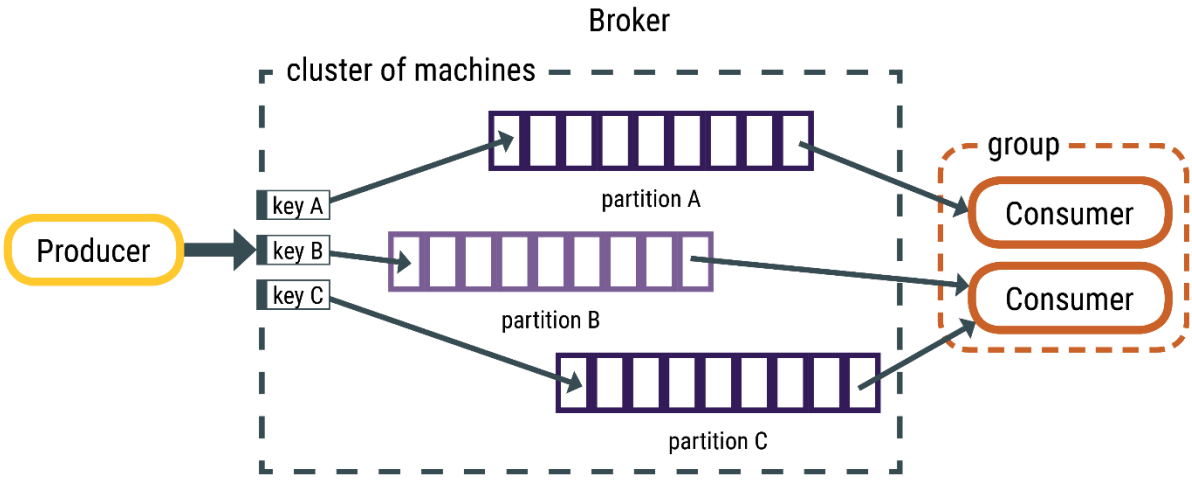
Εικόνα 1 : Αρχιτεκτονική Apache Kafka [4]

Το Kafka cluster αποθηκεύει ροές αρχείων σε κατηγορίες που ονομάζονται topics. Κάθε εγγραφή αποτελείται από ένα κλειδί, μια τιμή και μια χρονική σήμανση.



Εικόνα 2 Kafka Cluster with Consumer and producers

Ένα topic είναι μια κατηγορία ή ένα όνομα μιας ροής δεδομένων, στην όποια εγγράφονται τα δεδομένα. Επιπρόσθετα, τα topics αποτελούνται από συνδρομητές (subscribers). Άρα ένα topic μπορεί να έχει μηδέν, έναν ή πολλούς καταναλωτές (consumers) που παρακολουθούν τα δεδομένα που εγγράφονται σε αυτό.



Εικόνα 3 Περιγραφή partitions και broker

Οι εγγραφές στα partition καταχωρούνται με έναν διαδοχικό αριθμό ταυτότητας που ονομάζεται offset, ο οποίος προσδιορίζει μοναδικά κάθε εγγραφή εντός του διαμερίσματος. Το Kafka cluster διατηρεί όλα τα δημοσιευμένα αρχεία, είτε έχουν καταναλωθεί από τον consumer ή όχι μέχρι να περάσει η περίοδος διατήρησης. Μετά το πέρας της περιόδου τα δεδομένα θα απορριφθούν, για να ελευθερωθεί χώρος.

Πλεονεκτήματα : Πολλαπλοί Παραγωγοί (Producers).

Το Kafka μπορεί να χειριστεί πολλούς consumers, είτε οι πελάτες χρησιμοποιούν πολλά topics, είτε το ίδιο topic. Αυτό το καθιστά ιδανικό για συγκέντρωση δεδομένων από πολλά συστήματα. Για παράδειγμα, ένας ιστότοπος που φιλοξενεί περιεχόμενο σε χρήστες μέσω ενός αριθμού μικροϋπηρεσιών μπορεί να έχει ένα μόνο topic για προβολές σελίδας, στο οποίο μπορούν να δημιουργούν εγγραφές όλες οι υπηρεσίες χρησιμοποιώντας μια κοινή μορφή. Οι εφαρμογές καταναλωτών μπορούν στη συνέχεια να λάβουν μία ροή προβολών σελίδας για όλες τις εφαρμογές στον ιστότοπο χωρίς να χρειάζεται να συντονίσουν την κατανάλωση από πολλά topics. [5]

Υψηλή απόδοση: Το Apache Kafka είναι σε θέση να χειρίζεται δεδομένα υψηλής ταχύτητας και μεγάλου όγκου. Επίσης, μπορεί να υποστηρίξει τη διακίνηση χιλιάδων μηνυμάτων ανά δευτερόλεπτο πράγμα που το καθιστά κατάλληλο για μεγάλες επιχειρήσεις.

Χαμηλή καθυστέρηση: Είναι ικανό να χειριστεί χιλιάδες μηνύματα με πολύ λίγο χρόνο, που απαιτείται από την πλειοψηφία των νέων περιπτώσεων χρήσης.

Ανοχή σε σφάλματα: Ένα από τα μεγαλύτερα πλεονεκτήματα του είναι η ανοχή σφαλμάτων. Υπάρχει μια ικανότητα στο Kafka, να είναι ανθεκτικό σε αστοχίες κόμβου και μηχανήματος εντός cluster.

2.3 Διαχείριση Ροών Δεδομένων

Λόγω του τεράστιου όγκου δεδομένων και της ραγδαίας παραγωγής δεδομένων είναι αναγκαίο τα δεδομένα να επεξεργάζονται και να αναλύονται με υψηλή απόδοση και χαμηλή καθυστέρηση. Έτσι, εισάγεται η έννοια του Συστήματος Διαχείρισης Ροών Δεδομένων (ΣΔΡΔ). Στα ΣΔΡΔ, τεράστιοι όγκοι δεδομένων αποθηκεύονται σε κατάλληλα διαμορφωμένα συστήματα, τέτοια που να υποστηρίζουν την άντληση, την επεξεργασία και τον μετασχηματισμό των δεδομένων καθώς και την αποθήκευση τους σε μορφές που να εξυπηρετούν συγκεκριμένες ανάγκες. Κάποιες από τις ανάγκες μπορεί να είναι ερωτήματα (queries). Υπάρχουν διάφορες μέθοδοι για την μείωση του όγκου πληροφορίας. Για παράδειγμα, υπάρχουν τεχνικές συμπίεσης που προσπαθούν να συνοψίσουν τα δεδομένα με ένα πέρασμα, το οποίο θα περιγραφεί στη συνέχεια.

2.3.1 Συνόψεις

Μία πολύ καλή τεχνική συμπίεσης είναι η δημιουργία σκίτσου (Sketch). Δηλαδή, η περίληψη των δεδομένων με προκαθορισμένο τρόπο από τον χρήστη σύμφωνα με τις απαιτήσεις του. Για την υλοποίηση της παρούσας διπλωματικής ήταν αναγκαία η χρήση μίας μεθόδου σαν αυτής, αφού ο όγκος πληροφορίας που έπρεπε να χρησιμοποιηθεί ήταν πολύ μεγάλος. Η μέθοδος δημιουργίας σύνοψης προσφέρει πολλά πλεονεκτήματα χωρίς το κόστος χαμένης πληροφορίας. Η χαμένη πληροφορία στην σύνοψη εξαρτάται από το μέγεθος της δειγματοληψίας, δηλαδή όσο μεγαλύτερος αριθμός δειγματοληψίας πραγματοποιηθεί τόσο πιο ακριβείς θα είναι οι υπολογισμοί. Ο αλγόριθμος για την κατασκευή της σύνοψης που χρησιμοποιήθηκε αποτελείται από 2 μεθόδους κατακερματισμού. Η ιδέα πίσω από τη χρήση κατακερματισμού στις συνόψεις είναι η εισαγωγή της συσχέτισης, που αυξάνει τις πιθανότητες 2 πλειάδες από διαφορετικές συνόψεις να έχουν τα ίδια κλειδιά. Ένα πολύ βασικό χαρακτηριστικό μίας σύνοψης είναι ότι κτίζεται σε μόνο ένα πέρασμα κατά την άφιξη των δεδομένων (single pass), πράγμα που προσφέρει χαμηλή καθυστέρηση.

2.4 Apache Flink

Πολλά στοιχεία είναι απαραίτητα για ένα ακμάζον οικοσύστημα μεγάλων δεδομένων :

- Ποικιλία δεδομένων: Διαφορετικοί τύποι δεδομένων από πολλαπλές πήγες εισάγονται και εξάγονται.
- Ταχύτητα: Γρήγορη απορρόφηση και επεξεργασία δεδομένων σε πραγματικό χρόνο
- Όγκος: Επεκτάσιμη αποθήκευση και επεξεργασία μεγάλων ποσοτήτων δεδομένων.
- Φτηνή αποθήκευση: Η ικανότητα να αποθηκεύει δεδομένα σε προσιτή τιμή στην αρχική τους μορφή.
- Ευέλικτη επεξεργασία: Δυνατότητα λειτουργίας διαφόρων μηχανών επεξεργασίας στα ίδια δεδομένα.
- Υποστήριξη για streaming analytics: Τα αναλυτικά στοιχεία ροής αναφέρονται στην παροχή χαμηλής καθυστέρησης για την επεξεργασία ροών δεδομένων σε σχεδόν πραγματικό χρόνο.
- Υποστήριξη για σύγχρονες εφαρμογές: Δυνατότητα ενεργοποίησης νεών τύπων εφαρμογών που απαιτούν γρήγορη, ευέλικτη επεξεργασία δεδομένων, όπως εργαλεία BI, συστήματα μηχανικής εκμάθησης, ανάλυσης αρχείων καταγραφής και άλλα.

Οι παραπάνω απαιτήσεις μπορούν να ικανοποιηθούν με το εργαλείο Apache Flink. Το Flink ανήκει στην ίδια εταιρεία με το Apache Kafka. Το Apache Flink είναι μια κατανεμημένη μηχανή επεξεργασίας δεδομένων. Μπορεί να διαχειριστεί τόσο πεπερασμένες (bounded) όσο και μη πεπερασμένες (unbounded) ροές δεδομένων [6]. Η Εικόνα 4 περιγράφει τα bounded και

unbounded streams. Τα bounded streams αποτελούνται από πεπερασμένα δεδομένα, δηλαδή έχουν προκαθορισμένη αρχή και τέλος σε αντίθεση με τα unbounded streams, τα οποία έχουν μόνο αρχή αλλά όχι τέλος. Τα unbounded streams με την πάροδο του χρόνου αυξάνονται σε μεγάλο ρυθμό και οι υπολογισμοί που πραγματοποιούνται σε αυτά είναι συνέχεις και δεν έχουν τέλος σε αντίθεση με τα bounded streams που έχουν προκαθορισμένο μέγεθος, άρα και προκαθορισμένους υπολογισμούς. Το Apache Flink υπερέχει στην επεξεργασία συνόλων για τον λόγο ότι παρέχει ακριβή έλεγχο χρόνου και κατάστασης (stateful), που επιτρέπει κάθε είδους εκτέλεση σε unbounded streams. Όσο αφορά τα bounded streams επεξεργάζονται εσωτερικά από αλγόριθμους και δομές δεδομένων που έχουν σχεδιαστεί ειδικά για τα σύνολα δεδομένων σταθερού μεγέθους αποφέροντας έτσι μέγιστη απόδοση.

Πλεονεκτήματα Stream Processing Έναντι Batch Processing :

- Χαμηλότερη καθυστέρηση: Από τη στιγμή που η επεξεργασία ροών δεδομένων γίνεται σχεδόν σε πραγματικό χρόνο, η συνολική καθυστέρηση είναι πολύ πιο μικρή και προσφέρει τη δυνατότητα για πολλαπλές περιπτώσεις ειδικής χρήσης που χρειάζονται ελέγχους εν κινήσει.
- Ευκαμψία: Τα δεδομένα ροών δεδομένων είναι γενικά πιο ευέλικτα από τα πεπερασμένα δεδομένα, καθώς μια ευρύτερη ποικιλία τελικών εφαρμογών τύπων δεδομένων και μορφών μπορεί εύκολα να χειριστεί. Μπορεί, επίσης, να φιλοξενήσει αλλαγές στις πηγές δεδομένων.
- Χαμηλότερο κόστος: Αφού η επεξεργασία ροών δεδομένων μπορεί να διαχειριστεί άπειρα δεδομένα το συνολικό κόστος είναι λιγότερο, αφού δεν υπάρχει η ανάγκη για αποθήκευση δεδομένων για την επεξεργασία τους.

Το Apache Flink είναι ένα κατανεμημένο σύστημα το οποίο απαιτεί υπολογιστικούς πόρους για την εκτέλεση των εφαρμογών του. Ενσωματώνεται με όλους τους κοινούς cluster recourse managers, για παράδειγμα Hadoop Yarn και Kubernetes, αλλά μπορεί να λειτουργήσει εξίσου καλά και σαν αυτόνομο cluster. Αυτό επιτυγχάνεται με τον τρόπο ανάπτυξης που σχετίζεται με τον διαχειριστή πόρων που επιτρέπουν στο Flink να αλληλοεπιδρά με κάθε διαχειριστή πόρων με τον ιδιωματικό του τρόπο.

Κατά την ανάπτυξη μιας εφαρμογής, το Flink προσδιορίζει αυτόματα τους απαιτούμενους πόρους με βάση τον διαμορφωμένο παραλληλισμό της εφαρμογής και τους ζητά από τον resource manager. Σε περίπτωση αποτυχίας, το Flink αντικαθιστά το αποτυχημένο κοντέινερ ζητώντας νέους πόρους. Όλη η επικοινωνία για την υποβολή ή τον έλεγχο μιας αίτησης γίνεται μέσω κλήσεων REST. Αυτό διευκολύνει την ενσωμάτωση του Flink σε πολλά περιβάλλοντα.

Το Flink έχει σχεδιαστεί να εκτελεί εφαρμογές συνεχούς ροής σε οποιαδήποτε κλίμακα. Οι εφαρμογές παραλληλίζονται σε πιθανώς χιλιάδες εργασίες που διανέμονται και εκτελούνται

ταυτόχρονα σε ένα cluster. Επιπρόσθετα, το Flink διατηρεί πολύ μεγάλη κατάσταση εφαρμογής. Ο ασύγχρονος και incremental checkpointing αλγόριθμος εξασφαλίζει ελάχιστο αντίκτυπο σε καθυστερήσεις επεξεργασίας (processing latencies) και ταυτόχρονα εξασφαλίζει exactly-once state consistency.

Μια από τις βασικότερες λειτουργίες είναι οι τελεστές (operators). Οι τελεστές μετασχηματίζουν μία ή περισσότερες ροές δεδομένων σε μία νέα ροή δεδομένων. Τα προγράμματα μπορούν να συνδυάσουν πολλαπλούς μετασχηματισμούς. Κάποιες από τις λειτουργίες που προσφέρουν οι τελεστές του Flink και χρησιμοποιήθηκαν στην διπλωματική είναι το φιλτράρισμα, mapping, grouping, aggregating επάνω σε ροές ή σύνολα δεδομένων.

Περιγραφή Τελεστών:

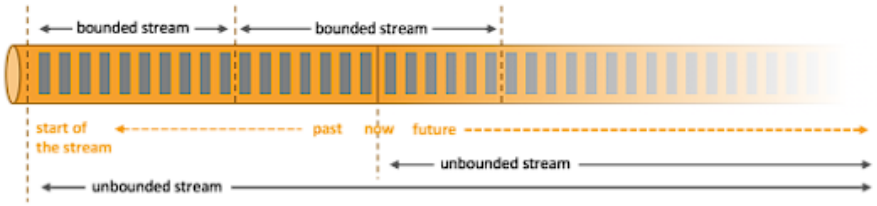
Αρχικά, ο τελεστής **Map** έχει σαν είσοδο ένα δεδομένο και παράγει σαν έξοδο ακριβώς ένα δεδομένο. Μπορεί να χρησιμοποιηθεί για μαθηματικές πράξεις σε κάποια συγκεκριμένη είσοδο. Αντιθέτως, τα **FlatMap** είναι συναρτήσεις που έχουν σαν είσοδο δεδομένα και τα μετασχηματίζουν σε μηδέν, ένα ή περισσότερα δεδομένα. Ένα πολύ απλό παράδειγμα είναι το split δεδομένων. Ο τελεστής **filter**, όπως είναι ξεκάθαρο και από το όνομα του, είναι υπεύθυνος για το φιλτράρισμα δεδομένων. Η λειτουργία του επιτυγχάνεται υπολογίζοντας μια Boolean συνάρτηση για κάθε δεδομένο διατηρώντας μόνο όσα η συνάρτηση επιστρέψει true. Επίσης, ο τελεστής **KeyBy** έχει σαν βασική λειτουργία να διαχωρίζει μία ροή δεδομένων σε περισσότερες ροές δεδομένων. Όλες οι εγγραφές με το ίδιο κλειδί μπαίνουν στο ίδιο partition και εσωτερικά ο τελεστής υλοποιείται με hash partitioning. Τέλος, ίσως ο σημαντικότερος τελεστής όσο αφορά τις κατανεμημένες εφαρμογές είναι ο τελεστής **Reduce**. Ο τελεστής reduce συνδυάζει το παρόν δεδομένο με το προηγούμενο reduced value και παράγει μία νέα τιμή. Για παράδειγμα, μία συνάρτηση, η οποία έχει σαν είσοδο δύο τιμές και σαν έξοδο το άθροισμα των δύο τιμών.

