ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Comparison of Different Clustering Algorithms for Diagnosing Memory-Related Performance Issues Using a Distributed Computing System

Στυλιανός Ι. Πουλακάκης

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Comparison of Different Clustering Algorithms for Diagnosing Memory-Related Performance Issues Using a Distributed Computing System

Στυλιανός Ι. Πουλακάκης

Α.Μ.: Μ1441

ΕΠΙΒΛΕΠΩΝ: Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

Μάρτιος 2017

ΠΕΡΙΛΗΨΗ

Αποτυχίες δημοφιλών συστημάτων μεγάλων τεχνολογικών κολοσσών καθιστούν την εφαρμογή των δοκιμών φορτίου (load tests) απαραίτητη για τον έλεγχο των συστημάτων λογισμικού. Παρόλα αυτά, η διάγνωση των προβλημάτων που σχετίζονται με την μνήμη αποτελεί μια σημαντική πρόκληση για τους προγραμματιστές. Για την αντιμετώπιση τους, εφαρμόζονται συχνά αυτοματοποιημένες τεχνικές ανάλυσης οι οποίες όμως απαιτούν σημαντική χειροκίνητη προσπάθεια και υψηλό βαθμό γνώσης του συστήματος. Μια λύση στο πρόβλημα αυτό, αποτελεί η χρήση τεχνικών της μηχανικής μάθησης (machine learning) με σκοπό την διάγνωση της υπάρχουσας ανώμαλης συμπεριφοράς του συστήματος. Οι Mark D. Syer et al. προτείνουν μια νέα αυτοματοποιημένη προσέγγιση συνδυάζοντας τους μετρητές απόδοσης (performance counters) και τα αρχεία εκτέλεσης (execution logs) εφαρμόζοντας την ιεραρχική ομαδοποίηση (hierarchical clustering) για την συσταδοποίηση των δεδομένων. Η ομαδοποίηση αυτή όμως, αποτυγχάνει σε περιπτώσεις μεγάλων δεδομένων (big data) καθώς παρουσιάζει μεγάλη πολυπλοκότητα. Εμείς, εφαρμόζουμε μια διαφορετική προσέγγιση του αλγορίθμου του Syer εκμεταλλευόμενοι το πλεονέκτημα του παραλληλισμού των ποικίλων διεργασιών που μας παρέχει το Spark framework. Βασιζόμενοι σε μια προηγούμενη εταιρική υλοποίηση του αλγόριθμου, στη φάση της συσταδοποίησης, εφαρμόζουμε τον k-means αλγόριθμο, αντί της ιεραρχικής, ώστε να αξιολογήσουμε τη συμπεριφορά των δυο αλγορίθμων για μεγάλα δεδομένα αλλά και αλγοριθμικά ως κομμάτι της προσέγγισης του Syer. Για την αξιολόγηση χρησιμοποιούμε συνθετικά δεδομένα από ένα πρόγραμμα υλοποίησης της Software Competitiveness International αλλά και πραγματικά δεδομένα από την εφαρμογή του Apache Tomcat έχοντας εισάγει ένα memory spike. Όσον αφορά τα αποτελέσματα, η προσέγγιση μας ανιχνεύει με ικανοποιητική ακρίβεια memory spikes ή συστάδες οι οποίες τα περιέχουν. Τέλος, σε περιπτώσεις μεγάλων σετ δεδομένων, τα αποτελέσματα που προκύπτουν, καθιστούν τον k-means αλγόριθμο καλύτερο ως προς τον χρόνο εκτέλεσης και την απόδοση σε σχέση με την ιεραρχική ομαδοποίηση.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Απόδοσης Λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Δοκιμές Φορτίου, Μετρητές Απόδοσης, Αρχεία Εκτέλεσης, Κατανεμημένα Υπολογιστικά Συστήματα, Ανάλυση Συστάδων

ABSTRACT

Failures in popular systems of technological giants illustrate load testing is a necessary procedure for the quality of software systems. However, the diagnosis of memory-related issues is a major challenge for developers. To address them, they often apply automated analysis techniques which require considerable manual effort and a high degree of system knowledge. One solution to this problem is the application of machine learning techniques to diagnose the existing abnormal system behavior. Mark D. Syer et al. propose a new automated approach combining performance counters and executing files by applying hierarchical clustering for clustering data. This grouping, however, fails in the case of large data sets as it generates greater complexity. We apply a different approach to the algorithm of Syer by using the Spark framework which offers parallelism of processes. Based on a previous corporate implementation of the algorithm, we apply the k-means algorithm in the clustering phase instead of the hierarchical clustering. This is done in order to evaluate the behavior of the two algorithms for large data sets and validate the k-means algorithm as part of the overall Syer approach. Our case studies use performance counters and execution logs from two systems. For the evaluation, we use synthetic data from one program created by Software Competitiveness International and actual data from the implementation of Apache Tomcat with an injection of a memory spike. Our approach identifies memory spikes corresponding to log lines with a high degree of precision. The approach detects a fairly accurate number of individual memory spikes or the clusters containing them. Finally, in the case of large data sets, the k-means algorithm performs better in terms of execution time and performance than hierarchical clustering.

SUBJECT AREA: Performance Software Engineering

KEYWORDS: Load Testing, Performance Counters, Execution Logs, Distributed Computing Systems, Cluster Analysis

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα εργασία αποτελεί διπλωματική εργασία στα πλαίσια του μεταπτυχιακού διπλώματος ειδίκευσης «Διαχείριση Πληροφορίας και Δεδομένων» του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Καποδιστριακού Πανεπιστημίου Αθηνών. Πριν την παρουσίαση των αποτελεσμάτων της παρούσας διπλωματικής εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω ορισμένους από τους ανθρώπους που γνώρισα, συνεργάστηκα μαζί τους και έπαιξαν πολύ σημαντικό ρόλο στην πραγματοποίησή της. Πρώτο από όλους θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής εργασίας, Αναπληρωτή Καθηγητή Ευστάθιο Χατζηευθυμιάδη για την πολύτιμη καθοδήγηση και εκτίμηση που μου έδειξε. Αισθάνομαι ιδιαίτερη ανάγκη να ευχαριστήσω την κ. Ζωή Αικατερινίδη, CEO της εταιρίας Software Competitiveness International, η οποία με εμπιστεύτηκε και μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο σημαντικό και ενδιαφέρον project. Ιδιαίτερες ευχαριστίες θέλω να απευθύνω στους συνεργάτες μου, Πασχάλη Βέσκο και Στάθη Κουκουβίνο για την καθοριστική τους βοήθεια, οι οποίοι στάθηκαν σημαντικοί αρωγοί στην προσπάθειά μου και με υποστήριξαν σε κάθε φάση της πορείας μου. Τέλος, θέλω να ευχαριστήσω τους γονείς μου Ιωάννη και Παρασκευή, καθώς και τον αδερφό μου Χάρη, που με υπομονή και κουράγιο πρόσφεραν την απαραίτητη ηθική συμπαράσταση για την ολοκλήρωση της διπλωματικής μου εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 12

1. ΕΙΣΑΓΩΓΗ 13

1.1 Η σημασία του Performance Engineering 13

1.2 Το πρόβλημα που εντοπίζεται 13

1.3 Μια άλλη προσέγγιση 14

1.4 Η συνεισφορά μας 14

1.5 Η οργάνωση της εργασίας 15

2. BACKGROUND-ΣΧΕΤΙΚΗ ΔΟΥΛΕΙΑ 16

2.1 Software Performance Engineering 16

2.2 Performance Testing 16

2.2.1 Load Testing 19

2.2.2 Stress Testing 21

2.3 Memory Related Issues 23

2.3.1 Persistent Memory Issues 23

2.3.2 Transient Memory Issues 26

2.4 Performance Counters 27

2.5 Execution Logs 28

2.6 Ο αλγόριθμος του Syer 32

2.6.1 Ένα απλό παράδειγμα 32

2.6.2 Data Preparation 36

2.6.3 Hierarchical Clustering 38

2.6.4 Cluster Analysis 44

3. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ 50

3.1 Distributed Computing 50

3.1.1 Apache Hadoop 52

3.1.2 Apache Spark 56
3.2 Java 8 65
3.2.1 Streams 65
3.2.2 Lambda Expressions 67
3.3 K-Means 68
3.3.1 Κριτήρια επιλογής του k 72
3.3.2 Διαφορές k-means αλγορίθμου και ιεραρχικής ομαδοποίησης 76
3.4 Το Cluster των υπολογιστών 76
3.5 Performance Counter Collection 78
3.6 Εφαρμογή ενός μικρού σετ δεδομένων 81
4. **ΑΠΟΤΕΛΕΣΜΑΤΑ** 85
4.1 Χρήση της Elbow Μεθόδου 88
4.2 Memory Spike Size Case Study 90
4.3 Memory Spike Weight Case Study 96
4.4 Stress Test Case Study 103
4.5 Apache Tomcat Real Case Study 109
4.5.1 Διαδικασία εκχώρησης ενός Memory Spike 110
4.5.2 Αποτελέσματα της εκτέλεσης 111
5. ΣΥΜΠΕΡΑΣΜΑΤΑ 114
ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ 116
ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ 118
ΠΑΡΑΡΤΗΜΑ Ι 119
ΠΑΡΑΡΤΗΜΑ ΙΙ 122
ΑΝΑΦΟΡΕΣ 123

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means με και χωρίς χρήση της elbow μεθόδου 89

Σχήμα 2: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means με και χωρίς χρήση της elbow μεθόδου 90

Σχήμα 3: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο ... 91

Σχήμα 4: Υπολογισμός των precision και recall εκτελώντας την ιεραρχική ομαδοποίηση 92

Σχήμα 5: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 93

Σχήμα 6: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 94

Σχήμα 7: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση για το 10KB data set με εφαρμογή διαφορετικών χρόνων δειγματοληψίας. 95

Σχήμα 8: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 96

Σχήμα 9: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο ... 98

Σχήμα 10: Υπολογισμός των precision και recall εκτελώντας την ιεραρχική ομαδοποίηση 99

Σχήμα 11: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 100

Σχήμα 12: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 101

Σχήμα 13: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 102

Σχήμα 14: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο 104

Σχήμα 15: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 105

Σχήμα 16: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση 106

Σχήμα 17: Εύρεση των memory spikes του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης 107

Σχήμα 18: Χρόνοι εκτέλεσης του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης 108

Σχήμα 19: Χρόνοι εκτέλεσης της φάσης συσταδοποίησης του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης 109

Σχήμα 20: Το χρησιμοποιημένο μέγεθος του σωρού και το memory spike που διαγνώστηκε 113

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1: Τεχνικές βαθμολόγησης για τον εντοπισμό των συστάδων που σχετίζονται με κάποιο θέμα διευθέτησης της μνήμης.....46

Πίνακας 2: Αριθμός των memory spikes στα W τεστ δεδομένων 97

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Χαρακτηριστικά μιας δοκιμής απόδοσης....17
Εικόνα 2: Βήματα των δοκιμών φορτίου....20
Εικόνα 3: Περιγραφή Stress Testing....22
Εικόνα 4: Σχεδιάγραμμα μιας διαρροής μνήμης....24
Εικόνα 5 :Σχέδιο ενός memory bloat....25
Εικόνα 6: Memory bloat vs memory leak....26
Εικόνα 7: Γραφική παράσταση μετρητών απόδοσης....28
Εικόνα 8: Βήματα τροποποίησης των αρχείων καταγραφής....28
Εικόνα 9: Παράδειγμα των γεγονότων εκτέλεσης....30
Εικόνα 10: Παράδειγμα των transaction logs....31
Εικόνα 11: Παράδειγμα των message logs....32
Εικόνα 12: Οι υποχρεώσεις ενός αναλυτή απόδοσης....33
Εικόνα 13: Βήματα εκτέλεσης του αλγορίθμου των Mark D. Syer et al. με χρήση της ιεραρχικής ομαδοποίησης....35
Εικόνα 14: Παράδειγμα απαλοιφής της δυναμικής πληροφορίας των γεγονότων εκτέλεσης....36
Εικόνα 15: Παράδειγμα συσταδοποίησης δεδομένων....38
Εικόνα 16: Αθροιστική και διαιρετική ιεραρχική ομαδοποίηση....39
Εικόνα 17: Δενδρόγραμμα των συστάδων....40
Εικόνα 18: Διαφορετικές μορφές σύνδεσης της ιεραρχικής ομαδοποίησης....43
Εικόνα 19: Δημιουργία διαφορετικού clustering με κοπή του δενδρογράμματος....44
Εικόνα 20: Εύρεση μίας απομακρυσμένης συστάδας....45
Εικόνα 21: Το επίπεδο επιρροής που ασκεί μια απομακρυσμένη συστάδα....48
Εικόνα 22: Σύνδεση ενός κατανεμημένου συστήματος υπολογιστών....51
Εικόνα 23: Λειτουργία του Hadoop....53
Εικόνα 24: Στάδια της MapReduce διαδικασίας....53
Εικόνα 25: Τα κύρια συστατικά της MapReduce λειτουργίας....55
Εικόνα 26: Χρήση του HDFS και της MapReduce λειτουργίας....56
Εικόνα 27: Τα κύρια συστατικά της Spark εφαρμογής....58
Εικόνα 28: Η σύνδεση του Hadoop με το Spark framework....59
Εικόνα 29: Διαίρεση ενός RDD σε partitions στην μνήμη....60
Εικόνα 30: Transformations και actions σε ένα RDD....61
Εικόνα 31: Ο κατευθυνόμενος άκυκλος γράφος στο Spark....63
Εικόνα 32: Η σύνδεση των κύριων συστατικών μιας Spark εφαρμογής....64
Εικόνα 33: Η βιβλιοθήκη MLlib του Spark framework....65

Εικόνα 34: Ένα παράδειγμα ροών.....66
Εικόνα 35: Τα κύρια συστατικά μιας ροής.....67
Εικόνα 36: Μία λάμδα έκφραση της Java 8 67
Εικόνα 37: Βήματα εκτέλεσης του αλγορίθμου του Mark D. Syer εφαρμόζοντας την k-means μέθοδο. 71
Εικόνα 38: Ο σχηματισμός μια μορφής «αγκώνα» 73
Εικόνα 39: Η διαδικασία της Cross Validation μεθόδου.....74
Εικόνα 40: Η σιλουέτα σε συνάρτηση με τον αριθμό των συστάδων.....75
Εικόνα 41: Το cluster των υπολογιστών που χρησιμοποιήσαμε για την εκτέλεση των σετ δεδομένων και η μεταξύ τους σύνδεση.....78
Εικόνα 42: Τα κύρια χαρακτηριστικά των επεκτάσεων JMX.....80
Εικόνα 43: Η γραφική παράσταση του αριθμού k των συστάδων σε συνάρτηση με το άθροισμα των τετραγώνων.....83
Εικόνα 44: Ο οθόνη του HDFS περιλαμβάνοντας όλα τα στοιχεία εξόδου της εφαρμογής μας.....86
Εικόνα 45: Οι μετρήσεις του used heap size που λάβαμε χάρις τους μετρητές απόδοσης 112

ΠΡΟΛΟΓΟΣ

Η διπλωματική αυτή εργασία πραγματοποιήθηκε στο χώρο της Software Competitiveness International (Softcom International). Η Softcom International είναι μια ταχέως αναπτυσσόμενη Εταιρεία, που ειδικεύεται σε λύσεις Έρευνας & Ανάπτυξης Λογισμικού και Εξατομικευμένες Υπηρεσίες Πληροφορικής, με γραφεία στην Αθήνα, έδρα στη Σητεία της Κρήτης και σημεία πώλησης στη Γερμανία. Σήμερα η Εταιρεία έχει παρουσία στην Ελληνική αγορά καθώς και έχει συνάψει μακροχρόνιες συνεργασίες με μεσαίες και μεγάλες πολυεθνικές Εταιρείες τόσο στη Γερμανία όσο και την Ελβετία και Γαλλία.

1. ΕΙΣΑΓΩΓΗ

1.1 Η σημασία του Performance Engineering

Η αύξηση των κατ' εξοχήν συστημάτων λογισμικού μεγάλης κλίμακας (ultra large-scale software systems) [1] θέτει πολλές νέες προκλήσεις όσον αφορά το πεδίο της μηχανικής απόδοσης λογισμικού (software performance engineering). Αποτυχίες δημοφιλών συστημάτων μεγάλων τεχνολογικών κολοσσών (όπως η Apple και η Google), είχαν καταστροφικές οικονομικές επιπτώσεις. Οι αποτυχίες αυτές συνδέονται περισσότερο με ζητήματα απόδοσης και κλιμάκωσης (scalability) παρά με σφάλματα κάποιων συγκεκριμένων χαρακτηριστικών (feature bugs) [2] [3].

Οι δοκιμές απόδοσης κρίνονται απαραίτητες ώστε να προσδιοριστεί η σταθερότητα ενός συστήματος και το πόσο καλά ανταποκρίνεται κάτω από ένα συγκεκριμένο φόρτο εργασίας [4], [5]. Οι αναλυτές απόδοσης (performance analysts) είναι υπεύθυνοι για την εκτέλεση δοκιμών φορτίου (load testing) χάρις τις οποίες παρακολουθούν την συμπεριφορά του συστήματος δεχόμενο ρεαλιστικά φορτία δεδομένων, και πρέπει να εξασφαλίσουν ότι τα ULS (ultra large-scale) συστήματα θα είναι σε θέση να αποδίδουν υπό τις συγκεκριμένες συνθήκες. Τέτοιες δοκιμές φορτίου επιτρέπουν στους αναλυτές να προσδιορίζουν την μέγιστη λειτουργική ικανότητα ενός συστήματος, τις λειτουργικές απαιτήσεις αλλά και τα σημεία συμφόρησης που εντοπίζονται .

1.2 Το πρόβλημα που εντοπίζεται

Παρά τη σημασία των δοκιμών φορτίου, οι τρέχουσες τεχνικές ανάλυσης απαιτούν σημαντική χειροκίνητη προσπάθεια και υψηλό βαθμό γνώσης του συστήματος [2], [6], [7]. Οι αναλυτές απόδοσης οφείλουν να εξετάσουν εκατοντάδες megabytes ή gigabytes μετρητών απόδοσης (performance counters) για να ελέγξουν τη χρήση των πόρων και αρχείων καταγραφής εκτέλεσης (execution logs) για να κατανοήσουν πλήρως τη συμπεριφορά του συστήματος.

Υπάρχουν πολλά εργαλεία και τεχνικές που αφορούν την μεμονωμένη χρήση είτε των μετρητών απόδοσης, είτε των αρχείων εκτέλεσης. Παρόλα αυτά, μικρή έρευνα έχει διεξαχθεί όσον αφορά τον συνδυασμό αυτών των δυο συνόλων για την διάγνωση σχετιζόμενων με την μνήμη προβλημάτων (memory-related issues) τα οποία είναι πιθανό να προκαλέσουν την κατάρρευση συστημάτων ή την δυσλειτουργία εφαρμογών σε κάποια στιγμή της εκτέλεσης τους. Ο συνδυασμός των μετρητών απόδοσης και των αρχείων εκτέλεσης μπορεί να προσφέρει την δυνατότητα στους αναλυτές απόδοσης να εστιάσουν την προσοχή τους σε ένα μικρό κομμάτι του κώδικα εξοικονομώντας έτσι αρκετό χρόνο και κόπο και να αντιμετωπίσουν αποτελεσματικά τα προβλήματα τα οποία εντοπίζονται.

Οι ειδικοί απόδοσης βρίσκονται συχνά αντιμέτωποι με προβλήματα διευθέτησης της μνήμης τα οποία μπορούν να επηρεάσουν την συμπεριφορά ενός συστήματος μειώνοντας τις επιδόσεις του και πολλές φορές προκαλώντας τους σύντριβες (crushes). Τα προβλήματα αυτά είναι δύσκολο να διαγνωστούν από τους αναλυτές και μπορούν να χωριστούν σε δυο κατηγορίες, σε παροδικά (transient memory-related issues) και στα επίμονα (persistent memory-related issues).

1.3 Μια άλλη προσέγγιση

Ο Mark D. Syer et al. [8] προτείνουν μια νέα αυτοματοποιημένη προσέγγιση ώστε να υποστηρίξουν τους αναλυτές απόδοσης στην διάγνωση προβλημάτων σχετικών με την μνήμη, συνδυάζοντας τους μετρητές απόδοσης και των αρχείων εκτέλεσης. Σε πρώτο στάδιο, η δυναμική πληροφορία των αρχείων εκτέλεσης μετατρέπεται σε στατικά γεγονότα εκτέλεσης (execution events). Τα δεδομένα προετοιμάζονται με τέτοιον τρόπο ώστε να μπορούν έπειτα να χρησιμοποιηθούν για αυτοματοποιημένη, στατιστική ανάλυση. Οι μετρητές απόδοσης συνδυάζονται με τα γεγονότα εκτέλεσης και κατηγοριοποιούνται σε χρόνο-φέτες προφίλ (time-slice profiles).

Στην συνέχεια, υπολογίζεται ο πίνακας αποστάσεων (distance matrix). Το κάθε time-slice profile παριστάνεται ως ένα σημείο σε έναν χώρο πολλών διαστάσεων. Ο υπολογισμός του πινάκα αυτού γίνεται με την χρήση της Pearson απόστασης. Έπειτα, χρησιμοποιείται η μέθοδος της ιεραρχικής ομαδοποίησης για την ανάθεση των time-slice προφίλ σε συστάδες. Το τελικό στάδιο αποτελεί η χρήση αυτοματοποιημένων στατιστικών τεχνικών για τον εντοπισμό του συνόλου των γραμμών καταγραφής αρχείου (log lines) που σχετίζεται με προβλήματα διευθέτησης της μνήμης.

Η προσέγγιση αυτή παρουσιάζει πολλές δυσκολίες καθώς απαιτεί υψηλή υπολογιστική ισχύ και χαρακτηρίζεται από μεγάλη πολυπλοκότητα. Σε ένα σύστημα ή εφαρμογή, που περιέχει τεράστιο όγκο δεδομένων και λειτουργιών, η εύρεση των log lines που σχετίζονται με κάποιο συγκεκριμένο πρόβλημα διευθέτησης της μνήμης αποτελεί μία δύσκολη έως αδύνατη διαδικασία.

1.4 Η συνεισφορά μας

Η διπλωματική εργασία αυτή, χρησιμοποιεί ως σύστημα αναφοράς μια προηγούμενη υλοποίηση του αλγορίθμου από την Software Competitiveness International που προτείνουν οι Syer et al. παρουσιάζοντας έναν τρόπο εκτέλεσης και ανάλυσης της προσέγγισης αυτής χρησιμοποιώντας εργαλεία εξόρυξης δεδομένων (data mining) και μεγάλων δεδομένων. Τα στάδια υλοποίησης είναι τα ίδια με αυτά τα οποία παρουσιάζονται στην δημοσίευση του Syer με την κύρια διαφορά ότι παρέχονται λύσεις κατανεμημένων υπολογιστικών συστημάτων και η δυνατότητα παραλληλισμού των ποικίλων διεργασιών.

Η υλοποίηση αυτή έγινε με βάση το Apache Spark framework και των βιβλιοθηκών που παρέχει η MLlib χρησιμοποιώντας την έκδοση 8 της Java η οποία παρέχει την δυνατότητα χρήσης ροών (streams) και λάμδα εκφράσεων (lambda expressions). Βασιζόμενη σ’ αυτήν την υλοποίηση, όσον αφορά το στάδιο της ομαδοποίησης, εφαρμόζεται ο k-means αλγόριθμος συσταδοποίησης σε αντίθεση με την ιεραρχική ομαδοποίηση. Τα επί μέρους αποτελέσματα καθώς και οι κύριες διαφορές της εκτέλεσης των σετ δεδομένων, παρουσιάζονται στο κεφάλαιο των αποτελεσμάτων.

Για την εκτέλεση των σετ δεδομένων, χρησιμοποιήθηκε ένα cluster συνολικά τριών υπολογιστών. Τα δεδομένα που χρησιμοποιήθηκαν (μετρητές απόδοσης καθώς και αρχεία εκτέλεσης) προέρχονται από δύο πηγές. Η πρώτη αποτελεί ένα πρόγραμμα παραγωγής συνθετικών δεδομένων υλοποιημένο από τη Software Competitiveness International ενώ η δεύτερη, η εφαρμογή Apache Tomcat. Περισσότερες πληροφορίες για τα σετ δεδομένων που εφαρμόστηκαν, την υλοποίηση και τα χαρακτηριστικά των υπολογιστών που χρησιμοποιήθηκαν παρατίθενται στα επόμενα κεφάλαια.

1.5 Η οργάνωση της εργασίας

Η διπλωματική αυτή εργασία, οργανώνεται ως εξής:

Το 2ο Κεφάλαιο, περιλαμβάνει ένα background των δεδομένων που θα χρησιμοποιήσουμε στην εργασία αυτή, δηλαδή πληροφορίες σχετικά με θέματα που σχετίζονται με την μνήμη και την αντιμετώπιση τους με χρήση μετρητών απόδοσης και γεγονότων εκτέλεσης. Επιπλέον, παρατίθεται ο αλγόριθμος του Mark D. Syer και αναλύονται τα βήματα εκτέλεσης του χρησιμοποιώντας την ιεραρχική ομαδοποίηση.

Το 3ο Κεφάλαιο, παρουσιάζει μια αναλυτική περιγραφή της υλοποίησης του αλγορίθμου με χρήση του k-means αλγορίθμου εφαρμόζοντας παράλληλους υπολογισμούς χάρις το Spark framework.

Το 4ο Κεφάλαιο περιλαμβάνει τα αποτελέσματα της εκτέλεσης του αλγορίθμου καθώς επίσης και μια σύγκριση των αποτελεσμάτων αυτών, μεταξύ της υλοποίησης με χρήση της ιεραρχικής ομαδοποίησης και της δικής μας προσέγγισης, του k-means αλγορίθμου.

Τέλος, το 5ο Κεφάλαιο συνοψίζει την διπλωματική εργασία.

2. BACKGROUND-ΣΧΕΤΙΚΗ ΔΟΥΛΕΙΑ

2.1 Software Performance Engineering

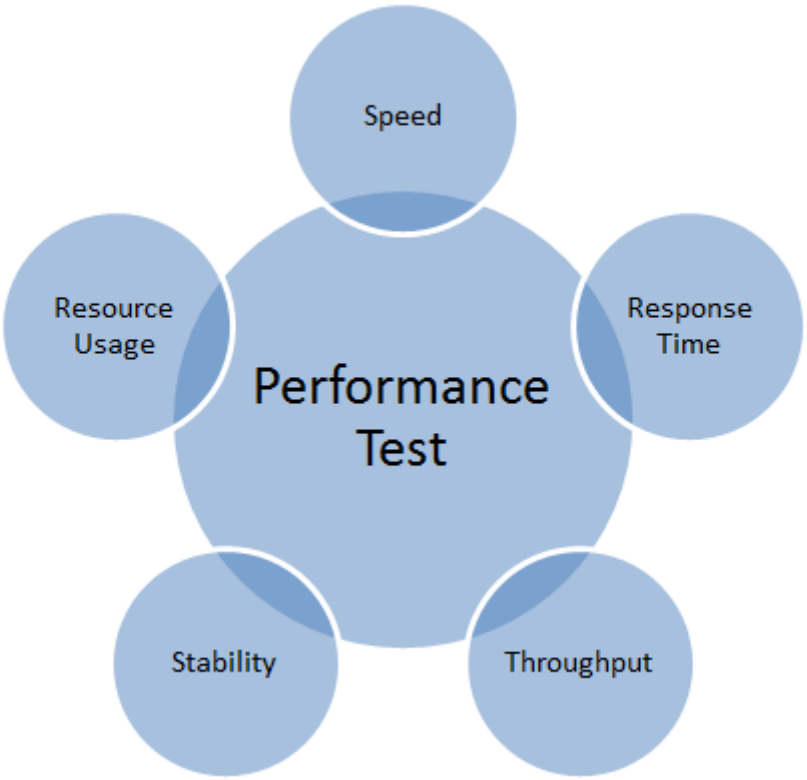
Στο πλαίσιο της μηχανικής συστημάτων, η μηχανική απόδοσης λογισμικού ή SPE (software performance engineering), περιλαμβάνει το σύνολο των ρόλων, των δεξιοτήτων, των δραστηριοτήτων, των πρακτικών και των εργαλείων που εφαρμόζονται σε κάθε φάση του κύκλου ζωής ανάπτυξης των συστημάτων. Η SPE εξασφαλίζει τον σχεδιασμό και την υλοποίηση της λειτουργικής λύσης η οποία θα πληροί τις μη λειτουργικές απαιτήσεις απόδοσης (όπως το throughput, η λανθάνουσα κατάσταση, ή η χρήση της μνήμης).

Στο πλαίσιο της μηχανικής λογισμικού, η SPE μπορεί να αναφέρεται εναλλακτικά ως μηχανική απόδοσης των εφαρμογών [9]. Καθώς η σύνδεση μεταξύ της επιτυχίας των εφαρμογών και της επιτυχίας των επιχειρήσεων συνεχίζει να κερδίζει την αναγνώριση, η μηχανική απόδοσης λογισμικού έχει αποκτήσει έναν πολύ σημαντικό ρόλο στο πλαίσιο ανάπτυξης του κύκλου ζωής του λογισμικού. Ως εκ τούτου, ο όρος χρησιμοποιείται συνήθως για να περιγράψει τις διαδικασίες, τους ανθρώπους και τις τεχνολογίες που απαιτούνται για την αποτελεσματική δοκιμή μη λειτουργικών απαιτήσεων, στην εξασφάλιση της τήρησης των υπηρεσιών και τέλος στην βελτιστοποίηση της απόδοσης των εφαρμογών πριν από την ανάπτυξη.

Η απόδοση είναι ένα σημαντικό χαρακτηριστικό της ποιότητας του κάθε συστήματος λογισμικού. Πολλά συστήματα λογισμικού, με αντικειμενοστραφή και μη προσανατολισμό (object και non-object-oriented software systems) δεν μπορούν να λειτουργήσουν λόγω προβλημάτων απόδοσης. Για παράδειγμα, το σύστημα ή μία εφαρμογή η οποία βρίσκεται στο τελικό στάδιο, μπορεί να μην ανταποκρίνεται αρκετά γρήγορα στις ενέργειες του χρήστη, ή να αδυνατεί να διαχειριστεί τον αριθμό των συναλλαγών που λαμβάνουν χώρα κατά την ώρα αιχμής όπου ο φόρτος εργασίας είναι αρκετά μεγάλος. Μία άλλη περίπτωση, είναι ένα ενσωματωμένο σύστημα το οποίο μπορεί να μην ανταποκρίνεται αρκετά γρήγορα σε ένα εξωτερικό ερέθισμα, ή να είναι σε θέση να μην μπορεί να επεξεργαστεί γεγονότα τα οποία συμβαίνουν με υψηλή συχνότητα.

2.2 Performance Testing

Στην τεχνολογία λογισμικού (software engineering), ο έλεγχος απόδοσης (performance testing) αποτελεί ένα υποσύνολο της μηχανικής απόδοσης λογισμικού και σε γενικές γραμμές είναι μια εφαρμογή δοκιμών για να προσδιοριστεί η σταθερότητα ενός συστήματος και το πόσο καλά ανταποκρίνεται κάτω από ένα συγκεκριμένο φόρτο εργασίας [10]. Μπορεί επίσης να εφαρμοστεί με κύριο σκοπό τη διερεύνηση ή την επαλήθευση ποιοτικών χαρακτηριστικών του συστήματος, όπως η επεκτασιμότητα, η αξιοπιστία και η χρήση των πόρων.



Εικόνα 1: Χαρακτηριστικά μιας δοκιμής απόδοσης [11]

Περά από το γεγονός ότι ο έλεγχος απόδοσης μπορεί να επαληθεύσει ότι το σύστημα πληροί τις προδιαγραφές που αξιώνονται από τον κατασκευαστή ή τον προμηθευτή του, μπορεί επίσης να συγκρίνει δύο ή περισσότερες συσκευές ή προγράμματα όσον αφορά παραμέτρους όπως η ταχύτητα, η ταχύτητα μεταφοράς δεδομένων, το εύρος ζώνης, η διακίνηση των δεδομένων, η απόδοση ή αξιοπιστία τους.

Έτσι, κρίνεται ζωτικής σημασίας ο λεπτομερής προσδιορισμός των προδιαγραφών απόδοσης (απαιτήσεις) και η τεκμηρίωση τους σε οποιοδήποτε σχέδιο δοκιμών απόδοσης. Ιδανικά, αυτό επιτυγχάνεται κατά τη διάρκεια της φάσης δημιουργίας των απαιτήσεων της ανάπτυξης του συστήματος του κάθε έργου, πριν από οποιαδήποτε άλλη προσπάθεια σχεδιασμού.

Ωστόσο, ο έλεγχος της απόδοσης συχνά δεν εκτελείται σύμφωνα με μία μόνο προδιαγραφή: π.χ. κανείς δεν μπορεί να εκφράσει ποιος πρέπει να είναι ο μέγιστος αποδεκτός χρόνος απόκρισης για ένα συγκεκριμένο πληθυσμό χρηστών. Ο έλεγχος της απόδοσης χρησιμοποιείται συχνά ως μέρος της διαδικασίας της ρύθμισης του προφίλ απόδοσης και η κύρια ιδέα είναι να προσδιοριστεί ο «πιο αδύναμος κρίκος». Υπάρχει αναπόφευκτα ένα μέρος του συστήματος, το οποίο αν ανταποκριθεί πιο γρήγορα, θα έχει ως αποτέλεσμα το συνολικό σύστημα να εκτελείται πιο γρήγορα. Μερικές φορές είναι αρκετά δύσκολο να προσδιοριστεί ποιο ακριβώς μέρος του συστήματος αντιπροσωπεύει αυτό το κρίσιμο μονοπάτι. Μερικά εργαλεία δοκιμών περιλαμβάνουν ή μπορεί να διαθέτουν πρόσθετα στοιχεία (add-ons) που παρέχουν πράκτορες (agents) οι οποίοι τρέχουν στον server και αναφέρουν στοιχεία όπως τους χρόνους συναλλαγής, τους χρόνους πρόσβασης σε βάσεις δεδομένων και άλλες οθόνες διακομιστή (server

monitors), τα οποία μπορούν να αναλυθούν από κοινού με τα στατιστικά στοιχεία των ακατέργαστων αποδόσεων.

Ο πρωταρχικός στόχος των δοκιμών απόδοσης είναι να κατανοήσουμε τον τρόπο με τον οποίο συμπεριφέρεται ένα σύστημα κατά τη φάση των δοκίμων και πώς αυτή η συμπεριφορά διαφέρει από το ιστορικό της συμπεριφοράς του (δηλαδή, η συμπεριφορά του συστήματος κατά τη διάρκεια μιας προηγούμενης δοκιμής απόδοσης ή σ' έναν συγκεκριμένο τομέα). Η κατανόηση της συμπεριφοράς ενός συστήματος, διαφέρει από το ιστορικό του και παρέχει στους αναλυτές απόδοσης χρήσιμες πληροφορίες για το αν υπάρχουν παλινδρομήσεις απόδοσης (δηλαδή, αν η απόδοση του συστήματος έχει υποβαθμιστεί μετά από μια αλλαγή στον πηγαίο κώδικα ή διαμόρφωση) και για το αν η δοκιμή απόδοσης είναι αντιπροσωπευτική του τομέα στον οποίο εφαρμόζεται (δηλαδή, αν η συμπεριφορά κατά τη διάρκεια της δοκιμής απόδοσης είναι παρόμοια με τη συμπεριφορά του συστήματος στο συγκεκριμένο τομέα-πεδίο).

Η ικανοποίηση αυτού του στόχου παρέχει στους αναλυτές απόδοσης την δυνατότητα να σχεδιάζουν δοκιμές φόρτου εργασίας (test workloads) για να εξασκούν το σύστημα και στη συνέχεια να παρακολουθούν τη συμπεριφορά του συστήματος τους κάτω από αυτόν το φόρτο εργασίας. Οι φόρτοι εργασίας συνήθως προέρχονται από ένα σημείο αναφοράς (benchmark) ή δημιουργούνται εκ νέου, χαρακτηρίζοντας ένα προηγούμενο φόρτο εργασίας (δηλαδή, μια έκδοση alpha ή beta testing).

Ένας φόρτος εργασίας [12] μπορεί να καθοριστεί ως benchmark κατά την αξιολόγηση ενός συστήματος ηλεκτρονικού υπολογιστή από την άποψη της απόδοσης (πόσο εύκολα ο υπολογιστής χειρίζεται το φόρτο εργασίας). Ο φόρτος εργασίας με τη σειρά του χωρίζεται σε χρόνο απόκρισης (ο χρόνος μεταξύ ενός αιτήματος χρήστη και της αντίστοιχης απάντησης στο αίτημα του από το σύστημα) και απόδοσης (πόση δουλειά επιτυγχάνεται κατά τη διάρκεια μιας συγκεκριμένης χρονικής περιόδου).

Ο χαρακτηρισμός του φόρτου εργασίας ενός συστήματος αποτελεί ζωτικής σημασίας στοιχείο και προσφέρει μία καλύτερη κατανόηση του τρόπου με τον οποίο οι χρήστες ενός συστήματος αλληλεπιδρούν με το σύστημα και πώς το ίδιο το σύστημα συμπεριφέρεται σαν αποτέλεσμα. (π.χ. κατανοώντας περιπτώσεις χρήσης και τις προτιμήσεις των χρηστών). Όπως αναφέρθηκε και προηγουμένως, η πιο κοινή εφαρμογή του χαρακτηρισμού ενός φόρτου εργασίας είναι η δημιουργία ενός benchmark workload για τις δοκιμές απόδοσης (π.χ. χαρακτηρίζοντας το field workload για να δημιουργήσουμε ένα performance test workload το οποίο θα είναι αντιπροσωπευτικό του field).

Ωστόσο, η συμπεριφορά του συστήματος βασίζεται στην συμπεριφορά των χιλιάδων ή εκατομμύρια χρηστών οι οποίοι αλληλεπιδρούν με το σύστημα. Μ΄ αυτόν τον τρόπο, το σύστημα συνεχώς εξελίσσεται εκμεταλλευόμενο τις μεταβολές που πραγματοποιούν οι χρηστές, τα χαρακτηριστικά τα οποία προστίθενται ή τροποποιούνται καθώς επίσης και την αλλαγή των προτιμήσεων των χρηστών. Ως εκ τούτου, οι αναλυτές χρειάζονται όλο και μεγαλύτερη υποστήριξη για να παρέχουν τον έλεγχο της απόδοσης εντός του κόστους και των χρονικών περιορισμών που εμφανίζονται.

Ο έλεγχος της απόδοσης μπορεί επίσης να χρησιμοποιηθεί ως διαγνωστικό βοήθημα στον εντοπισμό των σημείων συμφόρησης των επικοινωνιών (communications bottlenecks). Συχνά ένα σύστημα θα λειτουργήσει πολύ καλύτερα εάν ένα πρόβλημα έχει επιλυθεί σε ένα μόνο σημείο ή σε ένα μόνο συστατικό. Επιπλέον, το σύστημα έχει την δυνατότητα να υπολογίσει ποιο μέρος του συστήματος ή του φόρτου εργασίας το αναγκάζει να μην εκτελείται σωστά.

Νέες συστηματικές προσεγγίσεις απαιτούνται, οι οποίες θα περιέχουν συνολικά, χαρακτηριστικά της συμπεριφοράς ενός συστήματος και θα συσχετίζουν αυτόν τον πλουσιότερο χαρακτηρισμό της τρέχουσας συμπεριφοράς ενός συστήματος με το ιστορικό συμπεριφοράς. Μ' αυτόν τον τρόπο, θα παρέχεται στους αναλυτές απόδοσης μια βαθύτερη κατανόηση της συμπεριφοράς του συστήματος που διαχειρίζονται.

Μέσα από σχετικές μελέτες συστημάτων λογισμικού μεγάλης κλίμακας, συμπεριλαμβανομένων συστημάτων ανοιχτού κώδικα (open source) και επιχειρήσεων, οι προσεγγίσεις αυτές μπορούν να βοηθήσουν τους αναλυτές απόδοσης να κατανοήσουν καλύτερα τις διαφορές μεταξύ της τρέχουσας συμπεριφοράς του συστήματος σε σχέση με το ιστορικό του. Πιο συγκεκριμένα, άριστη η γνώση των διαφορών αυτών, θα βοηθήσει τους αναλυτές να εντοπίσουν και να διαγνώσουν τις επιδόσεις παλινδρόμησης (regression performance) καθώς και να βελτιώσουν τις επιδόσεις των ελέγχων που εκτελούν.

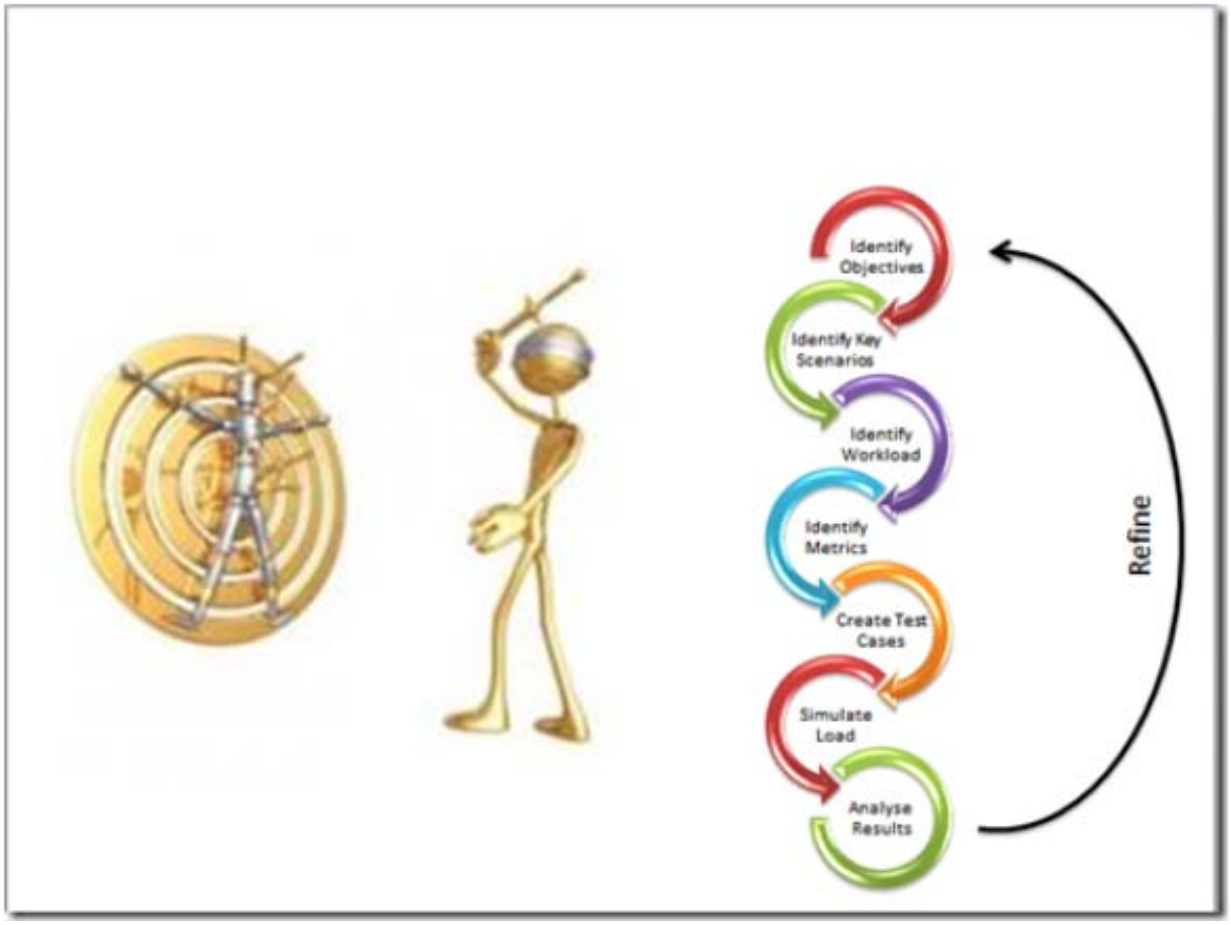
2.2.1 Load Testing

Ο έλεγχος φορτίου [10] ονομάζεται η διαδικασία της εφαρμογής εικονικών δεδομένων και αιτήσεων σε μια μορφή λογισμικού, μια εφαρμογή ή έναν διαδικτυακό τόπο και ελέγχει την απόδοση και συμπεριφορά του κάτω από διάφορες συνθήκες. Μέχρι την στιγμή που κάποιο εργασία (project) λογισμικού πλησιάζει την ολοκλήρωση του, κατά πάσα πιθανότητα θα έχει περάσει από πολλές δοκιμές, ιδιαίτερα σ' ένα ευέλικτο (agile) περιβάλλον δοκίμων όπου οι δοκιμές και η ανάπτυξη (development) πραγματοποιούνται ταυτόχρονα.

Παρ' όλες τις δοκιμές τις οποίες το νέο λογισμικό μπορεί να έχει υποστεί, υπάρχει πραγματικά μόνο ένας τρόπος για να γνωρίζουμε αν το λογισμικό μπορεί να διαχειριστεί έναν μεγάλο φόρτο αιτήσεων από τους χρηστές και αυτός αποτελεί ο έλεγχος φορτίου. Οι διαχειριστές του νέου λογισμικού μπορούν να χρησιμοποιήσουν ειδικά εργαλεία έλεγχου φορτίου τα οποία τους επιτρέπουν να εξετάσουν τη χρήση του υπό συγκεκριμένες συνθήκες (π.χ. ταυτόχρονα μεγάλος αριθμός αιτήσεων, μεγάλοι όγκοι δεδομένων κ.α.).

Ο έλεγχος φορτίου αποτελεί ένα κρίσιμο συστατικό όσον αφορά την αποφυγή αποτυχιών των ULS συστημάτων λογισμικού. Τα ULS συστήματα αποτελούν εξαιρετικά μεγάλης κλίμακας εντατικά συστήματα λογισμικού με πάρα πολλές γραμμές κώδικα, μεγάλο αριθμό χρηστών, και τεράστιο όγκο δεδομένων. Η κλίμακα αυτών των συστημάτων απαιτεί σχεδόν τέλειο συγχρονισμό (near-perfect up-time) χιλιάδων ταυτόχρονων συνδέσεων και λειτουργιών. Η αδυναμία των συστημάτων αυτών να ανταποκριθούν στις απαιτήσεις της απόδοσης, έχει οδηγήσει σε αρκετές αποτυχίες υψηλού προφίλ εταιρειών, συμπεριλαμβανομένης της έναρξης του MobileMe της Apple και η κυκλοφορία του Firefox 3.0 με σημαντικές οικονομικές επιπτώσεις.

Οι αναλυτές απόδοσης είναι υπεύθυνοι για την εκτέλεση δοκιμών φορτίου που παρακολουθούν τον τρόπο με τον οποίο το σύστημα συμπεριφέρεται δεχόμενο ρεαλιστικά φορτία δεδομένων. Τέτοιες δοκιμές φορτίου επιτρέπουν στους αναλυτές να προσδιορίζουν την μέγιστη λειτουργική ικανότητα ενός συστήματος, τις λειτουργικές απαιτήσεις της απόδοσης του αλλά και τα σημεία συμφόρησης που εντοπίζονται.



Εικόνα 2: Βήματα των δοκιμών φορτίου [13]

Παρακάτω παρατίθεται ένα απλό παράδειγμα χρήσης των δοκιμών φορτίου. Ας υποθέσουμε ότι είμαστε στη φάση ανάπτυξης μιας νέας διαδικτυακής πλατφόρμας ψηφοφοριών, και θα θέλαμε η εφαρμογή να είναι σε θέση να διαχειρίζεται ενδεχομένως έως και 10.000 υποβολές χρηστών ανά λεπτό στην ώρα αιχμής. Η ανάπτυξη του λογισμικού χαρακτηρίζεται από την εκτέλεση δοκιμών μονάδας (unit testing), καθώς οι προγραμματιστές γράφουν τον κώδικα και εκτελούν δοκιμές παλινδρόμησης για να βεβαιωθούν ότι δεν έχουν πειραχθεί οι υφιστάμενες λειτουργίες κατά την διάρκεια της κάθε νέας τροποποίησης. Παρόλα αυτά όμως, καθώς η ανάπτυξη λογισμικού βρισκόταν σε εξέλιξη, οι developers δεν εξέτασαν αν τελικά η εφαρμογή μπορεί να δεχτεί εκατοντάδες ή ακόμα και χιλιάδες χρήστες και μαζί τις πολλαπλές αιτήσεις τους. Σ' αυτό το σημείο, οι προγραμματιστές οφείλουν να πραγματοποιήσουν load testing και να ελέγξουν αν τελικά η εφαρμογή είναι σε θέση να εξυπηρετήσει τον μεγάλο αυτόν αριθμό των αιτήσεων.

Από τεχνικής απόψεως, οι δοκιμές φορτίου μίας αίτησης δεν μπορούν να εκτελεστούν έως ότου ένα έργο είναι σχεδόν στο τέλος του κύκλου παραγωγής του, στο οποίο η πραγματική εμπλοκή των χρηστών και η απόδοση του συστήματος μπορούν να προσομοιωθούν με ακρίβεια και να τεθούν σε δοκιμασία [14]. Είναι ανάλογο με ένα αυτοκίνητο: μπορούμε να επιδιορθώσουμε και να δοκιμάσουμε τον κινητήρα, αλλά αν ο κινητήρας δεν έχει ακόμη εγκατασταθεί, δεν μπορούμε να ελέγξουμε την απόδοση του αυτοκινήτου στο δρόμο. Σε ένα έργο ανάπτυξης λογισμικού, αν σε οποιαδήποτε σημείο

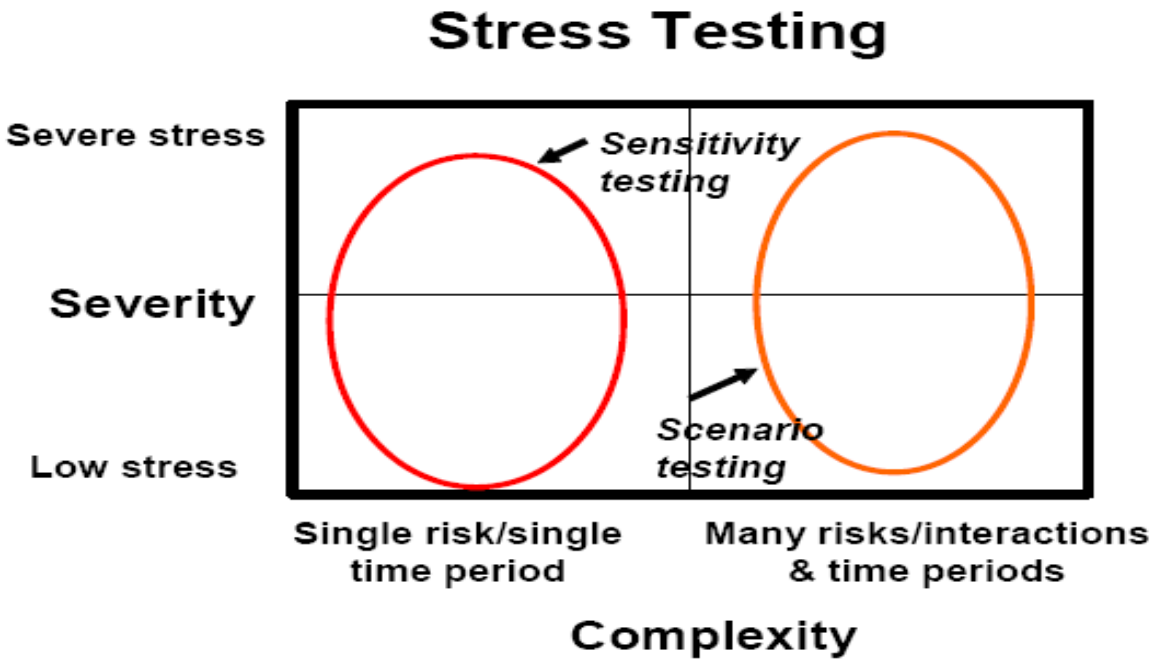
της εφαρμογής εμφανίζονταν προβλήματα όπως ζητήματα υποστήριξης (backend) επιδόσεων, ταυτόχρονης εισόδου χρηστών ή οτιδήποτε άλλο που θα μπορούσε να προσθέσει πίεση (stress) στο σύστημά όπως διαρροές μνήμης (memory leaks), ή κάποια μειωμένη λειτουργικότητα, τότε θα ήταν βέβαιο, ότι ακόμα και μία πρόωρη περιορισμένη μορφή load testing θα προετοίμαζε ήδη τους προγραμματιστές για τις επιπτώσεις της εμπλοκής πολλών χρηστών στο σύστημά.

Οι δοκιμές φορτίου αποτελούν τον πιο γνωστό και πιο συχνά χρησιμοποιημένο τύπο του των δοκιμών απόδοσης (performance testing) και συνίστανται στην εφαρμογή συνηθισμένης πίεσης (ρουτίνας) σε μια εφαρμογή ή IT συστήματος για να εξεταστεί αν μπορεί να αποδώσει υπό κατάλληλες συνθήκες. Σχετίζεται με έναν παρόμοιο τρόπο διαχείρισης δοκίμων που ονομάζονται δοκιμές πίεσης ή δοκιμές άγχους (stress testing).

2.2.2 Stress Testing

Σε αντίθεση με το load testing το οποίο εξασφαλίζει ότι μια συγκεκριμένη λειτουργία μπορεί να εκτελεστεί απλά, έτσι όπως έχει αρχικά σχεδιαστεί να εκτελείται, οι δοκιμές πίεσης [10] αποτελούν έναν πιο «βάναυσο» τρόπο δοκίμων και αφορούν την εφαρμογή μη ρεαλιστικών και ακραίων καταστάσεων όπως για παράδειγμα την υπερφόρτωση δεδομένων, μέχρι η συγκεκριμένη εφαρμογή ή σύστημα να σπάσει. Οι δοκιμές αυτές προσπαθούν να σπάσουν το σύστημα με την χρήση συντριπτικών πόρων ή με την απομάκρυνση πόρων από αυτό (στην περίπτωση αυτή ονομάζονται μερικές φορές αρνητικές δοκιμές). Ο κύριος σκοπός αυτής της διαδικασίας είναι να βεβαιωθούμε ότι σε περίπτωση που το σύστημα αποτυγχάνει, η λειτουργία του ανακτάται ξανά ομαλά. Η κατάσταση αυτή, είναι γνωστή ως ποιότητα ανάκτησης.

Και τα δυο συστήματα μπορούν να παίξουν σημαντικό ρόλο στον ακριβή καθορισμό της απόδοσης ενός συγκεκριμένου κομματιού του λογισμικού διεπαφής, όπως μια ιστοσελίδα η ενός backend συστήματος, δηλαδή με το πόσο καλά μπορεί το λογισμικό να ασχοληθεί με πραγματικά φορτία και τακτική χρήση.



Εικόνα 3: Περιγραφή Stress Testing [15]

Οι δοκιμές πίεσης προκαλούν σκόπιμα βλάβες, έτσι ώστε ο διαχειριστής του λογισμικού να μπορεί να αναλύσει τους κινδύνους που εμφανίζονται και εμπλέκονται στα σημεία θραύσης και στην συνέχεια να βρει τους κατάλληλους τρόπους και προγράμματα ώστε αυτή η θραύση να γίνει όσο το δυνατόν πιο ομαλή. Οι προσομοιώσεις αυτές είναι χρήσιμες για την προετοιμασία και αντιμετώπιση απροσδόκητων καταστάσεων, δηλαδή για να καθοριστεί ακριβώς το πόσο ένα συγκεκριμένο σύστημα μπορεί να πιεστεί και μ' αυτόν τον τρόπο να εξερευνηθούν τα όρια απόδοσης του.

Οι προσομοιώσεις ακραίων καταστάσεων μπορούν να επιτύχουν υψηλότερη κάλυψη με την παραγωγή σφαλμάτων. Η κάλυψη αυτή μπορεί να βελτιωθεί περαιτέρω με τη χρήση ένεσης σφαλμάτων (fault injection). Κατά τη δοκιμή του λογισμικού, η fault injection είναι μια τεχνική η οποία βελτιώνει την κάλυψη μιας δοκιμής με την εισαγωγή σφαλμάτων για να ελέγξουμε μονοπάτια κώδικα και να αντιμετωπίσουμε συγκεκριμένα λάθη, τα οποία σπάνια θα εμφανίζονταν. Συχνά χρησιμοποιείται σε συνδυασμό με τη προσομοίωση ακραίων καταστάσεων και θεωρείται ευρέως ένα σημαντικό μέρος της ανάπτυξης ισχυρού λογισμικού.

Είναι σημαντικό επίσης το γεγονός ότι η εφαρμογή ενός απλού load test σε ένα λογισμικό ή σύστημα το οποίο δεν είναι πραγματικά έτοιμο για τις αναμενόμενες απαιτήσεις, μπορεί ξαφνικά να μετατραπεί σε stress test ενώ είναι σε λειτουργία. Μόλις το φορτίο αρχίσει να προκαλεί τα πράγματα να σπάσουν, από αυτήν την στιγμή, εξορισμού, το φορτίο θα στρεσάρει το σύστημα. Αυτός είναι ο κύριος λόγος που μερικές φορές αυτοί οι δυο τύποι διαχείρισης απόδοσης συγχέονται, επειδή ακριβώς ένα απλό φορτίο μπορεί να αποδεδειχθεί ότι αποτελεί ένα load test κάτω από ορισμένες συνθήκες και stress test κάτω από κάποιες άλλες.

Παρόλα αυτά, αν μας ενδιαφέρει μόνο αν μια εφαρμογή λογισμικού μπορεί να αντέξει τα αιτήματα των χρηστών και των δράσεων που είναι πιθανό να συναντήσει σε κανονικές

συνθήκες, τότε το load testing είναι η σωστή μέθοδος δοκιμών που πρέπει να εφαρμοστεί.

2.3 Memory Related Issues

Τα προβλήματα που αφορούν τη μνήμη, μπορούν να υποβαθμίσουν την απόδοση (με την αύξηση του overhead της μνήμης και εξαντλώντας τη διαθέσιμη μνήμη) και να προκαλέσουν συντριβές (crushes) με την πλήρη εξάντληση της διαθέσιμης μνήμης. Τέτοιου είδους θέματα είναι δύσκολο να εντοπιστούν ή να διαγνωστούν και πολλές φορές μπορούν να οφείλονται στην γήρανση του λογισμικού (software aging) [16], [17]. Σε γενικές γραμμές, τα θέματα που σχετίζονται με την κατανάλωση της μνήμης κατατάσσονται ως παροδικά (transient) ή επίμονα (persistent).

Τα παροδικά προβλήματα διευθέτησης της μνήμης (memory spikes) αποτελούν μεγάλες απότομες αυξήσεις στην χρήση μνήμης σε σχετικά σύντομο χρονικό διάστημα ενώ τα επίμονα προβλήματα είναι σταθερές αυξήσεις της χρήσης της μνήμης στην πάροδο του χρόνου. Τα persistent προβλήματα που αφορούν την μνήμη, μπορούν να διαιρεθούν περαιτέρω σε πρήξιμο της μνήμης (memory bloat) το οποίο προκαλείται από αναποτελεσματικές εφαρμογές και σε διαρροές της μνήμης (memory leaks) που προκαλούνται λόγω αποτυχίας απελευθέρωσης της αχρείαστης μνήμης.

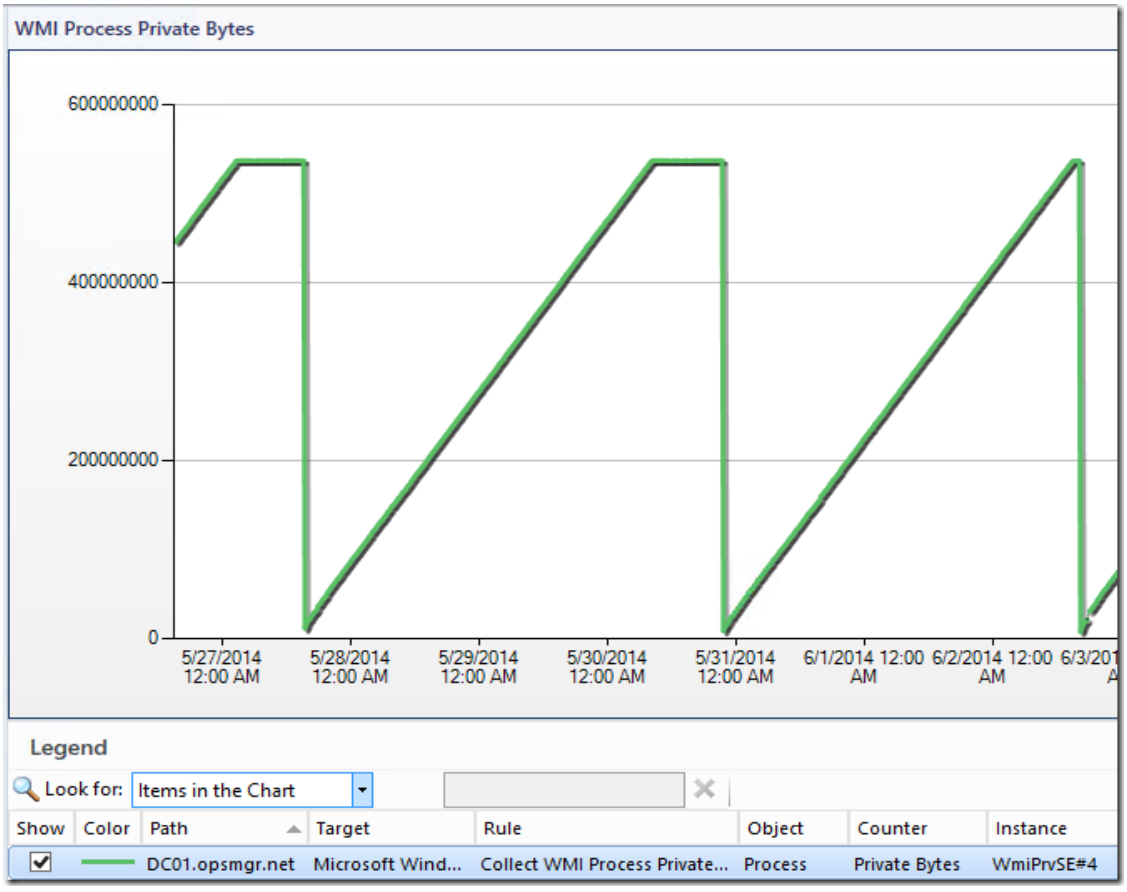
Οι εφαρμογές με αυτά τα προβλήματα, όσο το σύνολο των διεργασιών τους αυξάνεται, γίνονται πιο αργές στην πάροδο του χρόνου και μπορούν να σταματήσουν να αποκρίνονται. Ταυτόχρονα, τα memory leaks και οι memory bloat περιπτώσεις παραμένουν εμφανώς δύσκολες να εντοπιστούν και αποτελούν ένα μεγάλο αριθμό των αναφερθέντων σφαλμάτων που εμφανίζονται στις εφαρμογές.

Η προσέγγισή που θέτουν οι Syer et al. εστιάζει στην διάγνωση (diagnosis), σε αντίθεση με την ανίχνευση (detection) των ζητημάτων που σχετίζονται με τη μνήμη. Οι αναλυτές απόδοσης μπορούν να χρησιμοποιήσουν μια ποικιλία τεχνικών για τον εντοπισμό θεμάτων που σχετίζονται με προβλήματα διευθέτησης της μνήμης πριν από τη εφαρμογή της προσέγγισης αυτής για τη διάγνωση τους. Για παράδειγμα, αναλυτές απόδοσης μπορούν να σχεδιάσουν τη χρήση της μνήμης σε συνάρτηση με τον χρόνο για να ελέγξουν αν εμφανίζονται επίμονα προβλήματα μνήμης (όπου η χρήση της μνήμης αυξάνεται συνεχώς) ή να συγκρίνουν την ελάχιστη, μέση και μέγιστη χρήση της μνήμης για να προσδιορίσουν αν υπάρχουν παροδικά (transient) προβλήματα μνήμης.

2.3.1 Persistent Memory Issues

Memory Leak

Όταν ένα πρόγραμμα χρειάζεται να αποθηκεύσει προσωρινά κάποιες πληροφορίες κατά τη διάρκεια της εκτέλεσης του, μπορεί να ζητήσει δυναμικά ένα μεγάλο κομμάτι της μνήμης από το σύστημα. Ωστόσο, το σύστημα έχει ένα σταθερό ποσό της συνολικής διαθέσιμης μνήμης. Αν μία εφαρμογή χρησιμοποιήσει όλο το σύνολο της ελεύθερης μνήμης του συστήματος, άλλες εφαρμογές δεν θα είναι σε θέση να αποκτήσουν τη μνήμη που απαιτούν και να εκτελέσουν τις δικές τους λειτουργίες. Οι συνέπειες της απώλειας αυτής, είναι μία εφαρμογή η οποία πλέον δε μπορεί να δεσμεύσει κάποιο ποσό της μνήμης, να τερματίζεται ομαλά ή να κλείνει απρόσμενα.



Εικόνα 4: Σχεδιάγραμμα μιας διαρροής μνήμης [18]

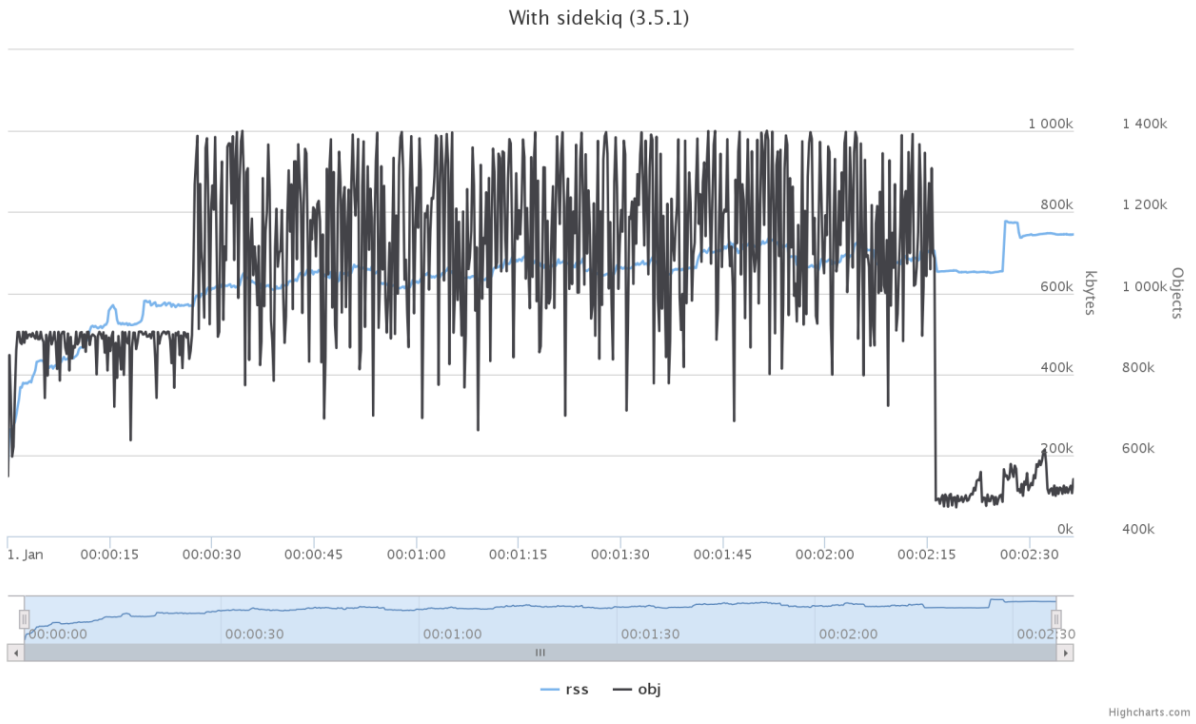
Οι περισσότερες εφαρμογές μεγάλης κλίμακας αιτούνται τακτικά τη μνήμη με αποτέλεσμα να εξαντλείται η μνήμη του συστήματος και να δημιουργείται μια ντόμινο επίδραση (domino effect). Σε συνθήκες ύπαρξης χαμηλής μνήμης, ακόμα και αν η εφαρμογή δε τερματίσει, το σύστημα είτε θα επιβραδύνει είτε πλέον δε θα αποκρίνεται. Όταν μία εφαρμογή κατανέμει δυναμικά μνήμη και δεν την απαλλάσσει όταν τελειώσει την εκτέλεση της, τότε δημιουργείται μια διαρροή μνήμης (memory leak). Τα αποτελέσματα αυτά δεν είναι επιθυμητά για την ομαλή λειτουργία μίας εφαρμογής ή ενός λειτουργικού συστήματος.