Τα προγράμματα αναπαρίστανται από ένα γράφημα ροής δεδομένων, που εκτελείται στον πυρήνα του Flink. Η Εικόνα 5 αναπαριστά ένα παράδειγμα του γραφήματος ροής δεδομένων για την εφαρμογή του Flink. Επίσης, ένας πολύ σημαντικός τελεστής που χρησιμοποιήθηκε σε μεγάλο βαθμό είναι ο CoFlatMap, ο οποίος αποτελείται από 2 FlatMap, που έχουν κοινή μνήμη. Χρησιμοποιείται για να ενώσει 2 streams, τα οποία χρησιμοποιούν διαφορετικούς map τελεστές.

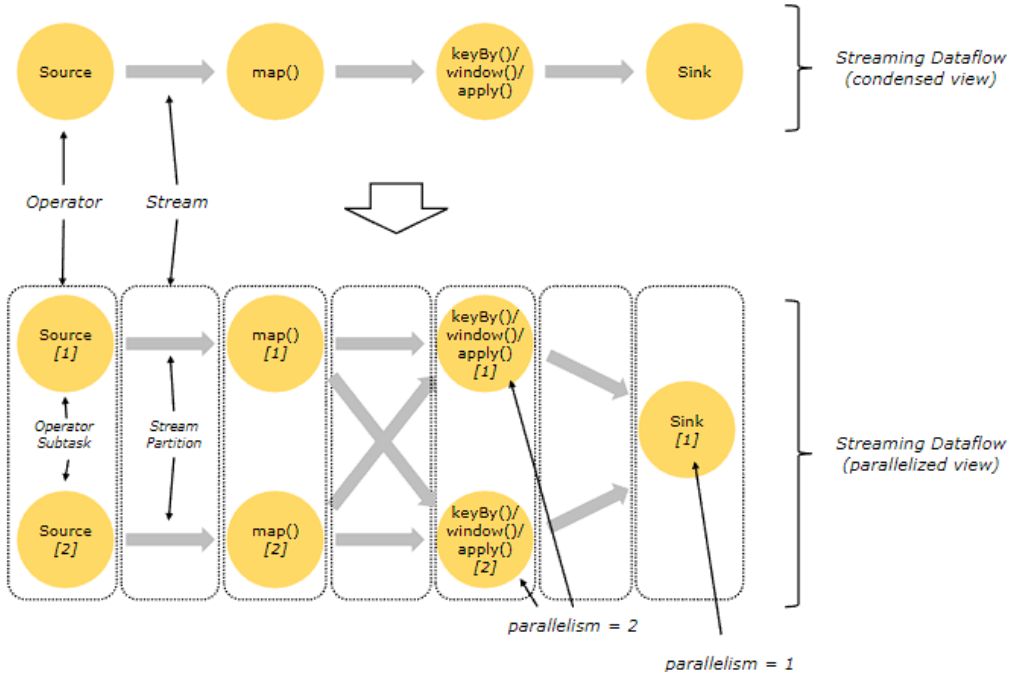
Επίσης παρέχονται μερικά βασικά data sources και sinks, τα οποία είναι build-in στο Flink. Τα προκαθορισμένα data sources περιέχουν την δυνατότητα διαβάσματος από αρχεία, sockets και directories. Τα προκαθορισμένα data sinks υποστηρίζουν γράψιμο σε αρχεία, stdout, stderr και σε sockets. Στην διπλωματική δεν χρησιμοποιήθηκαν κάποια από τα προκαθορισμένα, καθώς χρησιμοποιήθηκε το Apache Kafka για την εισαγωγή και εξαγωγή δεδομένων.

Ο ασύγχρονος και incremental checkpointing αλγόριθμος, που αναφέρθηκε προηγουμένως, λειτουργεί με το να θέτει σταθερά snapshots από τα κατανεμημένα δεδομένα

του operator state. Αυτά τα snapshots συμπεριφέρονται σαν σταθερά checkpoints και σε περίπτωση δυσλειτουργίας του συστήματος επιστρέφουν στο προηγούμενο checkpoint.



Εικόνα 4: Bounded και Unbounded Streams [7]



Εικόνα 5: Γράφημα Ροής Δεδομένων στο Apache Flink

Μία εφαρμογή κτισμένη επάνω στο Apache Flink μπορεί να αξιοποιήσει σχεδόν όλους τους πόρους του επεξεργαστή και της μνήμης. Διανέμεται σε δεκάδες χιλιάδες διεργασίες που διανέμονται και εκτελούνται ταυτόχρονα σε έναν cluster. Επιπρόσθετα, οι χρήστες του

ανέφεραν εντυπωσιακή επεκτασιμότητα για τις εφαρμογές, όπως για παράδειγμα εφαρμογές που εκτελούνται σε πολλούς πυρήνες και επεξεργάζονται terabytes δεδομένων (Alibaba).

Πλεονεκτήματα Apache Flink:

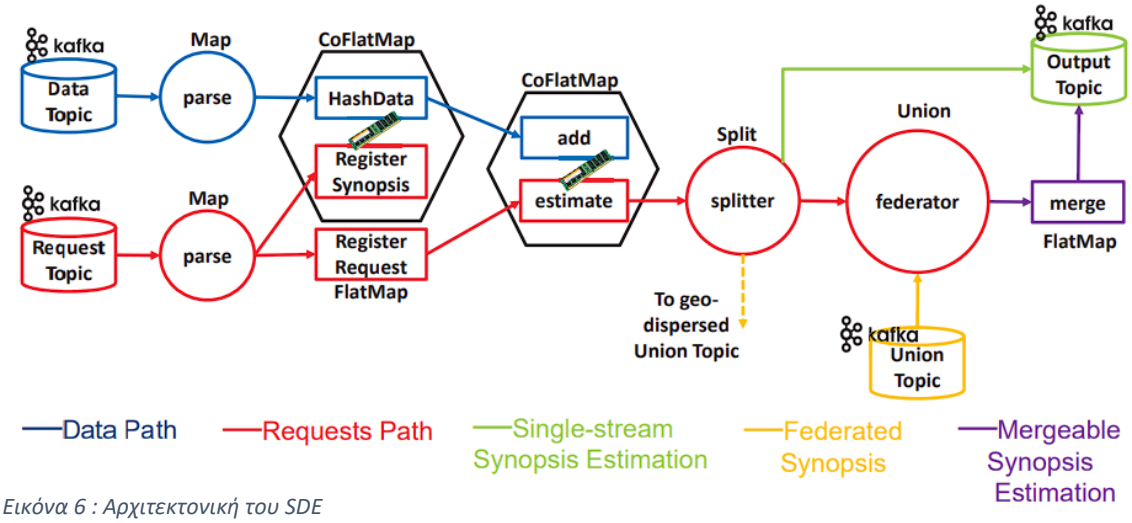
- Stateful επεξεργασίας ροής: Επιτρέπει στους χρήστες να ορίζουν κατανεμημένους υπολογισμούς σε συνεχείς ροές δεδομένων. Αυτό επιτρέπει σύνθετες αναλύσεις επεξεργασίας συμβάντων, όπως windowed joins, aggregation, pattern matching, fraud detection.
- Stream και batch processing: Το Apache Flink είναι μια εξαιρετική επιλογή για εφαρμογές ροής σε πραγματικό χρόνο, που πρέπει να επεξεργάζονται από κοινού stream και batch δεδομένα.
- Επεκτασιμότητα: Μπορεί να κλιμακώσει έως και χιλιάδες κόμβους με ελάχιστη καθυστέρηση και απώλεια απόδοσης, λόγω των αποτελεσματικών πρωτοκόλλων επικοινωνίας δικτύου του.
- Υποστήριξη API : Υποστηρίζει API για τη σύνταξη εφαρμογών ροής σε Java και Scala.
- Fault tolerance και availability: Το framework του Apache Flink είναι κτισμένο πάνω από το ισχυρό Akka actor, το οποίο παρέχει εγγενή ανοχή σφαλμάτων. Η μηχανή κατανεμημένου χρόνου εκτέλεσης του Apache Flink εξασφαλίζει υψηλή διαθεσιμότητα και ανοχή σε σφάλματα επεξεργασίας ροής καθιστώντας το εξαιρετική επιλογή για εφαρμογές mission-critical.
- Χαμηλή καθυστέρηση και υψηλή απόδοση: Η αστραπιαία ταχύτητα και υψηλή απόδοση του Apache Flink το καθιστούν ιδανικό για αναλύσεις σε πραγματικό χρόνο ή για επεξεργασία δεδομένων από πηγές, όπως μετρήσεις αισθητήρων από συσκευές IoT, αρχεία καταγραφής μηχανημάτων, δεδομένα συναλλαγών πιστωτικών καρτών ή ροές κλικ ιστού.
- Ευέλικτες μορφές δεδομένων: Η διαχείριση δεδομένων σε διαφορετικές μορφές μπορεί να είναι δύσκολή, αλλά το Apache Flink υποστηρίζει διαφορετικές μορφές δεδομένων, όπως CSV, Apache Avro, JSON.
- Βελτιστοποίηση: Το εργαλείο βελτιστοποίησης ερωτημάτων παρέχει πολλές ενσωματωμένες βελτιστοποιήσεις, όπως διοχέτευση και συγχώνευση δεδομένων για μείωση του χρόνου υπολογισμού. Το Flink Table API και SQL παρέχουν πρόσθετες βελτιστοποιήσεις ερωτημάτων και συντονισμένες υλοποιήσεις χειριστή.
- Ευέλικτο Deployment: Το Apache Flink προσφέρει υποστήριξη πρώτης κατηγορίας για αρκετούς κοινούς clustered deployment targets, όπως για παράδειγμα YARN, Apache Mesos, Docker και Kubernetes. Μπορεί, επίσης, να ρυθμιστεί, ώστε να εκτελείται σαν ένας αυτόνομος cluster.

2.5 Μηχανή Δεδομένων Συνόψεων

Το SDE (Synopses Data Engine) είναι μια μηχανή παράλληλης επεξεργασίας δεδομένων τόσο για unbounded streams όσο και για bounded streams χτισμένη επάνω στο Apache Flink. Είναι μια εφαρμογή ΔΣΡΔ. Λόγω του ότι η αρχιτεκτονική και η λειτουργικότητα του προγράμματος είναι κτισμένη πάνω στο Apache Flink, προσφέρει στον χρήστη αρκετά προτερήματα. Κάποια από αυτά είναι :

1) Δυνατότητα ενσωμάτωσης και μεταφοράς νέων συνόψεων εν κινήσει.
2) Επαναχρησιμοποίηση των ήδη εγγεγραμμένων συνόψεων στο πρόγραμμα.
3) Η ταυτόχρονη διατήρηση εκατοντάδων συνόψεων διαφόρων τύπων για μεγάλο όγκο ροών δεδομένων.
4) Συνόψισή δεδομένων για Big Data streams.
5) Βελτιστοποίηση εκτέλεσης ροής εργασίας.

Είναι δυνατή η οριζόντια επεκτασιμότητά του, αρά το SDE είναι κατάλληλο για την ανάλυση δεδομένων μεγάλης κλίμακας. Αυτό επιτυγχάνεται, αφού οι υπολογισμοί δεν κατανέμονται μόνο σε προκαθορισμένες μονάδες επεξεργασίας, αλλά σε ένα cluster. Επίσης, αξιοσημείωτο είναι και η παροχή κάθετης και ενοποιημένη επεκτασιμότητας. Η κάθετη επεκτασιμότητα είναι υπεύθυνη για την κλιμάκωση του υπολογισμού σε πολύ μεγάλο αριθμό επεξεργασμένων ροών. Η σύνοψη που υλοποιήθηκε για την παρούσα πτυχιακή ενσωματώθηκε στο SDE και προσαρμόστηκε με σκοπό την βέλτιστη και ταχύτερη απάντηση ερωτημάτων. Απαραίτητο για τη χρήση του ήταν η κατανόηση της λειτουργικότητας του, η οποία αποτελείται, αρχικά, από 2 CoFlatMap για την εισαγωγή ερωτήματος και δεδομένων, όπως φαίνεται στην εικόνα 6.



Εικόνα 6 : Αρχιτεκτονική του SDE

2.6 KVM Συνόψεις

Οι συνόψεις συσχετίσεων απορρίπτουν τα όρια αφελών προσεγγίσεων ενεργοποιώντας την ανακατασκευή ενός ομοιόμορφου τυχαίου δείγματος. Η μέθοδος αύτη με χρήση μεθόδων κατακερματισμού, που επιλέγουν ένα προκαθορισμένο παραμετροποιημένο αριθμό πλειάδων <k,Xk> από έναν πίνακα Tx = <Kx,X>, χτίζει μία σύνοψη, η οποία επιτρέπει σε δεδομένα από διαφορετικούς πίνακες να είναι στοιχισμένα και να εφαρμοστούν σε αυτά υπολογισμοί συσχετίσεων.

$\mathcal{T}_X$

K_X	X
2021-01	6.0
2021-02	4.0
2021-03	2.0
2021-04	3.0
2021-05	0.5
2021-06	4.0
2021-07	2.0

$\mathcal{T}_Y$

K_Y	Y
2021-01	5.5
2021-01	4.5
2021-02	3.9
2021-02	2.0
2021-03	4.0
2021-03	1.0
2021-04	4.0

$\mathcal{T}_{X \bowtie Y}$

$K_{X \bowtie Y}$	$X_{X \bowtie Y}$	$Y_{X \bowtie Y}$
2021-04	3.0	4.0
2021-03	2.0	2.5
2021-02	4.0	3.0
2021-01	6.0	5.0

Εικόνα 7 Πλήρες ένωση μεταξύ των πινάκων Tx και Ty

Στην εικόνα 7 μπορούμε να διακρίνουμε ένα πλήρες join χωρίς την χρήση συνόψεων μεταξύ 2 πινάκων Tx Και Ty οπού Kx και Ky η τιμή της στήλης και X,Y η αριθμητική τιμή. Σε αυτή τη περίπτωση όμως το μέγεθος των πινάκων είναι αρκετά μικρό. Για μεγάλους πίνακες το κόστος ενός πλήρες join θα κόστιζε πάρα πολύ. Αν χρησιμοποιήσουμε την τεχνική του Paper για της KMVSynopsis το κόστος μειώνεται πολύ επειδή πλέον εξαρτάτε από το μέγεθος των συνόψεων.

2.6.1 Σύνοψη

Για την κατασκευή της σύνοψης L<Kx,X> γίνεται χρήση 2 διαφορετικών μεθόδων κατακερματισμού. Η πρώτη μέθοδος, h, είναι μια μέθοδος που χωρίς συγκρούσεις, τυχαία και ομοιόμορφα αντιστοιχεί τις τιμές των κλειδιών k ∈ Kx σε μοναδικούς ακέραιους αριθμούς. Δεδομένου ότι οι ακέραιοι h(k) είναι μοναδικοί χρησιμοποιούνται ως αναγνωριστικά κλειδιά πλειάδας στη σύνοψη L<Kx,X>. Όσο αφορά την 2^{η} μέθοδο κατακερματισμού hu αντιστοιχεί τους μοναδικούς χαρακτηριστικούς αριθμούς h(k) σε πραγματικούς αριθμούς στο εύρος τιμών [0,1],

ομοιόμορφα και τυχαία. Η 2^{η} μέθοδος κατακερματισμού hu είναι ο παράγοντας απόφασης στην αφαίρεση και προσθήκη των πλειάδων κατά την κατασκευή της σύνοψης. Πιο συγκεκριμένα, οι n πιο μικρές τιμές hu εμπεριέχονται στη σύνοψη, περισσότερα σε αυτό θα αναφερθούν αργότερα. Έχοντας αναφέρει τα προηγούμενα μπορούμε να συντάξουμε ένα ορισμό, ο οποίος μας οδηγεί στην κατασκευή της σύνοψης.