Είναι ευθύνη της κάθε εφαρμογής να απελευθερώσει την δυναμική μνήμη που έχει δεσμευτεί από τα διάφορα συστατικά της. Με αυτόν τον τρόπο, η μνήμη απελευθερώνεται και επιστρέφει στο σύστημα, όπου μπορεί να ανακατανεμηθεί και να χρησιμοποιηθεί από άλλες εφαρμογές, όπου απαιτείται. Η μνήμη δεν χρησιμοποιείται από την εφαρμογή πια, όπως επίσης, δεν μπορεί να χρησιμοποιηθεί από το σύστημα ή οποιοδήποτε άλλο πρόγραμμα.

Οι διαρροές μνήμης προσθέτονται στην πάροδο του χρόνου και σε περίπτωση που δεν καθαριστούν έγκαιρα, το σύστημα μένει εκτός διαθέσιμης μνήμης [18]. Συνήθως η εφαρμογή συνοδεύεται από φρικτά αργό χρόνο απόκρισης και συχνά ο χρήστης δεν μπορεί καν να την τερματίσει εξαιτίας αυτής της ατονίας.

Memory Bloat

Αποτελεί μία μορφή επίμονου προβλήματος διαχείρισης της μνήμης. Αφορά την υπερβολική κατανάλωση της μνήμης και προκαλείται από ανεπαρκείς εφαρμογές. Είναι ένα πιο χρονικά ευαίσθητο πρόβλημα σε σχέση με τις διαρροές μνήμης (memory leak). Οι εφαρμογές οι οποίες χαρακτηρίζονται από bloat προβλήματα, μπορούν να περιορίσουν τον αριθμό των ενεργειών (αιτήσεων) ενός χρήστη τις οποίες μπορεί να τρέξει καθώς επίσης και να υποβαθμίσουν την ανταπόκριση του συστήματος ,αναγκάζοντας άλλες εφαρμογές να τερματιστούν (page out) πρόωρα.



Εικόνα 5 :Σχέδιο ενός memory bloat [19]

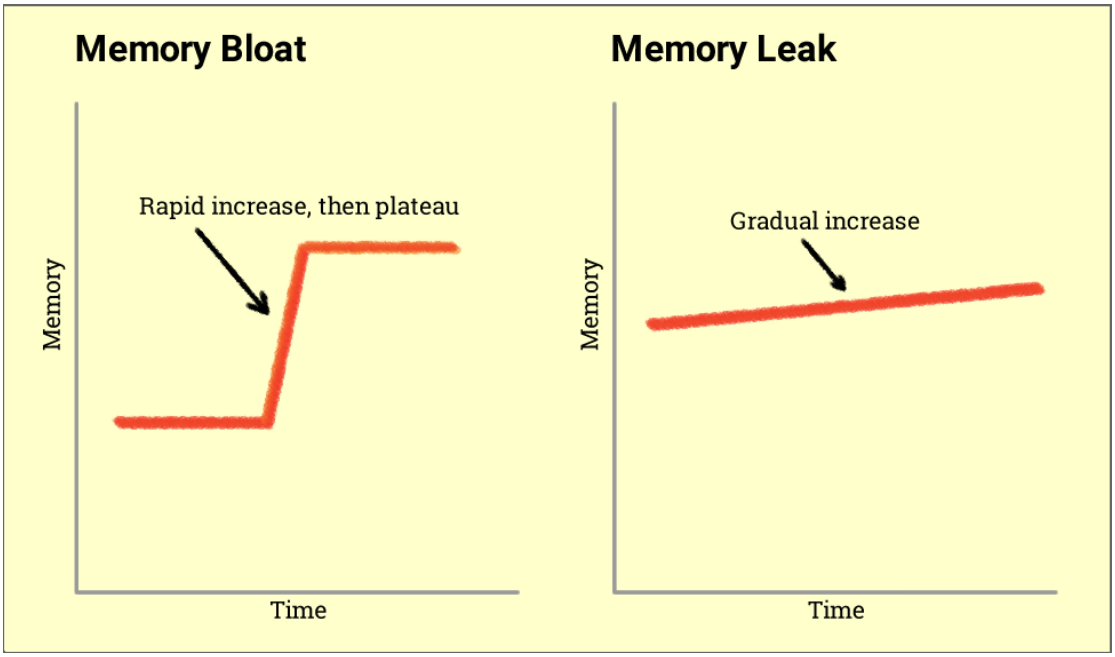
Τα bloat θέματα μνήμης είναι αρκετά περίπλοκα, και πρέπει να εξεταστούν από διάφορες οπτικές γωνίες: τη δοκιμή (testing), την αναπαραγωγή (reproducing), διαρροή (leaking) κ.α. Ο ευκολότερος τρόπος για έναν αναλυτή απόδοσης να ανιχνεύσει bloat καταστάσεις είναι να παρακολουθήσει την αύξηση του μεγέθους της μνήμης κατά την διαδικασία εκτέλεσης της εφαρμογής ή του συστήματος.

Η συμφόρηση η οποία προκαλείται από την μεγάλη συσσώρευση δεδομένων και η αύξηση της μνήμης κάνει την εφαρμογή να αποκρίνεται αργά. Παρόλα αυτά όμως, υπάρχει πιθανότητα η εφαρμογή να ανακάμψει γρήγορα σε περίπτωση που εξυπηρετήσει κάποιο αργό αίτημα, δηλαδή ένα αίτημα το οποίο δεν έχει μακροχρόνιες απαιτήσεις.

Επιπλέον, τα περισσότερα bloat προβλήματα προκαλούνται από τους χρήστες δύναμης (power users): δηλαδή ενέργειες που δουλεύουν μια χαρά για τους περισσότερους απλούς χρήστες, θα λυγίσουν υπό το βάρος των χρηστών αυτών. Ένας power user ή έμπειρος χρήστης είναι ένας χρήστης υπολογιστή που μπορεί να χρησιμοποιεί προηγμένες δυνατότητες του υλικού (hardware) ενός υπολογιστή, λειτουργικών συστημάτων, εφαρμογών ή ιστοσελίδων που δεν χρησιμοποιούνται από τον μέσο

χρήστη. Ένας power user μπορεί να μην έχει εκτεταμένες τεχνικές γνώσεις των συστημάτων που χρησιμοποιεί και δεν είναι κατ 'ανάγκην σε θέση να γνωρίζει τα κύρια συστατικά του προγραμματισμού και διαχείρισης του συστήματος. Εντούτοις, χαρακτηρίζεται από την ικανότητα να πραγματοποιεί μια ευρύτερη ή πιο γενική χρήση του συστήματος ή εφαρμογής.

Τέλος, ενώ τα bloat προβλήματα μπορούν να «ακρωτηριάσουν» γρήγορα μια ιστοσελίδα, στην πραγματικότητα είναι πιο εύκολο να εντοπιστεί η αιτία τους σε σχέση με τις διαρροές μνήμης. Εάν η εφαρμογή πάσχει από υψηλή χρήση της μνήμης, είναι προτιμότερο να εξεταστούν τα δεδομένα της για την ύπαρξη memory bloat και έπειτα για την ύπαρξη memory leaks.



Εικόνα 6: Memory bloat vs memory leak [20]

2.3.2 Transient Memory Issues

Memory Spike

Αποτελεί ένα παροδικό πρόβλημα διευθέτησης της μνήμης όπου εντοπίζεται μία σημαντική, ταχεία και παρατεταμένη αύξηση της μνήμης. Το αποτέλεσμα της αύξησης αυτής είναι μια μικρή μείωση της απόκρισης της εφαρμογής ή του συστήματος. Ένα memory spike παρουσιάζει μια μορφή σαν ένα αγκάθι, δηλαδή περιλαμβάνει μια υπερβολική τιμή της μνήμης η οποία «ξεφεύγει» από την ομαλή λειτουργία της. Ένα παράδειγμα εμφάνισης ενός spike προβλήματος είναι το εξής: Ο υπολογιστής χρησιμοποιεί το 15-20% της κύριας μνήμης , αλλά κάθε λεπτό ή κάποια άλλη στιγμή, ξαφνικά εντοπίζονται αιχμές γύρω στο 95-100%, ακόμα και με μια απλή πληκτρολόγηση ενός μηνύματος. Προκαλείται μια τεράστια επιβράδυνση που δε κάνει τίποτα, αλλά η

πλοήγηση στο διαδίκτυο ή ακόμα και μια εγγραφή ενός απλού ηλεκτρονικού μηνύματος είναι πρακτικά αδύνατη.

Σε αντίθεση με τα persistent προβλήματα διευθέτησης της μνήμης, το memory spike δεν έχει μεγάλη διάρκεια και δεν έχει τόσο μεγάλη επιρροή στην εκτέλεση. Ωστόσο, σε περίπτωση που εντοπιστεί ένας μεγάλος και συχνός αριθμός memory spikes στο σύστημα, τα αποτελέσματα θα είναι αρνητικά όσον αφορά την ομαλή λειτουργία του.

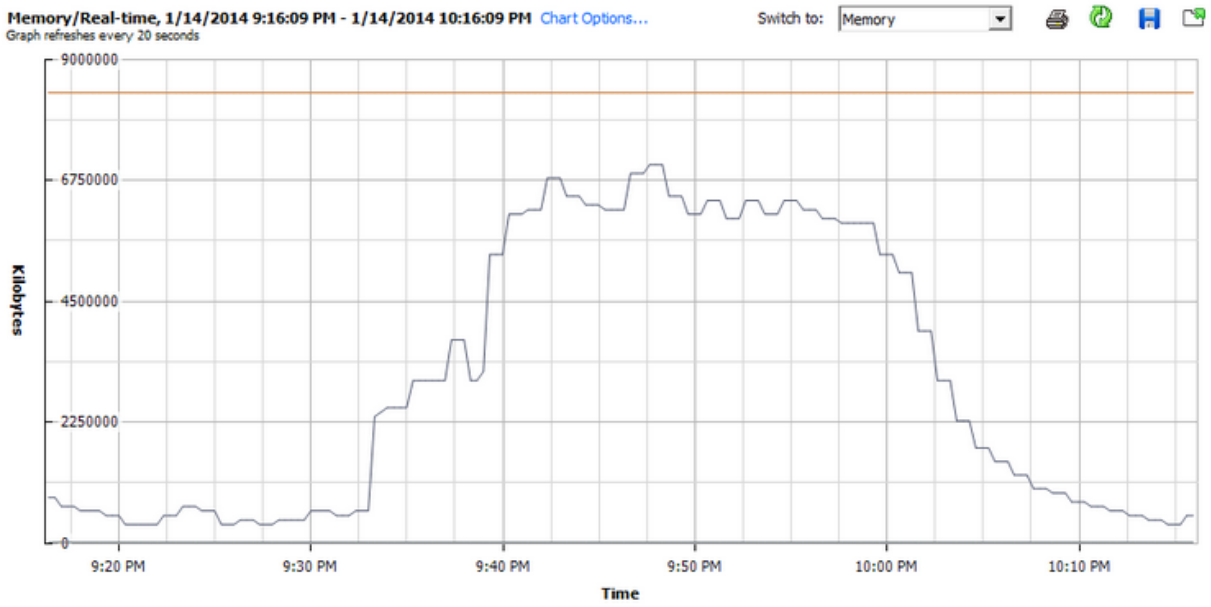
2.4 Performance Counters

Οι μετρητές απόδοσης αποτελούν ευέλικτες μετρήσεις οι οποίες μπορούν να παρακολουθούν τα στοιχεία ενός συστήματος, όπως τους επεξεργαστές (CPU), την μνήμη και το δίκτυο εισόδου εξόδου (I / O). Για παράδειγμα, ένας Web διακομιστής (server) μπορεί να αναφέρει πόσες συνεδρίες διαχειρίζεται ή ένας DB διακομιστής (database server) μπορεί να αναφέρει τον αριθμό των συνδέσεων και των ερωτημάτων (queries) τα οποία πραγματοποιήθηκαν μέσα σε ένα συγκεκριμένο χρονικό διάστημα.

Χάρις τη χρήση των μετρητών απόδοσης, μία εφαρμογή ή ένα λειτουργικό σύστημα, έχει την δυνατότητα να εξάγει χρήσιμα στατιστικά δεδομένα τα οποία είναι απαραίτητα για την σύγκριση και ανάλυση των επιδόσεων του. Επιπλέον, παρέχονται στον διαχειριστή του συστήματος σημαντικές πληροφορίες σχετικά με τη διαχείριση της μνήμης (δέσμευση - αποδέσμευση) που πραγματοποιείται κατά τη διάρκεια της εκτέλεσης μιας εφαρμογής ή κατά τις διάφορες φάσεις λειτουργίας του λειτουργικού συστήματος.

Η Ανάλυση των μετρητών απόδοσης έχει δύο σημαντικούς περιορισμούς. Κατ 'αρχάς, οι μετρητές απόδοσης ενδιαφέρονται μόνο για την προοπτική του συστήματος και όχι για την πλευρά του χρήστη. Ωστόσο, η οπτική του χρήστη των επιδόσεων μπορεί να διαφέρει σημαντικά από την συνολική απόδοση. Για παράδειγμα, ο μέσος χρόνος απόκρισης για 1.000 χρήστες μπορεί να είναι 100 χιλιοστά του δευτερολέπτου, αλλά η απόκριση για ορισμένους χρήστες μπορεί να είναι σημαντικά υψηλότερη σε σχέση με άλλους. Δεύτερον, η ανάλυση των μετρητών απόδοσης δεν λαμβάνει υπόψη τα αρχεία εκτέλεσης. Έτσι, αυτές οι προσεγγίσεις μπορούν να προσδιορίσουν αν η συμπεριφορά του συστήματος διαφέρει από το ιστορικό της συμπεριφοράς του, αλλά δεν μπορούν να βρουν τον ακριβή λόγο που συμβαίνει κάτι τέτοιο.

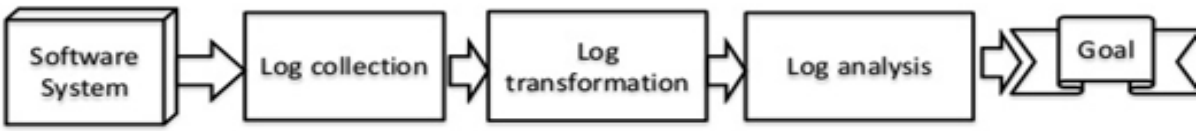
Οι τωρινές προσεγγίσεις οι οποίες συγκρίνουν την τρέχουσα συμπεριφορά ενός συστήματος με το ιστορικό της συμπεριφοράς του, δεν εκμεταλλεύονται πλήρως τα αρχεία καταγραφής εκτέλεσης επειδή ο δυναμικός χαρακτήρας των πληροφοριών που περιέχονται στα αρχεία καταγραφής απορρίπτεται όταν τα logs (κούτσουρα) μετατρέπονται σε γεγονότα καταγραφής. Επιπλέον, αυτές οι προσεγγίσεις δε λαμβάνουν υπόψη ταυτόχρονες πολλαπλές πηγές πληροφοριών (δηλαδή, λαμβάνουν υπόψη, είτε τα αρχεία καταγραφής εκτέλεσης, είτε τους μετρητές απόδοσης, αλλά όχι και τα δύο). Ως εκ τούτου, νέες προσεγγίσεις απαιτούνται για τη σύγκριση της τρέχουσας συμπεριφοράς ενός συστήματος με το ιστορικό της συμπεριφοράς του οι οποίες θα εμπλουτίζουν τα γεγονότα εκτέλεσης με πρόσθετες πηγές πολύτιμων πληροφοριών (δηλαδή, τους μετρητές απόδοσης) και θα αξιοποιούν πλήρως τις πληροφορίες των αρχείων καταγραφής εκτέλεσης (δηλαδή, τα στατικά και δυναμικά στοιχεία των αρχείων εκτέλεσης).



Εικόνα 7: Γραφική παράσταση μετρητών απόδοσης [21]

2.5 Execution Logs

Τα αρχεία ή κούτσουρα καταγραφής στο πλαίσιο της επιστήμης των υπολογιστών, είναι η αυτόματη παραγωγή, χρονοσήμανση και τεκμηρίωση των γεγονότων που σχετίζονται με ένα συγκεκριμένο σύστημα. Τα αρχεία αυτά μπορούν να περιγράφουν γεγονότα που συμβαίνουν σε ένα λειτουργικό σύστημα, σε μια εφαρμογή ή μηνύματα μεταξύ διαφορετικών χρηστών του λογισμικού επικοινωνίας. Logging ονομάζεται η διαδικασία της διατήρησης ενός αρχείου καταγραφής. Ένα log αρχείο μπορεί να είναι χρήσιμο στην παρακολούθηση της χρήσης ενός υπολογιστή και στην ανάκτηση δεδομένων σε περίπτωση λάθους ή κάποιας ανάγκης.



Εικόνα 8: Βήματα τροποποίησης των αρχείων καταγραφής [22]

Τα αρχεία εκτέλεσης καταγράφονται και αποθηκεύονται σε συγκεκριμένους φακέλους που ονομάζονται φάκελοι καταγραφής (log files) [23]. Οι φάκελοι καταγραφής, ως επί των πλείστων είναι απλά αρχεία κειμένου (.txt) και είναι άχρηστα για το μέσο χρήστη.

Αποτελούν όμως, το καλύτερο βοήθημα για έναν προγραμματιστή ώστε να τον βοηθήσουν να διορθώσει την εφαρμογή όταν αυτή κρίνεται προβληματική. Πέρα από τους προγραμματιστές οι οποίοι χρησιμοποιούν τα log files για την αποσφαλμάτωση του συστήματος (system debugging), χρησιμοποιούνται επίσης και από τους διαχειριστές του συστήματος (για την παρακολούθηση της λειτουργίας του συστήματος). Σχεδόν όλα τα λειτουργικά συστήματα και οι εφαρμογές λογισμικού παράγουν αρχεία καταγραφής.

Ένα αρχείο καταγραφής περιγράφει λεπτομερώς το τι συμβαίνει μέσα και έξω από ένα συγκεκριμένο διακομιστή ή εφαρμογή. Είναι μια έννοια πολύ όμοια με το μαύρο κουτί του αεροπλάνου που καταγράφει όλα όσα συμβαίνουν κατά την διάρκεια μιας πτήσης σε περίπτωση προβλήματος. Οι πληροφορίες συχνά καταγράφονται χρονολογικά και βρίσκονται στον ριζικό κατάλογο, ή περιστασιακά σε ένα δευτερεύοντα φάκελο, ανάλογα με το πώς έχει ρυθμιστεί το σύστημα. Το μόνο πρόσωπο που έχει τακτική πρόσβαση στα αρχεία καταγραφής είναι ο διαχειριστής του συστήματος. Τα αρχεία καταγραφής προστατεύονται γενικά με έναν κωδικό πρόσβασης, έτσι ώστε ο διαχειριστής ή διαχειριστές του να είναι οι μοναδικοί με πλήρη πρόσβαση σ' αυτά.

Τα αρχεία καταγραφής πρόσβασης διατηρούνται σε ένα Web διακομιστή, περιλαμβάνοντας όλα τα μεμονωμένα αρχεία που οι χρήστες έχουν ζητήσει από μια συγκεκριμένη ιστοσελίδα. Το σύνολο αυτό περιλαμβάνει τα HTML (HyperText Markup Language) αρχεία, ενταγμένες γραφικές εικόνες και οποιαδήποτε άλλα σχετικά αρχεία που μεταδίδονται μεταξύ του διακομιστή και του χρήστη. Από τα αρχεία καταγραφής του εξυπηρετητή, ένας διαχειριστής του συστήματος μπορεί να προσδιορίσει τον αριθμό των επισκεπτών, τις πιο δημοφιλείς περιοχές της ιστοσελίδας που επισκέπτονται οι χρήστες, τον αριθμό των αιτήσεων για κάθε σελίδα και άλλες χρήσιμες στατιστικές πληροφορίες όπως τις ώρες της ημέρας, εβδομάδας, μήνα ή έτους που είναι πιο δημοφιλείς. Η χρήση των μπισκότων (cookies) επιτρέπει τους υπεύθυνους διαχειριστές της ιστοσελίδας (webmasters) να αντλήσουν ακόμα πιο λεπτομερείς πληροφορίες σχετικά με την πρόσβαση που έχουν οι μεμονωμένοι χρήστες.

Ο ρόλος ενός αρχείου καταγραφής είναι να παρακολουθεί το τι συμβαίνει με το διακομιστή. Αν κάτι δυσλειτουργεί μέσα σε ένα πολύπλοκο σύστημα, τότε μπορεί να μην υπάρχει άλλος τρόπος εντοπισμού του προβλήματος. Αρχεία καταγραφής χρησιμοποιούνται για την παρακολούθηση πολύπλοκων συστημάτων. Όταν εντοπιστεί ένα πρόβλημα, τότε είναι εύκολο να αντιμετωπιστεί και να διορθωθεί. Τα αρχεία καταγραφής είναι επίσης σημαντικά για τις εφαρμογές που έχουν λίγη έως καμία ανθρώπινη αλληλεπίδραση, όπως τις εφαρμογές του διακομιστή.

Υπάρχουν φορές, που τα αρχεία καταγραφής είναι πάρα πολύ δύσκολο να διαβαστούν ή δε βγάζουν κάποιο νόημα [24]. Τότε, κρίνεται απαραίτητη η ανάλυση του αρχείου καταγραφής η οποία εκτελείται γενικά από κάποιο είδος προγράμματος υπολογιστή που μετατρέπει τις πληροφορίες που περιέχει σε μια πιο συνοπτική και αναγνώσιμη μορφή. Τα αρχεία καταγραφής μπορούν επίσης να χρησιμοποιηθούν για τη συσχέτιση των δεδομένων μεταξύ των διακομιστών, ώστε να βρεθούν τα κοινά προβλήματα μεταξύ των διαφόρων συστημάτων τα οποία απαιτούν μια κοινή λύση για την επισκευή όλων τους.

Ο εξυπηρετητής δεν είναι το μόνο σύστημα που χρησιμοποιούνται τα αρχεία καταγραφής. Διάφορα συστήματα ελέγχου, καθώς και τα λειτουργικά συστήματα υπολογιστών, διαθέτουν υποσυστήματα που έχουν ακριβώς την ίδια λειτουργία όπως ένα αρχείο καταγραφής. Ενώ η λειτουργία τους είναι πιο περίπλοκη από ότι ένα απλό αρχείο καταγραφής, τις περισσότερες φορές πράττουν παρόμοια, καταγράφοντας ένα

log μήνυμα στο αρχείο και αποθηκεύοντας το μέχρι να χρησιμοποιηθεί. Ένα τέτοιο υποσύστημα καταγραφής επιτρέπει σε έναν διαχειριστή να δημιουργήσει, να χρησιμοποιήσει φίλτρα, καθώς επίσης και να αναλύσει τα log μηνύματα εξάγοντας χρήσιμα συμπεράσματα [25]. Άλλες μορφές καταγραφής χρησιμοποιούν πιο εξελιγμένα συστήματα, ορισμένα από τα οποία αναλύουν ακόμη και τα αρχεία καταγραφής, πριν τα χρειαστούν. Τα συστήματα αυτά εξαρτώνται απ' το που ακριβώς έχουν αποθηκευτεί τα αρχεία καταγραφής.

Τέλος, η εκτέλεση των αρχείων καταγραφής αντιμετωπίζεται συνήθως ως μια ακολουθία γεγονότων, χωρίς την πλήρη εκμετάλλευση πρόσθετων πηγών πολύτιμων σε πληροφορίες που σχετίζονται με τέτοια γεγονότα (π.χ. τις δυναμικές πληροφορίες αρχείων εκτέλεσης και οι μετρητές απόδοσης που συλλέγονται κατά την εκτέλεση του συστήματος). Γι' αυτόν τον λόγο κρίνονται απαραίτητες νέες προσεγγίσεις όπως αυτή των Syer et al. που θα προσθέσουν στα γεγονότα εκτέλεσης χρήσιμες πληροφορίες εκμεταλλευόμενες τόσο την στατική όσο και την δυναμική πληροφορία των αρχείων εκτέλεσης.

Event Logs

Τα event logs αποτελούν αρχεία καταγραφής συμβάντων και είναι ένας βασικός πόρος που βοηθά στην παροχή πληροφοριών σχετικά με την κυκλοφορία του δικτύου, τη χρήση του και άλλες συνθήκες. Ένα αρχείο καταγραφής συμβάντων αποθηκεύει τα δεδομένα κατάλληλα ώστε να χρησιμοποιηθούν από τους υπεύθυνους της ασφαλείας ή τα αυτοματοποιημένα συστήματα ασφαλείας για να βοηθήσουν τους διαχειριστές του δικτύου σε σημαντικά θέματα όπως η ασφάλεια, η απόδοση και η διαφάνεια.

07/13/12 14:28:39	Informational	Service Control Manager	7036	600
07/13/12 14:35:13	Informational	Service Control Manager	7036	599
07/13/12 14:43:32	Informational	Service Control Manager	7036	598
07/13/12 14:43:59	Informational	Service Control Manager	7036	597
07/13/12 14:44:05	Informational	Service Control Manager	7036	596
07/13/12 14:44:11	Informational	Service Control Manager	7036	595
07/13/12 14:44:12	Informational	Service Control Manager	7036	594
07/13/12 14:44:12	Informational	Service Control Manager	7036	593
07/13/12 14:53:53	Informational	Microsoft-Windows-Kernel-General	12	592
07/13/12 14:53:56	Critical	Microsoft-Windows-Kernel-Power	41	587
07/13/12 14:53:58	Informational	Microsoft-Windows-Kernel-Processor-Power	26	584
07/13/12 14:53:58	Informational	Microsoft-Windows-Kernel-Processor-Power	26	581

Εικόνα 9: Παράδειγμα των γεγονότων εκτέλεσης [26]

Τα event logs μπορούν επίσης να χρησιμοποιηθούν για τον συνδυασμό των καταχωρήσεων αρχείων καταγραφής από πολλαπλές πηγές. Η προσέγγιση αυτή, σε συνδυασμό με τη στατιστική ανάλυση, μπορεί να δώσει συσχετισμούς μεταξύ φαινομενικά άσχετων καταχωρήσεων σε διαφορετικούς διακομιστές.

Transaction Logs

Τα περισσότερα συστήματα βάσεων δεδομένων διατηρούν κάποιο είδος καταγραφής συναλλαγών, οι οποίες δεν προορίζονται ως έλεγχος για μετέπειτα ανάλυση και οι οποίες πληροφορίες είναι συνήθως μη αναγνώσιμες από τον άνθρωπο. Αυτά τα αρχεία καταγραφής συναλλαγών αποθηκεύονται σαν δεδομένα και επιτρέπουν στην βάση να ανακτήσει χαμένα δεδομένα από συντριβές ή άλλα σφάλματα και να διατηρήσει τα αποθηκευμένα δεδομένα σε μια σταθερή κατάσταση. Έτσι, τα συστήματα βάσεων δεδομένων πέρα από αρχεία καταγραφής συμβάντων, περιλαμβάνουν συνήθως και αρχεία καταγραφής συναλλαγών.

	Current LSN	Operation	Context	Xact ID	Transaction ID	Log Record Fixed Length	Log Record Length	Log Reserve	AllocUnitId
1	000005cd:00000021:0038	LOP_BEGIN_CKPT	LCX_NULL	NULL	0000:00000000	96	96	0	NULL
2	000005cd:00000038:0001	LOP_XACT_CKPT	LCX_BOOT_PAGE_CKPT	NULL	0000:00000000	24	28	0	NULL
3	000005cd:00000039:0001	LOP_END_CKPT	LCX_NULL	NULL	0000:00000000	136	136	0	NULL
4	000005cd:0000003a:0001	LOP_BEGIN_XACT	LCX_NULL	74018	0000:00014690	76	140	9162	NULL
5	000005cd:0000003a:0002	LOP_LOCK_XACT	LCX_NULL	NULL	0000:00014690	24	40	0	NULL
6	000005cd:0000003a:0003	LOP_MODIFY_ROW	LCX_IAM	NULL	0000:00014690	62	104	204	72057594047561
7	000005cd:0000003a:0004	LOP_MODIFY_ROW	LCX_PFS	NULL	0000:00014690	62	92	198	72057594047561
8	000005cd:0000003a:0005	LOP_MODIFY_ROW	LCX_PFS	NULL	0000:00014690	62	92	198	6488064
9	000005cd:0000003a:0006	LOP_INVALIDATE_CACHE	LCX_NULL	NULL	0000:00014690	26	26	32	NULL
10	000005cd:0000003a:0007	LOP_COUNT_DELTA	LCX_CLUSTERED	NULL	0000:00000000	208	208	0	458752
11	000005cd:0000003a:0008	LOP_COUNT_DELTA	LCX_CLUSTERED	NULL	0000:00000000	208	208	0	327680
12	000005cd:0000003a:0009	LOP_COUNT_DELTA	LCX_CLUSTERED	NULL	0000:00000000	208	208	0	196608
13	000005cd:0000003a:000a	LOP_COUNT_DELTA	LCX_CLUSTERED	NULL	0000:00000000	208	208	0	196608
14	000005cd:0000003a:000b	LOP_HOBT_DDL	LCX_NULL	NULL	0000:00014690	36	36	42	NULL
15	000005cd:0000003a:000c	LOP_MODIFY_ROW	LCX_CLUSTERED	NULL	0000:00014690	62	184	243	458752
16	000005cd:0000003a:000d	LOP_HOBT_DDL	LCX_NULL	NULL	0000:00014690	36	36	42	NULL
17	000005cd:0000003a:000e	LOP_MODIFY_ROW	LCX_CLUSTERED	NULL	0000:00014690	62	120	213	327680
18	000005cd:0000003a:000f	LOP_MODIFY_ROW	LCX_CLUSTERED	NULL	0000:00014690	62	124	215	281474979397632
19	000005cd:0000003a:0010	LOP_COMMIT_XACT	LCX_NULL	74018	0000:00014690	80	84	90	NULL

Εικόνα 10: Παράδειγμα των transaction logs [27]

Τα transaction logs λειτουργούν με τον ίδιο τρόπο που οι μετατροπείς (transducers) μετατρέπουν τα φαινόμενα σε μετρήσιμα σήματα. Το βασικό μοντέλο απαιτεί μετατροπείς οι οποίοι μετατρέπουν το ρεύμα ή την τάση σε ένα σήμα. Επιπλέον, συστήματα μπορεί να απαιτούν ωμικά δίκτυα για να παράγουν ένα σήμα. Κατά την ανάλυση της λειτουργίας των αρχείων καταγραφής συναλλαγών, είναι σημαντικό το γεγονός ότι τα σήματα μπορούν να μετρηθούν χρησιμοποιώντας διάφορες μεθόδους. Μια περίπλοκη κατανόηση των αρχείων καταγραφής συναλλαγών απαιτεί την εξέταση του δοθέντος σήματος και των χαρακτηριστικών του. Τέλος, τα transaction logs χωρίζονται σε δύο τύπους σημάτων: ψηφιακά και αναλογικά.

Message Logs

Τα message logs αποτελούν αρχεία καταγραφής μηνυμάτων τα οποία συναντώνται σε εφαρμογές όπως διαδικτυακές προωθήσεις μηνυμάτων (internet relay chat), άμεσων μηνυμάτων (instant messaging), και ομότιμων συστημάτων (P2P systems) (π.χ. chat, multiplayer παιχνίδια) που συνήθως έχουν τη δυνατότητα αυτόματης αποθήκευσης κειμένου, όπως επίσης, δημόσια αλλά και ιδιωτικά μηνύματα συνομιλίας μεταξύ των

χρηστών. Τα αρχεία καταγραφής μηνυμάτων είναι σχεδόν καθολικά αρχεία απλού κειμένου τα οποία μπορούν να αποθηκευτούν σε HTML μορφή ή σε μια άλλη προσαρμοσμένη μορφή για να διευκολύνουν την ανάγνωση και την κρυπτογράφηση των διάφορων μηνυμάτων.

Level	Time	Source	ID	Message
Error	9/23/08 11:35:21.433	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -
Error	9/23/08 11:35:21.433	Network tunnel	0000020a	<KERNEL> auth_final:pac_file_url
Error	9/23/08 10:06:14.232	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -
Error	9/23/08 10:06:14.232	Network tunnel	0000020a	<KERNEL> auth_final:pac_file_url
Error	9/23/08 10:05:42.959	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -
Error	9/23/08 10:05:42.959	Network tunnel	0000020a	<KERNEL> auth_final:pac_file_url
Error	9/23/08 08:39:59.670	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -
Error	9/23/08 08:39:59.670	Network tunnel	0000020a	<KERNEL> auth_final:pac_file_url
Error	9/23/08 08:29:19.043	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -
Error	9/23/08 08:29:19.043	Network tunnel	0000020a	<KERNEL> auth_final:pac_file_url
Error	9/23/08 08:22:47.305	Network tunnel	0000020a	<KERNEL> auth_final:equipment_id -

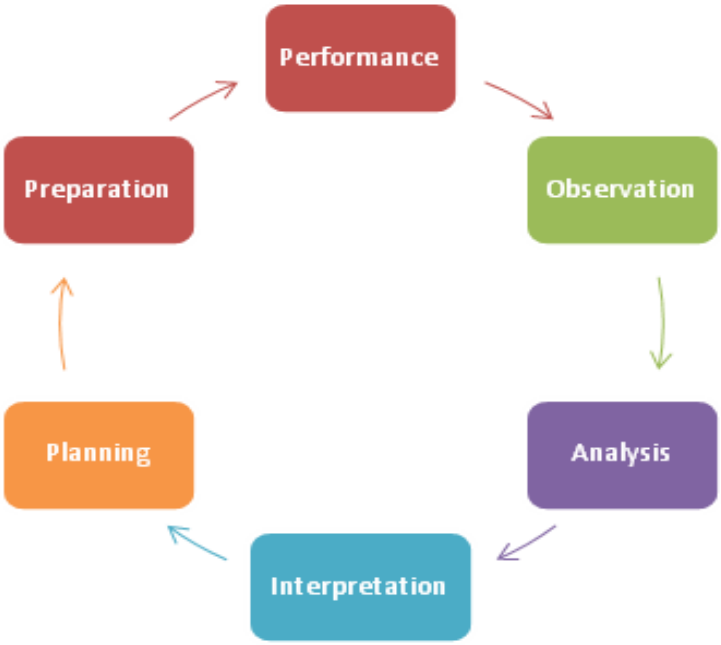
Εικόνα 11: Παράδειγμα των message logs [28]

2.6 Ο αλγόριθμος του Syer

2.6.1 Ένα απλό παράδειγμα

Ο Τζάκ, ένας αναλυτής απόδοσης, είναι υπεύθυνος για τον συνεχή έλεγχο της απόδοσης ενός συστήματος τηλεπικοινωνιών μεγάλης κλίμακας. Λαμβάνοντας υπόψη τους συνεχώς εξελισσόμενους φόρτους εργασίας (field workloads), συχνά χρειάζεται να ενημερώνει τις δοκιμές του για να εξασφαλίσει ότι οι δόκιμες του φόρτου (test workloads) που πραγματοποιεί θα αντιπροσωπεύουν, όσο το δυνατόν περισσότερο, τα field workloads. Ένα workload [12] στον τομέα της Πληροφορικής, αποτελεί το ποσό του φόρτου εργασίας που έχει δοθεί στον υπολογιστή να εκτελέσει σε μια δεδομένη στιγμή. Ο φόρτος εργασίας συνήθως χαρακτηρίζεται από κάποιο αριθμό χρηστών που συνδέονται και αλληλεπιδρούν με τις εφαρμογές του υπολογιστή.

Ο Τζάκ παρακολουθεί αυτά τα φορτία χρησιμοποιώντας μετρητές απόδοσης (π.χ. τον χρόνο απόκρισης και τη χρήση μνήμης). Σε περίπτωση που ένας ή περισσότεροι από αυτούς τους μετρητές αποκλίνει από το καθορισμένο ή αναμενόμενο εύρος (π.χ. ο χρόνος απόκρισης υπερβαίνει τον μέγιστο χρόνο απόκρισης ο οποίος καθορίζεται στο επίπεδο των υπηρεσιών ή η χρήση της μνήμης υπερβαίνει τη μέση χρήση μνήμης του ιστορικού της), ο Τζάκ πρέπει να διερευνήσει την αιτία της απόκλισης. Τότε, οφείλει να ενημερώσει τις δοκιμές του.



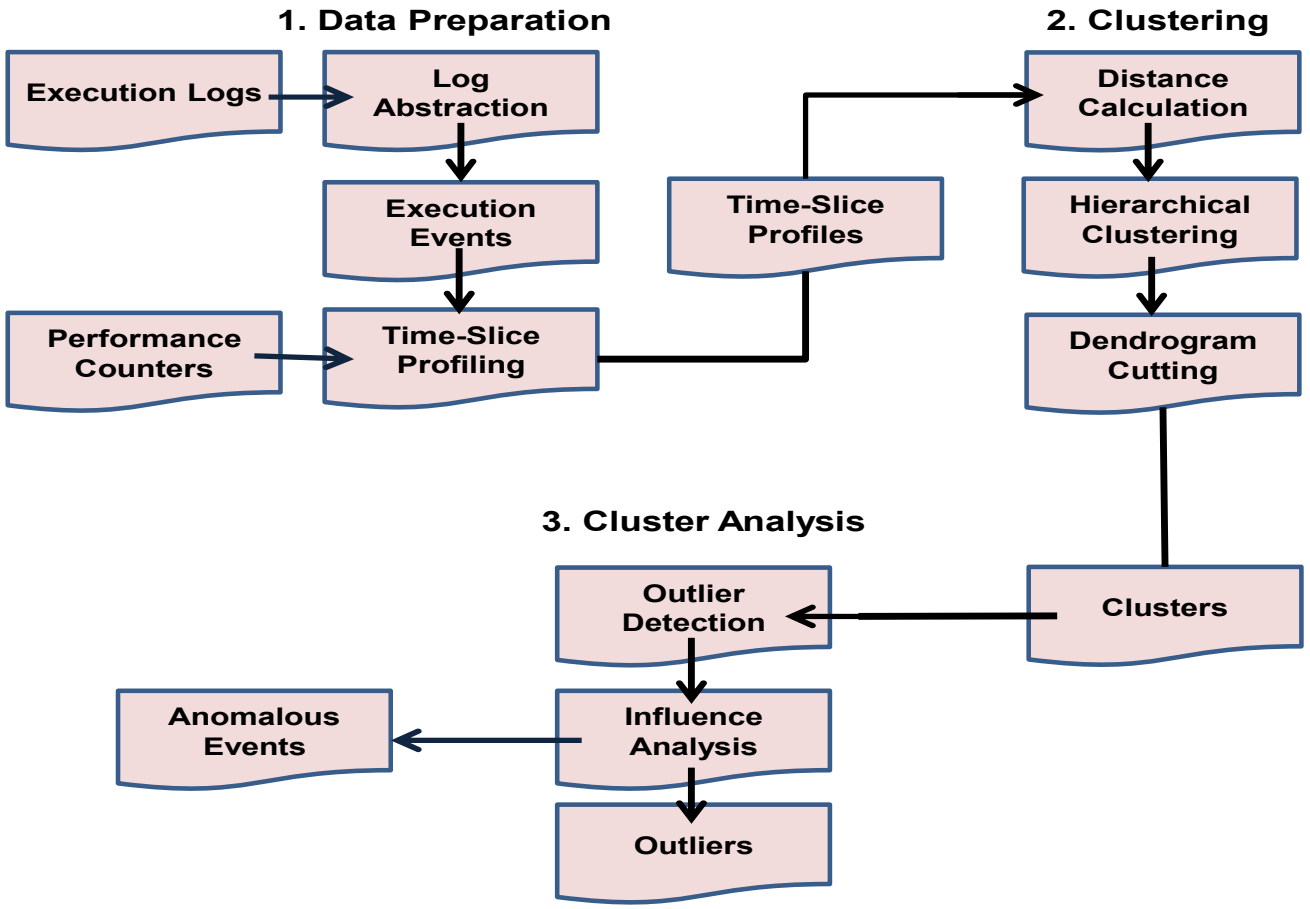
Εικόνα 12: Οι υποχρεώσεις ενός αναλυτή απόδοσης [29]

Ο ίδιος παρακολουθεί τις επιδόσεις του συστήματος στο χώρο και ανακαλύπτει ότι η χρήση της μνήμης υπερβαίνει τη μέση χρήση μνήμης του ιστορικού της. Ο Τζάκ οφείλει να ενημερώσει γρήγορα την απόδοσή των δοκιμών που διεξάγει για να αναπαράγει αυτό το πρόβλημα στο περιβάλλον δοκιμών του. Μπορεί στη συνέχεια να καθορίσει τον λόγο που το σύστημα χρησιμοποιεί περισσότερη μνήμη από ότι αναμενόταν. Παρά το γεγονός ότι οι μετρητές απόδοσης ανέφεραν ότι ο φόρτος εργασίας στον τομέα έχει αλλάξει (οδηγώντας σε αυξημένη χρήση της μνήμης), οι μόνες πληροφορίες που ο Τζάκ μπορεί να χρησιμοποιήσει για να αποδείξει κάτι τέτοιο, και ως εκ τούτου, να βρει τον τρόπο με τον οποίο θα πρέπει να ενημερώσει τις δοκιμές τις οποίες εφαρμόζει, αποτελούν τα γεγονότα εκτέλεσης.

Αυτά τα αρχεία καταγραφής περιγράφουν τη συμπεριφορά του συστήματος, δηλαδή την εκτέλεση σημαντικών γεγονότων όπως για παράδειγμα την έναρξη μιας ουράς (queuing) ή την ολοκλήρωση μιας εργασίας, στα οποία κατά τη διάρκεια εκτέλεσης τους εμφανίζεται μια σημαντική αύξηση της χρησιμοποιημένης μνήμης σε σχέση με αυτήν που αναμενόταν. Ο Τζάκ προσπαθεί να συγκρίνει αυτά τα execution logs, εξετάζοντας πόσο συχνά εμφανίζονται σημαντικά γεγονότα (π.χ. η παραλαβή ενός αιτήματος (request) στον τομέα σε σύγκριση με τις δοκιμές του). Ωστόσο, terabytes από execution logs συλλέγονται και κάποια γεγονότα συμβαίνουν εκατομμύρια φορές. Είναι αδύνατο ο Τζάκ να είναι σε θέση να μπορεί να κατανοήσει πλήρως τις διαφορές μεταξύ των field και test workloads με την απλή σύγκριση του πόσο συχνά εμφανίζεται η κάθε περίπτωση (δηλαδή το κάθε event). Για παράδειγμα, συγκρίνοντας απλά το πόσο συχνά κάθε event συμβαίνει, αγνοείται η σημαντικότερη λεπτομέρεια που είναι η αιτία που δημιούργησε τα γεγονότα (δηλαδή, το πλαίσιο).

Έτσι για να ξεπεράσει αυτό το πρόβλημα, ο Τζάκ πρέπει πέρα από τη γνώση που προσφέρουν τα execution logs, να βρει έναν τρόπο να συνδυάσει τις πληροφορίες που προσφέρουν οι μετρητές απόδοσης και να βρει ακριβώς την αιτία που προκαλεί τα γεγονότα τα οποία ευθύνονται για την κατάχρηση της μνήμης του συστήματος.

Ο Syer et al. προτείνουν μια αυτοματοποιημένη προσέγγιση ώστε τα γεγονότα εκτέλεσης να εμπλουτιστούν συνδυάζοντας τα αρχεία καταγραφής εκτέλεσης και τους μετρητές απόδοσης και να συγκρίνουν την απόδοση του συστήματος κατά τη διάρκεια των δοκιμών απόδοσης και των δοκιμών πιστοποίησης (certification testing). Η προσέγγιση αυτή αποτελείται από τρία κύρια σταδία, την προετοιμασία των δεδομένων (data preparation), την εφαρμογή της συσταδοποίησης (clustering) των δεδομένων με χρήση της ιεραρχικής ομαδοποίησης και τέλος την ανάλυση των συστάδων (cluster analysis).



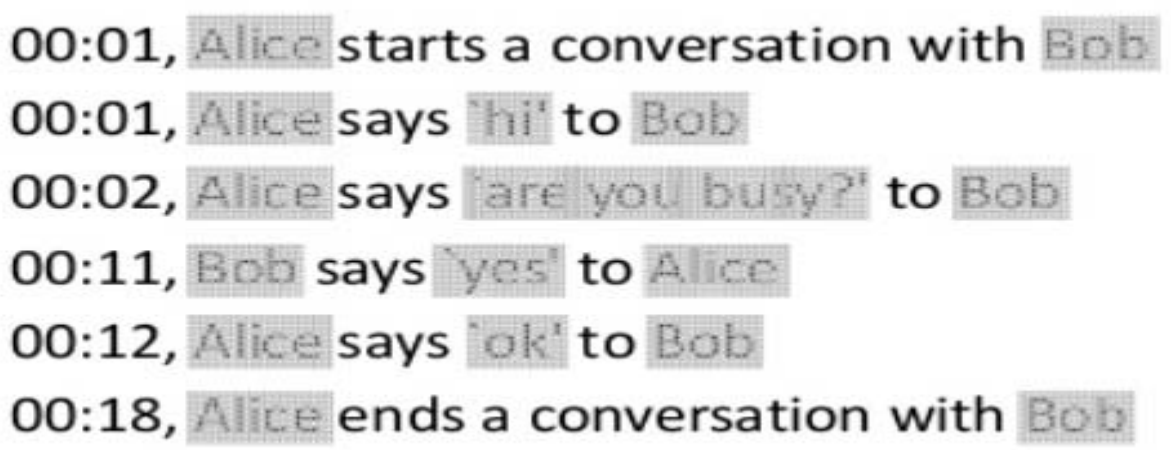
Εικόνα 13: Βήματα εκτέλεσης του αλγορίθμου των Mark D. Syer et al. με χρήση της ιεραρχικής ομαδοποίησης

2.6.2 Data Preparation

Το πρώτο βήμα στην προσέγγισή μας είναι να τροποποιήσουμε τα αρχεία καταγραφής εκτέλεσης και τους μετρητές απόδοσης για αυτοματοποιημένη ανάλυση. Η προετοιμασία των δεδομένων είναι μια διαδικασία δύο σταδίων. Πρώτα, αφαιρούμε την υλοποίηση (implementation) και συγκεκριμένες λεπτομέρειες από τα αρχεία καταγραφής εκτέλεσης προκειμένου να δημιουργήσουμε ένα σύνολο από execution events. Δεύτερον, δημιουργούμε υπογραφές απόδοσης (signature performance) τις οποίες ονομάζουμε time-slice profiles μετρώντας τον αριθμό των γεγονότων εκτέλεσης και την αλλαγή στην συνολική χρήση των πόρων σε κάθε διάστημα δειγματοληψίας.

Log Abstraction

Τα execution logs δεν είναι συνήθως σχεδιασμένα για αυτοματοποιημένη ανάλυση. Κάθε εκτέλεση ενός γεγονότος οδηγεί σε μια ελαφρώς διαφορετική γραμμή του αρχείου καταγραφής. Αυτό συμβαίνει διότι κάθε γραμμή του αρχείου καταγραφής περιέχει στατικά συστατικά, καθώς και δυναμικές πληροφορίες οι οποίες μπορεί να είναι διαφορετικές για κάθε εμφάνιση της γραμμής αυτής. Ως εκ τούτου, πριν αρχίσει η ανάλυση, θα πρέπει να αφαιρέσουμε αυτές τις δυναμικές πληροφορίες από το αρχείο καταγραφής γραμμών ώστε να είμαστε σε θέση να κατηγοριοποιήσουμε τα γεγονότα εκτέλεσης [30].



Εικόνα 14: Παράδειγμα απαλοιφής της δυναμικής πληροφορίας των γεγονότων εκτέλεσης [8]

Επιπλέον μερικές γραμμές καταγραφής συνεχίζουν να διατηρούν κάποιες δυναμικές πληροφορίες. Ο λόγος της διατήρησης αυτής είναι ότι μερικές δυναμικές πληροφορίες μπορεί να σχετίζονται με την διευθέτηση της μνήμης (π.χ. το μέγεθος μιας ουράς η ενός αρχείου). Επομένως, οι δυναμικές πληροφορίες διατηρούνται τοποθετώντας τους αριθμούς σε εύρη (π.χ. σε ποσοστιαία βάση). Ένας μοναδικός αριθμός ID (identification) τοποθετείται σε κάθε ένα από τα execution events για αυτοματοποιημένη ανάλυση και για επίτευξη συντομίας.

Time-slice profiling

Το επόμενο βήμα της διαδικασίας αποτελεί η ομαδοποίηση των time-slice προφίλ σε ομάδες με κοινή log activity (π.χ. κάποιες συγκεκριμένες στιγμές όπου ένα κοινό σετ γεγονότων έχει πραγματοποιηθεί). Αυτό γίνεται επειδή αναμένουμε ότι παρόμοια log δράση θα οδηγήσει σε παρόμοιο δέλτα της μνήμης. Θέματα τα οποία σχετίζονται με την μνήμη θα επηρεάζουν αυτά τα δέλτα μνήμης. Στην δίκη μας προσέγγιση, όσον αφορά την συσταδοποίηση των δεδομένων (time-slice profiles), έχει εφαρμοστεί ο k-means αλγόριθμος.

Σε πρώτο στάδιο, εμπλουτίζουμε τα execution events συνδυάζοντας τα αρχεία καταγραφής εκτέλεσης με τους μετρητές απόδοσης για να δημιουργήσουμε υπογραφές απόδοσης οι οποίες ονομάζονται time-slice profiles. Οι υπογραφές αυτές χαρακτηρίζονται από την επίδραση που έχει η συνολική συμπεριφορά του χρήστη πάνω στους πόρους του συστήματος (resource usage) όσον αφορά την χρήση λειτουργιών οι οποίες εκφράζονται από τα execution events κατά τη διάρκεια ενός μικρού χρονικού διαστήματος. Συνδυάζουμε τους μετρητές απόδοσης με τα γεγονότα εκτέλεσης χρησιμοποιώντας σφραγίδες χρόνου (time stamps). Όταν μια γραμμή καταγραφής αρχείου (log line) παράγεται ή πραγματοποιείται δειγματοληψία ενός μετρητή απόδοσης, τότε αυτά αποθηκεύονται σε ένα αρχείο καταγραφής μαζί με την ημερομηνία και την ώρα της δειγματοληψίας.

Παρόλο που οι μετρητές απόδοσης και τα αρχεία καταγραφής της εκτέλεσης περιέχουν time stamps, ο συνδυασμός αυτών των δύο είναι μια μεγάλη πρόκληση. Αυτό συμβαίνει επειδή οι μετρητές απόδοσης λαμβάνονται σε τακτά χρονικά διαστήματα, ενώ τα αρχεία καταγραφής εκτέλεσης προκύπτουν συνεχώς. Ως εκ τούτου, θα πρέπει να διαχωρίσουμε την εκτέλεση των logs με τέτοιο τρόπο ώστε να συνυπάρχουν με τους μετρητές απόδοσης. Ο διαχωρισμός αυτός μας επιτρέπει επίσης να αιτιολογήσουμε την καθυστερημένη επίδραση ορισμένων λειτουργιών όσον αφορά τους μετρητές απόδοσης. Για παράδειγμα, μπορεί να υπάρχει μια μικρή καθυστέρηση μεταξύ της στιγμής που μια γραμμή καταγραφής παράγεται και της στιγμής που η αντίστοιχη λειτουργία εκτελείται. Ο διαχωρισμός αυτός βοηθά επίσης στο να μειωθεί η επιβάρυνση που επιβλήθηκε στο σύστημα κατά τη φάση του performance testing, διότι η συχνότητα της δειγματοληψίας μετρητών απόδοσης μπορεί να μειωθεί.

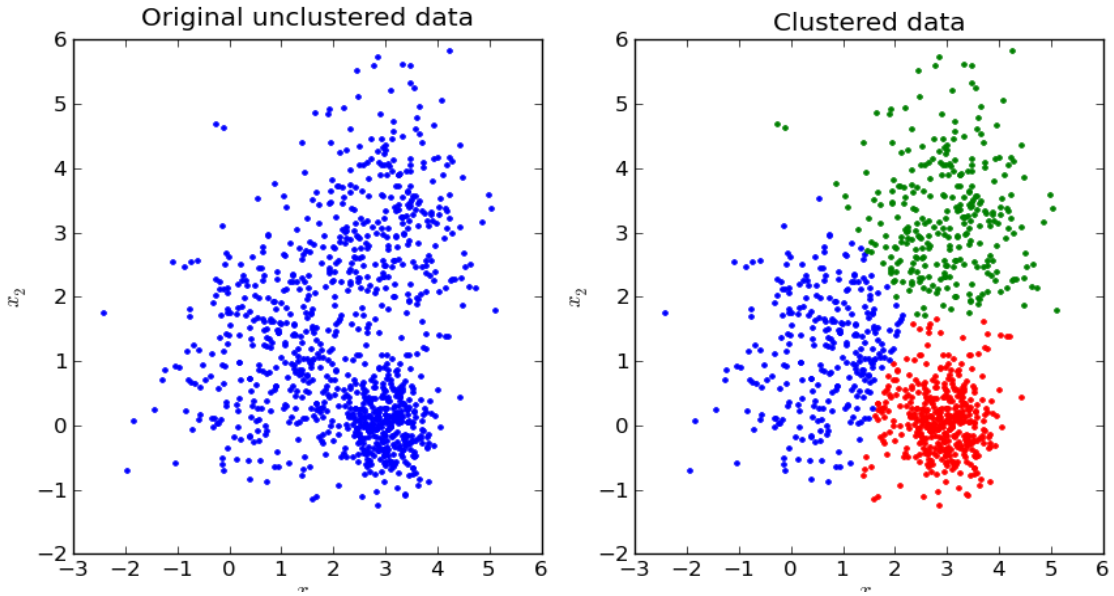
Κατά τη διάρκεια της χρονικής περιόδου μεταξύ δύο διαδοχικών δειγμάτων μετρητών απόδοσης (δηλαδή, μιας χρόνο-φέτας), μηδέν ή περισσότερες γραμμές καταγραφής μπορούν να παραχθούν από τα γεγονότα που συμβαίνουν εντός του συστήματος. Για παράδειγμα, μια γραμμή καταγραφής μπορεί να δημιουργηθεί όταν ένα νέο στοιχείο διεργασίας έχει ξεκινήσει, βρίσκεται στην ουρά ή έχει ολοκληρωθεί, ή όταν εμφανιστεί κάποιο σφάλμα κατά τη διάρκεια της χρόνο-φέτας. Τα execution events τότε διαχωρίζονται δημιουργώντας μια υπογραφή απόδοσης για κάθε time-slice.

Αυτή η υπογραφή δημιουργείται μετρώντας τον αριθμό των φορών που κάθε execution event εκτελείται κατά τη διάρκεια της χρόνο-φέτας και υπολογίζοντας την μεταβολή της χρήσης των πόρων μεταξύ της έναρξης και της λήξης της. Ως εκ τούτου, κάθε time-slice προφίλ έχει δύο συνιστώσες: μια δραστηριότητα καταγραφής, η οποία αποτελεί μια μέτρηση του κάθε execution event που έχει συμβεί κατά τη διάρκεια της χρόνο-φέτας και δεύτερον τη διάφορα της συνολικής χρήσης των πόρων του συστήματος στην πάροδο του time-slice, δηλαδή το δέλτα (delta) της μνήμης.

Τα execution logs και οι μετρητές απόδοσης συλλέγονται από το ίδιο μηχάνημα, με αποτέλεσμα να μην χρειάζεται να συγχρονίσουμε τη συλλογή των δεδομένων (data collection). Επιπλέον, η τεχνική παραγωγής υπογραφών που χρησιμοποιείται, δεν εξαρτάται από το περιεχόμενο και τη μορφή των μετρητών απόδοσης και των αρχείων καταγραφής εκτέλεσης. Δε βασίζεται σε κάποια συναλλαγή (transaction) / νήμα (thread) ή διεργασία και δεν δέχεται καμία ετικέτα, εκτός από μία χρονοσήμανση.

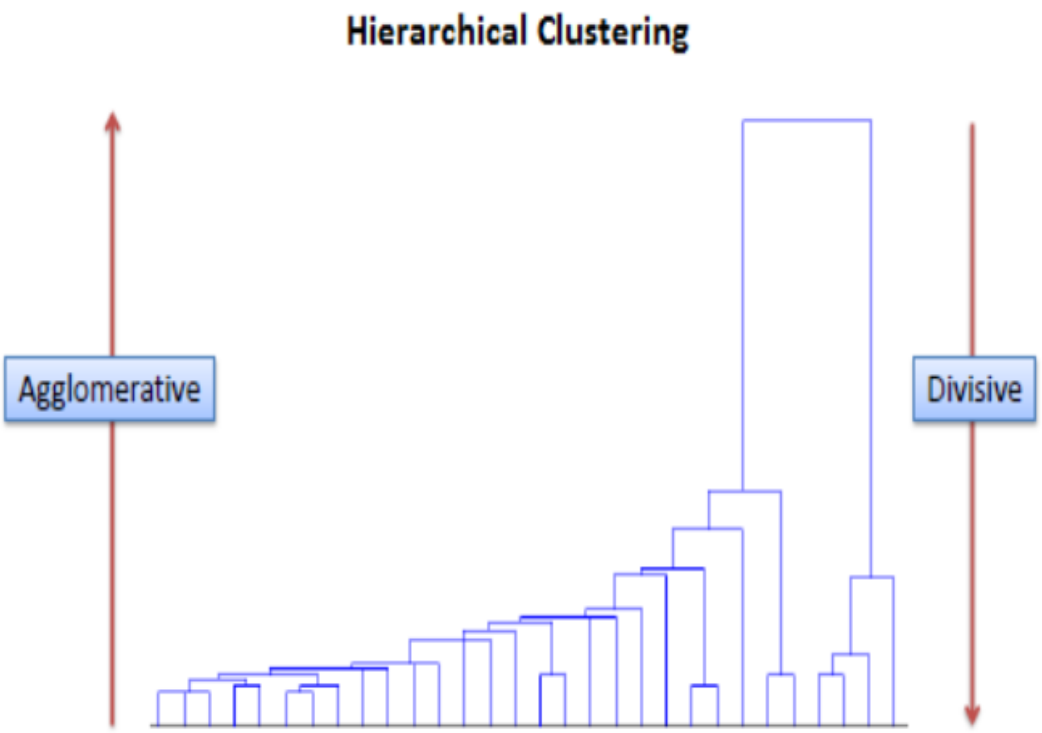
2.6.3 Hierarchical Clustering

Η ιεραρχική ομαδοποίηση αποτελεί έναν αλγόριθμο ομαδοποίησης που οδηγεί σε ένα καλό σχήμα συσταδοποίησης για ένα σύνολο δεδομένων. Για τη επιλογή του αλγορίθμου χρησιμοποιείται το μέτρο γειτνίασης και το κριτήριο συσταδοποίησης τα οποία ορίζουν απόλυτα τον αλγόριθμο, καθώς επίσης και η δυνατότητά του να καθορίσει ένα σχήμα συσταδοποίησης που να προσαρμόζεται στο συγκεκριμένο σύνολο δεδομένων.

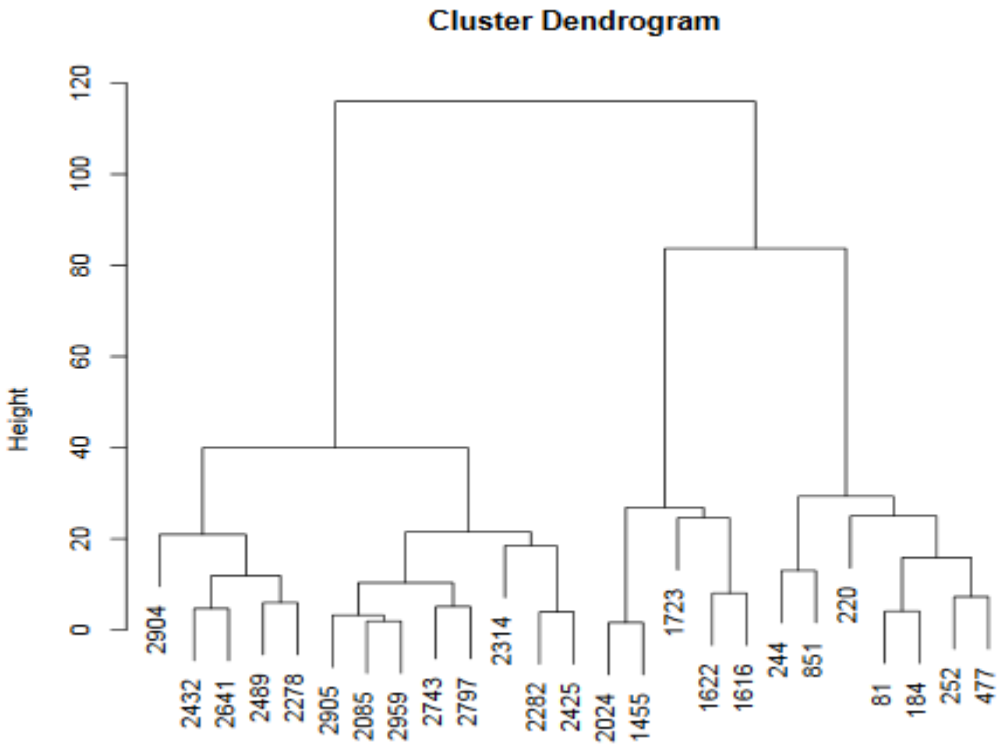


Εικόνα 15: Παράδειγμα συσταδοποίησης δεδομένων [32]

Στην ιεραρχική ομαδοποίηση, τα δεδομένα δεν χωρίζονται σε ένα συγκεκριμένο σύμπλεγμα σε ένα μόνο βήμα. Αντίθετα, μια σειρά από διαχωρισμούς λαμβάνει χώρα, η οποία μπορεί να έχει σαν αποτέλεσμα είτε μία μόνο συστάδα που περιέχει όλα τα αντικείμενα είτε n συστάδες οι οποίες περιέχουν η κάθε μια, ένα μόνο αντικείμενο. Η ιεραρχική ομαδοποίηση υποδιαιρείται σε δυο μεθόδους, στην αθροιστική μέθοδο (agglomerative), η οποία εφαρμόζει μια σειρά συγχωνεύσεων των n αντικειμένων σε ομάδες, και στην διαιρετική μέθοδο (divisive), η οποία διαχωρίζει τα n αντικείμενα διαδοχικά σε μικρότερες ομάδες. Η ιεραρχική ομαδοποίηση μπορεί να εκπροσωπείται από ένα δισδιάστατο διάγραμμα γνωστό ως δενδρόγραμμα, το οποίο απεικονίζει τις συγχωνεύσεις ή διασπάσεις οι οποίες πραγματοποιούνται σε κάθε διαδοχικό στάδιο της ανάλυσης.



Εικόνα 16: Αθροιστική και διαιρετική ιεραρχική ομαδοποίηση [33]



Εικόνα 17: Δενδρόγραμμα των συστάδων [34]

Distance Calculation

Κάθε time-slice προφίλ εκπροσωπείται από ένα σημείο σε ένα πολυδιάστατο χώρο. Οι διαδικασίες ομαδοποίησης βασίζονται στον εντοπισμό σημείων που βρίσκονται "κοντά" σε αυτό το πολυδιάστατο χώρο. Ως εκ τούτου, θα πρέπει να προσδιοριστεί με ποιον τρόπο ακριβώς θα μετρηθεί η απόσταση σε αυτό το χώρο. Όσο μεγαλύτερη είναι η απόσταση μεταξύ δύο σημείων, τόσο μεγαλύτερη θα είναι η ανομοιότητα μεταξύ των time-slice προφίλ τα οποία αντιπροσωπεύουν. Αρχικά υπολογίζονται όλες οι μεταξύ τους αποστάσεις των σημείων στο χώρο και αποθηκεύονται σε έναν πίνακα αποστάσεων (distance matrix).

Η μέτρηση της απόστασης υπολογίζεται σύμφωνα με την Pearson απόσταση καθώς η μετρική αυτή οδηγεί τις περισσότερες φορές σε μια συσταδοποίηση που είναι πιο κοντά στην αληθινή συσταδοποίηση (true clustering) [35], [36]. Χρησιμοποιείται η Pearson συσχέτιση (correlation) για να υπολογιστεί η ομοιότητα μεταξύ δυο προφίλ. Αυτή η μετρική δίνει τιμές μεταξύ -1 και +1, όπου η τιμή του 1 υποδεικνύει ότι δύο προφίλ είναι όμοια, η τιμή του 0 υποδεικνύει ότι δεν υπάρχει σχέση μεταξύ των προφίλ και η τιμή του -1 ότι υπάρχει μία αντίθετη σχέση μεταξύ των προφίλ (π.χ. όσο αυξάνεται η συχνότητα ενός συγκεκριμένου γεγονότος εκτέλεσης στο ένα προφίλ, τόσο μειώνεται στο άλλο) [37].

$$\rho = \frac{n\sum_i^n x_i \times y_i - \sum_i^n x_i \times \sum_i^n y_i}{\sqrt{(n\sum_i^n x_i^2 - (\sum_i^n x_i)^2) \times (n\sum_i^n y_i^2 - (\sum_i^n y_i)^2)}} \quad (2.1)$$

Linkage criteria

Η ιεραρχική ομαδοποίηση ενημερώνει τον πίνακα αποστάσεων με βάση συγκεκριμένων κριτήριων σύνδεσης. Τα κριτήρια σύνδεσης (linkage criteria) καθορίζουν την απόσταση μεταξύ των συνόλων των παρατηρήσεων και αποτελούν μία συνάρτηση των αποστάσεων μεταξύ των ζευγών των παρατηρήσεων. Τέλος, χρησιμοποιούνται στην εκτέλεση της ιεραρχικής ομαδοποίησης και τα σημαντικότερα από αυτά παραθέτονται παρακάτω:

Average Linkage

Στην ιεραρχική ομαδοποίηση με χρήση της μέσης σύνδεσης (average linkage), η απόσταση μεταξύ δύο συστάδων ορίζεται ως ο μέσος όρος των αποστάσεων μεταξύ όλων των ζευγών των αντικειμένων, όπου κάθε ζεύγος αποτελείται από ένα μόνο αντικείμενο από κάθε ομάδα.

Στη μέθοδο μέσης σύνδεσης, η απόσταση $D(r,s)$ υπολογίζεται ως:

$$D(r,s) = T_{rs} / (N_r * N_s) \quad (2.2)$$

Όπου T_{rs} είναι το άθροισμα όλων των ζευγών των αποστάσεων μεταξύ του συμπλέγματος r και συμπλέγματος s. N_r και N_s είναι τα μεγέθη των συστάδων r και s, αντίστοιχα. Σε κάθε στάδιο της ομαδοποίησης, οι συστάδες r και s, για τις οποίες η απόσταση $D(r,s)$ είναι η ελάχιστη, συγχωνεύονται.

Σε αντίθεση με πολλά άλλα κριτήρια σύνδεσης (linkage criteria), αυτός ο τρόπος σύνδεσης είναι ο πιο κατάλληλος σε περίπτωση που λίγες πληροφορίες είναι διαθέσιμες για την αναμενόμενη ομαδοποίηση (π.χ. τα σχετικά μεγέθη των αναμενόμενων συστάδων).

Κάθε φορά που οι δύο συστάδες συγχωνεύονται, η ιεραρχική μέθοδος μέσης σύνδεσης αφαιρεί τις νέες συστάδες από τον πίνακα αποστάσεων και προσθέτει το νέο σύμπλεγμα υπολογίζοντας την απόσταση μεταξύ του νέου συμπλέγματος και όλων των υφιστάμενων συστάδων. Η απόσταση μεταξύ δύο συστάδων ορίζεται ως η μέση απόσταση (όπως υπολογίζεται από την απόσταση Pearson) μεταξύ των προφίλ της πρώτης συστάδας και των προφίλ της δεύτερης [38], [39].

Single linkage

Μια από τις απλούστερες αθροιστικές ιεραρχικές μεθόδους ομαδοποίησης είναι αυτή της μονής σύνδεσης (single linkage), επίσης γνωστή ως η τεχνική του πλησιέστερου γείτονα (nearest neighbor) [40]. Το καθοριστικό χαρακτηριστικό της μεθόδου είναι ότι η απόσταση μεταξύ των ομάδων ορίζεται ως η απόσταση μεταξύ του πλησιέστερου ζεύγους αντικειμένων, όπου θεωρούνται ζεύγη αυτά τα οποία αποτελούνται από ένα αντικείμενο από κάθε ομάδα.

Στην μέθοδο μονής σύνδεσης η απόσταση $D(r,s)$ ορίζεται ως:

$D(r,s)$ = Min { $d(i,j)$: όπου το αντικείμενο i ανήκει στο cluster r και το αντικείμενο j στο cluster s}

Υπολογίζεται η απόσταση μεταξύ κάθε πιθανού ζεύγους αντικείμενων (i,j), όπου το αντικείμενο i βρίσκεται στο σύμπλεγμα r και το αντικείμενο j σε ένα άλλο σύμπλεγμα s. Η ελάχιστη τιμή των αποστάσεων αυτών αποτελεί την απόσταση μεταξύ των συστάδων r και s. Με άλλα λόγια, η απόσταση μεταξύ δύο συστάδων δίνεται από την τιμή της μικρότερης σχέσης (link) μεταξύ των συστάδων.