Ορισμός 2. *Επιλέγουμε n δείγματα από ζευγάρια <h(k),Xk> με μικρότερες τιμές του hu(k) π.χ. L<Kx,X> = {<h(k), Xk> : k ϵ min(k,hu(k))}, όπου min είναι μία συνάρτηση που επιστρέφει ένα σετ που περιέχει τα κλειδιά k με τις μικρότερες τιμές του hu(k).[8]*

2.6.2 Διαχείριση Επαναλαμβανόμενων Κλειδιών

Η μέθοδος που παρουσιάστηκε πιο πάνω προϋποθέτει ότι τα κλειδιά είναι μοναδικά σε κάθε πίνακα και περιγράφουν την κάθε γραμμή ξεχωριστά. Αντιθέτως, όμως, τα δεδομένα σε πραγματικά σύνολα δεδομένων σε μεγάλο βαθμό εμφανίζονται επανειλημμένα σε κατηγορικές τιμές. Σε τέτοιες περιπτώσεις υπάρχει μόνο ένα ζεύγος τιμών σχετικό με κάθε μοναδικό κλειδί k. Καθώς, η συσχέτιση ορίζεται μόνο για ζεύγη τιμών οι εφαρμογές που χρησιμοποιούν υπολογισμό της συσχέτισης προσπαθούν με τη χρήση aggregation function να μετατρέψουν τις πολλαπλές τιμές με το κοινό κλειδί σε ένα ζεύγος. Συνήθως, κατορθώνεται με τη χρήση συναρτήσεων άθροισης, συναρτήσεις εύρεσης μεγαλύτερου, μικρότερου κλπ. Στην παρούσα εργασία αποφασίστηκε να χρησιμοποιηθεί η συνάρτηση για την εύρεση μέσου όρου (mean). Τα επαναλαμβανόμενα κλειδιά μπορούν να διαχειριστούν κατά τη δημιουργία της σύνοψης σε πραγματικό χρόνο. Όταν ένα κλειδί k, που προϋπάρχει στη σύνοψη ξαναβρεθεί σε χρόνο t μία aggregate μέθοδος f μπορεί να εφαρμοστεί για τον υπολογισμό της νέας τιμής για χρόνο t κάνοντας aggregate το υπάρχον xk^{t-1} με το νέο Xk. Εφόσον η μέθοδος υπολογισμού μέσου όρου μπορεί να εφαρμοστεί σε streaming, τότε το κτίσιμο της σύνοψης είναι εφικτό με μόνο ένα πέρασμα από τα δεδομένα. Η μέθοδος είναι ανεξάρτητη από τέτοιες μεθόδους aggregation και μπορεί ευκολά να επεκταθεί παίρνοντας σαν όρισμα μία ή περισσότερες μεθόδους ανάλογα με τις ανάγκες του συστήματος. Η παραπάνω επεξήγηση αφορούσε το κτίσιμο σύνοψης από ένα δυαδικό πίνακα για σκοπούς απλότητας, αλλά επεκτείνεται σε πίνακες με πολλαπλές στήλες. Για παράδειγμα αν ο πίνακας περιέχει πολλαπλές στήλες, TX Z = {KX Z ,X,Z}, η σύνοψη θα επεκταθεί σε : L⟨KX Z ,X,Z ⟩ = {⟨h(k), xk , zk ⟩ : k ∈ min(k,hu (k))}.

$\mathcal{T}_X$

K_X	X
2021-01	6.0
2021-02	4.0
2021-03	2.0
2021-04	3.0
2021-05	0.5
2021-06	4.0
2021-07	2.0

$\mathcal{T}_Y$

K_Y	Y
2021-01	5.5
2021-01	4.5
2021-02	3.9
2021-02	2.0
2021-03	4.0
2021-03	1.0
2021-04	4.0

$L_{\langle K_X, X \rangle}$

$h(k)$	$h_u(k)$	x_k
bac52e98	0.48	2.0
16dab449	0.34	2.0
26f79756	0.47	3.0
4da33cf5	0.34	6.0

$L_{\langle K_Y, Y \rangle}$

$h(k)$	$h_u(k)$	y_k
16dab449	0.34	2.5
bd5a7c1f	0.89	3.0
26f79756	0.47	4.0
4da33cf5	0.34	5.0

$L_{\langle X \bowtie Y \rangle}$

$h(k)$	x_k	y_k
16dab449	2.0	2.5
26f79756	3.0	4.0
4da33cf5	6.0	5.0

Εικόνα 8 Οι πίνακες L<Kx,X> και L<Ky,Y> αναπαριστούν συνόψεις συσχέτισης από τους πίνακες Tx και Tx. Οι συνόψεις έχουν μέγεθος 4 και εφαρμόζεται στα κλειδιά aggregation μέσου όρου

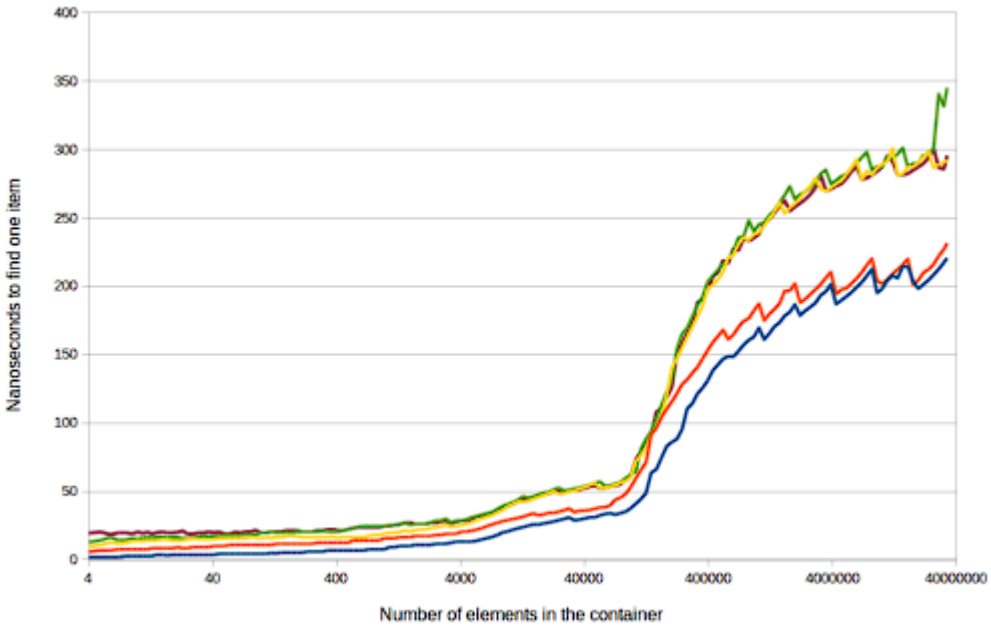
Στο παράδειγμα της εικόνας 8 οι πίνακες L<Kx,X> και L<Ky,Y> αναπαριστούν συνόψεις συσχέτισης από τους πίνακες Tx και Ty με μέγεθος 4. Η τιμή h(k) και hu(k) είναι τα αποτελέσματα των μεθόδων κατακερματισμού που θα περιγράφουν στην συνέχεια. Η τιμές Xk και Yk είναι η numeric τιμές των πινάκων όπου και διατηρούνται στην αρχική τους μορφή για του υπολογισμούς συσχετίσεων. Στα δεξιά είναι το αποτέλεσμα join μεταξύ των 2 συνόψεων.

2.6.3 Μέθοδοι Κατακερματισμού

Για το κτίσιμο της συσχέτισης ανάμεσα στις συνόψεις χρησιμοποιήθηκαν 2 μέθοδοι κατακερματισμού. Για τον υπολογισμό της μεταβλητής h χρησιμοποιήθηκε η μέθοδος κατακερματισμού 32 bits MurmurHash3, η οποία έδειξε να συμπεριφέρεται με πραγματικές τυχαίες μεθόδους κατακερματισμού. Η μέθοδος κατακερματισμού MurmurHash3 υπόσχεται μηδενικές συγκρούσεις και παρέχει πολύ καλή στατιστική κατανομή. Είναι μια non-cryptographic μέθοδος κατακερματισμού ιδανική για γενικό hash-based lookup. Το όνομα προέρχεται από την λειτουργία του, που αποτελείται από 2 απλές διεργασίες: τον πολλαπλασιασμό (MU) και την περιστροφή (R) που χρησιμοποιούνται στο εσωτερικό loop. Αντιθέτως, με τους cryptographic μεθόδους κατακερματισμού, δεν είναι σχεδιασμένο να είναι

δύσκολο για reverse by an adversary. Για την παρούσα διπλωματική χρησιμοποιήθηκε η μέθοδος MurmurHash3_x86_32 [9] .

Για τον υπολογισμό της μεταβλητής hu χρησιμοποιήθηκε η μέθοδος κατακερματισμού Fibonacci Hashing. Με απλά λόγια είναι μια πολλαπλασιαστική μέθοδος κρύβοντας, όμως, τεράστια προοπτική. Η συγκεκριμένη μέθοδος είναι ευρέως διαδεδομένη και όπως αποδείχτηκε με πειράματα μπορεί να αποφέρει εξαιρετικά αποτελέσματα [10].



Εικόνα 9 Benchmark Αναζήτηση σε Hash Table

Στην εικόνα 9 συγκρίνονται διάφοροι τρόποι αναζήτησης. Στον Χ άξονα είναι το μέγεθος του πίνακα και στον Υ άξονα ο χρόνος να βρεθεί το αντικείμενο. Η μέθοδος του πειράματος είναι ένα απλός επαναληπτικός βρόγχος, ο όποιος καλεί μία μέθοδο αναζήτησης. Για το τελικό αποτέλεσμα στο τέλος διαιρείται ο χρόνος στο loop με τον αριθμό επαναλήψεων. Όπως διαπιστώνεται από το πείραμα με την μπλε γραμμή, η μέθοδος κατακερματισμού Fibonacci είναι η βέλτιστη. Η συγκεκριμένη μέθοδος είναι επίσης και εξαιρετικά γρήγορη.

Η μέθοδος Fibonacci είναι κτισμένη πάνω στο golden ratio φ. Το φ ισούται με την αριθμό 1.6180399, ο οποίος χρησιμοποιήθηκε σε πολλούς τομείς της επιστήμης. Η ιδιότητα του golden ration είναι ότι μπορείς να το υποδιαιρέσεις σε οποιοδήποτε εύρος περίπου ομοιόμορφα, χωρίς ποτέ να επιστρέψει στην αρχική θέση. Για παράδειγμα, σε ένα hash table 1024 θέσεων

χρειάζεται να αντιστοιχηθεί μία αυθαίρετα μεγάλη τιμή κατακερματισμού σε αυτό το εύρος. Το πρώτο βήμα είναι να αντιστοιχηθεί σε όλο το εύρος των 64 bit αριθμών πολλαπλασιάζοντας τον αριθμό κατακερματισμού με τον αριθμό.

Ο πολλαπλασιασμός με αυτόν τον αριθμό θα προκαλέσει υπερχείλιση, αλλά θα «τυλίξει» όλο το εύρος των 64 bit σε ένα μοτίβο δίνοντας μία ομοιόμορφη κατανομή.

2.6.4 Κτίσιμο Σύνοψης

Για το κτίσιμο της σύνοψης χρησιμοποιήθηκαν οι 2 πιο πάνω μέθοδοι κατακερματισμού. Με την χρήση της $2^{ης}$ μεθόδου κατακερματισμού hu(k) τα δεδομένα μεταξύ των συνόψεων γίνονται εξαρτημένα καθώς αυξάνεται η πιθανότητα 2 συνόψεις να περιλαμβάνουν το ίδιο κλειδί. Αυτό συμβαίνει για τον λόγο ότι γίνεται αντιστοίχιση στο δεδομένα μεταξύ 0,1 και στην συνέχεια εφαρμόζεται σε αυτά το sampling. Έτσι ένα δείγμα hu(k) που θα περιλαμβάνεται σε μία σύνοψη με μεγάλη πιθανότητα θα εμφανιστεί και σε μία άλλη σύνοψη. Επίσης αφού τα δεδομένα είναι στοιχισμένα με τον ίδιο τρόπο κατακερματισμού κλειδιού μπορεί αν εφαρμοστούν υπολογισμοί συσχετίσεων. Για το κτίσιμο της σύνοψη υλοποιήθηκε ένας αλγόριθμος δένδρου. Συνοπτικά ο αλγόριθμος κάνει μία ανάγνωση των δεδομένων διατηρώντας ένα δένδρο, το όποιο κρατά τα n tuple <h(k),hu(k),xk> με τα ελάχιστα hu(k) δεδομένα. Εάν το μέγεθος της σύνοψης είναι μεγαλύτερο από την χωρητικότητά της, τότε αποφασίζεται εάν θα εισαχθεί το tuple στην σύνοψη σύμφωνα με το μέγεθος του hu και αφαιρείται το μεγαλύτερο. Στην περίπτωση που βρεθεί το ίδιο κλειδί χειρίζεται με μία mean average function. Μετά την κατασκευή της σύνοψης τα μεγέθη hu(k) δεν διατηρούνται καθώς δεν προσφέρουν κάτι στον υπολογισμό της συσχέτισης. Στην συνέχεια, θα αναλυθεί περισσότερο ο αλγόριθμος. Τέλος, σε συνδυασμό με τις μεθόδους κατακερματισμού χρησιμοποιείται και η σχέση για τον υπολογισμό της συσχέτισης Pearson Correlation (εξίσωση 2) και η σχέση για την κατάταξη στηλών (εξίσωση 3).

2.6.5 Εκτίμηση Συσχετισμένων σετ Μέσω Συνόψεων

2.6.5.1 Τύπος για Υπολογισμό Συσχέτισης Ανάμεσα στα Διανύσματα.

Το πολύ-μελετημένο πρόβλημα για την εύρεση συσχέτισης μεταξύ δύο διανυσμάτων έχει μελετηθεί και αναλυθεί σε μεγάλο βαθμό τον τελευταίο αιώνα. Η μέθοδος Pearson's Correlation Coefficient είναι μια από τις παλαιότερες και ευρέως χρησιμοποιημένες μεθόδους για υπολογισμό της συσχέτισης. Τα αποτελέσματα της εξίσωσης είναι ανάμεσα στο διάστημα (-1,1) σε όλες τις περιπτώσεις, αφού αφορά μία κανονικοποιημένη μέτρηση της συν-διακύμανσης.

$$\rho_{XY} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Εξίσωση 1

Όταν εφαρμόζεται σε πληθυσμό, η τιμή του Pearson correlation coefficient συχνά αναφέρεται ως *ρ*. Η τιμή *ρ* μπορεί να υπολογιστεί με ένα πεπερασμένο δείγμα κατανομής X και Y χρησιμοποιώντας την κοινώς αναφερόμενη Pearson's sample correlation (r) :

$$r_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}.$$

Εξίσωση 2

Στην Εξίσωση 2 η μεταβλητή n αναφέρεται στο μέγεθος δειγματοληψίας, οι μεταβλητές xi και yi είναι μεμονωμένα δείγματα. Τα διανύσματα $\bar{x}$ και $\bar{y}$ είναι τα μέσα των δειγμάτων X και Y.

2.6.5.2 Βαθμολογία Συσχέτισης Μεταξύ Στηλών.

Για την υποβολή ερωτημάτων σε μεγάλες συλλογές συνόλων δεδομένων, εστιάζουμε την προσοχή μας σε μία παραλλαγή ερωτημάτων συσχέτισης, που ανακτούν τα αποτελέσματα των k πιο συσχετισμένων διανυσμάτων και μπορούν να οριστούν με τον εξής ορισμό :

Ορισμός 1 (Top-k Join-Correlation Query). *Δεδομένου μίας στήλης Q και μίας στήλης ένωσης KQ από ένα query πίνακα TQ, βρες τους κ πιο συσχετισμένους πίνακες TX σε ένα σύνολο δεδομένων τέτοιο, ώστε ο πίνακας TX να είναι joinable με τον πίνακα TQ στην στήλη KQ και να έχει την μεγαλύτερη συσχέτιση μετά την ένωση μεταξύ μίας στήλης C ∈ TX και της Q. [7]*

Η πιο ακριβής αλλά μη αποδοτική μέθοδος για την υλοποίηση του παραπάνω ορισμού θα ήταν εφαρμόζοντας ένα πλήρες join μεταξύ των 2 πινάκων και ο υπολογισμός της συσχέτισης. Αύτη η μέθοδος, όμως, για μεγάλα σύνολα δεδομένων θα ήταν πάρα πολύ χρονοβόρα και με μεγάλο κόστος. Για την απάντηση των ερωτημάτων θα μπορούσαν να χρησιμοποιηθούν συνόψεις συσχέτισης (KMV Synopsis).