Σε κάθε στάδιο της ιεραρχικής ομαδοποίησης [42] [43], οι συστάδες r και s, για τις οποίες η απόσταση $D(r,s)$ είναι ελάχιστη, συγχωνεύονται. Στην περίπτωση αυτή, οι εν λόγω δύο συστάδες συγχωνεύονται έτσι ώστε η πρόσφατα σχηματισμένη συστάδα, κατά μέσο όρο, θα έχει την ελάχιστη απόσταση ζευγών μεταξύ των σημείων.

Ο παραλληλισμός της ιεραρχικής ομαδοποίησης είναι μια πολύπλοκη διαδικασία καθώς εμφανίζει μια εξάρτηση εγγενών στοιχείων κατά την διάρκεια της κατασκευής του ιεραρχικού δέντρου. Η εφαρμογή της ιεραρχικής ομαδοποίησης σε Spark είναι εφικτή χάρις έναν ιεραρχικό αλγόριθμο βασισμένο σε single linkage ο οποίος προτείνεται από τους Chen Jin et al [41].

Complete linkage

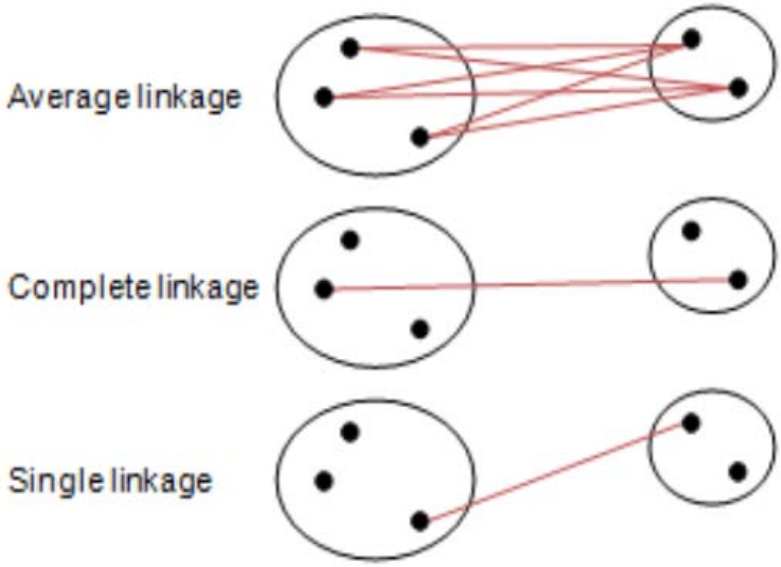
Στην μέθοδο πλήρους σύνδεσης, η οποία ονομάζεται επίσης μέθοδος απομακρυσμένου γείτονα (farthest neighbor), αποτελεί την αντίθετη μορφή με σε σχέση με αυτήν της ενιαίας σύνδεσης. Η απόσταση μεταξύ των ομάδων ορίζεται ως η απόσταση ανάμεσα στο πιο απομακρυσμένο ζεύγος αντικειμένων, ένα από κάθε ομάδα.

Στην πλήρη μέθοδο σύνδεσης, η απόσταση $D(r,s)$ υπολογίζεται ως:

$D(r,s)$ = Max { $d(i,j)$: όπου το αντικείμενο i ανήκει στο cluster r και το αντικείμενο j στο cluster s}

Υπολογίζεται η απόσταση μεταξύ κάθε πιθανού ζεύγους αντικείμενων (i,j), όπου i το αντικείμενο στο σύμπλεγμα r και j το αντικείμενο στο σύμπλεγμα s και η μέγιστη τιμή των αποστάσεων αυτών ορίζεται ως η απόσταση μεταξύ των συστάδων r και s. Η απόσταση μεταξύ δύο συστάδων δίνεται από την τιμή της μακρύτερης σχέσης μεταξύ των συστάδων.

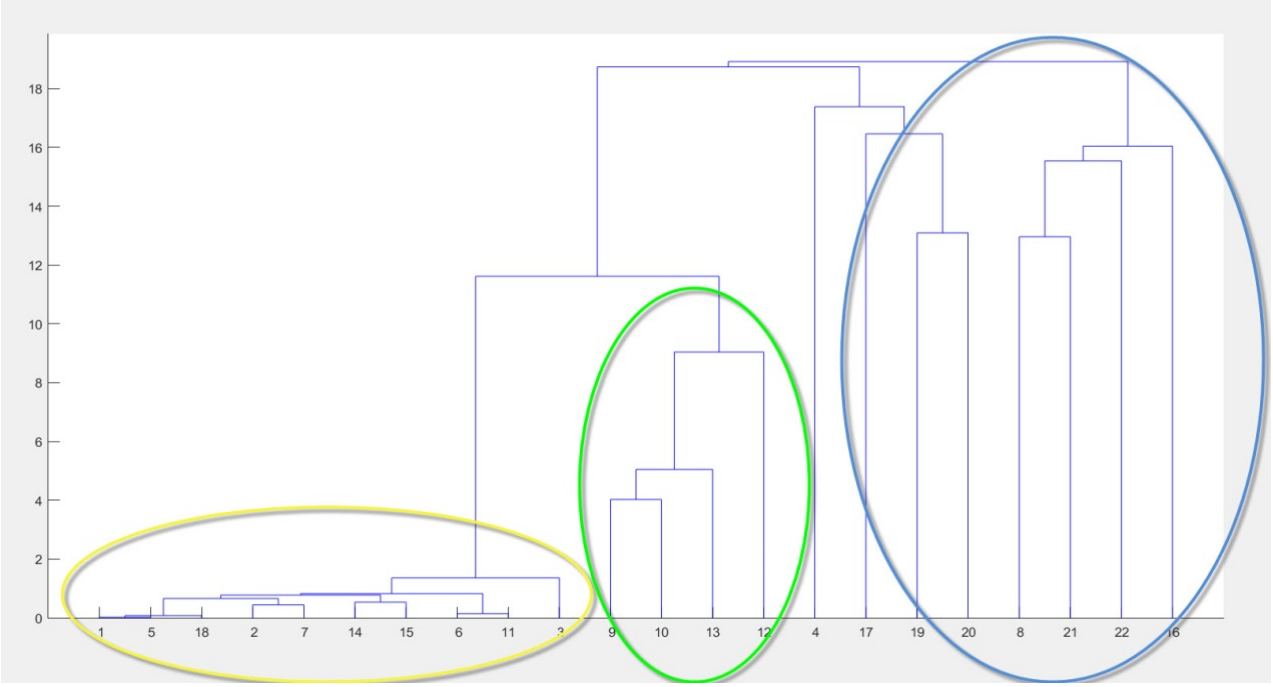
Σε κάθε στάδιο της ιεραρχικής ομαδοποίησης, οι συστάδες r και s, για τις οποίες η απόσταση $D(r,s)$ είναι η ελάχιστη, συγχωνεύονται.



Εικόνα 18: Διαφορετικές μορφές σύνδεσης της ιεραρχικής ομαδοποίησης

Dendrogram Cutting

Το αποτέλεσμα της εφαρμογής της ιεραρχικής ομαδοποίησης αποτελεί μια ιεραρχία συστάδων που συνήθως οπτικοποιείται χρησιμοποιώντας ιεραρχικά δενδρογράμματα συστάδων. Αυτά είναι διαγράμματα με την μορφή δυαδικών δέντρων και παρουσιάζουν κάθε στάδιο της ομαδοποίησης ως ένθετες μεμονωμένες συστάδες. Για να ολοκληρωθεί η διαδικασία της ομαδοποίησης, το δενδρόγραμμα πρέπει να κοπεί σε κάποιο ύψος. Αυτό οδηγεί σε μια ομαδοποίηση όπου κάθε time-slice προφίλ έχει ανατεθεί σε ένα μόνο σύμπλεγμα. Η κοπή του δενδρογράμματος γίνεται είτε χειροκίνητα (με οπτική επιθεώρηση), είτε με στατιστικές δοκιμασίες οι οποίες αναφέρονται ως κανόνες διακοπής (stopping rules).



Εικόνα 19: Δημιουργία διαφορετικού clustering με κοπή του δενδρογράμματος [44]

Παρόλο που μία οπτική επιθεώρηση του δενδρογράμματος είναι ευέλικτη και γρήγορη, υπόκειται σε ανθρώπινο σφάλμα και δεν μπορεί να είναι αξιόπιστη. Έτσι στον αλγόριθμο που περιγράφεται στην δημοσίευση των Mark D. Syer et al., εφαρμόζεται ο κανόνας διακοπής Calinski-Harabasz. Ο Calinski-Harabasz κανόνας είναι pseudo-F-statistic και αποτελεί μία αναλογία που αντικατοπτρίζει την ομοιότητα εντός της συστάδας και την ανομοιότητα μεταξύ του συμπλέγματος [45]. Η βέλτιστη ομαδοποίηση παρουσιάζει υψηλή ομοιότητα εντός συστάδας (δηλαδή, τα time-slice προφίλ μέσα σε μια συστάδα θα είναι πολύ παρόμοια) και υψηλή ανομοιότητα μεταξύ συμπλέγματος (δηλαδή, τα προφίλ μεταξύ δύο διαφορετικών ομάδων θα είναι πολύ ανόμοια).

Όσον αφορά την ομαδοποίηση των time-slice προφίλ, ο Mark D. Syer et al. εφαρμόζουν την αθροιστική ιεραρχική μέθοδο χρησιμοποιώντας το average linkage σαν τρόπο σύνδεσης και τον Calinski-Harabasz όσον αφορά τους κανόνες διακοπής της ιεραρχικής ομαδοποίησης.

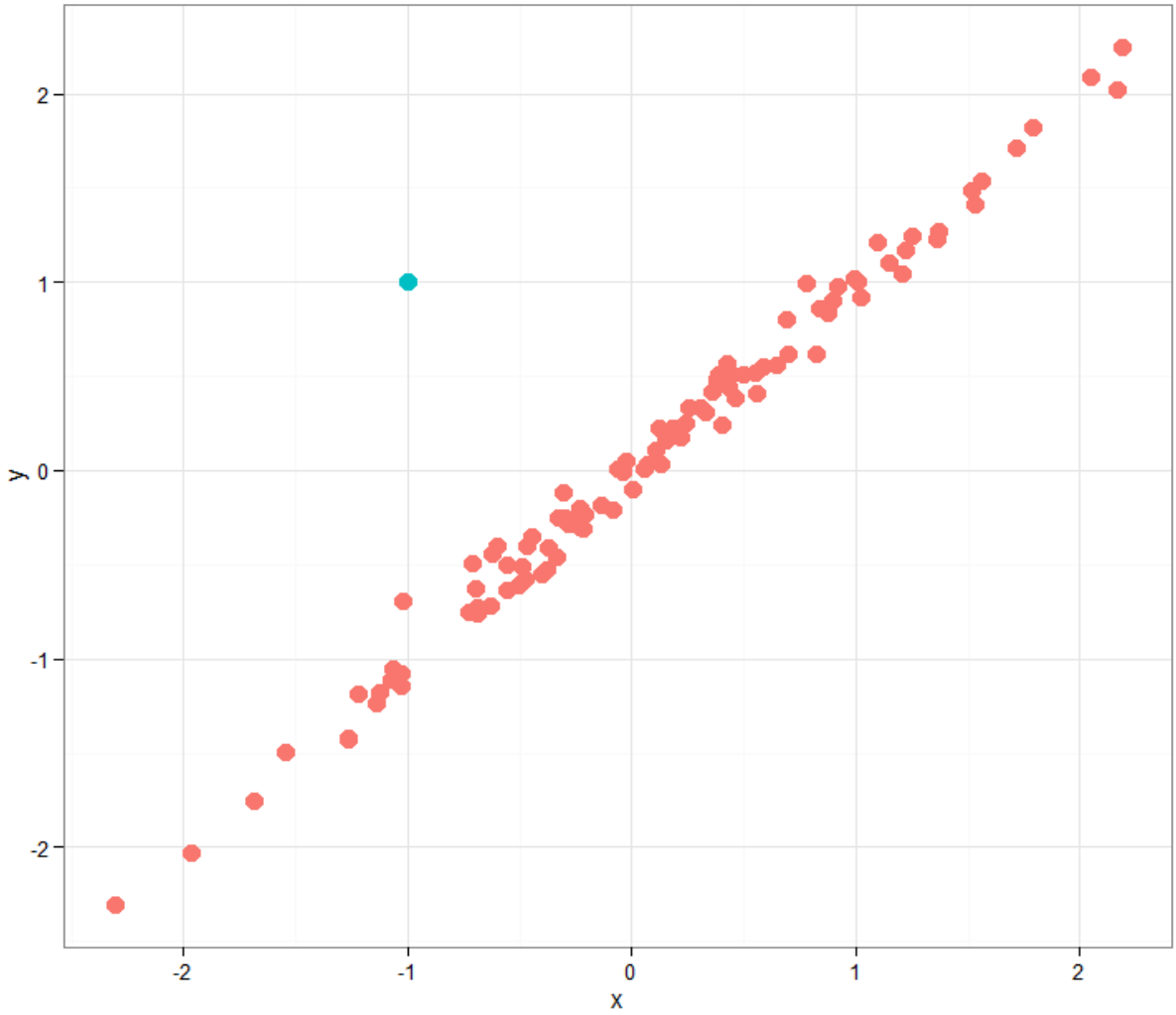
2.6.4 Cluster Analysis

Το τρίτο και τελευταίο βήμα στην προσέγγισή των Syer et al. είναι να εντοπιστούν οι log γραμμές οι οποίες αντιστοιχούν στη λειτουργικότητα που είναι υπεύθυνη για θέματα μνήμης που προέρχονται από το σύνολο των δεδομένων. Αρχικά, εντοπίζονται οι απομακρυσμένες συστάδες (outlying clusters). Στην συνέχεια, γίνεται αναγνώριση των βασικών γραμμών καταγραφής των απομακρυσμένων συμπλεγμάτων. Όπως και στο προηγούμενο βήμα, χρησιμοποιούνται ισχυρές στατιστικές τεχνικές για την αυτοματοποίηση αυτού του βήματος.

Outlier Detection

Σ' αυτό το στάδιο εντοπίζονται οι απομακρυσμένες συστάδες. Στον τομέα της στατιστικής, ένα ακραίο γεγονός (outlier) αποτελεί ένα σημείο παρατήρησης το οποίο είναι απομακρυσμένο από άλλες παρατηρήσεις [46]. Μια ακραία τιμή μπορεί να οφείλεται στην μεταβλητότητα της μέτρησης ή μπορεί να υποδείξει πειραματικό σφάλμα. Το τελευταίο μερικές φορές αποκλείεται από το σύνολο δεδομένων.

Οι ακραίες τιμές μπορεί να έχουν προκύψει κατά τύχη σε οποιαδήποτε κατανομή, αλλά συχνά δείχνουν, είτε ότι υπάρχει κάποιο σφάλμα στις μετρήσεις, είτε ότι ο πληθυσμός έχει μια κατανομή βαριάς ουράς (heavy-tailed). Στην πρώτη περίπτωση επιθυμούμε να απορρίψουμε τα απομακρυσμένα σημεία ή να χρησιμοποιήσουμε στατιστικά στοιχεία που είναι ανθεκτικά σε ακραίες τιμές, ενώ στη δεύτερη περίπτωση αποδεικνύεται ότι η κατανομή έχει υψηλή ασυμμετρία και ότι πρέπει να είμαστε προσεκτικοί όσον αφορά τη χρήση εργαλείων που αναλαμβάνουν μια κανονική κατανομή.



Εικόνα 20: Εύρεση μίας απομακρυσμένης συστάδας [47]

Στην περίπτωση μας, ο εντοπισμός των απομακρυσμένων συμπλεγμάτων επιτυγχάνεται χάρις την εξέταση των δέλτα μνήμης από κάθε ένα από τα time-slice προφίλ σε κάθε μία από τις συστάδες. Οι απομακρυσμένες συστάδες θα περιέχουν κάποια time-slice προφίλ τα οποία έχουν σημαντικό αντίκτυπο στην χρήση μνήμης (όπως αποδεικνύεται από τα δέλτα μνήμης). Δεδομένης της μεγάλης ποικιλίας των θεμάτων που σχετίζονται με την απόδοση της μνήμης, αποτελεί μεγάλη πρόκληση ο προσδιορισμός των προφίλ που έχουν το σωστό αντίκτυπο στην χρήση της μνήμης.

Οι Syer et al. στην προσέγγιση τους, προτείνουν κάποιες ανεπτυγμένες τεχνικές βαθμολόγησης για τον εντοπισμό των συστάδων που σχετίζονται με κάποιο θέμα διευθέτησης της μνήμης. Ο πίνακας 2.1 παρουσιάζει τις τεχνικές που χρησιμοποιούνται για να προσδιοριστούν τα παροδικά προβλήματα μνήμης (δηλαδή, τα memory spikes) καθώς επίσης και τα επίμονα προβλήματα μνήμης (δηλαδή, τα memory leaks και οι memory bloat περιπτώσεις).

Πίνακας 1: Τεχνικές βαθμολόγησης για τον εντοπισμό των συστάδων που σχετίζονται με κάποιο θέμα διευθέτησης της μνήμης

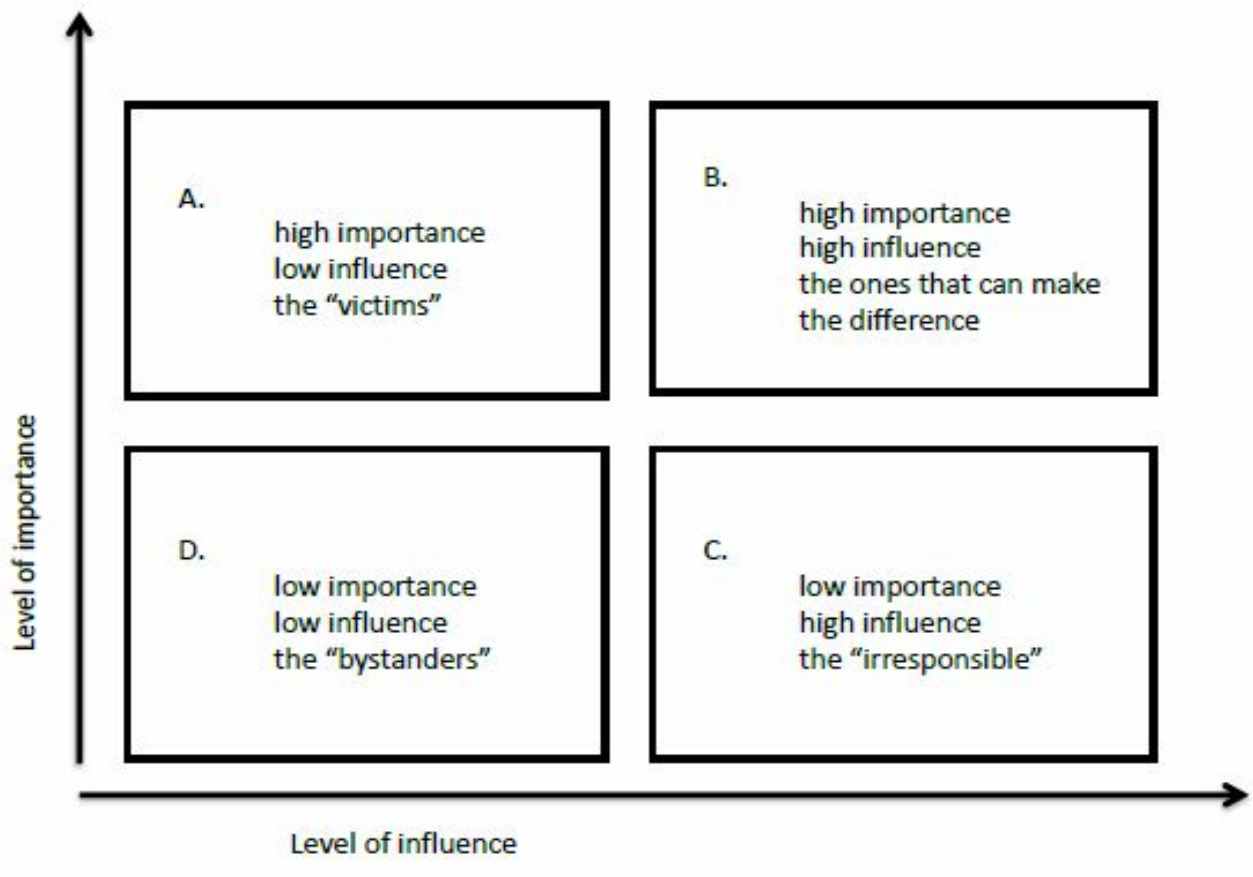
	Transient Memory Issues	Persistent Memory Issues	
Motivation	Η Λειτουργικότητα η οποία προκαλεί ένα memory spike πρόβλημα, χαρακτηρίζεται από ένα υψηλότερο από το μέσο όρο της μνήμης δέλτα σε μικρές συστάδες ή έναν συνδυασμό από ένα υψηλότερο από το μέσο όρο δέλτα μνήμης και μια υψηλότερη από το μέσο όρο τυπική απόκλιση του δέλτα μνήμης σε μεγαλύτερες συστάδες. Η τυπική απόκλιση κλιμακώνεται σύμφωνα με το μέγεθος της συστάδας για να τονίσει την τυπική απόκλιση σε μεγαλύτερες συστάδες.	Η Λειτουργικότητα η οποία προκαλεί ένα memory bloat θέμα, χαρακτηρίζεται από ένα υψηλότερο από το μέσο όρο δέλτα της μνήμης.	Η Λειτουργικότητα η οποία προκαλεί μια διαρροή μνήμης, χαρακτηρίζεται από έναν συνδυασμό ενός υψηλότερου από τον μέσο όρο δέλτα μνήμης και μια υψηλότερη από το μέσο όρο τυπική απόκλιση του δέλτα μνήμης. Προστίθεται η τυπική απόκλιση γιατί λειτουργικότητα με μεταβλητό δέλτα μνήμης μπορεί να υποδεικνύει επίσης διαρροές μνήμης.
Score	$\text{spike}_i = \frac{n_i \times \sigma_i}{\max(n \times \sigma)} + \frac{\mu_i}{\max \mu}$	$bloat_i = \frac{\mu_i}{\max \mu}$	$leak_i = \frac{\sigma_i}{\max \sigma} + \frac{\mu_i}{\max \mu}$
	$\mu_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \Delta memory_{i,j}$ $\sigma_i = \sqrt{\frac{1}{n_i - 1} \sum_{j=1}^{n_i} (\Delta memory_{i,j} - \mu_i)^2}$ Όπου i είναι ο αριθμός της συστάδας, n_i είναι το μέγεθος του $cluster_i$, μ_i είναι ο μέσος όρος του δέλτα μνήμης όλων των time-slice προφίλ, σ_i είναι η τυπική απόκλιση του δέλτα μνήμης του συμπλέγματος i και $spike_i$, $bloat_i$ και $leak_i$είναι οι βαθμολογίες που έχουν εκχωρηθεί στα συμπλέγματα i. Τα n, μ και σ είναι διανύσματα τα οποία περιέχουν το μέγεθος της συστάδας, τον μέσο όρο του δέλτα μνήμης και την τυπική απόκλιση του δέλτα μνήμης όλων των συστάδων. Η τυπική απόκλιση μια συστάδας που περιέχει ένα μόνο time-slice προφίλ αποδίδεται αυθαίρετα τη μέγιστη τυπική απόκλιση (δηλαδή, έτσι ώστε το πρώτο memory spike ή memory leak να έχουν βαθμολογία ίση με 1).		

Αρχικά υπολογίζονται οι βαθμολογίες των memory spikes ή των memory bloat και memory leak για κάθε συστάδα, ανάλογα με το αν το θέμα που έχει παρουσιαστεί όσον αφορά την μνήμη είναι παροδικό ή επίμονο. Οι συστάδες οι οποίες θεωρούνται outliers, είναι αυτές οι οποίες χαρακτηρίζονται από μια συνολική βαθμολογία μεγαλύτερη από το άθροισμα της διπλάσιας τυπικής απόκλισης (standard deviation) και της μέσης τιμής (δηλαδή $spike\ score > \mu_{spike} + 2\ x\ \sigma_{spike}$). Για την απλοποίηση των παραδειγμάτων εκτέλεσης, χρησιμοποιούμε μονή τυπική απόκλιση σε αντίθεση με την διπλή τυπική απόκλιση.

Influence Analysis

Η διαδικασία της ανάλυσης επιρροής (influence analysis) [48] αποτελεί ένα σημαντικό συστατικό της ανάλυσης των δεδομένων και έχει εφαρμοστεί ευρέως σε πολλά στατιστικά μοντέλα για τον εντοπισμό της επιρροής που έχουν οι παρατηρήσεις και την αξιολόγηση των διαταραχών που παρουσιάζουν τα μοντέλα.

Μια παρατήρηση μπορεί να θεωρηθεί ότι ασκεί σημαντική επιρροή, σε περίπτωση που οι προβλεπόμενες βαθμολογίες για άλλες παρατηρήσεις θα διαφέρουν, αν δεν είχε συμπεριληφθεί η εν λόγω παρατήρηση. Αν οι προβλέψεις είναι ίδιες με ή χωρίς την παρατήρηση, τότε η παρατήρηση δεν έχει καμία επίδραση στο μοντέλο της παλινδρόμησης. Αν οι προβλέψεις διαφέρουν κατά πολύ όταν η παρατήρηση δεν περιλαμβάνεται στην ανάλυση, τότε η παρατήρηση ασκεί επίδραση και ονομάζεται ισχυρή (influential). Η επιρροή μιας παρατήρησης εξαρτάται από δυο κύριους παράγοντες: πρώτον, το πόσο η τιμή που έχει η παρατήρηση σε σχέση με την προβλεπόμενη μεταβλητή διαφέρει από τη μέση τιμή της προβλεπόμενης μεταβλητής και δεύτερον από την διαφορά μεταξύ της προβλεπόμενης βαθμολογίας της παρατήρησης και της πραγματικής βαθμολογίας της [49].



Εικόνα 21: Το επίπεδο επιρροής που ασκεί μια απομακρυσμένη συστάδα [50]

Στην δίκη μας περίπτωση, πραγματοποιείται μια ανάλυση επιρροής στα απομακρυσμένα συμπλέγματα για να καθοριστεί ποια γεγονότα εκτέλεσης ξεχωρίζουν (differentiate) τα time-slice προφίλ σε απομακρυσμένες συστάδες από τον μέσο όρο των «κανονικών» προφίλ. Είναι πολύ πιθανό, αυτά τα γεγονότα εκτέλεσης να είναι υπεύθυνα για τα memory-related θέματα τα οποία προκαλούνται στο σύστημα ή την εφαρμογή.

Αρχικά, υπολογίζεται το κέντρο του συμπλέγματος των απομακρυσμένων συστάδων καθώς επίσης και το καθολικό κέντρο όλων των συστάδων. Το κέντρο των απομακρυσμένων συστάδων υπολογίζεται με χρήση του μέσου όρου της καταμέτρησης του κάθε event σε όλα τα time-slice προφίλ μιας συστάδας. Ομοίως, το καθολικό κέντρο υπολογίζεται με την χρήση του μέσου όρου της καταμέτρησης του κάθε event σε όλα τα time-slice προφίλ όλων των συστάδων. Τα κέντρα αυτά αντιπροσωπεύουν τη θέση όλων των συστάδων, σε ένα χώρο n-διαστάσεων, (όπου η είναι ο αριθμός των μοναδικών γεγονότων εκτέλεσης) καθώς επίσης και τον μέσο όρο των «φυσιολογικών» time-slice προφίλ.

Στην συνέχεια, υπολογίζεται η Pearson απόσταση (εξίσωση 2.1) μεταξύ του κέντρου των απομακρυσμένων συμπλεγμάτων και του καθολικού κέντρου. Αυτή η απόσταση ποσοτικοποιεί την διαφορά μεταξύ των time-slice προφίλ που βρίσκονται σε απομακρυσμένες συστάδες και τον μέσο όρο αυτών που βρίσκονται σε όλες τις συστάδες. Το επόμενο βήμα της διαδικασίας είναι ο υπολογισμός της μεταβολής της

βασικής απόστασης ανάμεσα στο κέντρο των απομακρυσμένων συμπλεγμάτων και του καθολικό κέντρου με και χωρίς χρήση κάθε γεγονότος εκτέλεσης.

Αυτό ποσοτικοποιεί την επιρροή που έχει το κάθε γεγονός εκτέλεσης. Όταν μια περίπτωση υπερβολικά μεγάλης επιρροής γεγονότος εκτέλεσης αφαιρείται, τα απομακρυσμένα συμπλέγματα αρχίζουν να μοιάζουν όλο και περισσότερο με το καθολικό μέσο όρο των time-slice προφίλ (δηλαδή, πιο κοντά ως προς το καθολικό κέντρο).

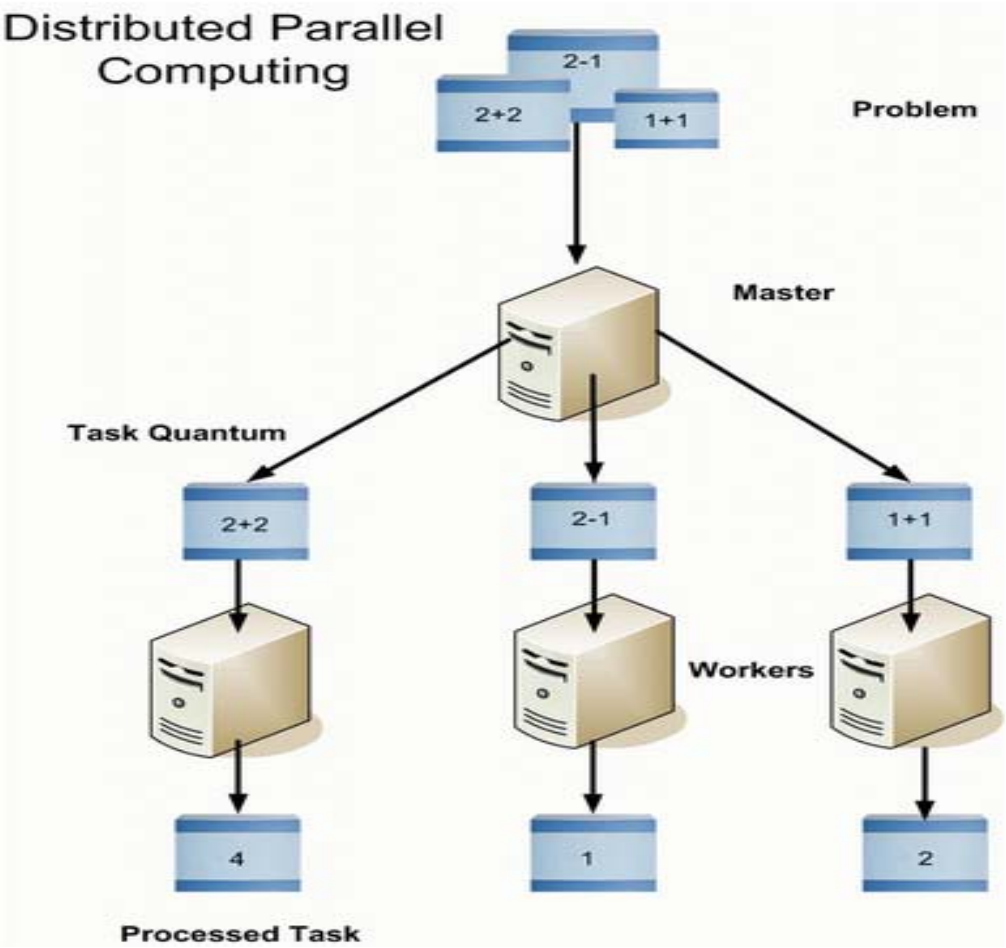
Συνεπώς, μια υπερβολικά μεγάλη επιρροή ενός γεγονότος εκτέλεσης προσδιορίζεται ως μια περίπτωση εκτέλεσης η οποία αν απομακρυνθεί από τον υπολογισμό της απόστασης, τότε θα μειωθεί η απόσταση μεταξύ του κέντρου των απομακρυσμένων συμπλεγμάτων και του καθολικού κέντρου περισσότερο από το διπλάσιο του μέσου όρου της τυπικής απόκλισης πάνω από τον μέσο όρο της αλλαγής της απόστασης. Χάρις τα αποτελέσματα αυτά, οι αναλυτές απόδοσης θα έχουν ένα χειροπιαστό σημείο εκκίνησης για την έρευνα τους και την αναζήτηση ενός memory-related προβλήματος.

3. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

3.1 Distributed Computing

Η υπολογιστική κατανομή (distributed computing) είναι μια έννοια της επιστήμης των υπολογιστών η οποία αναφέρεται σε πολλαπλά συστήματα πληροφορικής που εργάζονται ταυτόχρονα για την αντιμετώπιση ενός ενιαίου προβλήματος. Είναι ένα μοντέλο στο οποίο οι συνιστώσες ενός συστήματος λογισμικού διαμοιράζονται μεταξύ πολλών υπολογιστών για τη βελτίωση της αποτελεσματικότητας της απόδοσης και του χρόνου εκτέλεσης. Στα κατανεμημένα υπολογιστικά συστήματα, ένα ενιαίο πρόβλημα χωρίζεται σε πολλά τμήματα, και κάθε τμήμα επιλύεται σε διαφορετικούς υπολογιστές. Όσο οι υπολογιστές είναι συνδεμένοι στο δίκτυο και έχουν την δυνατότητα να επικοινωνούν μεταξύ τους για να επιλύσουν το πρόβλημα. Αν η διαδικασία πραγματοποιηθεί σωστά, τότε οι υπολογιστές εκτελούν το σύνολο των διεργασιών σαν μια ενιαία οντότητα.

Ο απώτερος στόχος των κατανεμημένων συστημάτων πληροφορικής είναι η μεγιστοποίηση της απόδοσης έτσι ώστε η σύνδεση των χρηστών και των πόρων πληροφορικής να γίνει οικονομικά αποτελεσματική, διαφανή και με αξιόπιστο τρόπο. Επίσης, μ’ αυτόν τον τρόπο, εξασφαλίζεται η ανοχή σε σφάλματα και επιτρέπεται η προσβασιμότητα των πόρων σε περίπτωση που ένα από τα κύρια δομικά στοιχεία αποτύχει.



Εικόνα 22: Σύνδεση ενός κατανεμημένου συστήματος υπολογιστών [51]

Η ιδέα της διανομής των πόρων μέσα σε ένα δίκτυο υπολογιστών δεν είναι καινούρια. Η πρώτη διαμοίραση των πόρων ξεκίνησε με τη χρήση των τερματικών δεδομένων εισόδου (data entry terminals) σε μεγάλα συστήματα υπολογιστών (mainfram), δηλαδή υπολογιστές που χρησιμοποιούνται κυρίως από μεγάλους οργανισμούς για σημαντικές εφαρμογές, μαζική επεξεργασία δεδομένων κ.α., στη συνέχεια εφαρμόστηκε σε μικρούς υπολογιστές (minicomputers) και στις μέρες μας μπορεί να χρησιμοποιηθεί σε προσωπικούς υπολογιστές και σε αρχιτεκτονικές της μορφής πελάτη-διακομιστή (client-server) με περισσότερα επίπεδα (βαθμίδες).

Μια κατανεμημένη αρχιτεκτονική υπολογιστών [52] αποτελείται από μια σειρά από υπολογιστές-πελάτες (client machines) με πολύ ελαφρούς πράκτορες (agents) λογισμικού που έχουν εγκατασταθεί σε έναν ή περισσότερους ειδικούς διανεμόμενους διακομιστείς διαχείρισης υπολογιστών. Οι πράκτορες που λειτουργούν με τους υπολογιστές-πελάτες συνήθως ανιχνεύουν πότε ακριβώς το μηχάνημα είναι σε αδράνεια και αποστέλλουν μια ειδοποίηση στο εξυπηρετητή διαχείρισης ενημερώνοντας τον ότι η μηχανή δεν είναι σε χρήση και δεν είναι διαθέσιμη για δουλειά επεξεργασίας.

Οι πράκτορες, στην συνέχεια αιτούνται ένα πακέτο εφαρμογής (application package). Όταν ο υπολογιστής-πελάτης λαμβάνει αυτό το πακέτο από το εξυπηρετητή διαχείρισης της διαδικασίας, τρέχει το λογισμικό εφαρμογής σε περίπτωση που έχει ελεύθερους CPU κύκλους και στέλνει το αποτέλεσμα πίσω στον εξυπηρετητή διαχείρισης. Όταν ο

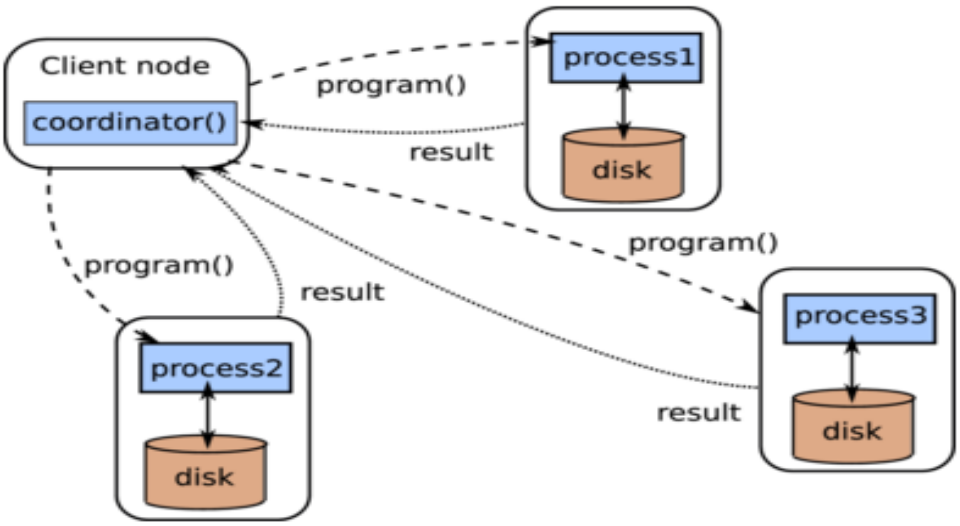
χρήστης επιστρέψει και απαιτήσει πάλι τους πόρους, ο εξυπηρετητής διαχείρισης επιστρέφει τους πόρους που χρησιμοποιούσε για να εκτελέσει διάφορες εργασίες κατά την απουσία του χρήστη.

Ένα απλό παράδειγμα παράλληλης επεξεργασίας ενός κατανεμημένου υπολογιστικού συστήματος αποτελεί μια εφαρμογή επεξεργασίας κειμένου [53] η οποία μπορεί να αποτελείται από έναν διορθωτή κειμένου (editor) σε έναν υπολογιστή, έναν ορθογράφο (spell-checker) σε ένα δεύτερο υπολογιστή, και ένα λεξικό με συνώνυμα και αντώνυμα (thesaurus) σε έναν τρίτο υπολογιστή. Σε ορισμένα κατανεμημένα υπολογιστικά συστήματα, κάθε ένα από τα τρία υπολογιστικά συστήματα θα μπορούσε ακόμη και να τρέχει ένα διαφορετικό λειτουργικό σύστημα.

3.1.1 Apache Hadoop

Το Apache Hadoop αποτελεί μια δομή ανοιχτού κώδικα (open source framework) η οποία χρησιμοποιείται για την αποθήκευση των δεδομένων και των τρεχουσών εφαρμογών σε συστάδες [54]. Το Hadoop είναι διαθέσιμο για παραπάνω από 10 χρόνια και αποτελεί μια αξιόπιστη λύση για την επεξεργασία μεγάλων συνόλων δεδομένων. Παρέχει πολλές δυνατότητες αποθήκευσης για κάθε είδος δεδομένων, τεράστια επεξεργαστική ισχύ και την ικανότητα να χειριστεί σχεδόν απεριόριστες ταυτόχρονες εργασίες η διαδικασίες.

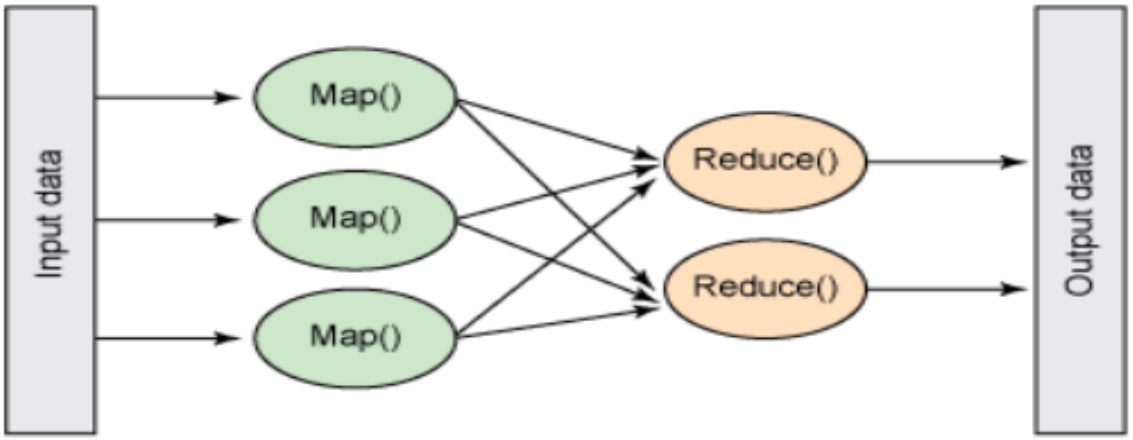
Το Hadoop είναι μια κατανεμημένη υποδομή επεξεργασίας η οποία μπορεί να χρησιμοποιηθεί σε ένα υπολογιστικό σύστημα. Παρόλα αυτά, για την πλήρη εκμετάλλευση όλης της υπολογιστικής του ισχύος, πρέπει να εφαρμοστεί σε εκατοντάδες ή ακόμα και χιλιάδες υπολογιστικά συστήματα το καθένα με αρκετούς πυρήνες επεξεργασίας (processor cores). Παρόλο που η MapReduce λειτουργία δε μπορεί να λύσει όλα τα προβλήματα [55], το Hadoop αποτελεί μια καλή λύση για έρευνα, την διεξαγωγή πειραμάτων και την διαχείριση των δεδομένων [56].



Εικόνα 23: Λειτουργία του Hadoop [57]

MapReduce

Η MapReduce λειτουργία είναι η καρδιά του Hadoop. Ο όρος MapReduce στην πραγματικότητα αναφέρεται σε δύο ξεχωριστές και διακριτές εργασίες που εκτελούνται από τα Hadoop προγράμματα. Η πρώτη είναι η map διαδικασία (χαρτογράφησης), η οποία λαμβάνει ένα σύνολο δεδομένων και το μετατρέπει σε ένα άλλο σύνολο δεδομένων, όπου τα επιμέρους στοιχεία αναλύονται σε πλειάδες (ζεύγη κλειδιού / τιμής) και η δεύτερη αποτελεί η reduce διαδικασία (μείωση) η οποία δέχεται ως είσοδο την έξοδο της map λειτουργίας και συνδυάζει αυτές τις πλειάδες δεδομένων σε ένα μικρότερο σύνολο των πλειάδων. Όπως είναι φυσικό, η reduce διαδικασία εκτελείται πάντα μετά το πέρας της map λειτουργίας.



Εικόνα 24: Στάδια της MapReduce διαδικασίας [58]

Το σημαντικότερο πλεονέκτημα της MapReduce λειτουργίας [59] είναι ότι καθιστά εύκολη την επεξεργασία των δεδομένων χρησιμοποιώντας πολλαπλούς κόμβους υπολογιστών. Σύμφωνα με το μοντέλο, τα αρχέτυπα επεξεργασίας δεδομένων ονομάζονται χαρτογράφοι (mappers) και μειωτές (reducers). Η αποσύνθεση μιας εφαρμογής επεξεργασίας δεδομένων σε mappers και reducers μερικές φορές είναι τετριμμένη. Παρόλα αυτά, από τη στιγμή που γράφουμε μια εφαρμογή έχοντας μία MapReduce μορφή, η κλιμάκωση της εφαρμογής για να τρέξει πάνω από τις εκατοντάδες, χιλιάδες, ή ακόμη και δεκάδες χιλιάδες μηχανήματα σε ένα cluster πραγματοποιείται απλώς με μια αλλαγή ρυθμίσεων. Αυτή η απλή επεκτασιμότητα είναι ο κύριος λόγος που έχει προσελκύσει πολλούς προγραμματιστές να χρησιμοποιούν το MapReduce μοντέλο.

Το MapReduce μοντέλο αποτελείται από διάφορα στοιχεία [60], μεταξύ των οποίων:

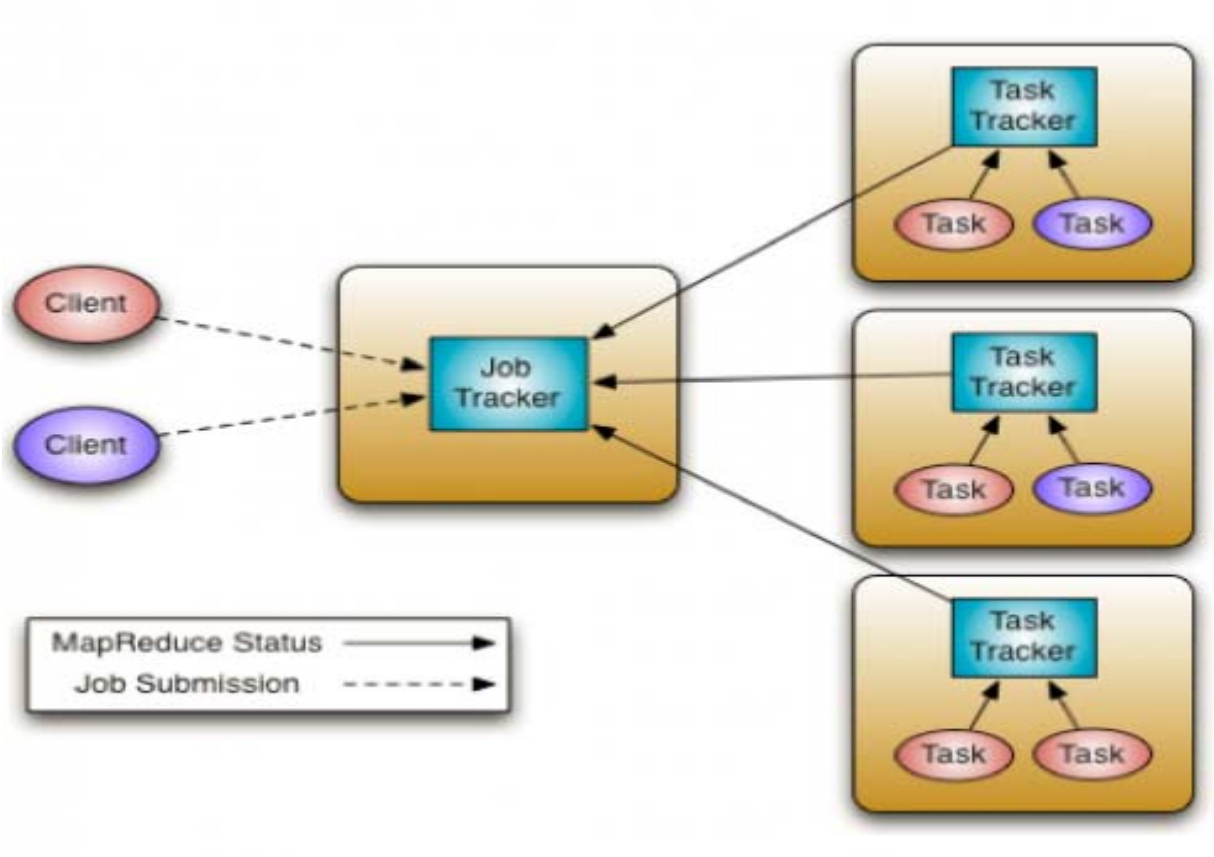
- JobTracker: ο κύριος κόμβος που διαχειρίζεται όλες τις εργασίες και τους πόρους σε μια συστάδα.

- TaskTrackers: οι πράκτορες οι οποίοι έχουν αναπτυχθεί σε κάθε μηχάνημα της συστάδας για να τρέξουν το map και να μειώσει τα καθήκοντα.

- JobHistoryServer: ένα συστατικό που παρακολουθεί τις ολοκληρωμένες εργασίες, και συνήθως αναπτύσσεται ως μια ξεχωριστή λειτουργία ή με χρησιμοποιείται μαζί με τον JobTracker.

Όσον αφορά τις διαδικασίες που ολοκληρώνονται γρήγορα, και όπου τα δεδομένα ταιριάζουν στην κύρια μνήμη μίας μηχανής ή μιας μικρής συστάδας, η χρήση του MapReduce μοντέλου συνήθως δεν είναι αποτελεσματική. Δεδομένου ότι αυτά τα πλαίσια έχουν σχεδιαστεί για την ανάκτηση μετά από απώλεια ολόκληρων κόμβων κατά τη διάρκεια του υπολογισμού, κάποια ενδιάμεσα αποτελέσματα γράφονται σε αποθηκευτικά μέσα. Η ανάκαμψη μιας συντριβής έχει μεγάλο κόστος, και μπορούμε να την εκμεταλλευτούμε μόνο όταν ο υπολογισμός υλοποιείται από πολλούς υπολογιστές και μεγάλο χρόνο λειτουργίας. Ένα έργο που ολοκληρώνει την εκτέλεση του μέσα σε λίγα δευτερόλεπτα μπορεί απλά να ξαναρχίσει σε περίπτωση που προκύψει κάποιο σφάλμα. Η πιθανότητα να καταρρεύσει τουλάχιστον ένα μηχάνημα μεγαλώνει γρήγορα ανάλογα με το μέγεθος της συστάδας. Σε τέτοιου είδους προβλήματα, οι εφαρμογές διατηρούν όλα τα δεδομένα τους στη μνήμη και απλά σε περίπτωση κάποιας αποτυχίας κόμβων, εκτελούν πάλι τους υπολογισμούς. Όταν τα δεδομένα δεν είναι αρκετά μεγάλα, τότε μη κατανεμημένες (non distributed) λύσεις θα είναι συχνά πιο γρήγορες σε σχέση με ένα MapReduce σύστημα.

Η MapReduce λειτουργία είναι μια πολύ καλή λύση για τους υπολογισμούς ενός περάσματος, αλλά δεν είναι πολύ αποτελεσματική για τις περιπτώσεις χρήσης που απαιτούν υπολογισμούς σε πολλά περάσματα και αλγορίθμους. Κάθε βήμα στη ροή εργασιών επεξεργασίας δεδομένων έχει μία map φάση και μια reduce φάση και απαιτείται η μετατροπή οποιαδήποτε περίπτωσης σε μοτίβο MapReduce ώστε να αξιοποιηθεί αυτή η λύση.



Εικόνα 25: Τα κύρια συστατικά της MapReduce λειτουργίας [61]

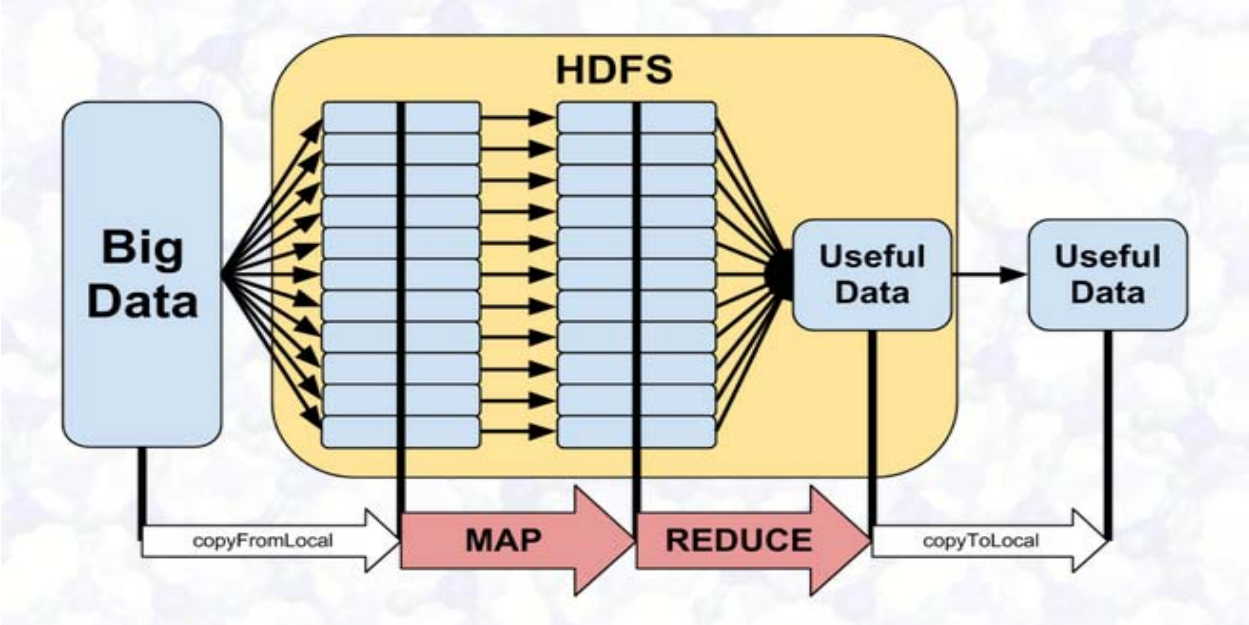
HDFS

Το Hadoop Distributed File System (HDFS) [62] είναι ένα κατανεμημένο σύστημα αρχείων σχεδιασμένο για να λειτουργεί με το υλικό (hardware). Έχει πολλές ομοιότητες με τα υπάρχοντα κατανεμημένα συστήματα αρχείων. Ωστόσο, οι διαφορές από τα άλλα συστήματα είναι σημαντικές. Το HDFS είναι εξαιρετικά ανεκτικό σε σφάλματα και έχει σχεδιαστεί να εφαρμόζεται σε υλικό χαμηλού κόστους. Παρέχει πρόσβαση υψηλής απόδοσης για τα δεδομένα της εφαρμογής και είναι κατάλληλο για εφαρμογές που έχουν μεγάλα σύνολα δεδομένων. Το HDFS ελαφρύνει μερικές POSIX (Portable Operating System Interface for Unix) απαιτήσεις για να επιτρέψει τη ροή πρόσβασης στο αρχείο δεδομένων του συστήματος.

Το HDFS χαρακτηρίζεται από μία master / slave αρχιτεκτονική [54]. Μια HDFS συστάδα αποτελείται από ένα ενιαίο NameNode, ένα κύριο διακομιστή (master server) που διαχειρίζεται το χώρο ονομάτων του συστήματος αρχείων και ρυθμίζει την πρόσβαση σε αρχεία από τους πελάτες. Επιπλέον, υπάρχει ένας αριθμός από DataNodes, συνήθως ένα κόμβο ανά συστάδα, οι οποίοι διαχειρίζονται τον αποθηκευτικό χώρο των κόμβων που συνδέονται και τρέχουν σ' αυτούς. Το HDFS εκθέτει ένα χώρο ονομάτων του συστήματος αρχείων και επιτρέπει στα δεδομένα των χρηστών να αποθηκεύονται σε αρχεία. Εσωτερικά, ένα αρχείο διαχωρίζεται σε ένα ή περισσότερα μπλοκ και τα εν λόγω μπλοκ αποθηκεύονται σε ένα σύνολο από DataNodes. Ο NameNode εκτελεί

λειτουργίες συστήματος αρχείων namespace, όπως το άνοιγμα, το κλείσιμο και τη μετονομασία αρχείων και καταλόγων. Επίσης, καθορίζει τη χαρτογράφηση των μπλοκ στους DataNodes οι οποίοι με την σειρά τους είναι υπεύθυνοι για την εξυπηρέτηση ανάγνωσης (read) και γραφής (write) αιτημάτων από τους πελάτες του συστήματος αρχείων. Οι DataNodes εκτελούν επίσης τη δημιουργία μπλοκ, διαγραφή και αντιγραφή κατόπιν εντολής από τον NameNode.

Τα δεδομένα εξόδου των διεργασιών μεταξύ κάθε βήματος πρέπει να αποθηκεύονται στο κατανεμημένο σύστημα αρχείων του Hadoop πριν ξεκινήσει το επόμενο βήμα. Ως εκ τούτου, αυτή η προσέγγιση τείνει να είναι αργή λόγω αντιγραφής και αποθήκευσης δεδομένων στον δίσκο. Επίσης, οι λύσεις του Hadoop περιλαμβάνουν συνήθως συστάδες που είναι δύσκολο να εγκαθιδρυθούν και να διαχειριστούν. Απαιτείται επίσης η ενσωμάτωση διαφόρων εργαλείων για τις περιπτώσεις χρήσης μεγάλων δεδομένων (όπως το Mahout για μηχανική μάθηση και το Storm για τη συνεχή ροή επεξεργασίας δεδομένων).



Εικόνα 26: Χρήση του HDFS και της MapReduce λειτουργίας [62]

3.1.2 Apache Spark

Το Apache Spark είναι ένα σύστημα υπολογισμού συστάδων (clustering computing system) το οποίο ξεκίνησε ως ένα ερευνητικό έργο στο UC (university of California) Berkeley AMPLab το 2009, και μετατράπηκε σε έργο ανοιχτού κώδικα το 2010. Το 2013, έγινε ένα Apache project με αποτέλεσμα να γίνει πιο δημοφιλές και να κερδίσει την ευρύτερη κοινότητα [63]. Το Spark ανήκει στην ομάδα ανοιχτού κώδικα του Hadoop και έχει δημιουργηθεί με βάση το HDFS.

Πρώτα απ 'όλα, το Spark προσφέρει μια ολοκληρωμένη, ενιαία δομή για τη διαχείριση και επεξεργασία μεγάλων δεδομένων με μια ποικιλία συνόλων δεδομένων (data sets)

διαφορετικής φύσεως (δεδομένα κειμένου, γραφήματα κλπ). Το Spark επιτρέπει στις εφαρμογές, όσον αφορά τις Hadoop συστάδες, να τρέξουν έως και 100 φορές πιο γρήγορα στη μνήμη και 10 φορές πιο γρήγορα στον δίσκο.

Πέρα από τις Map και Reduce λειτουργίες, υποστηρίζει ερωτήματα (queries) SQL, συνεχούς ροής δεδομένων (streaming data), βιβλιοθήκες μηχανικής μάθησης (MLlib) και την επεξεργασία δεδομένων γραφήματος. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν αυτές τις δυνατότητες αυτόνομα (stand-alone) ή να τις συνδυάσουν και να τις τρέξουν σε μια ενιαία περίπτωση χρήσης (single data pipeline).

Παρ' όλες τις ομοιότητες που έχει με το Hadoop, το Spark μπορεί να αποδώσει καλυτέρα σε ειδικές περιπτώσεις και εφαρμογές. Το Spark δεν περιορίζεται μόνο στο στάδιο Map-Reduce και μπορεί να εκτελεστεί μέχρι και 100 φορές πιο γρήγορα από το Hadoop σε συγκεκριμένες εφαρμογές. Παρέχει την δυνατότητα της in-memory υπολογισμού συστάδας η οποία επιτρέπει στις εφαρμογές να φορτώνουν στην μνήμη της συστάδας.

Τα δεδομένα τα οποία φορτώνονται στην κύρια μνήμη μπορούν επανειλημμένα να χρησιμοποιηθούν από βάσεις δεδομένων και χάρις τη χρήση του Spark, επιτυγχάνεται η μείωση του χρόνου εκτέλεσης. Αυτή η ιδιότητα, καθιστά το Spark κατάλληλο για τη χρήση του στον τομέα της μηχανικής μάθησης. Το Spark μπορεί να εκτελεστεί σε τρεις γλώσσες, σε Scala, Python ή Java. Στη συγκεκριμένη εργασία, θα χρησιμοποιήσουμε την αντικειμενοστραφή γλώσσα Java για την υλοποίηση και κατασκευή του αλγορίθμου που προτείνεται από τους Syer et al.

Το Spark αποτελείται από τρία κύρια συστατικά τα οποία παρατίθενται παρακάτω:

Driver program

Το Πρόγραμμα οδήγησης είναι η βασική εφαρμογή που συντονίζει την εκτέλεση σε όλη την συστάδα. Πρόκειται για μια εφαρμογή που γράφτηκε για το συγκεκριμένο πρόβλημα, για παράδειγμα, μια εφαρμογή που μετρά τον αριθμό των φορών που η κάθε μια λέξη εμφανίζονται σε ένα κείμενο (wordcount application). Το Spark αρχικοποιεί πρώτα μια σύνδεση με την συστάδα και εν συνεχεία, δημιουργεί καθήκοντα και αιτήματα τα οποία εκτελούνται από αυτήν.

Worker Nodes – Executor

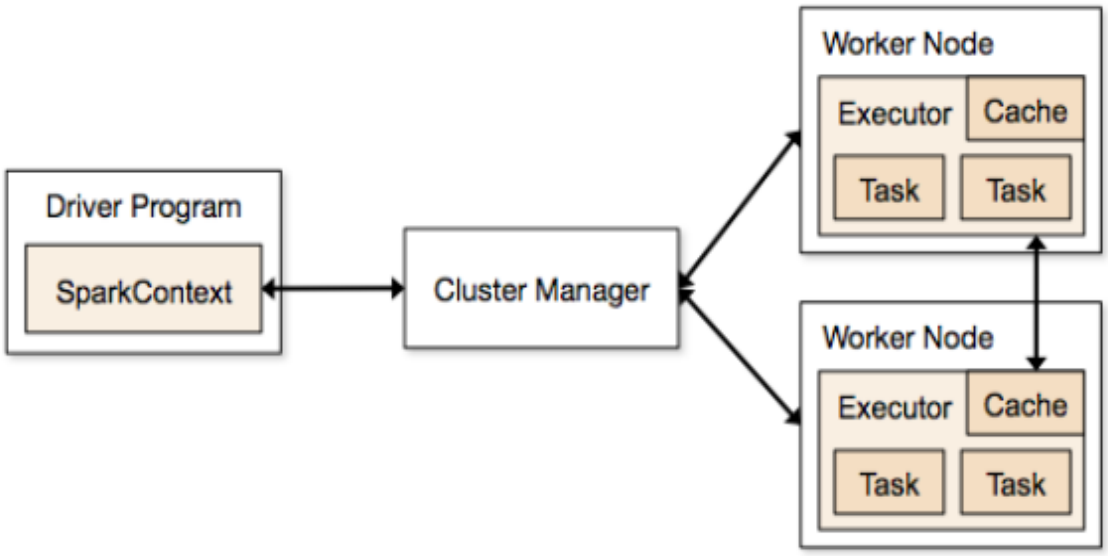
Οι worker nodes αναφέρονται σε φυσικές συσκευές που μπορούν να εκτελέσουν υπολογισμούς. Σε κάθε εργαζόμενο κόμβο, βρίσκεται σε λειτουργιά ένα έργο εκτέλεσης (execution task) το οποίο λαμβάνει εργασίες από το πρόγραμμα οδήγησης και τις εκτελεί. Επιπλέον, ένας εκτελεστής (executor) χειρίζεται τη μνήμη για αυτόν τον εργαζόμενο.

Cluster Manager

Ο διαχειριστής συστάδας (cluster manager) είναι υπεύθυνος για το χειρισμό όλων των διαθέσιμων μέσων για το σύμπλεγμα. Ο επεξεργαστής και η μνήμη είναι παραδείγματα των εν λόγω πόρων. Όταν το πρόγραμμα οδήγησης εκκινεί την εκτέλεσή του, συνδέεται σε έναν διαχειριστή συστάδας που υποστηρίζεται από το Spark και αιτείται τους

διαθέσιμους εκτελεστές στους κόμβους εργασίας. Όταν διατεθούν αυτοί οι πόροι, η αίτηση αποστέλλεται στους εκτελεστές με σκοπό να πραγματοποιηθούν οι τρέχοντες υπολογισμοί.

Σε όλους τους διαχειριστές συστάδας, οι εργασίες ή ενέργειες μέσα σε μια εφαρμογή Spark, έχουν χρονοπρογραμματιστεί από τον Spark scheduler μ' έναν FIFO (first in first out) τρόπο. Εναλλακτικά, ο χρονοπρογραμματισμός μπορεί να ρυθμιστεί με μια δίκαιη πολιτική χρονικού προγραμματισμού όπου το Spark εκχωρεί πόρους για την απασχόληση με round-robin τρόπο. Επιπλέον, η μνήμη που χρησιμοποιείται από μια εφαρμογή μπορεί να ελεγχθεί με τις ρυθμίσεις στο SparkContext. Οι πόροι που χρησιμοποιούνται από μια εφαρμογή Spark μπορεί να ρυθμίζονται δυναμικά με βάση το φόρτο εργασίας. Έτσι, η εφαρμογή μπορεί να απελευθερώσει αχρησιμοποίητους πόρους και να τους αιτηθεί πάλι σε περίπτωση που υπάρχει ξανά ζήτηση.

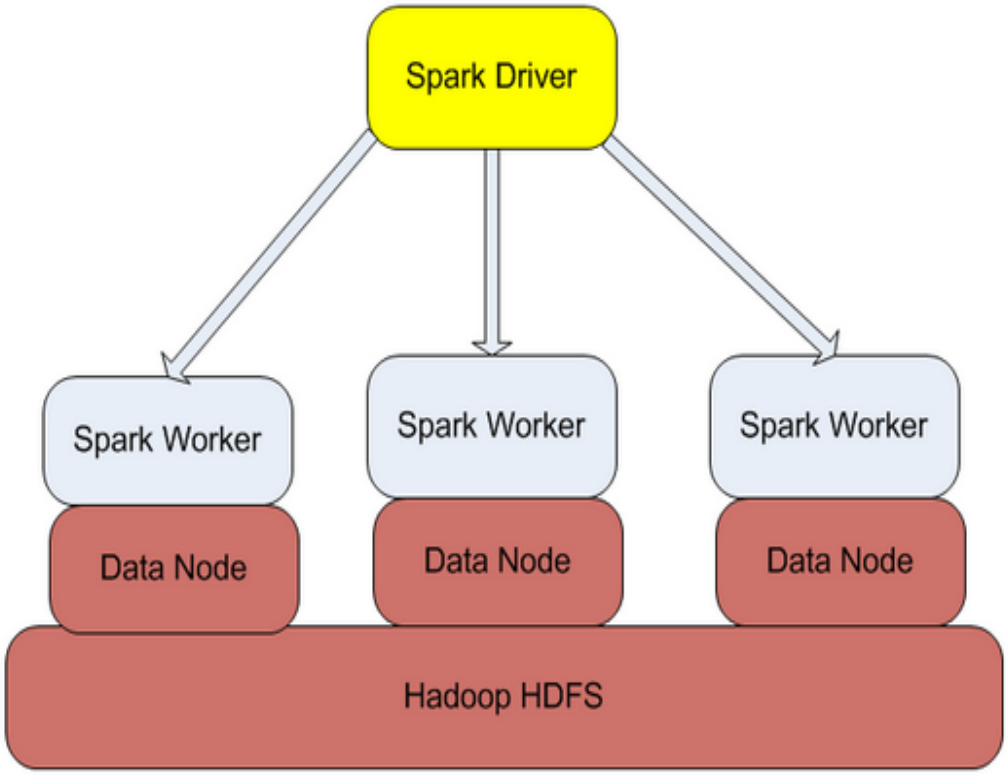


Εικόνα 27: Τα κύρια συστατικά της Spark εφαρμογής [64]

Το Spark δε χρειάζεται να γνωρίζει εκ των πρότερων τον διαχειριστή συστάδας που χρησιμοποιείται. Αυτό επιτρέπει στο Spark να τρέξει πάνω από τις υπάρχουσες συστάδες, παράλληλα με υπάρχουσες εφαρμογές όπως το Hadoop. Επιπλέον, επιτρέπει στο Spark να επωφεληθεί από χαρακτηριστικά του cluster manager. Για παράδειγμα, όταν σχηματιστεί μια συστάδα με τη χρήση του Apache Mesos cluster manager, το Spark μπορεί να μοιραστεί δυναμικά, επεξεργαστές μεταξύ διαφορετικών Spark εφαρμογών. Άλλα παραδείγματα των διαχειριστών συμπλέγματος είναι:

- Spark Standalone: Ένας βασικός διαχειριστής συστάδας ο οποίος γίνεται διαθέσιμος μαζί με την εγκατάσταση του Spark.