Για την κατάταξη των αποτελεσμάτων με βάση μόνο τη συσχέτιση μπορεί να αποφέρει χαμηλής απόδοσης αποτελέσματα καθώς με την χρήση συνόψεων έχουμε κάποια απώλεια πληροφορίας αφού δεν συμπεριλαμβάνονται όλα τα δεδομένα. Έτσι, χρησιμοποιείται μία μέθοδος για την κατάταξη των στηλών, που λαμβάνει σαν παράμετρο ένα ποσοστό λάθους. Λαμβάνοντας υπόψιν την αβεβαιότητα σε σχέση με τους υπολογισμούς εφαρμόζεται ένα σύστημα με παράμετρο το ρίσκο που επιλέγει k αποτελέσματα και μεγιστοποιεί :

$$\max \sum_{i=1}^{k} \left(|\hat{r}_{Q \triangleright\triangleleft C_i}| * (1 - risk(Q, C_i)) \right)$$

Εξίσωση 3

Όσο αφορά την απόλυτη τιμή $|\hat{r}_{Q \triangleright\triangleleft C_i}|$ είναι ο υπολογισμός της συσχέτισης χρησιμοποιώντας ένα υπολογιστή συσχέτισης, όπως για παράδειγμα την εξίσωση 2 εφαρμόζοντάς την στην ένωση. Η συνάρτηση $risk(Q, C_i)$ επιστρέφει έναν αριθμό ανάμεσα στο διάστημα [0,1] η οποία υπολογίζει τη διασπορά των εκτιμήσεων συσχέτισης χρησιμοποιώντας $L_{Q \triangleright\triangleleft C_i}$,

Όταν το ρίσκο που σχετίζεται με τον υπολογισμό είναι μη μηδενικό ένα ποσοστό ποινής εφαρμόζεται στον υπολογισμό ανάλογα με το ρίσκο. Όσο αφορά την απόλυτη τιμή εφαρμόζεται για τον λόγο ότι η αρνητική συσχέτιση μπορεί να είναι εξίσου χρήσιμη με την θετική συσχέτιση.

Για τον υπολογισμό του ρίσκου υπήρχαν αρκετές άλλες επιλογές στην ερευνητική μελέτη, όπως για παράδειγμα Confidence Interval Bound που ήταν η πιο αποδοτική, η μέθοδος Bootstrap CI και Hoeffding's CI. Στην παρούσα εργασία για τον λόγο ότι η υλοποίηση ήταν κατανεμημένη, η μέθοδος Confidence Interval Bound έκανε χρήση join στηλών και είχε σαν προϋπόθεση την ολοκλήρωση των συνόψεων και ανταλλαγή πολλών στηλών για τον υπολογισμό του ρίσκου.

$$se_z = 1 - \frac{1}{\sqrt{\max(4, n) - 3}} \qquad ci_b = 1 - \frac{\rho_{PM1}^{high} - \rho_{PM1}^{low}}{2}$$

$$ci_h = 1 - \frac{ci_{length} - ci_{min}^{high}}{ci_{max}^{high} - ci_{min}^{high}}$$

Εικόνα 10 Confidence Interval Bounds

Για τον λόγο αυτό αποφασίστηκε να χρησιμοποιηθεί η σχέση:

$$SE_z = 1/\sqrt{n - 3}.$$

Εξίσωση 4 Τυπικό λάθος της δειγματοληψίας του μετασχηματισμού Fisher's Z correlation transformation

Μία μέθοδος προσέγγισης για υπολογισμό του ρίσκου είναι η χρήση βασικών στατιστικών μεθόδων, όπως για παράδειγμα την εξίσωση 4. Στη εξίσωση η μεταβλητή n αφορά το μέγεθος της δειγματοληψίας. Παρότι υποθέτει bivariate κανονική κατανομή οι υπολογισμοί είναι αρκετά απλοί και δεν απαιτούν ανταλλαγές δεδομένων. Η μόνη προϋπόθεση είναι το γνωστό

μέγεθος δειγματοληψίας n. Η εξίσωση αναμένεται να έχει καλύτερη απόδοση και ακριβέστερα αποτελέσματα, καθώς το μέγεθος δειγματοληψίας αυξάνεται για οποιαδήποτε κατανομή δεδομένων. Για δεδομένα που αφορούν μη γνωστές κατανομές δεδομένων στον χρήστη δεν θα είχε τα βέλτιστα αποτελέσματα, καθώς υποθέτει κανονική κατανομή και δεν λαμβάνει σαν παράμετρο κάποια πληροφορία για την κατανομή των δεδομένων. Σε αυτήν την περίπτωση η μέθοδος Confidence Interval Bounds θα ήταν σε μεγαλύτερο βαθμό πιο αποδοτική.

Κεφάλαιο 3

Σχετικές Εργασίες

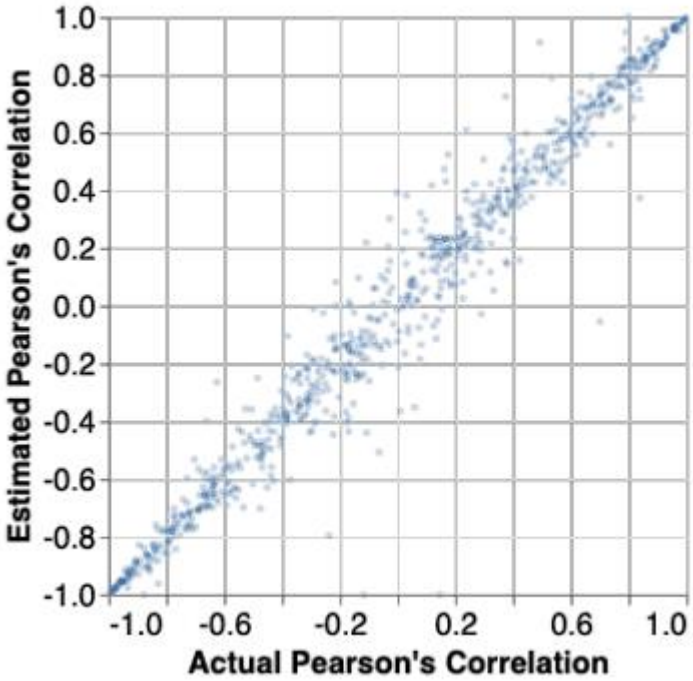
Στην 3η ενότητα της παρούσας διπλωματικής εργασίας περιγράφονται οι έρευνες και διπλωματικές που αποτέλεσαν τον ακρογωνιαίο λίθο αυτής της εργασίας. Για παράδειγμα οι αλγόριθμοι για την κατασκευή της σύνοψης, η ιδέα για την κατανεμημένη υλοποίηση, οι αλγόριθμοι εύρεσης και κατάταξης συσχετισμένων στηλών.

3.1 Υπολογισμός Συσχέτισης με την Βοήθεια Συνόψεων

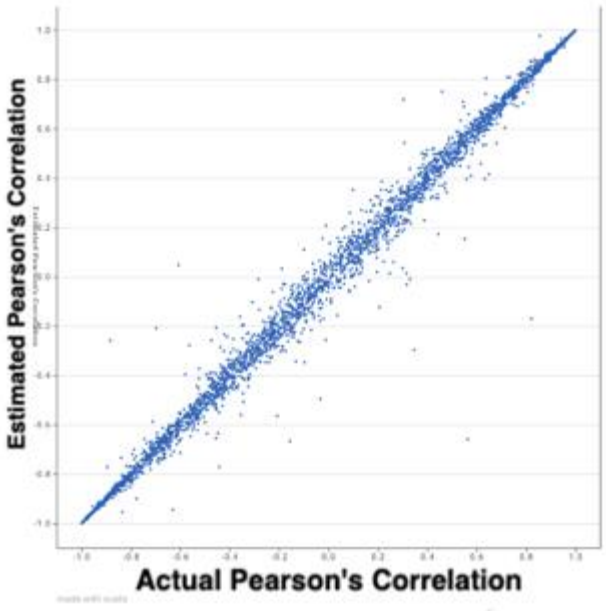
Η ιδέα και υλοποίηση της σύνοψης προήλθε από την ερευνητική μελέτη "Correlation Sketches for Approximate Join-Correlation Queries" [7]. Η συγκεκριμένη μελέτη και επεξεργασία, όμως, έγινε αρχικά τοπικά για σκοπούς απλότητας και ευκολίας και έπειτα κατανεμημένα. Στην παρούσα εργασία έγινε η υλοποίηση βάσει της περιγραφής στη δημοσίευση. Χρησιμοποιήθηκαν οι 2 μέθοδοι κατακερματισμού και μετά από έρευνα και μελέτη αποφασίστηκε η πιο αποδοτική μέθοδος για την κατάταξη συσχετισμένων στηλών. Παρόλο που στην έρευνα αναφέρει ότι η Fisher Z μέθοδος δεν ήταν η πιο αποδοτική και ακριβής, δεν λάμβανε σαν παράμετρο την κατανεμημένη υλοποίηση, που στην περίπτωσή μας είναι απαραίτητη για τον λόγο ότι δεν απαιτούν καθόλου ανταλλαγές δεδομένων μεταξύ τους αφού σαν μόνη παράμετρο έχει το μέγεθος της σύνοψης n που είναι άμεσα διαθέσιμη πληροφορία και δεν απαιτεί ανταλλαγές στηλών. Τα confidence interval bounds, που αναφέρει είναι πιο ακριβή και αποδοτικά σε μη κατανεμημένο περιβάλλον, αλλά λόγω των πολλαπλών ανταλλαγών μεταξύ στηλών που απαιτούν τα δεδομένα και λόγο της χρήσης μεγίστου και ελάχιστου της κάθε στήλης όπως περιγράφηκαν στο κεφάλαιο 2 θα καθιστούσε την κατανεμημένη υλοποίηση πολύ πιο χρονοβόρα. Αφού μελετήθηκε και αναλύθηκε η έρευνα με τις κατάλληλες παραμετροποιήσεις, για τις ανάγκες της παρούσας διπλωματικής υλοποιήθηκε αρχικά σε τοπικό περιβάλλον με σκοπό να διεξαχθούν τα πειράματα και να βεβαιωθεί ότι η λειτουργία hashing λειτουργεί σωστά, ώστε να έχουμε τα σωστά θεμέλια στην κατανεμημένη υλοποίηση. Στην έρευνα αναφέρονται κάποια πειράματα που διεξάχθηκαν και τα αντίστοιχα αποτελέσματα. Για να διαπιστωθεί η ορθή λειτουργία του αλγόριθμου προσομοιώθηκαν κάποια πειράματα και μετά από λεπτομερή σύγκριση με τα αποτελέσματα του paper βεβαιώθηκε η σωστή λειτουργία του προγράμματος.

Ένα από τα πειράματα που διεξάχθηκαν ήταν η κατασκευή ενός σετ δεδομένων με t πίνακες που περιέχουν n πλειάδες από <k,Xk,Yk>, όπου $k \in K$ είναι μία τυχαία λέξη και $Xk \in X$, $Yk \in Y$ πραγματική αριθμοί από μία διωνυμική κανονική κατανομή με μηδενικό μέσο. Η διασπορά από τους πίνακες X και Y επιλέχθηκαν με σκοπό το Pearson correlation coefficient μεταξύ X και Y να είναι περίπου ίσο με μία παράμετρο rxy. Έπειτα, κατασκευάστηκαν t ζευγάρια

από πίνακες Tx = <Kx,X> και Ty = <Ky,Y>. Τέλος, μειώθηκε ο αριθμός των πλειάδων στους πίνακες Ty από n σε n’ επιλέγοντας ένα τυχαίο ομοιόμορφα κατανεμημένο δείγμα n’ = n * c, όπου c τυχαία πραγματική μεταβλητή ανάμεσα το όριο (0,1) υποδεικνύοντας την πιθανότητα ένωσης μεταξύ X και Y. Το μέγεθος ζευγαριών πινάκων τέθηκε t = 3000. Για κάθε πίνακα το n τέθηκε ένας τυχαίος αριθμός παρμένος ομοιόμορφα από το διάστημα (0,500000) και η συσχέτιση rxy ήταν ένας τυχαίος αριθμός στο διάστημα (-1,1). Αρχικά, στο πείραμα τα αποτελέσματα δεν ήταν αντίστοιχά με αυτά του πειράματος, όμως, με τις απαραίτητες αλλαγές επιτεύχθηκε η προσαρμογή τους, έτσι ώστε να ληφθούν τα ίδια αποτελέσματα με το πείραμα που προσωμοιώθηκε στην έρευνα. Αφού επαληθεύτηκε η σωστή λειτουργία των μεθόδων κατακερματισμού και του αλγόριθμου για την κατασκευή του δένδρου, που χρησιμοποιήθηκαν για την κατασκευή της σύνοψης, είχε λάβει μέρος ο σχεδιασμό για την κατανεμημένη υλοποίηση του προγράμματος.



Εικόνα 11 Αποτελέσματα πειράματος από το paper



Εικόνα 12 Αποτελέσματα πειράματος από την υλοποίηση

Μπορούμε εύκολα να παρατηρήσουμε από τις εικόνες 11 και 12 ότι τα πειράματα της εργασίας είναι όμοια με αυτά της έρευνας. Ιδανικά εάν η σύνοψη εμπεριείχε όλα τα δεδομένα στο πείραμα θα διακρίναμε μία ευθεία γραμμή, καθώς η υπολογισμένη συσχέτιση θα ήταν ίδια με την πραγματική συσχέτιση.

3.2 Κατανεμημένη Υλοποίηση

Αφού ελέγχθηκε και πιστοποιήθηκε η ορθή λειτουργία του αλγόριθμου για κατασκευή της σύνοψης και των μεθόδων κατακερματισμού, ακολούθησε η σχεδίαση και η μελέτη για την κατανεμημένη υλοποίηση. Αρχικά, μελετήθηκε ο σχεδιασμός των διπλωματικών εργασιών «Πρόβλεψη Μετοχών στο Apache Flink» [11] και «Αποδοτική Πρόβλεψη Εξέλιξης Παράλληλων Καρκινικών Προσομοιώσεων στο Apache Flink»[12]. Στην πτυχιακή της αποδοτικής πρόβλεψης εξέλιξης παράλληλων καρκινικών προσομοιώσεων είχε χρησιμοποιηθεί η LSH σύνοψη που υλοποιήθηκε στο εσωτερικό του SDE με πρωταρχικό ρόλο της σύνοψης να είναι ο υπολογισμός των bitmaps και ο κατακερματισμός αυτών σε bucket σύμφωνα με παραμέτρους από τον χρήστη. Έγινε επίσης χρήση του Pearson Correlation Coefficient. Στην άλλη διπλωματική εργασία, που μελετήθηκε η υλοποίηση της εκτίμησης της συσχέτισης των εκάστοτε χρονοσειρών πραγματοποιήθηκε με βάση τον αλγόριθμο Discrete Fourier Transform (DFT) και βασίστηκε στο σύστημα StatStream. Είχαν χρησιμοποιηθεί παρόμοιοί αλγόριθμοι, αλλά είχαν αρκετές διαφορές, αφού και στις δύο άλλες διπλωματικές εργασίες είχε γίνει χρήση παραθύρων για την υλοποίηση στο Flink, καθώς είχαν σαν παράμετρο τον χρόνο, κάτι που καθιστούσε την υλοποίηση αρκετά διαφορετική αφού στην παρούσα εργασία ο χρόνος δεν είναι σαν παράμετρος. Παρόλα αυτά η υλοποίηση και ενσωμάτωση των συνόψεων, που υλοποιήθηκαν

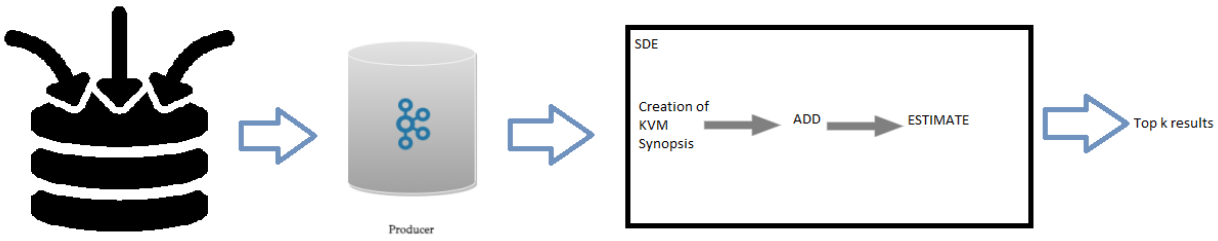
από τις 2 εργασίες στον SDE παρείχαν σημαντική βοήθεια, αφού με βάση αυτών έγινε δυνατή η πρόσθεση της σύνοψη στον SDE στα κατάλληλα σημεία με τις κατάλληλες παραμετροποιήσεις σύμφωνα με τις ανάγκες της εργασίας. Επιπρόσθετα, μελετήθηκε και ο τρόπος δημιουργίας request, estimate και ο τρόπος που στέλνονται τα δεδομένα για προσθήκη στη σύνοψη.

Τέλος, από τις 2 διπλωματικές εργασίες μελετήθηκε ο τρόπος εισαγωγής δεδομένων στο Apache Kafka. Ήταν αρκετά όμοιος, αφού με τον ίδιο τρόπο εισάγονται τα δεδομένα στον SDE. Με την βοήθεια της έρευνας και των δύο διπλωματικών εργασιών παρήλθε σημαντική βοήθεια για τον σχεδιασμό και την κατανόηση της παρούσας εργασίας.

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση Συστήματος

Στο 4ο κεφάλαιο παρουσιάζεται αναλυτικά η αρχιτεκτονική του συστήματος και η μελέτη που έλαβε μέρος πριν γίνει η υλοποίηση του.



Εικόνα 13 Αρχιτεκτονική συστήματος

4.1 Dataset

Τα δεδομένα που χρησιμοποιήθηκαν κατά την διάρκεια των πειραμάτων ως είσοδο στα Kafka topics δημιουργήθηκαν με βάση τα αντίστοιχα δεδομένα της δημοσίευσης ακολουθώντας την Synthetic Bivariate Normal (SBN) κατανομή για τα διεξοδικά πειράματα σε τοπικό περιβάλλον, έτσι ώστε να εξασφαλιστεί η σωστή λειτουργία του αλγόριθμου για τη δημιουργία συνόψεων.

Έπειτα, όσο αφορά τα πειράματα που έγιναν για την κατανεμημένη υλοποίηση τα δεδομένα δημιουργήθηκαν με τη χρήση βοηθητικού προγράμματος που δημιουργήθηκε εξ ολοκλήρου για τις ανάγκες της παρούσας εργασίας. Το dataset αποτελείτο από έναν αριθμό δεδομένων μεγάλου μεγέθους, έτσι ώστε οι υπολογισμοί να είναι ακριβείς. Πιο συγκεκριμένα η κάθε μέτρηση αφορούσε το dataset από όπου προέχονταν η μέτρηση, το όνομα του attribute, καθώς και η numeric τιμή του. Οι μετρήσεις καταγράφονταν σε ένα αρχείο txt για την εύκολη προσβασιμότητά του. Οι μετρήσεις ήταν της ακόλουθης μορφής :

```
animal:zwo:skulos:age:11
```

Εικόνα 14 Παράδειγμα δόμησης των δεδομένων για εισαγωγή στην σύνοψη

Για τη χρήση των δεδομένων πριν την εισαγωγή είναι σημαντικό να εφαρμοστεί ένα φιλτράρισμα στις τιμές με σκοπό όσες μετρήσεις δεν ακολουθούν την παραπάνω μορφή να παραλείπονται και να μη διαβάζονται από το topic του Kafka.

4.2 Λεπτομέρειες Υλοποίησης

4.2.1 Kafka & SDE

Αφού ακολουθήσει η δημιουργία των δεδομένων, τα δεδομένα διαβάζονται και εγγράφονται σε ένα Kafka topic. Το σύστημα SDE καταναλώνει τα δεδομένα από το Kafka topic, όπου και αρχίζει η παράλληλη επεξεργασία των δεδομένων από το πρόγραμμα. Καθώς αντλούνται τα δεδομένα, παράλληλα δημιουργείται ένα αίτημα για την δημιουργία των συνόψεων, όπως αποτυπώνεται παρακάτω :

1) AISkey: το κλειδί που προκύπτει από τον συνδυασμό του dataset και του attribute όπου θα επεξηγηθεί στην συνέχεια ο λόγος που έγινε αυτός ο συνδυασμός,
2) requestID: **1** στην προκειμένη περίπτωση είναι αίτημα για δημιουργία σύνοψης,
3) synopsisID: ο αναγνωριστικός αριθμός για την σύνοψη, όπου στην προκειμένη περίπτωση είναι το 14,
4) uID: ο αναγνωριστικός αριθμός κάθε αιτήματος,
5) parameters: οι παράμετροι που εισάγονται κατά την δημιουργία της σύνοψης είναι {“Attribute”,” Value”,” Metric”} για τη χρήση, όταν θα πάρει μέρος η εισαγωγή των δεδομένων,
6) noOfP: ο αριθμός παραλληλισμού του SDE.

Request rq = new Request (AISkey, 1, 14, 1110, parameters, 1);

Σε ένα επαναληπτικό βρόχο διαβάζονται το δεδομένα από το αρχείο και δημιουργούνται τα ερωτήματα για την δημιουργία των συνόψεων. Η παράμετρος AISkey αποτελείται από τον συνδυασμό του dataset και attribute (στήλης) για την αποθήκευση στην λίστα του FlatMap και μελλοντική χρήση για την εισαγωγή δεδομένων. Εάν βρεθεί ήδη υπάρχων κλειδί κατά την διάρκεια της δημιουργίας των ερωτημάτων, δεν δημιουργείται νέα σύνοψη, για τον λόγο ότι υπάρχει.

4.2.2 Σύνοψη

Κατά τη λήψη του αιτήματος για δημιουργία της σύνοψής, δημιουργείται μία νέα σύνοψη σύμφωνα με το κλειδί στην λίστα με τα AISkeys. Η μορφή της σύνοψης στο σύστημα SDE έχει σαν παραμέτρους τα εξής δεδομένα :

1) uID,
2) parameters: τις παραμέτρους από το ερώτημα,
3) Sketch: Το Sketch είναι μια δομή HashMap για την αποθήκευση των τιμών της σύνοψης,
4) Humap: Χρήση σε υπολογισμούς,
5) W: μέγιστο μέθοδος της σύνοψης,
6) Size: πόσα δεδομένα βρίσκονται στην σύνοψη.

Το μέγεθος W είναι μία σταθερά, της οποίας η αρχικοποίηση γίνεται σε όλες τις συνόψεις ανάλογα με το επιθυμητό μέγεθος της σύνοψης. Επιπρόσθετα, υλοποιήθηκαν 2 μέθοδοι στην κλάση της σύνοψης, οι οποίες θα επεξηγηθούν σε μετέπειτα στάδιο. Τα δεδομένα που εισάγονται στη σύνοψη διαμορφώνονται με τις μεθόδους κατακερματισμού, που έχουν αναφερθεί στο Κεφάλαιο 2.

4.2.3 Add

Μόλις πραγματοποιηθεί η λήψη του ερωτήματος για τη δημιουργία της σύνοψης και δημιουργηθεί η σύνοψη λαμβάνει μέρος η αποστολή δεδομένων στην κατάλληλη σύνοψη σύμφωνα με τον αλγόριθμο 1.

Algorithm 1 Add

while $line \neq null$ **do**
  $words \leftarrow line.split(" : ")$
  $key \leftarrow words[0]$
  for (k=1;k<words.length-3;k=k+4) **do**
    if $words.length > 1$ **then**
      $jsonString \leftarrow Attr : +words[k] + Value : +words[k+1] + Metric : +words[k+3]$
      $node \leftarrow mapper.readTree(jsonString)$
      $AISkey \leftarrow key + words[k]$
      $Datapoint \leftarrow Datapoint(AISkey, node)$
      $producer \leftarrow send(ProducerRecord(String, String)(topicName, dp.toJsonString()))$
    end if
  end for
end while

Αλγόριθμος 1 Λειτουργία αποστολής δεδομένων στις συνόψεις

Ο αλγόριθμος 1 αρχικά διαβάζει τα δεδομένα από ένα αρχείο και τα εισάγει στο topic. Έπειτα το σύστημα SDE καταναλώνει τα δεδομένα από το Kafka topic. Στην μεταβλητή key αποθηκεύεται το όνομα του dataset, το οποίο στη συνέχεια συνδυάζεται με το όνομα του attribute, για να δημιουργηθεί το κλειδί, που θα οδηγήσει τα δεδομένα στη σωστή σύνοψη που δημιουργήθηκε προηγουμένως. Επίσης, αποθηκεύονται σε ένα Json String το Attribute μαζί με το Value και τα Metric. Τέλος, δημιουργείται ένα datapoint και αποστέλλονται τα δεδομένα στο Kafka topic για να επεξεργαστούν από το FlatMap του SDE.

```
Datapoint dp = new Datapoint(AISkey, AISkey, node);
producer.send(new ProducerRecord<String, String>(topicName, dp.toJsonString()));
```

Εικόνα 15 Δημιουργία Datapoint

Στην εικόνα 15 μπορούμε να διακρίνουμε την δημιουργία ενός νέου data point με AISkeys τον συνδυασμό του dataset και της ονομασίας της στήλης. Στην επόμενη γραμμή κώδικα με την χρήση ενός producer στέλνεται στο κατάλληλο topic και το datapoint μετατρέπεται σε ένα JSON String.

Σύμφωνα με την Εικόνα 13, όπου περιγράφεται η αρχιτεκτονική του SDE τα δεδομένα βρίσκονται πλέον στο CoFlatMap, όπου υπάρχει το FlatMap add. Στο add FlatMap ελέγχεται εάν υπάρχει ήδη η σύνοψη με το AISkeys και έπειτα στέλνεται στην αντίστοιχη σύνοψη. Σε περίπτωση που το κλειδί δεν υπάρχει, αγνοείται. Στην συνέχεια καλείται η συνάρτηση add στην κλάση KMVSynopsis. Αρχικά, εξάγονται τα δεδομένα από το JSON Node και γίνεται η αρχικοποίηση τους σε τοπικές μεταβλητές. Στην συνέχεια αυξάνεται το μέγεθος της σύνοψης κατά 1. Τα δεδομένα κατακερματίζονται, όπως περιεγράφηκε στο Κεφάλαιο 2 και εισάγονται στη σύνοψη με τον ακόλουθο αλγόριθμο.

$humap \leftarrow TreeMap < Double, Integer >$
$Sketch \leftarrow HashMap < Integer, Double >$
$entry \leftarrow HashMap < Integer, List < Double >>$
$h \leftarrow h.murmurhash3$
$hu \leftarrow h.Fibonacci$
$values \leftarrow add(count)$
$values \leftarrow add(value)$
if $(humap.size() > 1)$ **then**
  $big\_hu \leftarrow humap.lastKey()$
  $h\_of\_bighu \leftarrow humap.get(big_hu)$
end if
if $(!entry.containsKey(h)\ \ and\ \ entry.size() < W)$ **then**
  $entry \leftarrow put(h, values)$
  $Sketch \leftarrow put(h, values.get(1))$
  $humap \leftarrow put(hu, h)$
else if $(!entry.containsKey(h)\ \ and\ \ hu < big\_hu)$ **then**
  $entry \leftarrow remove(h\_of\_bighu)$
  $Sketch \leftarrow remove(h\_of\_bighu)$
  $humap \leftarrow remove(big\_hu)$
  $humap \leftarrow put(hu, h)$
  $Sketch \leftarrow put(h, values)$
  $entry \leftarrow put(h, values.get(1))$
else if $(entry.containsKey(h))$ **then**
  $aggregate\_values \leftarrow entry.get(h)$
  $aggrnum1 \leftarrow aggregate\_values.get(1)$
  $counter\_for\_agg \leftarrow aggregate\_values.get(0)$
  $counter\_for\_agg++$
  $total\_value \leftarrow aggrnum1 + value$
  $value \leftarrow (total\_value)/counter\_for\_agg$
  $entry \leftarrow remove(h)$
  $Sketch \leftarrow remove(h)$
  $aggregate\_values \leftarrow clear()$
  $aggregate\_values \leftarrow add(counter\_for\_agg)$
  $aggregate\_values \leftarrow add(total\_value)$
  $entry \leftarrow put(h, aggregate\_values)$
  $Sketch \leftarrow put(h, value)$
end if

Αλγόριθμος 2 Λειτουργία add της σύνοψης

Περιγραφή Αλγορίθμου:

Αφού σταλούν τα δεδομένα και φτάσουν στην σύνοψη αρχικά γίνεται η αρχικοποίηση 3 δομών για την επεξεργασία των δεδομένων. Η δομή humap χρησιμοποιείται για τη διατήρηση του μεγαλύτερου hashed value, έτσι ώστε να γίνει η απόφαση αν θα αφαιρεθεί κάτι από το Sketch. Η δομή Sketch είναι υπεύθυνη για τη διατήρηση της τελικής σύνοψης. Στη δομή entry αποθηκεύονται τα δεδομένα που αντλούνται από το Kafka topic. Έπειτα, ελέγχεται εάν το κατακερματισμένο κλειδί δεν υπάρχει ήδη στο Sketch και εάν το μέγεθος του Sketch δεν είναι πλήρης. Εάν πληρούνται οι έλεγχοι, τότε εισάγονται τα δεδομένα στις 3 δομές. Σε περίπτωση που δεν πληρείται το $1^{ο}$, δηλαδή το μέγεθος του Sketch είναι πλήρες, τότε θα ακολουθήσει το $2^{ο}$. Σε αυτήν την περίπτωση κοιτάζει το μέγεθος hu δηλαδή το $2^{ο}$ hashed value, αν είναι μικρότερο από το ήδη υπάρχον αποθηκευμένο hu στην λίστα. Εφόσον πληρείται αυτή η συνθήκη, τότε αφαιρείται από τις 3 δομές το στοιχείο με το μεγαλύτερο hu που προϋπήρχε και εισάγεται το νέο. Στην τελευταία περίπτωση, που δεν πληρούνταν οι συνθήκες προηγουμένως, ελέγχεται εάν το κλειδί υπάρχει ήδη στην λίστα. Σε αυτήν την περίπτωση γίνονται aggregate τα values του ήδη υπάρχοντος value με του καινούργιου και διαιρείται το άθροισμα των values με τον αριθμό των εμφανίσεων του value μέχρι την παρούσα στιγμή στο Sketch. Αφαιρείται το προηγούμενο key,value και προστίθεται τα νέα aggregated values.

4.2.4 Find Column

Αφού πραγματοποιηθεί η εισαγωγή των δεδομένων στο σύστημα SDE και στις συνόψεις, τότε λαμβάνει μέρος η λειτουργία για εύρεση των top-k correlated columns. Για να έρθει εις πέρας αυτή η λειτουργία πρέπει πρώτα να γίνει η εύρεση της ζητούμενης στήλης, η όποια θα συγκριθεί με τις υπόλοιπες που ήδη υπάρχουν στον SDE. Για την εύρεση της στήλης ανάμεσα στις συνόψεις στον SDE χρησιμοποιήθηκε το παρακάτω ερώτημα:

1) Το όνομα του dataset μαζί με το όνομα της στήλης που πρέπει να βρεθεί
2) requestID: **3** στην προκείμενή περίπτωση είναι αίτημα για εύρεση της στήλης,
3) synopsisID: ο αναγνωριστικός αριθμός για τη σύνοψη, που στην προκειμένη περίπτωση είναι το 14,
4) uID: ο αναγνωριστικός αριθμός κάθε αιτήματος.

Request rq = new Request("animals whale", 3, 14, 1110)

Όταν φτάσει το ερώτημα στο FlatMap RqRouter, τότε ελέγχεται εάν το κλειδί που ζητείται υπάρχει ήδη στη λίστα με τα διαθέσιμα κλειδιά από τις συνόψεις που δημιουργήθηκαν προηγουμένως. Στην περίπτωση που βρεθεί ήδη υπάρχον κλειδί, τότε στέλνεται το ερώτημα στο επόμενο FlatMap, SDEcoFlatMap. Στο FlatMap καλείται η συνάρτηση estimate που υλοποιήθηκε στην κλάση KMVSynopsis. Η συνάρτηση estimation που υλοποιήθηκε στην σύνοψη έχει 2

διαφορετικές λειτουργίες. Η 1η λειτουργία αφορά την εύρεση στήλης και η 2η θα επεξηγηθεί αργότερα.

Όταν καλεστεί η συνάρτηση estimate ελέγχεται το requestID, για να αποφασιστεί ποια θα είναι η λειτουργία της συνάρτησης. Έπειτα, γίνεται η αρχικοποίηση μίας λίστας στο μέγεθος του Sketch και εισάγονται τα δεδομένα του Sketch στη λίστα διαχωρίζοντας τα με " : ". Όταν εγγραφούν στη λίστα όλα τα δεδομένα του Sketch τότε δημιουργείται ένα Estimation και επιστρέφει στο FlatMap. Το estimation περιέχει τις παρακάτω μεταβλητές :

1) Το όνομα του dataset μαζί με το όνομα της στήλης που βρέθηκε
2) requestID: **3** στην προκείμενή περίπτωση είναι αίτημα για εύρεση της στήλης,
3) synopsisID: ο αναγνωριστικός αριθμός για τη σύνοψη, που στην προκειμένη περίπτωση είναι το 14,
4) uID: ο αναγνωριστικός αριθμός κάθε αιτήματος,
5) parameters: Η λίστα με τα περιεχόμενα του Sketch.