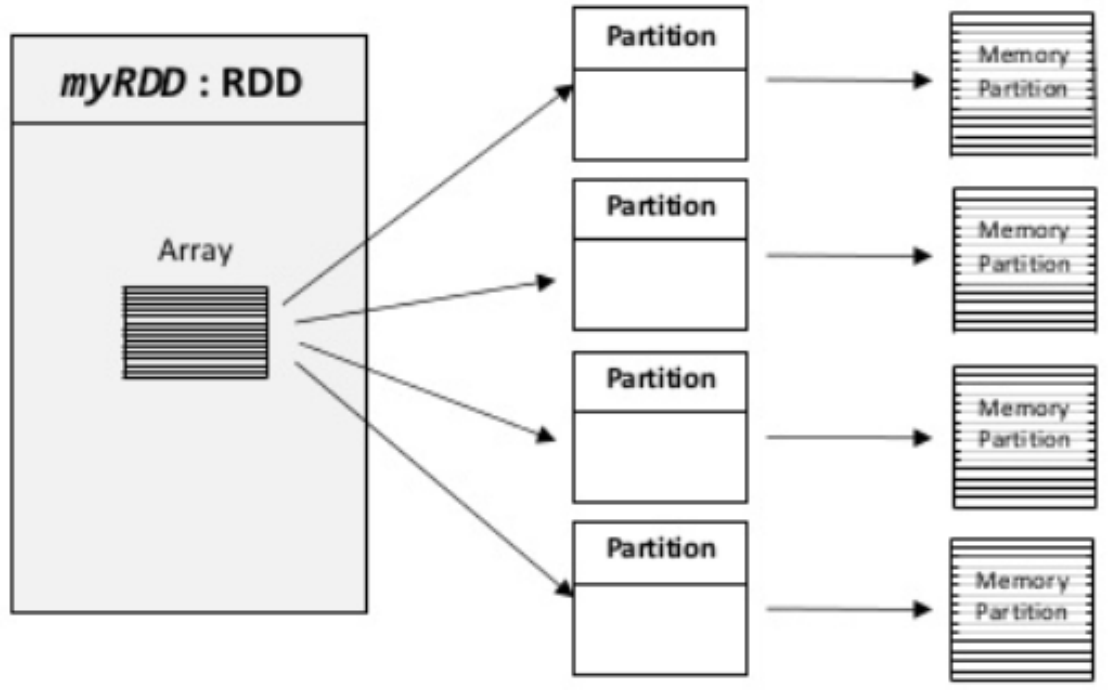
- Hadoop YARN: Ένας διαχειριστής συστάδας ο οποίος αποτελεί τον διαχειριστή πόρων (resource manager) του Hadoop.



Εικόνα 28: Η σύνδεση του Hadoop με το Spark framework [65]

Resilient Distributed Data sets

Σε υψηλό επίπεδο, κάθε Spark εφαρμογή αποτελείται από ένα πρόγραμμα οδήγησης που τρέχει τη κύρια συνάρτηση (main function) του χρήστη και εκτελεί διάφορες παράλληλες λειτουργιές σε μια συστάδα. Η κύρια αφαίρεση (abstraction) την οποία προσφέρει το Spark είναι ένα ανθεκτικό κατανεμημένο σύνολο δεδομένων (resilient distributed data set), το οποίο αποτελεί μία συλλογή κατανεμημένων στοιχείων μεταξύ των κόμβων της συστάδας η οποία μπορεί να λειτουργεί σε παραλληλία. Κάθε σύνολο δεδομένων RDD χωρίζεται σε λογικές κατατμήσεις και μπορεί να υπολογιστεί σε διαφορετικούς κόμβους του συμπλέγματος. Οι χρήστες μπορούν επίσης να αιτηθούν στο Spark να τοποθετήσει ένα RDD στη μνήμη, ώστε να μπορεί να επαναχρησιμοποιηθεί αποτελεσματικά σε παράλληλες λειτουργιές. Επιπρόσθετα, τα RDDs ανακτούνται αυτόματα σε περίπτωση αποτυχίας των κόμβων.



Εικόνα 29: Διαίρεση ενός RDD σε partitions στην μνήμη [66]

Μια δεύτερη αφαίρεση του Spark [67] αποτελούν οι κοινόχρηστες μεταβλητές οι οποίες μπορούν να χρησιμοποιηθούν σε παράλληλες διεργασίες. Από προεπιλογή, όταν το Spark εκτελεί μια συνάρτηση σε παραλληλία ως ένα σύνολο διεργασιών σε διαφορετικούς κόμβους, αποστέλλει ένα αντίγραφο της κάθε μεταβλητής που χρησιμοποιείται σε κάθε διεργασία. Επισήμως, τα RDDs είναι κατανεμημένες συλλογές εγγραφών μόνο γι' ανάγνωση. Μπορεί να δημιουργηθούν μέσω ντετερμινιστικών διαδικασιών πάνω σε δεδομένα σχετικά σταθερής αποθήκευσης ή άλλα RDDs.

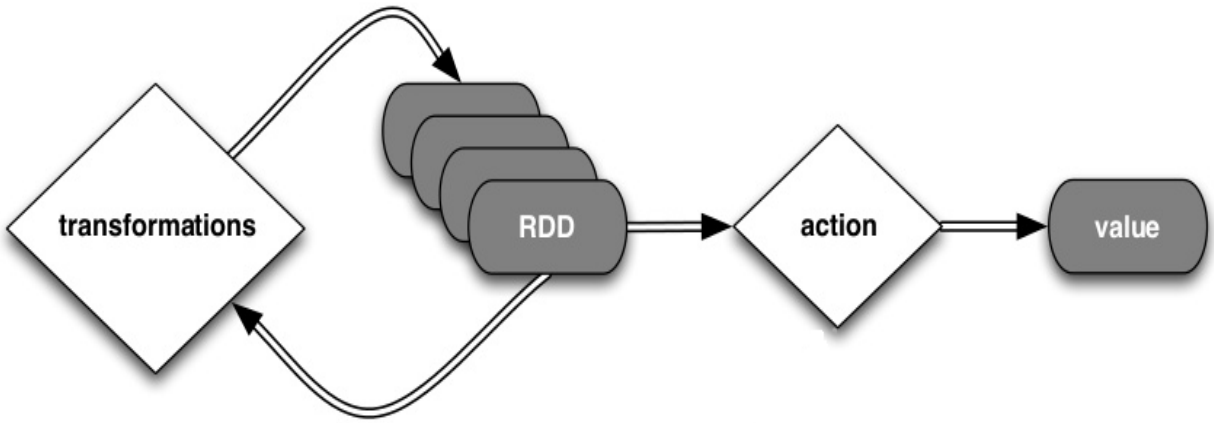
Από προεπιλογή, τα μετασχηματισμένα RDDs υπολογίζονται εκ νέου κάθε φορά που μια ενέργεια εκτελείται σ' αυτά. Το Spark υποστηρίζει τη διατήρηση των υπολογισμένων συνόλων δεδομένων γύρω από την συστάδα για ταχύτερη πρόσβαση, χωρίς να χρειάζεται να τα υπολογίζει ξανά. Υποστηρίζει την αποθήκευση τους είτε στην μνήμη (caching) είτε στον δίσκο. Σε περίπτωση της ανθεκτικότητας της μνήμης (memory persistence), αν η συστάδα δεν έχει αρκετή μνήμη για να κρατήσει όλο το RDD, τότε κάποια partitions (διαμερίσεις) του RDD απορρίπτονται από την cache, και υπολογίζονται εκ νέου από το αρχικό σετ δεδομένων.

Υπάρχουν δύο τρόποι για να δημιουργήσουμε RDDs Ο πρώτος είναι ο παραλληλισμός μιας υπάρχουσας συλλογής (collection) στο πρόγραμμα οδήγησης και ο δεύτερος η δημιουργία μιας αναφοράς σ' ένα σύνολο δεδομένων ενός εξωτερικού συστήματος αποθήκευσης, όπως ένα κοινό σύστημα αρχείων (shared filesystem), HDFS, HBase, ή οποιαδήποτε άλλη πηγή δεδομένων η οποία προσφέρει μία Hadoop μορφή εισόδου (input format).

RDD Operations

Τα RDDs αποτελούν αμετάβλητες συλλογές, πράγμα που σημαίνει ότι δεν υπάρχουν συναρτήσεις οι οποίες μπορούν να αλλάξουν άμεσα, τα περιεχόμενα ενός υπάρχοντος RDD. Αν κάποιος θέλει να αλλάξει μερικές από τις τιμές του (values), τότε ένα νέο RDD θα δημιουργηθεί έχοντας τις νέες αυτές τιμές.

Τα RDDs υποστηρίζουν δύο τύπους δραστηριοτήτων (operations): τους μετασχηματισμούς (transformations), οι οποίοι δημιουργούν ένα νέο σύνολο δεδομένων από ένα υπάρχον και τις δράσεις (actions), οι οποίες επιστρέφουν μια τιμή στο πρόγραμμα οδήγησης μετά την εκτέλεση ενός υπολογισμού που αφορά το σύνολο δεδομένων [68]. Για παράδειγμα, η συνάρτηση map είναι ένας μετασχηματισμός που περνάει κάθε στοιχείο του συνόλου δεδομένων μέσα από μια συνάρτηση και επιστρέφει έναν νέο RDD το οποίο αντιπροσωπεύει τα αποτελέσματα. Από την άλλη πλευρά, η reduce διαδικασία αποτελεί μία δράση η οποία αθροίζει όλα τα στοιχεία του RDD χρησιμοποιώντας κάποια λειτουργία και επιστρέφει το τελικό αποτέλεσμα στο πρόγραμμα οδηγού (αν και για όλη αυτήν την διαδικασία, υπάρχει επίσης μια παράλληλη συνάρτηση, η reduceByKey η οποία επιστρέφει ένα κατανεμημένο σύνολο δεδομένων).



Εικόνα 30: Transformations και actions σε ένα RDD [22]

Όλοι οι μετασχηματισμοί σε Spark είναι αργοί, πράγμα που σημαίνει ότι δεν υπολογίζουν αμέσως τα αποτελέσματά τους. Αντί γι 'αυτό, όλοι οι μετασχηματισμοί σε RDD παρακολουθούνται και οι λειτουργιές τους εκτελούνται μόνο όταν καλείται μία δράση. Συνολικά υπάρχουν δύο βασικά πλεονεκτήματα της συμπεριφοράς αυτής. Αρχικά, το Spark επιτρέπει τη βελτιστοποίηση της εκτέλεσης μίας εφαρμογής. Για παράδειγμα, αν το Spark έχει να εκτελέσει μια δράση μέτρησης (count action) μετά από την χρήση φίλτρου (filter transformation), δεν υπάρχει κανένας λόγος να φιλτράρει τις τιμές πρώτα και στη συνέχεια να εκτελέσει την καταμέτρηση, αλλά μπορεί να τα μετρήσει, κατά την διάρκεια του φιλτραρίσματος τους.

Δεύτερον, η παρακολούθηση των μετασχηματισμών καθιστά το RDD ανθεκτικό σε σφάλματα, που σημαίνει ότι ακόμα και αν ένας υπολογιστής που ανήκει σε ένα cluster, σταματήσει να λειτουργεί, όλο το RDD μπορεί να είναι ακόμα διαθέσιμο, χωρίς να

χρειάζεται να αναπαράγουμε ξανά τα δεδομένα και τους αντίστοιχους υπολογισμούς. Ακόμη και αν υπάρχει μια αποτυχία ενός worker, τα RDDs μπορούν να αναδημιουργηθούν στο υπόλοιπο της συστάδας με την προϋπόθεση ότι τα αρχικά δεδομένα είναι ακόμα διαθέσιμα.

Directed Acyclic Graphs

Στην Επιστήμη Υπολογιστών, ένας γράφος ονομάζεται κατευθυνόμενος άκυκλος γράφος αν είναι κατευθυνόμενος και δεν περιέχει βρόχους (loops) [69]. Δηλαδή, οι ακμές του δεν είναι αμφίδρομες, αλλά έχουν κατεύθυνση και για κανένα κόμβο δεν υπάρχει μονοπάτι (πέρα από το τετριμμένο) που να ξεκινά από αυτόν και να καταλήγει σ' αυτόν.

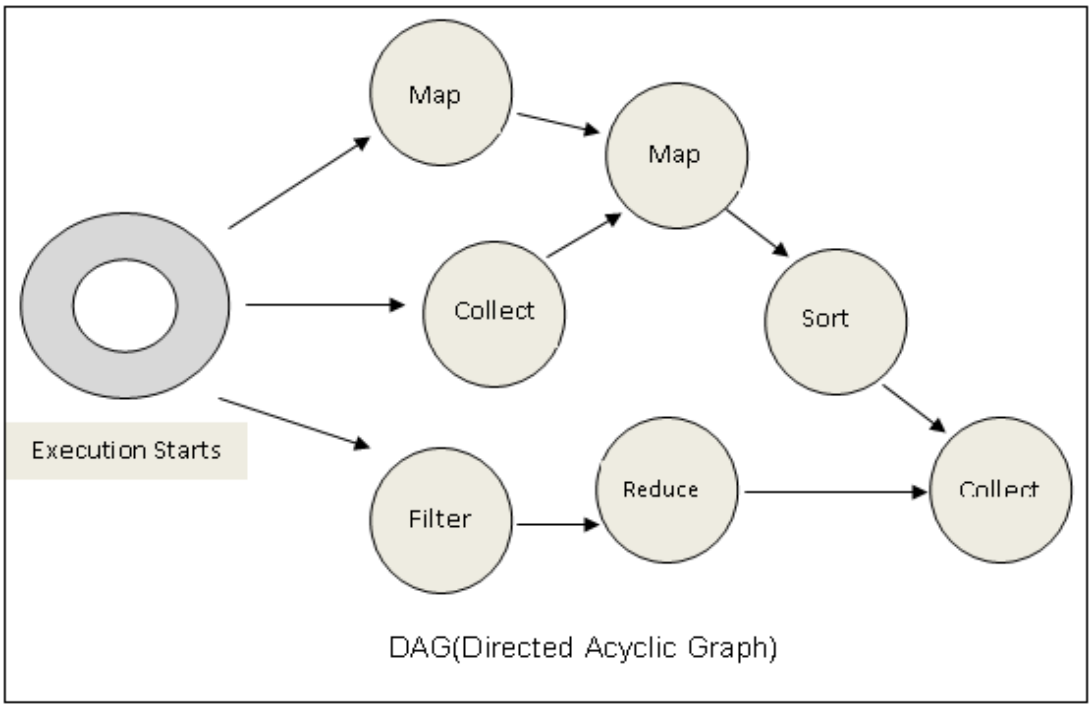
Ένας κατευθυνόμενος άκυκλος γράφος μπορεί να χρησιμεύσει στη μοντελοποίηση διαφόρων δομών. Σε ένα κατανεμημένο σύστημα, ένα συμβατικό πρόγραμμα δεν θα μπορούσε να λειτουργήσει σε περίπτωση που τα δεδομένα έχουν χωριστεί σε κόμβους. Ο κατευθυνόμενος άκυκλος γράφος αποτελεί έναν τρόπο προγραμματισμού για κατανεμημένα συστήματα. Μπορούμε να τον σκεφτούμε ως μία εναλλακτική λύση για την MapReduce λειτουργία. Ενώ η MapReduce έχει μόλις δύο βήματα (χαρτογράφηση και μείωση), ο DAG γράφος μπορεί να έχει πολλαπλά επίπεδα τα οποία μπορούν να σχηματίσουν μια δομή δέντρου. Στην περίπτωση εκτέλεσης ενός SQL ερωτήματος, ο DAG είναι πιο ευέλικτος με περισσότερες λειτουργίες, όπως χαρτογράφηση (map), φιλτράρισμα (filter), ένωση (join) κ.λπ. Επίσης, η εκτέλεση του DAG είναι πιο γρήγορη, καθώς τα ενδιάμεσα αποτελέσματα δεν γράφονται στο δίσκο.

Ο κατευθυνόμενος άκυκλος γράφος αναφέρεται σε ένα μοντέλο για τον προγραμματισμό μιας διεργασίας (job scheduling) στην οποία οι θέσεις της διεργασίας εκπροσωπούνται ως κορυφές σε ένα γράφημα. Στο γράφημα αυτό, η σειρά εκτέλεσης καθορίζεται από την κατεύθυνση που έχουν οι ακμές.

Σε ένα σύστημα το οποίο χρησιμοποιεί έναν DAG γράφο για τον χρονοπρογραμματισμό των διεργασιών, ανεξάρτητοι κόμβοι (υπολογιστικά βήματα) μπορούν να τρέξουν παράλληλα και όχι διαδοχικά στο γράφημα. Η προσέγγιση αυτή διευκολύνει τους προγραμματιστές καθώς καθιστά ευκολότερη την κατασκευή πιο πολύπλοκων υπολογισμών πολλαπλών σταδίων, και αποφεύγει το προγραμματιστικό κόστος (schedule scheduling) που επιβάλλεται γενικά από την παραδοσιακή MapReduce διαδικασία.

Φυσικά, η απλή μετάβαση σε έναν DAG γράφο για τον χρονοπρογραμματισμό των διεργασιών δεν μετριάζει τις μεγάλες καθυστερήσεις που συνδέονται με ένα μόνο βήμα των Hadoop MapReduce διεργασιών. Αυτός είναι ο λόγος που ακόμη και ροές εργασίας κατασκευασμένες ως DAGs που συνδέουν τις MapReduce λειτουργίες του Hadoop εξακολουθούν να υποφέρουν στο τομέα της αδράνειας (latency). Κάθε ροή εργασίας εξακολουθεί να έχει μεγάλο χρονικό κόστος εκκίνησης και μεγάλες καθυστερήσεις όσον αφορά μεμονωμένες εργασίες. Έτσι, προκειμένου να επιτευχθεί χαμηλή συνολική καθυστέρηση, συστήματα όπως το Spark, έχουν επίσης προσθέσει άλλες βελτιστοποιήσεις που αφορούν κυρίως την αντιγραφή των δεδομένων στη μνήμη και αποφεύγουν όσο περισσότερο γίνεται την χρήση του δίσκου I / O.

Εκτός από τη βελτίωση της λανθάνουσας κατάστασης (latency), τα DAG συστήματα παρουσιάζουν και άλλα πλεονεκτήματα. Για παράδειγμα, είναι πιο απλό να εφαρμόσουμε μια ανεκτική προσέγγιση βλάβης χρησιμοποιώντας ένα κατευθυνόμενο άκυκλο γράφο. Σε περίπτωση αποτυχίας μιας διεργασίας, μπορούμε εύκολα να επιστρέψουμε πίσω, μέσα από το γράφημα και να εκτελεστούν εκ νέου οποιεσδήποτε διεργασίες απέτυχαν ακόμη και σε ενδιάμεσα στάδια του υπολογισμού. Η επιβολή της σειράς του γραφήματος επιτρέπει πάντα να κατευθυνόμαστε μέσα στον γράφο από οποιονδήποτε κόμβο μέχρι να καταλήξουμε στον τελευταίο κόμβο.



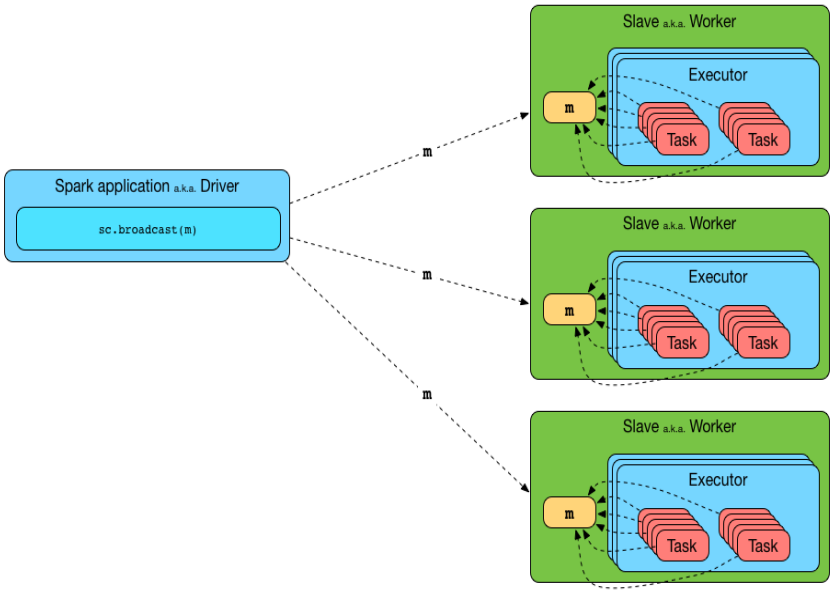
Εικόνα 31: Ο κατευθυνόμενος άκυκλος γράφος στο Spark [70]

Shared Variables

Το Spark παρέχει αυτόματα ένα αντίγραφο της κάθε μεταβλητής του προγράμματος οδήγησης που χρησιμοποιείται μέσα σε ένα μετασχηματισμό ή μια δράση στη συστάδα [64]. Δεδομένου ότι, σε περίπτωση μεγάλων πινάκων, αυτό μπορεί να μην είναι πολύ αποτελεσματικό. Το Spark υποστηρίζει, επίσης, χειροκίνητες μεταβλητές ευρυεκπομπής (manually broadcasting). Αυτές οι μεταβλητές αποτελούν αντίγραφα των αρχικών και σε περίπτωση που οι τιμές τους ενημερωθούν στην συστάδα, το πρόγραμμα οδήγησης δεν θα ανακτήσει τις νέες τιμές τους.

Επιπλέον, υποστηρίζεται ένα ιδιαίτερο είδος κοινών μεταβλητών οι οποίοι ονομάζονται συσσωρευτές (accumulators). Από τη πλευρά των workers, οι συσσωρευτές χρησιμοποιούνται μόνο για εγγραφή στους οποίους οι εκτελεστές μόνο μπορούν να προσθέσουν τιμές.

Από την άλλη πλευρά, το πρόγραμμα οδήγησης μπορεί να διαβάσει μόνο αυτές τις τιμές. Αυτό καθιστά τους συσσωρευτές ιδανικούς για εύρεση, για παράδειγμα, καθολικά αθροίσματα (global sum) σε ένα RDD. Αρχικά, θα υπολογιστεί το τοπικό άθροισμα για κάθε worker, και κατόπιν αιτήματος του οδηγού, θα υπολογιστεί το συνολικό άθροισμα (δηλαδή αυτό του συνόλου τον workers).



Εικόνα 32: Η σύνδεση των κύριων συστατικών μιας Spark εφαρμογής [71]

MLlib

Η MLlib (machine learning library) [72] αποτελεί μία συλλογή από αλγόριθμους μηχανικής μάθησης που τρέχουν στην κορυφή του Spark. Περιέχει αλγόριθμους για δυαδική ταξινόμηση, ομαδοποίηση, γραμμική παλινδρόμηση, συνεργατικό φιλτράρισμα κ.α.. Επιπλέον η MLlib διαθέτει γνωστούς αλγορίθμους συσταδοποίησης όπως την k-means και Bisecting k-means.

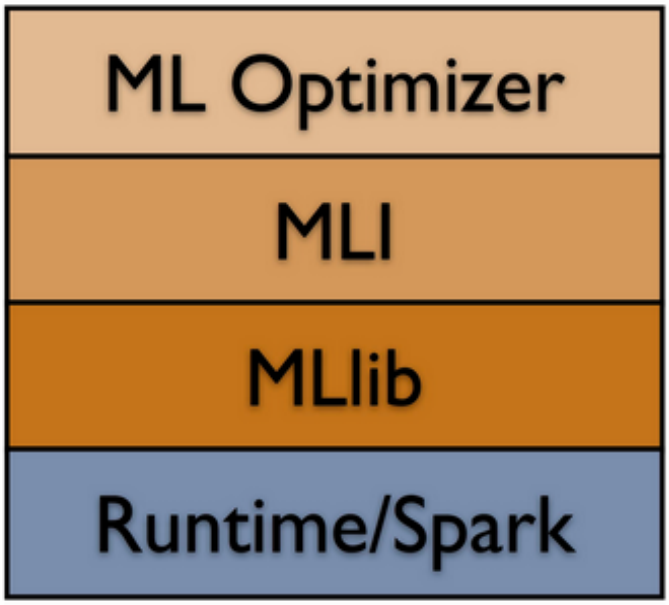
Η MLlib πέρα από τους αλγορίθμους που αναφέρθηκαν προηγουμένως βρίσκει εφαρμογές στα παρακάτω σημαντικά χαρακτηριστικά:

- Featurization: εξαγωγή χαρακτηριστικών, μετασχηματισμός και τεχνικές μείωσης διαστάσεων.

- Αγωγοί (pipelines): Εργαλεία για την κατασκευή, την αξιολόγηση και την ρύθμιση αγωγών μηχανικής μάθησης.

- Ανθεκτικότητα (persistence): αλγόριθμοι εξοικονόμησης και φορτίου, μοντέλα, και αγωγοί.

- Ωφελιμότητα (utilities): γραμμική άλγεβρα, στατιστική, διαχείριση δεδομένων κ.α.



Εικόνα 33: Η βιβλιοθήκη MLlib του Spark framework [72]

3.2 Java 8

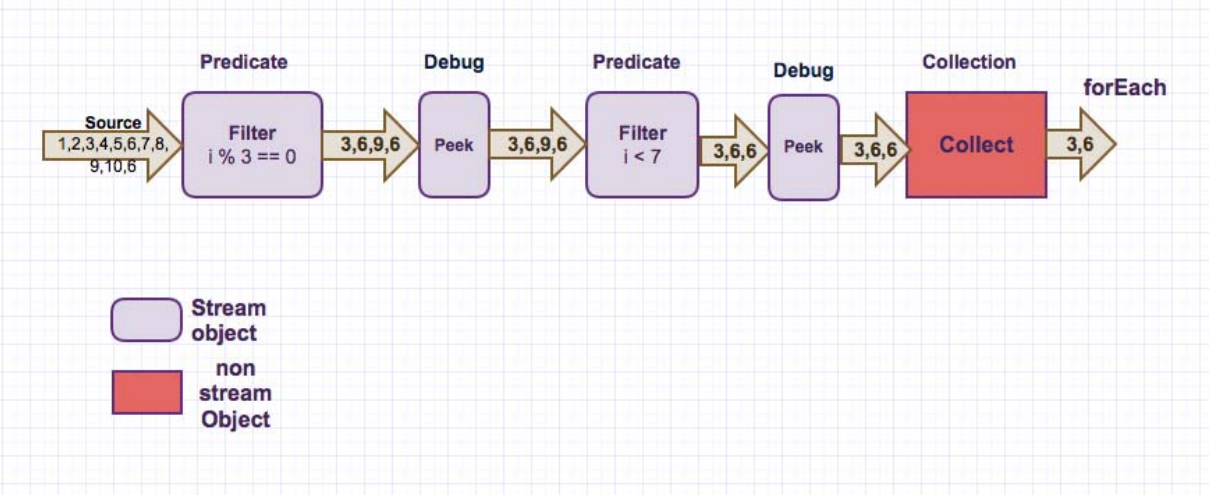
Η επίσημη κυκλοφορία της Java 8 προσφέρει μια ποικιλία νέων αξιόλογων χαρακτηριστικών [73], το σημαντικότερο από τα οποία είναι αναμφισβήτητα η API (application program interface), ροές (streams) και οι λάμδα εκφράσεις (lambda expressions). Τα χαρακτηριστικά αυτά όχι μόνο βελτιώνουν κάποιες υπάρχουσες λειτουργίες αλλά επιπλέον, προσθέτουν κάποιες νέες στην κατασκευή και υλοποίηση εφαρμογών. Επιπλέον, η Java γλώσσα προγραμματισμού από αντικειμενοστραφής αποκτά νέα στοιχεία συναρτησιακού προγραμματισμού.

Αρκετές από τις ευκολίες που μας δίνει η προσθήκη των νέων μηχανισμών της Java 1.8 έκδοσης αφορούν στη διαχείριση συλλογών. Πιο συγκεκριμένα έχουμε νέους τρόπους να διατρέξουμε τα στοιχεία μιας συλλογής, να μετατρέψουμε τα στοιχεία της καθώς επίσης να επιλέξουμε στοιχεία της και να τα ενώσουμε. Το Spark αποτελεί ένα framework στο οποίο μπορούμε να χρησιμοποιήσουμε την γλώσσα Java για την κατασκευή εφαρμογών χρησιμοποιώντας συχνά τις ροές και τις λάμδα εκφράσεις.

3.2.1 Streams

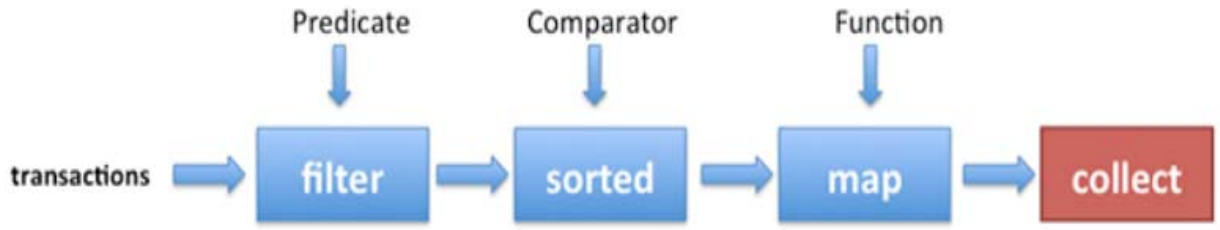
Οι ροές έχουν σαν κύριο ρόλο να επιτρέπουν λειτουργικές διεργασίες (functional style) σε μια ροη από στοιχεία. Μία ροή είναι μια αφηρημένη έννοια και όχι μια δομή δεδομένων. Δεν είναι μια συλλογή όπου μπορούμε να αποθηκεύσουμε τα στοιχεία. Η πιο σημαντική διαφορά ανάμεσα σε μια ροή και μια δομή είναι ότι μία ροή δεν κατέχει τα δεδομένα. Για παράδειγμα, δεν μπορούμε να δείξουμε σε μια τοποθεσία μιας ροής όπου υπάρχει ένα συγκεκριμένο στοιχείο. Μπορούμε να καθορίσουμε μόνο τις λειτουργίες που εφαρμόζονται σε αυτά τα δεδομένα. Μια ροή είναι μια αφαίρεση (abstraction) μιας μη-μεταβλητής συλλογής λειτουργιών που εφαρμόζονται με κάποια σειρά στα δεδομένα.

Οι διεργασίες ροών χωρίζονται σε ενδιάμεσες (intermediate) και τερματικές (terminal), και συνδυάζονται για να σχηματίσουν αγωγούς (pipelines) ροής. Ένας αγωγός ροής αποτελείται από μια πηγή (όπως μια συλλογή), μια σειρά, μια λειτουργία γεννήτριας (generator), ή ένα κανάλι I / O που ακολουθείται από μηδέν ή περισσότερες ενδιάμεσες εργασίες. Οι Ενδιάμεσες λειτουργίες επιστρέφουν ένα νέο ρεύμα. Εκτελώντας μια ενδιάμεση λειτουργία, όπως π.χ. την περίπτωση φιλτραρίσματος, όπου δεν εκτελείται στην πραγματικότητα κανένα φιλτράρισμα, αλλά αντιθέτως δημιουργείται μια νέα ροη η οποία όταν διασχίζεται, περιέχει τα στοιχεία της αρχικής ροής που ταιριάζουν με το συγκεκριμένο κατηγόρημα (predicate). Η διάσχιση του αγωγού δεν αρχίζει μέχρις ότου εκτελεσθεί η τερματική λειτουργία του.



Εικόνα 34: Ένα παράδειγμα ροών [74]

Οι τερματικές λειτουργίες μπορούν να διασχίσουν την ροη για να παράγουν ένα αποτέλεσμα [75]. Μετά το πέρας της τερματικής λειτουργίας, ο αγωγός ροής θεωρείται ότι έχει καταναλωθεί, και δεν μπορεί πλέον να χρησιμοποιηθεί. Σε περίπτωση που θα χρειαστεί να διασχίσουμε πάλι την ίδια πηγή δεδομένων, θα πρέπει να επιστρέψουμε στην πηγή δεδομένων για να πάρουμε μια νέα ροη. Σε όλες σχεδόν τις περιπτώσεις, οι τερματικές λειτουργίες είναι ανυπόμονες (eager), ολοκληρώνοντας το πέρασμα τους από την πηγή των δεδομένων και την επεξεργασία του αγωγού πριν από την επιστροφή.

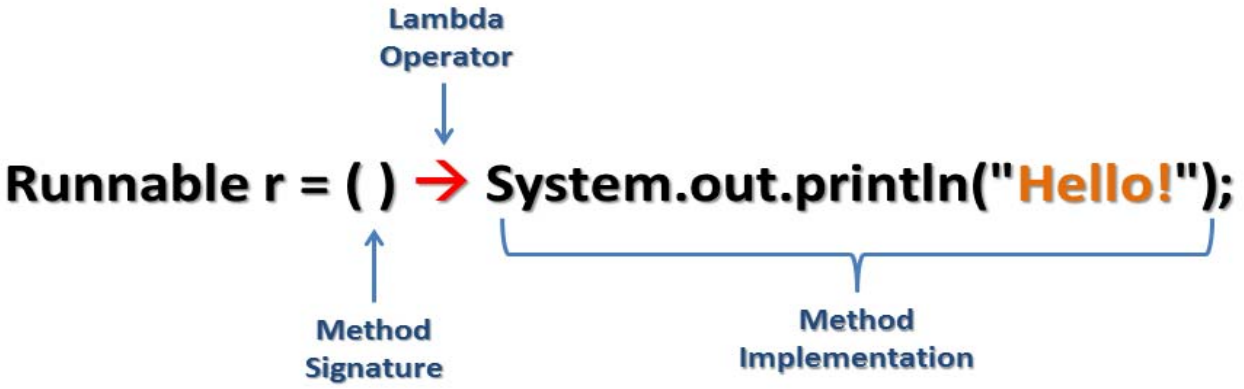


Εικόνα 35: Τα κύρια συστατικά μιας ροής [75]

3.2.2 Lambda Expressions

Για χρόνια η Java είχε λάβει την ετικέτα της μη κατάλληλης γλώσσας προγραμματισμού για τεχνικές συναρτησιακού προγραμματισμού καθώς εφαρμόζεται περισσότερο σαν αντικειμενοστραφής γλώσσα παρά σαν συναρτησιακή. Πράγματι, δεν υπήρχε ένας καθαρός και αποδεκτός τρόπος να αναφερθούμε σε ένα μπλοκ κώδικα μέσω ενός ονόματος. Η έκδοση 8 του JDK (java development kit) περιλαμβάνει τις λάμδα εκφράσεις και ταυτόχρονα μας παρέχει το προνόμιο χρήσης της γλώσσας ως συναρτησιακή.

Αντί της χρήσης του δυσανάγνωστου συντακτικού των ανώνυμων κλάσεων, χάρις της νέας Java έκδοσης, μπορούμε πλέον να χρησιμοποιήσουμε τις λάμδα εκφράσεις. Οι λάμδα εκφράσεις είναι ουσιαστικά συναρτήσεις χωρίς όνομα. Οι εκφράσεις αυτές υλοποιούνται κυρίως διατρέχοντας τα στοιχεία μιας συλλογής εφαρμόζοντας σ' αυτά διάφορες λειτουργίες. Επιπλέον, μία lambda έκφραση μπορεί να αποθηκευθεί σε μια μεταβλητή και να επαναχρησιμοποιηθεί.