Estimation es1 = new Estimation(1110,rq.getKey(),3,14,stringArray)

4.2.5 Find Correlation

Αφού επιστραφεί το estimation από τη λειτουργία για εύρεση στήλης, τότε θα πρέπει να συγκριθεί με όλες τις υπόλοιπες στήλες που υπάρχουν στο σύστημα SDE. Για την επίτευξη αυτής της λειτουργίας, μόλις επιστρέψει το estimation γίνεται mapping σε ένα καινούργιο request στην κλάση KafkaProducerEstimation για να μπορεί να χρησιμοποιηθεί σαν ερώτημα, καθώς σαν estimation δεν ήταν χρήσιμο.

```
public Request(Estimation element) {
    this.DataSetkey = element.getEstimationkey();
    RequestID = 6;
    SynopsisID = element.getSynopsisID();
    UID = element.getUID();
    Param = element.getParam();
}
```

Εικόνα 17 Mapping Estimation σε Request

Όταν πάρει μέρος το mapping του estimation σε request, όλες οι μεταβλητές παραμένουν ίδιες εκτός από το SynopsisID, που μετατρέπεται σε 6. Το 6 είναι ο χαρακτηριστικός αριθμός για εύρεση correlation. Έπειτα, το request γράφεται στο Kafka topic και παράλληλα το ερώτημα στέλνεται στο RqRouterFlatMap. Στο FlatMap, όπου διατηρείται η λίστα με όλες τις ήδη δημιουργημένες συνόψεις χρησιμοποιώντας ένα επαναληπτικό βρόγχο στέλνεται το

ερώτημα σε όλες τις συνόψεις. Στην συνέχεια, το ερώτημα καταφθάνει στο SDEcoFlatMap, όπου και καλείται η συνάρτηση estimate της κάθε σύνοψης.

Η συνάρτηση estimate, όπως αναφέρθηκε πιο πάνω έχει 2 διαφορετικές λειτουργίες. Η 2η λειτουργία, λοιπόν, αφορά την εύρεση συσχέτισης. Όταν καλεστεί η συνάρτηση estimate ελέγχεται ο χαρακτηριστικός αριθμός requestID, όπου διαχωρίζει τις λειτουργίες. Στην παρούσα περίπτωση, όπου ο αριθμός είναι 6 γίνεται η αρχικοποίηση ενός πίνακα με όλα τα περιεχόμενα του request, που υπήρχαν στο Sketch του προηγούμενου estimation. Στη συνέχεια, για τον υπολογισμό του correlation εφαρμόζεται join μεταξύ του παρόντος Sketch και του Sketch που υπήρχε στο estimation, έτσι ώστε να βρεθούν τα όμοια στοιχεία των 2 Sketch και να εφαρμοστεί ο τύπος για το correlation. Έπειτα, ακολουθεί ο αλγόριθμος που χρησιμοποιήθηκε για να υπολογιστεί το correlation:

```
for (val : final_Sketch) do
    values ← List < Double >
    values ← val.getValue
    xsum ← xsum + values.get(0)
    ysum ← ysum + values.get(1)
end for

X ← xsum/final_Sketch.size()
Y ← ysum/final_Sketch.size()

for (val : final_Sketch) do
    values ← List < Double >
    values ← val.getValue
    x ← values.get(0)
    y ← values.get(1)
    arithmitis ← arithmitis + ((x − X) ∗ (y − Y))
    paronomastis1 ← paronomastis1 + (Math.pow((x − X), 2))
    paronomastis2 ← (Math.pow((y − Y), 2))
end for

paronomastis1 ← Math.sqrt(paronomastis1)
paronomastis2 ← Math.sqrt(paronomastis2)
paronomastis ← paronomastis1 ∗ paronomastis2
rxy ← arithmitis/paronomastis
```

Αλγόριθμος 3 Υπολογισμός συσχέτισης μεταξύ των συνόψεων

Ο παραπάνω αλγόριθμος αρχικά χρησιμοποιεί τα 2 values που προέκυψαν από τα κοινά hashed values των 2 Sketch και ακολούθως κάνει τους υπολογισμούς σύμφωνα με την Εξίσωση 2. Στη συνέχεια, ελέγχεται εάν το value, που προκύπτει από την εξίσωση είναι πραγματικός αριθμός και δεν προέκυψε κάποιο error κατά την εκτέλεση για την αποφυγή Runtime Exception.

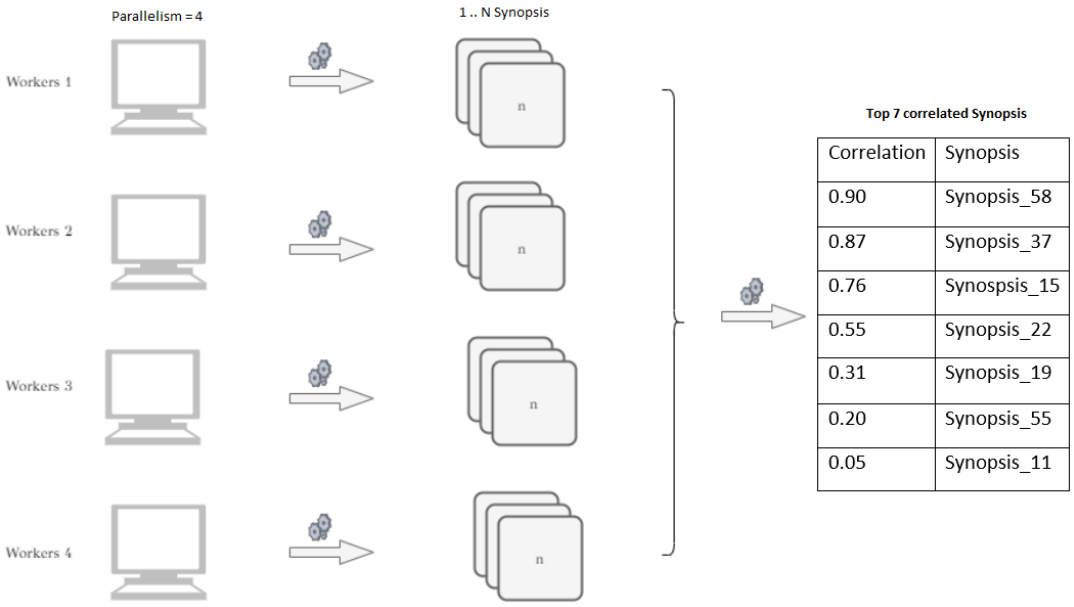
Αφού υπολογιστεί ο αριθμός της συσχέτισης μεταξύ των 2 συνόψεων σύμφωνα με την Εξίσωση 3, πρέπει να πολλαπλασιαστεί μαζί με ένα ποσοστό λάθους. Όπως αποφασίστηκε, σύμφωνα με τις απαιτήσεις της παρούσας διπλωματικής εργασίας για τον υπολογισμό του ρίσκου, επιλέχθηκε η Εξίσωση 4, αφού δεν χρησιμοποιεί ανταλλαγές δεδομένων μεταξύ στηλών, άρα είναι πολύ πιο κερδοφόρα όσο αφορά την χωρική και χρονική πολυπλοκότητα.

Όσον αφορά την ακρίβεια στις μετρήσεις δεν είχε την απόδοση των μεθόδων στο paper, αλλά λαμβάνοντας υπόψη τον τεράστιο όγκο δεδομένων και τις ανταλλαγές που θα χρειαζόταν να γίνουν ανάμεσα στις στήλες χρησιμοποιήθηκε η σχέση αυτή. Για τον υπολογισμό του ποσοστού λάθους χρησιμοποιήθηκε η μεταβλητή, που υπήρχε στη σύνοψη, η οποία ήταν αντίστοιχή με το μέγεθος της σύνοψης. Σύμφωνα με την Εξίσωση 4, όσο μεγαλύτερη ήταν η δειγματοληψία για την κατασκευή της σύνοψης, τόσο μικρότερο ήταν το ποσοστό λάθους, που επιβάλλεται σαν ποινή στη συσχέτιση των 2 συνόψεων.

Μετά από την εξαγωγή του τελικού αποτελέσματος συσχέτισης, το αποτέλεσμα επιστρέφεται μέσω της συνάρτησης στη σύνοψη σε μορφή Estimation πίσω από το FlatMap, όπου καλέστηκε. Η μορφή του estimation, που επιστρέφεται έχει την εξής μορφή :

```
Estimation(rq.getKey(),rxy);
```

Με τη χρήση του παραπάνω estimation επιστρέφεται το όνομα της σύνοψης που συγκρίθηκε με το column του ερωτήματος και η τελική συσχέτιση, που προέκυψε συμπεριλαμβανομένου και του τελεστή λάθους. Το estimation που φτάνει στο SDEcoFlatMap σπάει σαν key, value και αποθηκεύεται σε ένα tree map, το οποίο έχει σαν κλειδί το ποσοστό συσχέτισης και σαν value το όνομα της σύνοψης που συγκρίθηκε. Το tree map ταξινομεί της συσχετίσεις και κρατάει τις top k. Για την μέθοδο ταξινόμησης χρησιμοποιεί τη δομή δεδομένων με όνομα Red-Black tree.



Εικόνα 18 Λειτουργία estimate για top 7 Correlated Synopsis

Στην εικόνα 18 είναι ένα παράδειγμα υπολογισμού των top 7 συσχετισμένων στηλών. Η στήλη Synopsis αντιπροσωπεύει τις συνόψεις που δημιουργήθηκαν για τις στήλες από τα dataset. Η στήλη correlation έχει πολύ μικρές τιμές για τον λόγο ότι τα dataset αυτού του παραδείγματος αποτελούνταν από λίγες στήλες. Έτσι η $7^{η}$ στήλη έχει αρκετά μικρό αριθμό συσχέτισης.

Κεφάλαιο 5

Πειραματική Διαδικασία

5.1 Μεθοδολογία Ελέγχου

Ο έλεγχος του συστήματος για την ορθή λειτουργία πραγματοποιήθηκε αρχικά σε τοπικό περιβάλλον, το οποίο αποτελείτο από 1 μηχάνημα Ryzen 5 3600, 16GB RAM, 1TB SSD. Έπειτα, για την κατανεμημένη υλοποίηση οι έλεγχοι πραγματοποιήθηκαν 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

Τα πειράματα που διεξήχθησαν στην παρούσα διπλωματική χωρίστηκαν σε 2 μέρη. Το 1^{ο} μέρος αφορούσε τη διεξαγωγή πειραμάτων και μετρήσεων για την ορθή και αποδοτική λειτουργία του αλγόριθμου για τη δημιουργία της σύνοψης. Το 2^{ο} μέρος αφορούσε την διεξαγωγή πειραμάτων για το ζητούμενο της εργασία δηλαδή την εύρεση των top-k correlated columns. Διεξάχθηκαν πάνω από 150 πειράματα μέχρι την εξασφάλιση της ορθής λειτουργίας και την αφαίρεση διαφόρων σφαλμάτων που προέκυψαν.

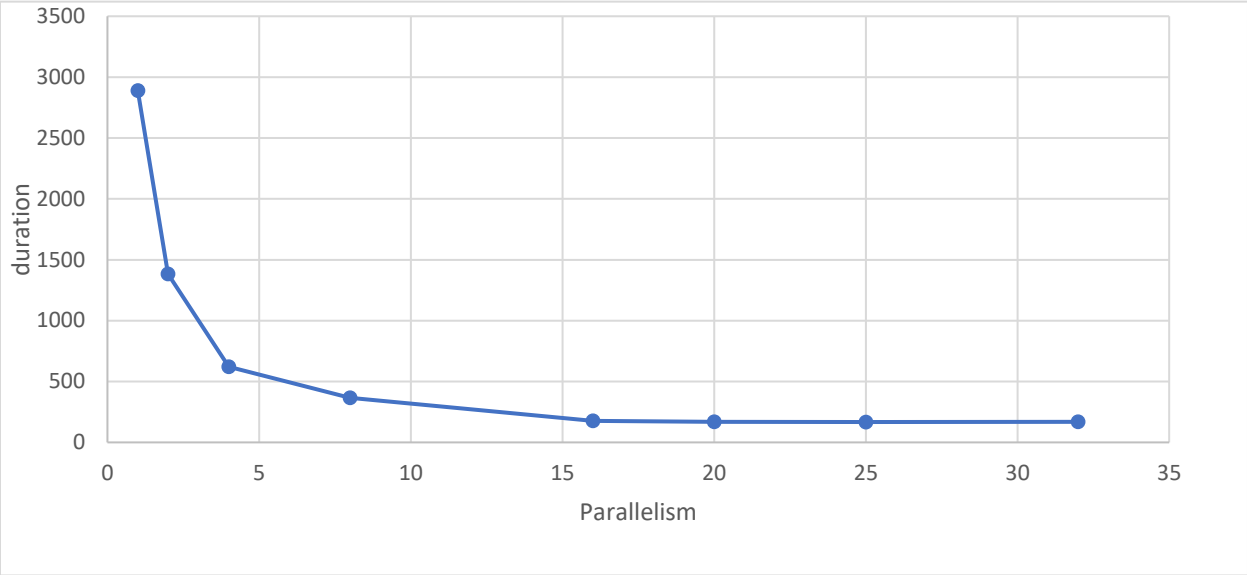
5.2.1 Πρώτο Μέρος Πειραμάτων

Για τη μέτρηση της απόδοσης του αλγορίθμου και τη διεξαγωγή των πειραμάτων αποφασίστηκε να μεταβάλλονται 4 βασικοί παράμετροι του προγράμματος, έτσι ώστε να εξεταστεί σε ένα πλήρες φάσμα. Σε κάθε πείραμα μεταβάλλεται και μία διαφορετική παράμετρος με σκοπό τη μέτρηση του duration, ενώ για τις υπόλοιπες επιλέχθηκαν κάποιες προκαθορισμένες τιμές των παραμέτρων που βρίσκονταν κάπου στο μέσο και έχουν έως εξής:

- Parallelism (default value 8)
- Tuples (Αριθμός δεδομένων στο dataset, default 50,000,000)
- Στήλες (Αριθμός στηλών που έχει το dataset, default 100)
- Dataset (Αριθμός διαφορετικών dataset που γράφονται στο Kafka topic, default 2)

Scalability Συστήματος

Με τη μεταβολή του μεγέθους του παραλληλισμού μπορεί να μετρηθεί το scalability του συστήματος. Η μεταβολή του μεγέθους μπορεί να επιτευχθεί στην αρχικοποίηση του προγράμματος όταν γίνεται η αρχικοποίηση του παραλληλισμού. Με την αύξηση του παραλληλισμού επιτυγχάνεται αύξηση του αριθμού των μηχανήματων (workers), που αναλαμβάνουν την κατανεμημένη και παράλληλη λειτουργία του συστήματος. Με την αύξηση αυτή αναμένεται και μεγάλη βελτίωση στη λειτουργία του συστήματος αφού οι διεργασίες εκτελούνται παράλληλα. Για το μέγεθος του παραλληλισμού στο πείραμα χρησιμοποιήθηκαν οι τιμές 1,2,4,8,16,20,25,32.



Εικόνα 19 Διάγραμμα Scalability για διαφορετικές τιμές παραλληλισμού

Μπορεί εύκολα να παρατηρηθεί στην εικόνα 19 μέχρι ο παραλληλισμός να γίνει ίσος με 16 ότι η διάρκεια δημιουργίας της σύνοψης για μέγεθος 50,000,000 στοιχείων μειώνεται γραμμικά. Κορυφώνεται στους 16 workers, καθώς ο κατακερματισμός των διεργασιών μεταξύ των workers δεν βελτιώνεται ουσιαστικά για παραλληλισμό ίσο με 32. Οι τιμές των υπόλοιπων παραμέτρων που χρησιμοποιήθηκαν ήταν οι default που προαναφέρθηκαν, όπως θα ακολουθήσει και στα επόμενα πειράματα.

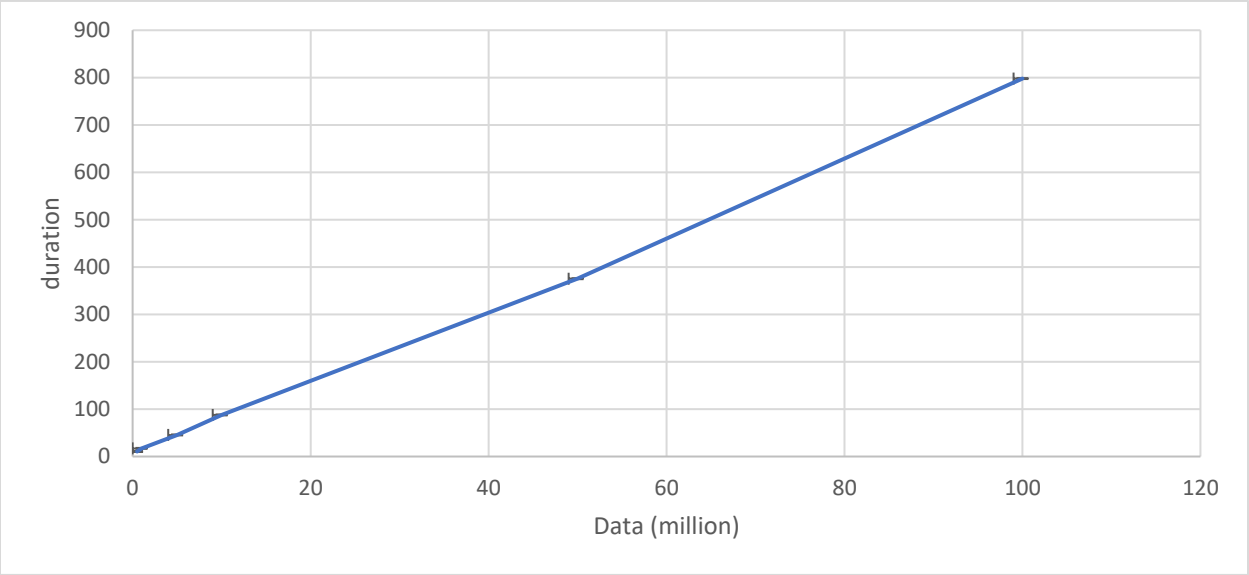
Οι τιμές που χρησιμοποιήθηκαν για την διάρκεια ήταν ο μέσος όρος από 10 μετρήσεις που προέκυψαν και παρατηρήθηκε μία μικρή απόκλιση, καθώς η απόδοση εξαρτάται από την απόδοση του worker, στον οποίο θα ανατεθεί η κάθε διεργασία.

Parallelism	**1**	**2**	**4**	**8**	**16**	**20**	**25**	**32**
Standard deviation	**58.3095**	**27.0185**	**19.2354**	**11.1803**	**4.61519**	**14.1598**	**9.03881**	**8.01873**

Πίνακας 1 Τυπική απόκλιση πειράματος με βάση τον παραλληλισμό

Αριθμός Δεδομένων

Στην εικόνα 21 τα δεδομένα του κάθε dataset στο κάθε πείραμα ξεχωριστά είχαν τις τιμές, όπως φαίνεται στην εικόνα 22. Οι υπόλοιπες παράμετροι είχαν τις τιμές.



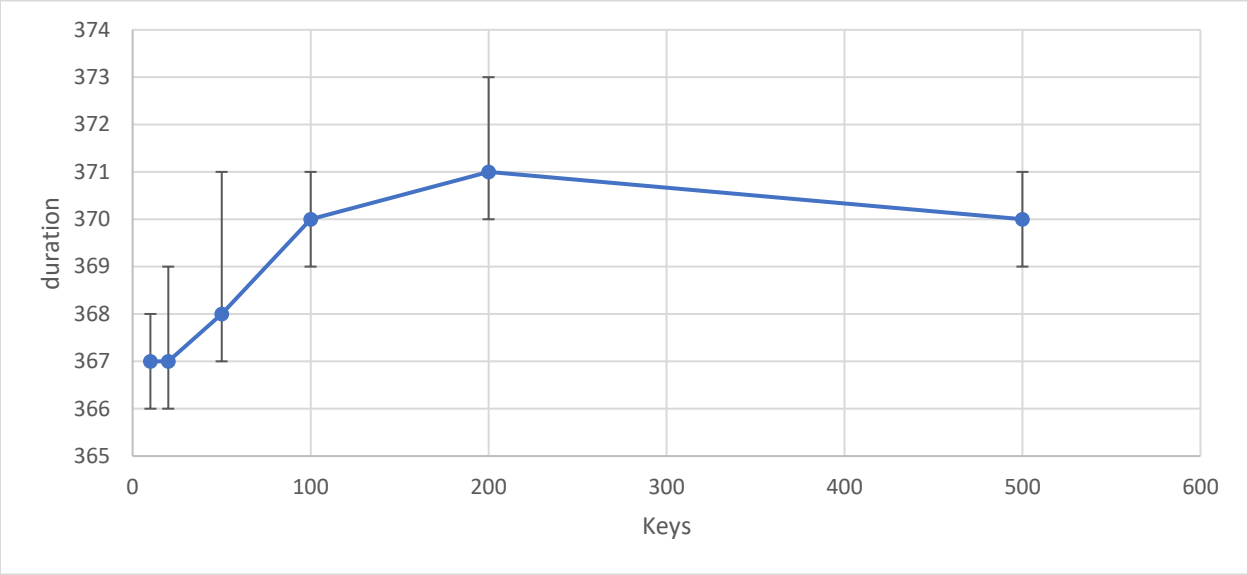
Εικόνα 20 Πείραμα για δημιουργία σύνοψης με βάση το μέγεθος των δεδομένων

Τα συμπεράσματα που μπορούμε να διεξάγουμε από την εικόνα 20, είναι ότι ανεξάρτητα από το μέγεθος της σύνοψης ο χρόνος εισαγωγής των δεδομένων είναι γραμμικός με τον χρόνο. Αυτό συμβαίνει για τον λόγο ότι η δημιουργία της σύνοψης γίνεται με ένα πέρασμα από τα δεδομένα. Αρά σύμφωνα με το μέγεθος των δεδομένων θα δημιουργηθούν αντίστοιχα ερωτήματα.

Tuples	**500,000**	**1,000,000**	**5,000,000**	**10,000,000**	**50,000,000**	**100,000,000**
Standard deviation	**2.7018**	**2.5884**	**4.0373**	**2.7748**	**11.2160**	**19.2353**

Πίνακας 2 Τυπική απόκλιση πειράματος 2 σύμφωνα με το κάθε μέγεθος των δεδομένων

Αριθμός Στηλών



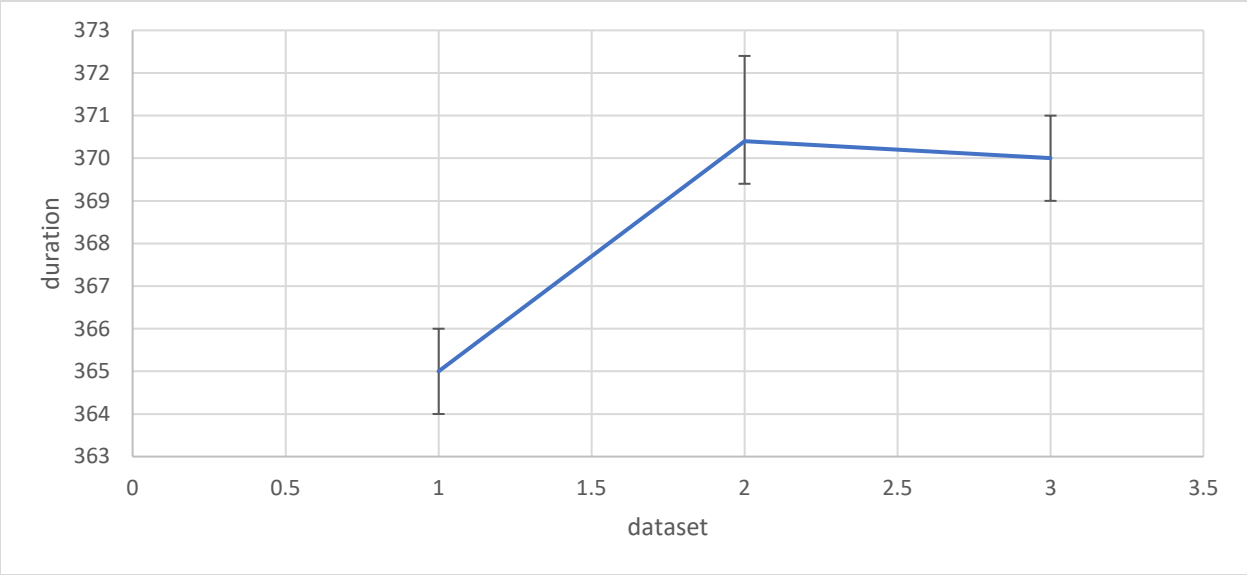
Εικόνα 21 Πείραμα για δημιουργία σύνοψης σύμφωνα με τις μοναδικές στήλες

Για τη διεξαγωγή του πειράματος αυτού πραγματοποιήθηκε η μεταβολή των στηλών στο κάθε dataset. Όπως μπορούμε να συμπεράνουμε από την εικόνα 21, το μέγεθος των μοναδικών κλειδιών δεν επιφέρει άμεσο αντίκτυπο στον χρόνο για την δημιουργία της σύνοψης. Αυτό συμβαίνει για τον λόγο ότι τα αιτήματα για την δημιουργία των συνόψεων είναι ανάλογα με το μέγεθος των δεδομένων και δεν είναι ανάλογα με το μέγεθος των στηλών. Δηλαδή για μέγεθος δεδομένων ίσο με 50,000,000 θα σταλούν 50,000,000 ερωτήματα για εισαγωγή δεδομένων ανεξάρτητα από τις μοναδικές στήλες.

Keys	**10**	**20**	**50**	**100**	**200**	**500**
Standard deviation	**1.0580**	**2.1803**	**3.4027**	**1.8294**	**2.5092**	**1.9050**

Πίνακας 3 Τυπική απόκλιση πειράματος 3 σύμφωνα με το μέγεθος των μοναδικών στηλών.

Αριθμός Dataset

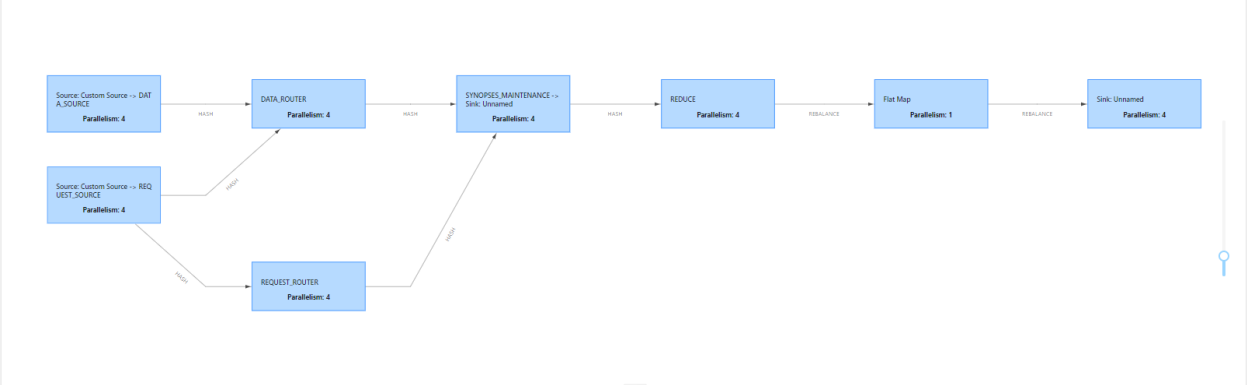


Εικόνα 22 Πείραμα για δημιουργία σύνοψης σύμφωνα με τα μοναδικά dataset

Στο τελευταίο πείραμα του 1ο μέρους διεξαγωγής πειραμάτων εξετάστηκε η διάρκεια δημιουργίας της σύνοψης σύμφωνα με τον αριθμό των μοναδικών dataset που δίνονταν σαν είσοδος στον SDE. Όπως φαίνεται στην εικόνα 22, ο χρόνος με την αύξηση των dataset παραμένει ο ίδιος. Όπως κα στο προηγούμενο παράδειγμα με την αύξηση του αριθμού των dataset το μέγεθος των δεδομένων παρέμεινε το ίδιο συνεπώς το μέγεθος ερωτημάτων παρέμεινε το ίδιο.

Dataset	**1**	**2**	**3**
Standard deviation	**1.04159458**	**2.264381405**	**1.536577473**

Πίνακας 4 Τυπική απόκλιση σύμφωνα με τα dataset



Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Duration	Tasks
Source: Custom Source -> DATA_SOURCE	RUNNING	0 B	0	254 MB	1,056,986	4	2023-06-11 11:53:03	30s	4
Source: Custom Source -> REQUEST_SOURCE	RUNNING	0 B	0	0 B	200	4	2023-06-11 11:53:03	30s	4
DATA_ROUTER	RUNNING	254 MB	1,056,075	254 MB	1,055,876	4	2023-06-11 11:53:03	30s	4
REQUEST_ROUTER	RUNNING	14.3 KB	200	0 B	200	4	2023-06-11 11:53:03	30s	4
SYNOPSES_MAINTENANCE -> Sink: Unnamed	RUNNING	254 MB	1,055,320	0 B	0	4	2023-06-11 11:53:03	30s	4
REDUCE	RUNNING	0 B	0	0 B	0	4	2023-06-11 11:53:03	30s	4
Flat Map	RUNNING	0 B	0	0 B	0	1	2023-06-11 11:53:03	30s	1
Sink: Unnamed	RUNNING	0 B	0	0 B	0	4	2023-06-11 11:53:03	30s	4

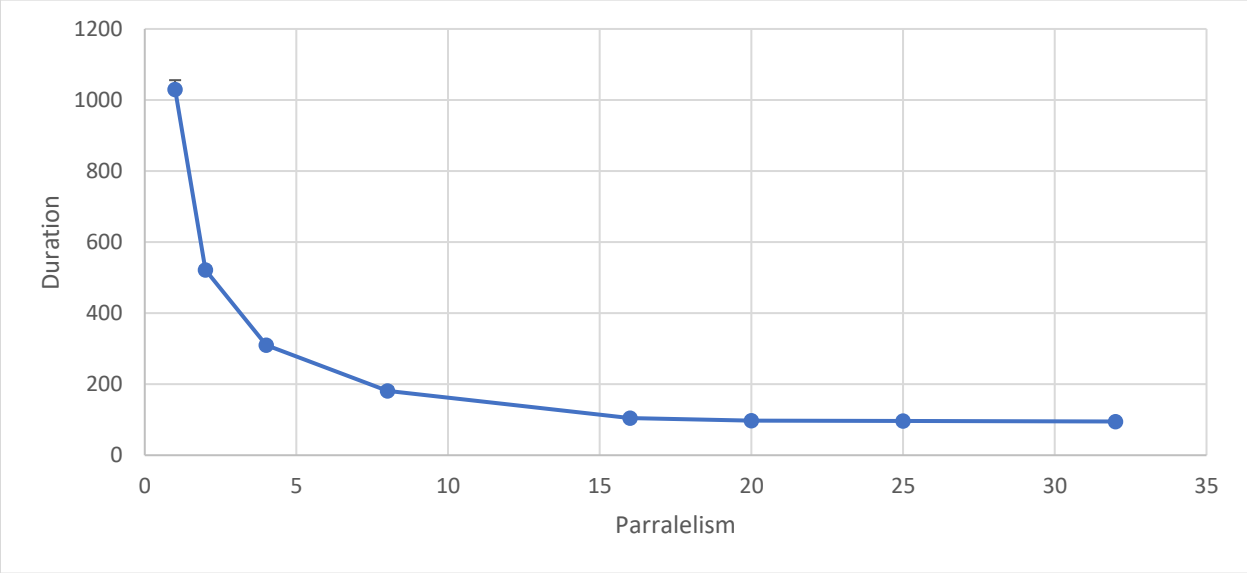
Εικόνα 23 Προσομοίωση πειράματος για παραλληλισμό = 4

Στην εικόνα 23 μπορούμε να διακρίνουμε την προσομοίωση του 1ου πειράματος (Εικόνα 19) στον cluster του Πολυτεχνείου Κρήτης με 200 μοναδικές στήλες και παραλληλισμό ίσο με 4. Την στιγμή που πάρθηκε το στιγμιότυπο είχα σταλεί περίπου 1000000 δεδομένα όπως μπορούμε να διακρίνουμε και από τον Data_Router.

5.2.2 Δεύτερο Μέρος Πειραμάτων

Όσο αφορά τη μελέτη της κύριας λειτουργίας της παρούσας εργασίας, που ήταν η εύρεση των top k συσχετισμένων στηλών οι παράμετροι που μεταβάλλονταν ήταν όμοιοι με το 1ο μέρος. Ο χρόνος που μετρήθηκε αφορούσε την αποστολή 1000 ερωτήσεων για την εύρεση μίας στήλη ανάμεσα στις συνόψεις που δημιουργήθηκαν και εν τέλει τον υπολογισμό της συσχέτισης. Για να πάρει μέρος η αποστολή των ερωτημάτων έπρεπε πρώτα να δημιουργηθούν όλες οι συνόψεις στο 1ο μέρος.

Scalability Συστήματος



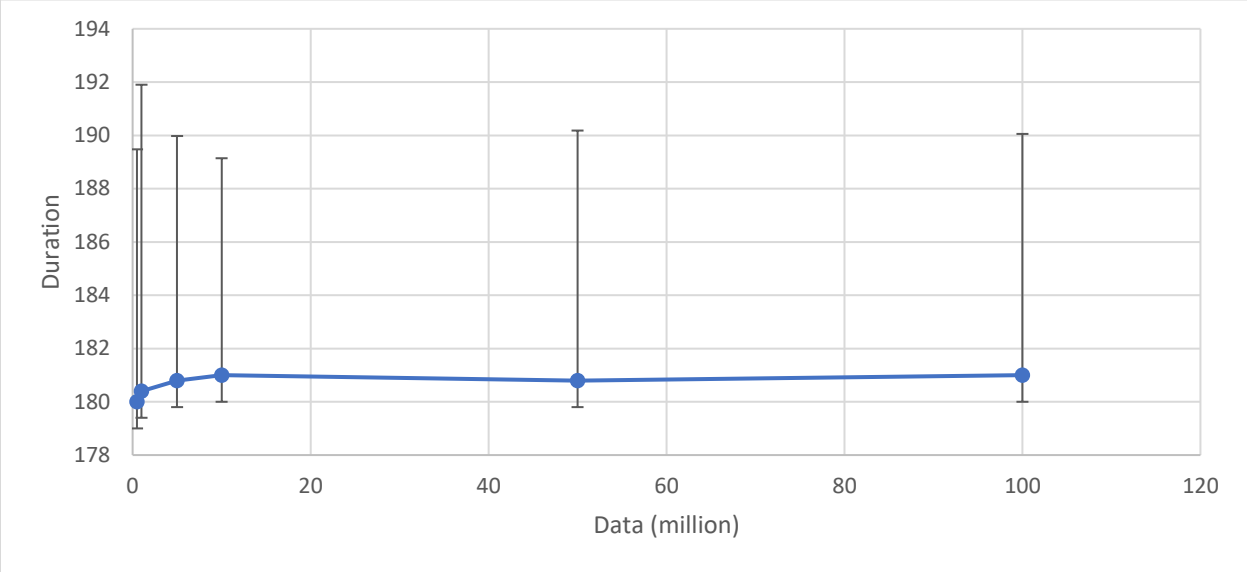
Εικόνα 24 Πείραμα για ερωτήματα συσχέτισης σύμφωνα με τον παραλληλισμό

Όπως μπορούμε να διαπιστώσουμε από την εικόνα 24, η διάρκεια αποστολής 1000 ερωτήσεων και ο υπολογισμός τους είναι ανάλογος του παραλληλισμού σαν το 1ο πείραμα αυξάνοντας γραμμικά μέχρι για παραλληλισμό 16. Κορυφώνεται στους 16 workers, καθώς ο κατακερματισμός των διεργασιών μεταξύ των workers δεν βελτιώνεται ουσιαστικά για παραλληλισμό ίσο με 32.

Parallelism	**1**	**2**	**4**	**8**	**16**	**20**	**25**	**32**
Standard deviation	**48.4767 9857**	**25.88 4358 21**	**9.633275 663**	**9.476286 192**	**11.18033 989**	**1.58113 883**	**3.435112 807**	**2.549 51**

Πίνακας 5 Τυπική απόκλιση σύμφωνα με το μέγεθος του παραλληλισμού

Αριθμός Δεδομένων



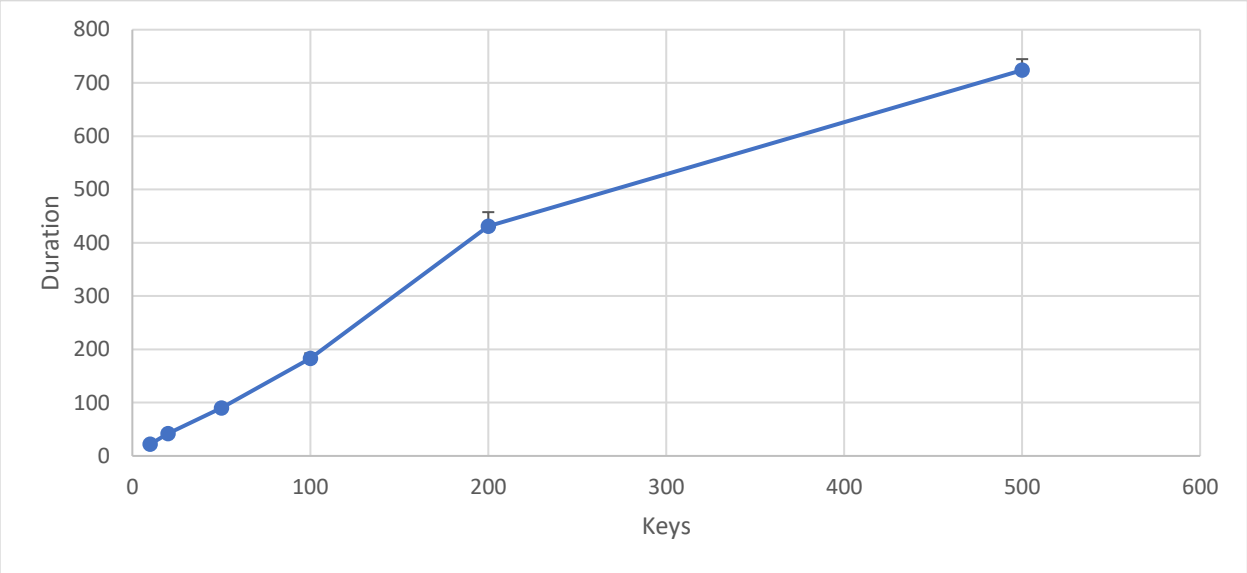
Εικόνα 25 Πείραμα για ερωτήματα συσχέτισης σύμφωνα με τα δεδομένα

Όσο αφορά τα αποτελέσματα της αναλογίας μεγέθους των δεδομένων που αποτέλεσαν το κτίσιμο της σύνοψης είναι αντίθετα από τη δημιουργία της σύνοψης όπως μπορούμε να διακρίνουμε από την εικόνα 25. Το αποτέλεσμα αυτό είναι λογικό, καθώς το μέγεθος της σύνοψης είναι σταθερό για οποιοδήποτε μέγεθος δεδομένων πριν τη δημιουργία της σύνοψης. Εάν η σύνοψη είναι γεμάτη και εξακολουθούν να έρχονται δεδομένα, τότε εφαρμόζεται ο αλγόριθμος, στον οποίο επιλέγονται τα κλειδιά με το μεγαλύτερο hu και τα άλλα αφαιρούνται. Έτσι, ανεξάρτητα από το μέγεθος του dataset το μέγεθος της σύνοψης είναι μία σταθερά από τον χρήστη. Μπορούμε να διακρίνουμε ότι ένα ερώτημα εκτελείτε με μέσο χρόνο 0,18 δευτερολέπτων.

Tuples	500,000	1,000,000	5,000,000	10,000,000	50,000,000	100,000,000
Standard deviation	9.476286192	11.50217371	9.176055798	8.142481194	9.380831522	9.055385138

Πίνακας 6 Τυπική απόκλιση σύμφωνα με το μέγεθος των δεδομένων

Αριθμός Στηλών



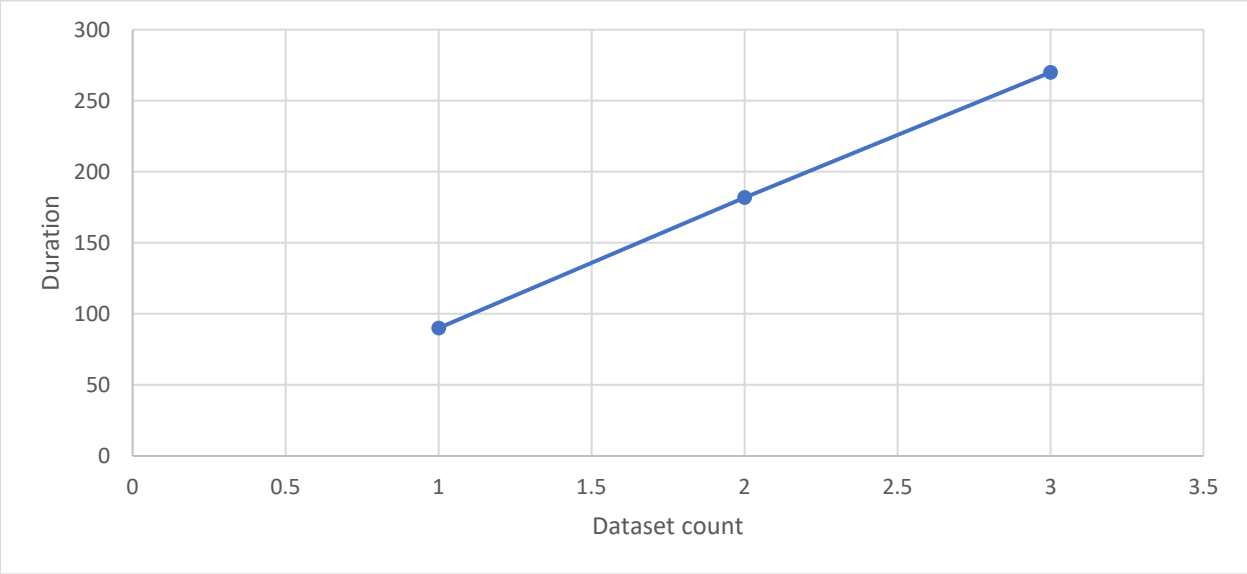
Εικόνα 26 Πείραμα για ερωτήματα συσχέτισης σύμφωνα με τις στήλες

Όσο αφορά τον αριθμό των μοναδικών στηλών, όπως μπορεί να διαπιστωθεί από την εικόνα 26, είναι ανάλογος με τη διάρκεια, αφού όσο περισσότερα μοναδικά κλειδιά υπάρχουν στο dataset τόσες περισσότερες συνόψεις θα πρέπει να δημιουργηθούν. Με τη δημιουργία περισσότερων συνόψεων απαιτούνται περισσότερες συγκρίσεις μεταξύ τους πράγμα που επιφέρει σημαντική καθυστέρηση στο σύστημα.

Keys	**10**	**20**	**50**	**100**	**200**	**500**
Standard deviation	**2.549509757**	**5.958187644**	**3.807886553**	**9.476286192**	**26.55183609**	**20.73644135**

Πίνακας 7 Τυπική απόκλιση σύμφωνα με το μέγεθος των στηλών

Αριθμός Dataset



Εικόνα 27 Πείραμα για ερωτήματα συσχέτισης σύμφωνα με τα dataset

Τα αποτελέσματα που εξάχθηκαν από τη διάρκεια σε συνάρτηση του αριθμού των dataset είναι παρόμοια με τα προηγούμενα (Εικόνα 26). Αυτό συμβαίνει για τον λόγο ότι στην περίπτωση δημιουργίας μία σύνοψης το κλειδί είναι ένας συνδυασμός dataset μαζί με το όνομα της στήλης. Έτσι για 2 dataset και 100 στήλες θα δημιουργηθούν 200 μοναδικά κλειδιά. Για 3 dataset 300 κλπ. Έτσι, θα υπάρχουν περισσότερες συνόψεις ,όπως στο προηγούμενο πείραμα με τις στήλες (κλειδιά).

Dataset	**1**	**2**	**3**
Standard deviation	**7.436396977**	**9.176055798**	**11.40175425**

Πίνακας 8 Τυπική απόκλιση σύμφωνα με το μέγεθος των dataset

Κεφάλαιο 6

6.1 Συμπεράσματα

Συμπερασματικά, στην παρούσα διπλωματική εργασία επιχειρήθηκε η κατάταξη συσχετισμένων στηλών στο Flink. Για να υλοποιηθεί προηγήθηκε η δημιουργία τις σύνοψης. Το 1^{ο} μέρος αποτελείτο από τη δημιουργία τις σύνοψης τοπικά για την ευκολότερη διεκπεραίωση πειραμάτων και την πιστοποίηση του αλγορίθμου, ο οποίος κατηγοριοποιεί τα δεδομένα εισόδου ανάλογα με την ομοιότητα τις μέσω τις σύνοψης KMV. Εν συνεχεία, έγινε εισαγωγή τις σύνοψης στο εσωτερικό του Synopsis Data Engine (SDE) για την κατανεμημένη και παράλληλη λειτουργεία του προγράμματος. Για την εισαγωγή και διαχείριση των δεδομένων γίνεται χρήση του εργαλείου Apache Kafka, το οποίο με τη σειρά του μετατρέπει τα δεδομένα σε streams μέσω του SDE. Η κάθε σύνοψη αποστέλλεται τις workers, όπου ανάλογα με τον παραλληλισμό προκύπτει και η πληθώρα των workers. Για το 2^{ο} μέρος, που ήταν και το ζητούμενο τις εργασίας πραγματοποιήθηκε η εύρεση των κ most correlated columns. Για την υλοποίηση πραγματοποιήθηκε η λειτουργία estimate. Για αυτήν τη λειτουργία αρχικά ζητείται από τον χρήστη η στήλη, η οποία θέλει να βρεθεί. Μέσω του ερωτήματος αυτού βρίσκεται η σύνοψη τις στήλης που έδωσε ο χρήστης και επιστρέφεται πίσω στο FlatMap. Το estimation, έπειτα, μετατρέπεται ξανά σε ερώτημα και αποστέλλεται σε τις τις συνόψεις, για να βρεθούν οι k πιο όμοιες στήλες. Για τη σύγκριση εφαρμόζεται η εξίσωση Pearson Correlation πολλαπλασιασμένη μαζί με ένα ποσοστό λάθους, που έχει σαν παράμετρο το μέγεθος τις σύνοψης. Για μεγαλύτερες συνόψεις η παράμετρος έχει μικρότερη τιμή. Τέλος, αφού βρεθούν τις οι τιμές συσχετίσεων μεταξύ των συνόψεων, κρατούνται οι k μεγαλύτερες και οι υπόλοιπες δεν αποθηκεύονται στον SDE.

Για να μετρηθεί η αποδοτικότητα και η σωστή λειτουργεία του αλγόριθμου ελέγχθηκε αρχικά τοπικά και έπειτα κατανεμημένα και απομακρυσμένα με τη χρήση των μηχανημάτων του cluster του Πολυτεχνείου Κρήτης για τις διαφορετικούς συνδυασμούς των παραμέτρων, που κρίθηκαν απαραίτητοι. Όσο αφορά την ακρίβεια των μετρήσεων τις συσχέτισης με τη χρήση των συνόψεων είναι αρκετά ακριβείς. Τις, όσο αφορά την απόδοση του μοντέλου κατά την κλιμάκωση του αλγορίθμου προέκυψαν αρκετά θετικά αποτελέσματα με την επιθυμητή συμπεριφορά με πολύ μικρό ποσοστό λάθους.

6.2 Μελλοντικές Επεκτάσεις

Για την καλύτερη λειτουργία του αλγορίθμου θα μπορούσαν να γίνουν βελτιώσεις στον υπολογισμό τις παραμέτρου που αφορά το ποσοστό λάθους, καθώς λαμβάνεται υπόψιν μόνο το μέγεθος τις σύνοψης. Οι αλγόριθμοι που παρουσιάζονται στο paper που βασίστηκε η εργασία θα μπορούσαν να εξελιχθούν, έτσι ώστε να μην χρειάζεται μεγάλος αριθμός στην ανταλλαγή στηλών πράγμα που θα καθιστούσε τις μετρήσεις συσχέτισής πιο ακριβείς.

Βιβλιογραφία

[1] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In J. D. P. Rolim and S. Vadhan, editors, Randomization and Approximation Techniques in Computer Science, pages 1–10, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[2] http://users.softnet.tuc.gr/~ngiatrakos/papers/cikm2020a.pdf

[3] Ροές Δεδομένων Δεκαπλασιάστηκε μέσα σε πέντε χρόνια ο όγκος δεδομένων των εταιρειών | Η ΚΑΘΗΜΕΡΙΝΗ

[4] Apache Kafka https://www.ibm.com/topics/apache-kafka\

[5] Neha Narkhede, 2017, pp. 39-44

[6] Data streans https://www.snowflake.com/guides/data-streams

[7] Apache Flink https://nexocode.com/blog/posts/what-is-apache-flink/

[8] A. Santos https://dl.acm.org/doi/10.1145/3448016.3458456?cid=99658620176

[9] MurmurHash3 https://github.com/aappleby/smhasher/blob/master/src/MurmurHash3.cpp

[10] Fibonacci Hashing https://probablydance.com/2018/06/16/fibonacci-hashing-the-optimization-that-the-world-forgot-or-a-better-alternative-to-integer-modulo/

[11] C. Manara. Πρόβλεψη Μετοχών στο Apache Flink

[12] M. Sotiria. Αποδοτική Πρόβλεψη καρκινικών κυττάρων στο Flink

[13] Apache Flink https://ibm-cloud-architecture.github.io/refarch-eda/technology/flink/

[14] Apache Kafka https://www.projectpro.io/article/apache-kafka-architecture-/442

[15] Bivariate Normal Distribution https://www.probabilitycourse.com/chapter5/5_3_2_bivariate_normal_dist.php

[16] SDE https://github.com/akontaxakis/INFORE_Synopses_Data_Engine

[17] KMV sketch https://www.youtube.com/watch?v=O3Z_ozOn6E8&ab_channel=dbislab

[18] Pearson Correlation https://www.youtube.com/watch?v=WpZi02ulCvQ&ab_channel=OneMinuteEconomics

[19] Fibonacci Hashing https://github.com/asamolov/fib-modulo-benchmark/blob/master/README.md

[20] KMV https://www.uni-mannheim.de/media/Einrichtungen/dws/Files_People/Profs/rgemulla/publications/beyer07distinct.pdf