Εικόνα 36: Μία λάμδα έκφραση της Java 8 [76]

Το JDK περιέχει μια ποικιλία από διεπαφές (interfaces) [73] οι οποίες δημιουργούνται για να εξυπηρετούν όλες τις πιθανές περιπτώσεις: άκυρες (void) συναρτήσεις, μη

παραμετρικές συναρτήσεις και φυσιολογικές συναρτήσεις οι οποίες περιέχουν τόσο παραμέτρους όσο και τιμές επιστροφής.

3.3 K-Means

Σε προβλήματα όπου τα δεδομένα ανήκουν σε πολλές διαστάσεις, η δημιουργία συστάδων είναι δύσκολη διαδικασία χωρίς τη χρήση της πληροφορικής μιας και η απεικόνιση των δεδομένων δεν παράγει σημαντική γνώση. Σε αυτήν την περίπτωση χρησιμοποιούνται αλγόριθμοι όπως ο k-means.Ο k-means αλγόριθμος είναι ένας από τους πιο ευρέως χρησιμοποιούμενους αλγορίθμους ομαδοποίησης [32]. Ο όρος «k-means» χρησιμοποιήθηκε για πρώτη φορά από τον James MacQueen το 1967, αν και η ιδέα πηγαίνει πίσω στο Hugo Steinhaus το 1957. Ο στάνταρ αλγόριθμος προτάθηκε για πρώτη φορά από τον Stuart Lloyd το 1957 ως μια τεχνική για διαμόρφωση του παλμού-κώδικα και δεν είχε δημοσιευτεί μέχρι το 1982. Το 1965, ο E. W. Forgy δημοσίευσε ουσιαστικά την ίδια μέθοδο και αποτελεί τον λόγο που μερικές φορές αναφέρεται ως Lloyd-Forgy.

Ο k-means αλγόριθμος αποτελεί ένα ευρέως χρησιμοποιημένο αλγόριθμο εκμάθησης χωρίς επίβλεψη (unsupervised learning) για την ανάλυση της διασποράς. Η εκμάθηση χωρίς επίβλεψη είναι η εκπαίδευση ενός αλγορίθμου τεχνητής νοημοσύνης (AI) χρησιμοποιώντας πληροφορίες που δεν είναι ταξινομημένες ούτε έχουν ταμπέλες και επιτρέπει στον αλγόριθμο να ενεργεί χωρίς καθοδήγηση πάνω σ' αυτές τις πληροφορίες.

Είναι μία μη-ντετερμινιστική και επαναληπτική μέθοδος και ο κύριος ρόλος της είναι η ανακάλυψη διαφορετικών κατηγοριών και η κατηγοριοποίηση τους σε συστάδες (clusters). Ο στόχος του αλγορίθμου είναι η ελαχιστοποίηση της διακύμανσης του intra-cluster η οποία δίνεται από τον παρακάτω τύπο:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_i - \mu_i)^2 \quad (3.1)$$

όπου k είναι ο συνολικός αριθμός των συστάδων, S_i το σύνολο των σημείων στην ι-οστή συστάδα και μ_i το κέντρο της ι-οστής συστάδας.

Όταν k-means αλγόριθμος έχει ελαχιστοποιήσει την διακύμανση του intra-cluster, μπορεί να μην έχει εντοπίσει το ολικό ελάχιστο (global minimum) της διακύμανσης. Γι' αυτόν τον λόγο, μια καλή ιδέα είναι να τρέξουμε τον αλγόριθμο αρκετές φορές, και να κρατήσουμε το αποτέλεσμα ομαδοποίησης με την καλύτερη διακύμανση του intra-cluster.

Ο αλγόριθμος χρησιμοποιεί κάποια αρχικά κέντρα για τις συστάδες και αναθέτει τα σημεία-δεδομένα στο κοντινότερο κέντρο από άποψη απόστασης. Στο πρώτο βήμα τα κέντρα αρχικοποιούνται τυχαία και όλα τα σημεία ανατίθενται στο κοντινότερο κέντρο. Στο δεύτερο βήμα υπολογίζονται εκ νέου τα κέντρα από τα σημεία τα οποία έχουν ανατεθεί στο κάθε ένα από αυτά. Έπειτα, τα σημεία ανατίθενται στα καινούρια κέντρα ανάλογα με την απόστασή τους και υπολογίζονται τα καινούρια πλέον κέντρα. Αυτά τα δύο βήματα επαναλαμβάνονται μέχρι να επιτευχθεί σύγκλιση. Στο τέλος, τα τελικά κέντρα των συστάδων ορίζονται από τα κέντρα που υπολογίζονται από τα σημεία που ανατέθηκαν σε αυτά στο τελευταίο βήμα πριν τη σύγκλιση.

Για παράδειγμα, θα εξετάσουμε την εφαρμογή της μεθόδου στα αποτελέσματα αναζήτησης στον ισότοπο Wikipedia. Ο όρος αναζήτησης "Jaguar" στη Wikipedia, θα επιστρέψει όλες τις ιστοσελίδες που περιέχουν τη λέξη Jaguar οι οποίες μπορούν να αναφέρονται σε ένα αυτοκίνητο μάρκας Jaguar, σε μια έκδοση του Mac OS η οποία επίσης ονομάζεται Jaguar, και στο ίδιο το ζώο. Ο k-means αλγόριθμος ομαδοποίησης μπορεί να εφαρμοστεί στην ομάδα των ιστοσελίδων που αναφέρονται στις παρόμοιες έννοιες αυτές. Έτσι, ο αλγόριθμος θα ομαδοποιήσει και θα χωρίσει όλες τις ιστοσελίδες που αναφέρονται στην λέξη Jaguar σε τρεις διαφορετικές συστάδες, την συστάδα των ιστοσελίδων που αναφέρονται στο αμάξι, σ' αυτήν που αναφέρονται στο λογισμικό και τέλος σ' αυτήν που αναφέρονται στο ζώο.

Ο k-means αλγόριθμος παρουσιάζει πολλά πλεονεκτήματα. Είναι γραμμικός ως προς τον αριθμό των δεδομένων (πολυπλοκότητα $O(n)$ για n αντικείμενα) και δουλεύει αρκετά καλά όταν το σχήμα των συστάδων είναι υπερ-σφαιρικό. Κάποια μειονεκτήματα του k-means είναι το ότι κάθε φορά που τρέχει εξαρχής, τα κέντρα είναι τυχαία (διαφορετικά), οπότε είναι πιθανό τα αποτελέσματα να είναι διαφορετικά. Ένα άλλο σημαντικό μειονέκτημα που παρουσιάζει είναι ότι για να εφαρμοστεί, απαιτείται να γνωρίζουμε εκ των προτέρων τον αριθμό των συστάδων που θα εφαρμόσουμε.

Ο k-means αλγόριθμος χρησιμοποιείται από τις περισσότερες μηχανές αναζήτησης όπως το Yahoo και το Google. Σκοπός του είναι να ομαδοποιεί ιστοσελίδες συγκρίνοντας την ομοιότητα τους και να προσδιορίζει το «ποσοστό ενδιαφέροντος» των αποτελεσμάτων αναζήτησης. Αυτό βοηθά τις μηχανές αναζήτησης να μειώσουν τον υπολογιστικό χρόνο όσον αφορά τους χρήστες.

Ο καθορισμός του αριθμού των συστάδων σε ένα σύνολο δεδομένων, δηλαδή ο αριθμός k αποτελεί ένα συχνό πρόβλημα όσον αφορά την ομαδοποίηση δεδομένων και είναι ένα σημαντικό θέμα όσον αφορά την πραγματική επίλυση ανάλυσης των δεδομένων.

Μια ορισμένη κατηγορία αλγορίθμων ομαδοποίησης (ιδίως k-means, k-medoids και ο αλγόριθμος μεγιστοποίησης) χρησιμοποιούν το k το όποιο καθορίζει τον αριθμό των συστάδων για την ανίχνευση. Άλλοι αλγόριθμοι όπως ο DBSCAN και ο αλγόριθμος OPTICS δεν απαιτούν την προδιαγραφή αυτής της παραμέτρου. Από την άλλη, η ιεραρχική ομαδοποίηση (Hierarchical Clustering) αποφεύγει εξαρχής την εφαρμογή αρχικής γνώσης αριθμού συστάδων.

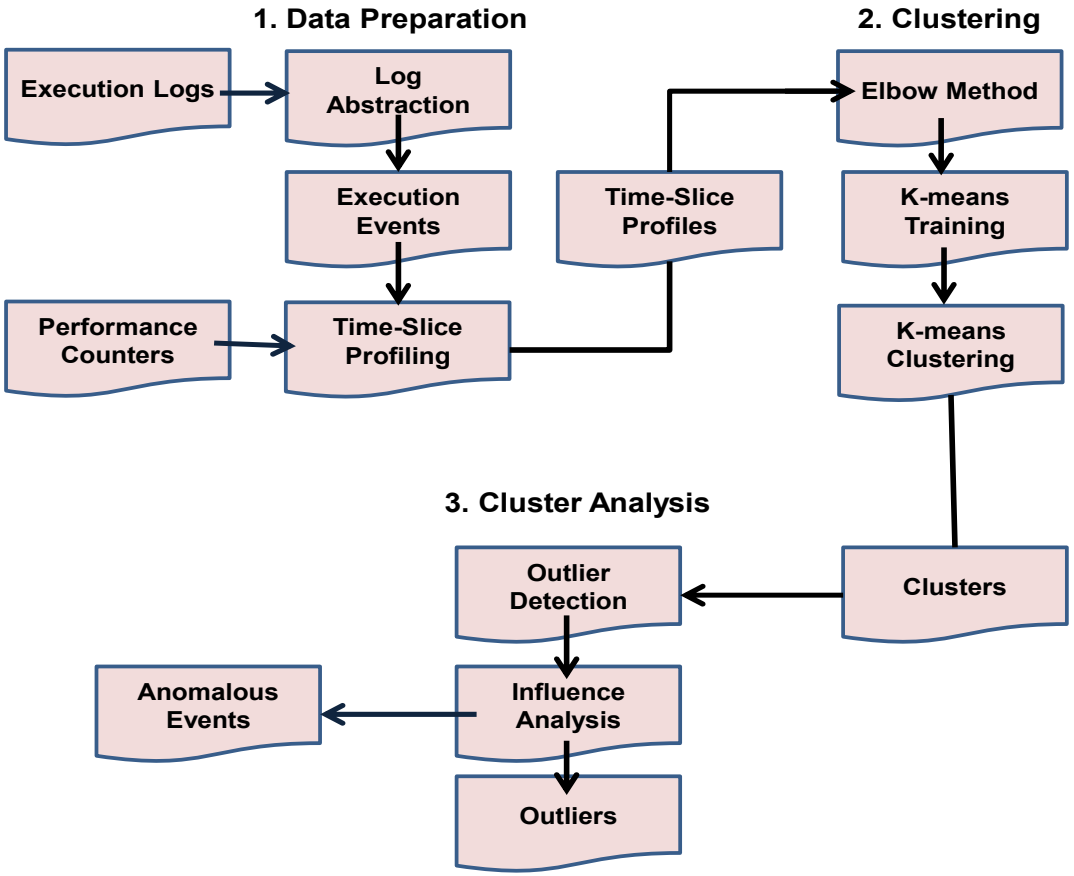
Η σωστή επιλογή του k είναι ένα συχνά διφορούμενο θέμα, με ερμηνείες οι οποίες εξαρτώνται από το σχήμα και το μέγεθος της κατανομής των σημείων σε ένα σύνολο δεδομένων και την επιθυμητή από τον χρηστή ανάλυση ομαδοποίησης. Επιπλέον, η αύξηση του k έχει πάντα σαν αποτέλεσμα την μείωση της ποσότητας του σφάλματος ομαδοποίησης που προκύπτει. Υπάρχει μία σπάνια περίπτωση να παρουσιάζεται μηδενικό σφάλμα αν κάθε σημείο δεδομένων ανατεθεί στο δικό του σύμπλεγμα (δηλαδή όταν ο αριθμός k ισούται με τον αριθμό των σημείων στα αρχικά δεδομένα n).

Μπορούμε να επιλέξουμε τον αριθμό των clusters από μια οπτική επιθεώρηση των σημείων στα αρχικά μας δεδομένα, αλλά σύντομα θα συνειδητοποιήσουμε ότι υπάρχει μεγάλη ασάφεια σε αυτή τη διαδικασία για όλα εκτός από τα απλούστερα σύνολα δεδομένων. Αυτό δεν είναι πάντα κακό, γιατί πραγματοποιούμε εκμάθηση χωρίς επίβλεψη και υπάρχει κάποια εγγενή υποκειμενικότητα στη διαδικασία σήμανσης.

Μ' αυτόν τον τρόπο, η παρακολούθηση ή η απεικόνιση της κατανομής των σημείων σε όλες τις ομάδες, αποτελεί έναν χρήσιμο και εύκολο τρόπο για να αποκτήσουμε μια

αρχική εικόνα για το πώς ο αλγόριθμος θα διαχωρίσει τα αρχικά δεδομένα στις k συστάδες.

Διαισθητικά η βέλτιστη επιλογή του k επιτυγχάνει μια ισορροπία μεταξύ της μέγιστης συμπίεσης των δεδομένων χρησιμοποιώντας ένα ενιαίο σύμπλεγμα και της μέγιστης ακρίβειας με την ανάθεση κάθε σημείου δεδομένων στο δικό του σύμπλεγμα. Αν μια κατάλληλη τιμή του k δεν είναι εμφανής από την ήδη γνώση των ιδιοτήτων του συνόλου δεδομένων, θα πρέπει να επιλεγεί είτε τυχαία, είτε χρησιμοποιώντας κάποια μέθοδο. Υπάρχουν διάφορες κατηγορίες μεθόδων για την πραγματοποίηση αυτής της απόφασης.



Εικόνα 37: Βήματα εκτέλεσης του αλγορίθμου του Mark D. Syer εφαρμόζοντας την k-means μέθοδο.

3.3.1 Κριτήρια επιλογής του k

Elbow Method

Μία μέθοδος για την επικύρωση του αριθμού των συστάδων αποτελεί η μέθοδος αγκώνα (elbow methpd) [77]. Η ιδέα της μεθόδου αυτής είναι η εκτέλεση μίας k-means ομαδοποίησης για το σύνολο δεδομένων για ένα εύρος τιμών του k (ορίζουμε για παράδειγμα το k από 1 έως 10) και για κάθε τιμή του k υπολογίζεται το άθροισμα των τετραγώνων των σφαλμάτων (error sum of squares) καθώς επίσης και η διακύμανση (variance).

Το SSE ορίζεται ως το άθροισμα των τετραγώνων της απόστασης ανάμεσα σε κάθε μέλος του συμπλέγματος και του αντίστοιχου κέντρου του. Το άθροισμα των τετραγώνων των σφαλμάτων υπολογίζεται από τον παρακάτω τύπο:

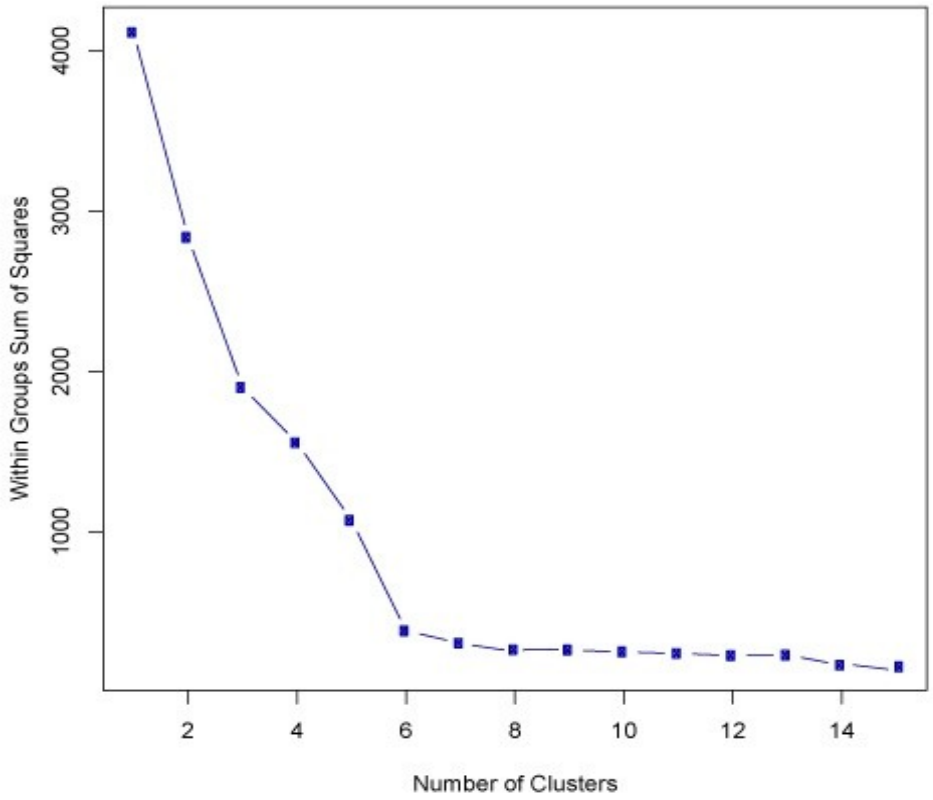
$$SSE = \sum_{i=0}^{k} \sum_{x \in c_i} \mathrm{dist}(x, c_i)^2 \qquad (3.2)$$

Παρακάτω παρατίθεται και ο αλγόριθμος υπολογισμού των SSE [78] για k = 1 έως maxk = 10:

```
var sse = {};
for (var k = 1; k <= maxk; ++k) {
    sse[k] = 0;
    clusters = kmeans(dataset, k);
    clusters.forEach(function(cluster) {
        mean = clusterMean(cluster);
        cluster.forEach(function(datapoint) {
            sse[k] += Math.pow(datapoint - mean,
2);
        });
    });
}
```

Στη συνέχεια, σχεδιάζουμε ένα γράφημα του αθροίσματος των τετραγώνων της απόστασης ή της διακύμανσης για κάθε τιμή του k. Εάν το γράφημα μοιάζει με ένα χέρι, τότε ο "αγκώνας" στο χέρι είναι η τιμή του k και αποτελεί την καλύτερη λύση και πρέπει να επιλεχτεί στην εκτέλεση του αλγορίθμου. Η κύρια ιδέα είναι ότι αναζητάμε ένα μικρό SSE. Το SSE τείνει να μειώνεται προς το 0 όσο αυξάνουμε τον αριθμό των k συστάδων. Το άθροισμα των τετραγώνων των σφαλμάτων έχει μηδενική τιμή όταν το k είναι ίσο με τον αριθμό των σημείων των αρχικών δεδομένων, γιατί τότε κάθε σημείο αποτελεί το δικό του σύμπλεγμα, και δεν υπάρχει σφάλμα μεταξύ αυτού του σημείου και του κέντρου του συμπλέγματος του. Έτσι, ο στόχος μας είναι να επιλέξουμε μια μικρή τιμή του k που εξακολουθεί να έχει χαμηλό SSE το όποιο μειώνεται απότομα και σχηματίζει

μια μορφή «αγκώνα». Ο «αγκώνας» στο γράφημα, εμφανίζεται όταν παρουσιάζονται φθίνουσες αποδόσεις σε συνδυασμό με την αύξηση του αριθμού k.



Εικόνα 38: Ο σχηματισμός μια μορφής «αγκώνα»

Ωστόσο, πρέπει να λάβουμε υπόψη μας ότι η elbow μέθοδος είναι μια ευρετική (heuristic) μέθοδος και, ως εκ τούτου, υπάρχει η πιθανότητα να μην μπορεί να λειτουργήσει καλά σε μια συγκεκριμένη περίπτωση. Μερικές φορές, υπάρχουν περισσότεροι από ένας «αγκώνας», ή δεν εμφανίζεται καθόλου. Σε αυτές τις περιπτώσεις συνήθως καταλήγουμε στον υπολογισμό του καλύτερου k αξιολογώντας πόσο καλά ο k-means λειτούργει στο πλαίσιο του συγκεκριμένου προβλήματος ομαδοποίησης που προσπαθούμε να επιλύσουμε.

Πέρα από την elbow μέθοδο, υπάρχει ένας αριθμός άλλων τεχνικών για την εύρεση του αρχικού αριθμού k [79] συμπεριλαμβανομένων της cross validation, information criteria, information theoretic jump μεθόδου, την silhouette μεθόδου καθώς επίσης και του G-means αλγορίθμου. Μερικοί από αυτούς τους αλγορίθμους αναφέρονται παρακάτω:

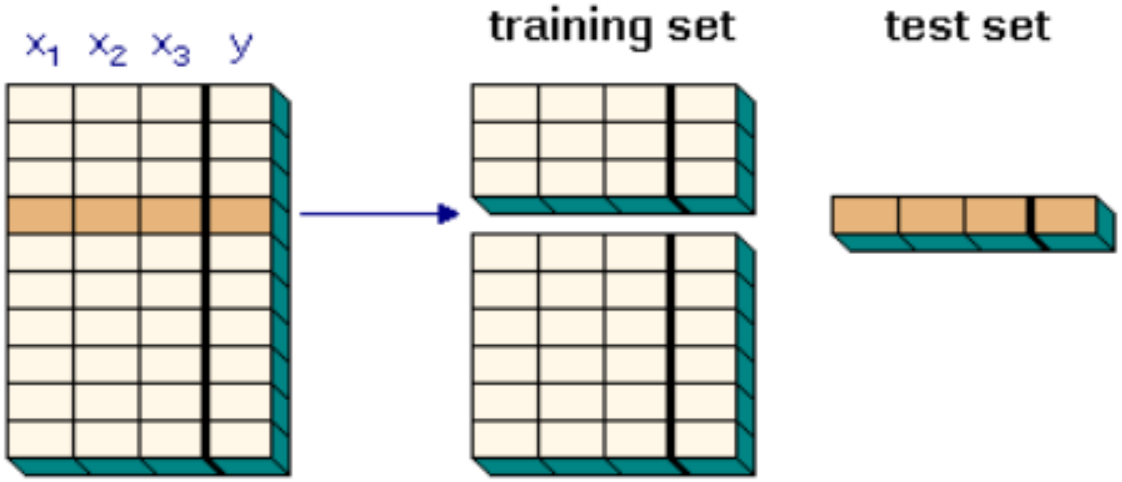
Cross-Validation

Μπορούμε επίσης, να χρησιμοποιήσουμε τη διαδικασία της διασταυρούμενης επικύρωσης (cross validation) για να βρούμε τον αριθμό k των συστάδων [80]. Η γενική ιδέα αυτής της μεθόδου είναι ότι έχουμε v παρατηρήσεις και σε κάθε επανάληψη, χρησιμοποιούμε τα v – 1 δείγματα για εκπαίδευση (training) και 1 για δοκιμή (testing).

Για μια καθορισμένη τιμή του k εφαρμόζουμε τον k-means αλγόριθμο και κάνουμε πρόβλεψη για την v-ιοστή παρατήρηση (έχοντας χρησιμοποιήσει τις v – 1 παρατηρήσεις του δείγματος) υπολογίζοντας κάθε φορά το λάθος.

Αυτή λοιπόν η διαδικασία εφαρμόζεται σε όλες τις πιθανές επιλογές των v παρατηρήσεων. Αφού επαναλάβουμε την διαδικασία αυτή για όλες τις v παρατηρήσεις, υπολογίζουμε τον μέσο όρο των λαθών ο οποίος αποτελεί ένα μέτρο για την σταθερότητα του μοντέλου (το πόσο καλά τα άγνωστα σημεία προβλέπονται από το μοντέλο). Η τελική επιλογή του k γίνεται για την τιμή που επιτυγχάνει την μικρότερη τιμή λάθους (μεγαλύτερη ακρίβεια ταξινόμησης). Σ' αυτήν την περίπτωση, το k παίρνει την μεγαλύτερη δυνατή τιμή (βέλτιστη με την έννοια της Cross-Validation μεθόδου).

Η Cross-Validation μέθοδος παρουσιάζει αρκετές αδυναμίες. Ως ευρετικός αλγόριθμος, δεν περιλαμβάνει καμία θεωρία δειγματοληψίας για στατιστικά συμπεράσματα σχετικά με το μέγεθος και τον αριθμό των συστάδων. Επίσης, δεν υπάρχουν εξωτερικές διαδικασίες επικύρωσης ώστε να εξασφαλιστεί ότι οι συστάδες που προέρχονται από μια συγκεκριμένη ανάλυση συμπλεγμάτων αποτελούν στην πραγματικότητα τις αληθινές συστάδες. Το πιθανό στατιστικό πρόβλημα της απόκτησης αντικειμένων σαν συστάδες περιπλέκεται ακόμα περισσότερο σε ορισμένες διαδικασίες οι οποίες απαιτούν εκ των προτέρων (a priori) υποθέσεις σχετικά με το μέγεθος και τον αριθμό των συστάδων.

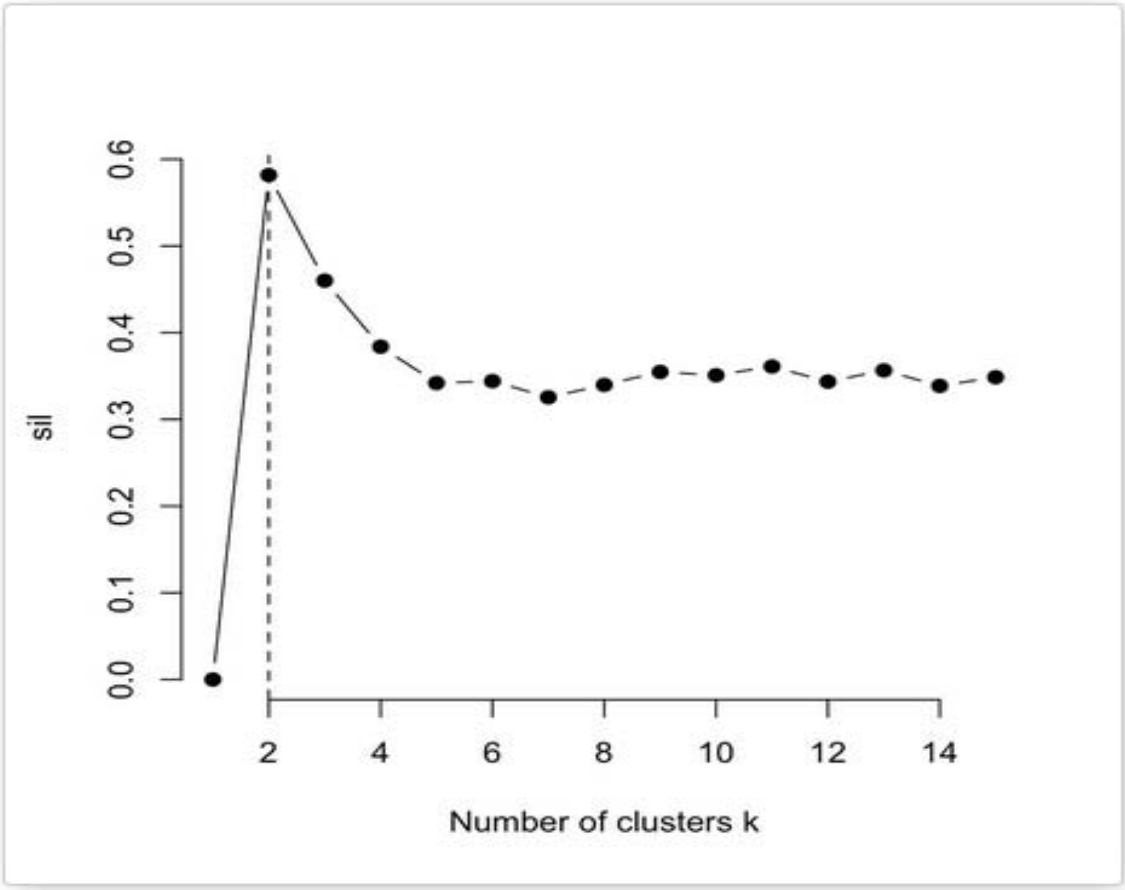


Εικόνα 39: Η διαδικασία της Cross Validation μεθόδου [81]

Silhouette

Η μέση σιλουέτα (silhouette) των δεδομένων είναι άλλο ένα χρήσιμο κριτήριο για την εύρεση του αριθμού k των συστάδων [82]. Η σιλουέτα αναφέρεται σε μια μέθοδο ερμηνείας και επικύρωσης της συνέπειας των δεδομένων μέσα στις συστάδες. Η τεχνική παρέχει μια συνοπτική γραφική αναπαράσταση του πόσο καλά κάθε αντικείμενο βρίσκεται μέσα στην συστάδα του. Η silhouette μπορεί να υπολογιστεί χρησιμοποιώντας οποιαδήποτε μετρική απόσταση, όπως την Ευκλείδεια απόσταση ή την απόσταση Manhattan. Περιγράφηκε για πρώτη φορά από Peter J. Rousseeuw το 1986 [83].

Πιο συγκεκριμένα, η σιλουέτα μιας περίπτωσης δεδομένων είναι ένα μέτρο το οποίο υπολογίζει πόσο στενά ή χαλαρά ταιριάζουν τα δεδομένα μέσα σε μια συστάδα, δηλαδή το σύμπλεγμα των οποίων η μέση απόσταση είναι μικρότερη από τα δεδομένα. Μια σιλουέτα κοντά στο 1 υποδηλώνει ότι τα δεδομένα είναι στο κατάλληλο σύμπλεγμα, ενώ μια σιλουέτα κοντά στο -1 υποδηλώνει ότι τα δεδομένα είναι σε λανθασμένο σύμπλεγμα. Τεχνικές βελτιστοποίησης όπως γενετικοί αλγόριθμοι είναι χρήσιμοι στον καθορισμό του αριθμού των συστάδων οι οποίοι δημιουργούν μία μεγαλύτερη σιλουέτα. Είναι επίσης δυνατόν να τροποποιήσουμε την κλίμακα των δεδομένων με τέτοιο τρόπο ώστε να επιτευχτεί μια μεγιστοποίηση της σιλουέτας προς τον σωστό αριθμό συστάδων.



Εικόνα 40: Η σιλουέτα σε συνάρτηση με τον αριθμό των συστάδων [84]

Τέλος, μία άλλη γρήγορη και απλή μέθοδος είναι να πάρουμε την τετραγωνική ρίζα του αριθμού των σημείων διαιρούμενο δια δύο, και να ορίσουμε το k ίσο με τον αριθμό αυτό. Δηλαδή:

$$k = \sqrt{(n/2)} \quad (3.3)$$

Σαφώς η μέθοδος αυτή έχει μικρότερη ακρίβεια από άλλες οι οποίες όμως μπορούν σε μερικές περιπτώσεις να αποτυγχάνουν.

3.3.2 Διαφορές k-means αλγορίθμου και ιεραρχικής ομαδοποίησης

Ο k-means αλγόριθμος και η ιεραρχική ομαδοποίηση ανήκουν στις μεθόδους με εκμάθηση χωρίς επίβλεψη. Η k-means μέθοδος παράγει μία ενιαία διαμέριση (partition), ενώ η ιεραρχική ομαδοποίηση μπορεί να δώσει διαφορετικές διαμερίσεις ανάλογα με την ανάλυση επίπεδου που εξετάζουμε. Ο k-means απαιτεί τον αρχικό αριθμό των συστάδων που θα δημιουργηθούν ενώ στην αθροιστική ή διαιρετική ιεραρχική ομαδοποίηση δεν απαιτείται ο αριθμός αυτός και μπορούμε να σταματήσουμε την εκτέλεση της σε οποιοδήποτε επίπεδο επιθυμούμε. Επιπλέον, ο k-means αλγόριθμος μπορεί να αποδώσει καλυτέρα σε σχέση με την ιεραρχική ομαδοποίηση όσον αφορά την συσταδοποίηση ενός μεγάλου συνόλου από παρατηρήσεις καθώς η δεύτερη έχει να πάρει πολλές αποφάσεις διαχωρισμού – συνένωσης (merge / split) [85].

Όσον αφορά τον χρόνο εκτέλεσης, η πρώτη μέθοδος είναι γραμμική ως προς τον αριθμό των αντικειμένων δεδομένων δηλ $O(n)$, όπου n είναι ο αριθμός των αντικειμένων δεδομένων. Αντίθετα, η πολυπλοκότητα ως προς τον χρόνο της πλειονότητας των ιεραρχικών αλγορίθμων ομαδοποίησης είναι τετραγωνική δηλαδή $O(n^2)$. Επομένως, για την ίδια ποσότητα δεδομένων, η ιεραρχική ομαδοποίηση θα λάβει τετραγωνικό χρονικό διάστημα.

Ο k-means αλγόριθμος ξεκινά με μια τυχαία επιλογή των κέντρων συμπλέγματος, ως εκ τούτου, μπορεί να δώσει διαφορετικά αποτελέσματα ομαδοποίησης σε διαφορετικές εκτελέσεις του αλγορίθμου. Έτσι, τα αποτελέσματα μπορεί να μην είναι επαναλαμβανόμενα και μ' αυτόν τον τρόπο να υπάρξει μία έλλειψη συνοχής. Αντίθετα, η ιεραρχική ομαδοποίηση, θα δώσει ακριβώς τα ίδια αποτελέσματα σε κάθε επανάληψη εκτέλεσης της.

Ο k-means αλγόριθμος υποθέτει ότι η διακύμανση (variance) της κατανομής του κάθε χαρακτηριστικού (μεταβλητής) είναι σφαιρική. Όλες οι μεταβλητές πρέπει να έχουν την ίδια διακύμανση και τα δεδομένα στοιχεία τα οποία εφαρμόζονται στην διαδικασία της ομαδοποίησης πρέπει να έχουν όμοιο αριθμό παρατηρήσεων. Αν δεν ισχύει κάποια από αυτές τις συνθήκες, τότε ο αλγόριθμος θα αποτύχει.

Ως γενικό συμπέρασμα, ο k-means αλγόριθμος είναι καλός για μεγάλο όγκο δεδομένων που παρουσιάζουν μία σφαιρική μορφή και η ιεραρχική μέθοδος είναι χρήσιμη για μικρά σύνολα δεδομένων. Παρόλα αυτά, και οι δυο τύποι ομαδοποίησης είναι ευρέως χρησιμοποιούμενοι και παρουσιάζουν αξιόλογη απόδοση σε περιπτώσεις εκμάθησης χωρίς επίβλεψη.

3.4 Το Cluster των υπολογιστών

Στα πειράματα (σετ δεδομένων) που εκτελέσαμε, χρησιμοποιήσαμε ένα cluster από συνολικά τρεις υπολογιστές. Τα χαρακτηριστικά των υπολογιστών και η μεταξύ τους σύνδεση παρουσιάζονται παρακάτω:

Developer Laptop

- CPU: Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz
- bogomips: 5387.76
- Launch Date: Q1'11
- Cores: 2
- Threads: 4

- OS: Linux

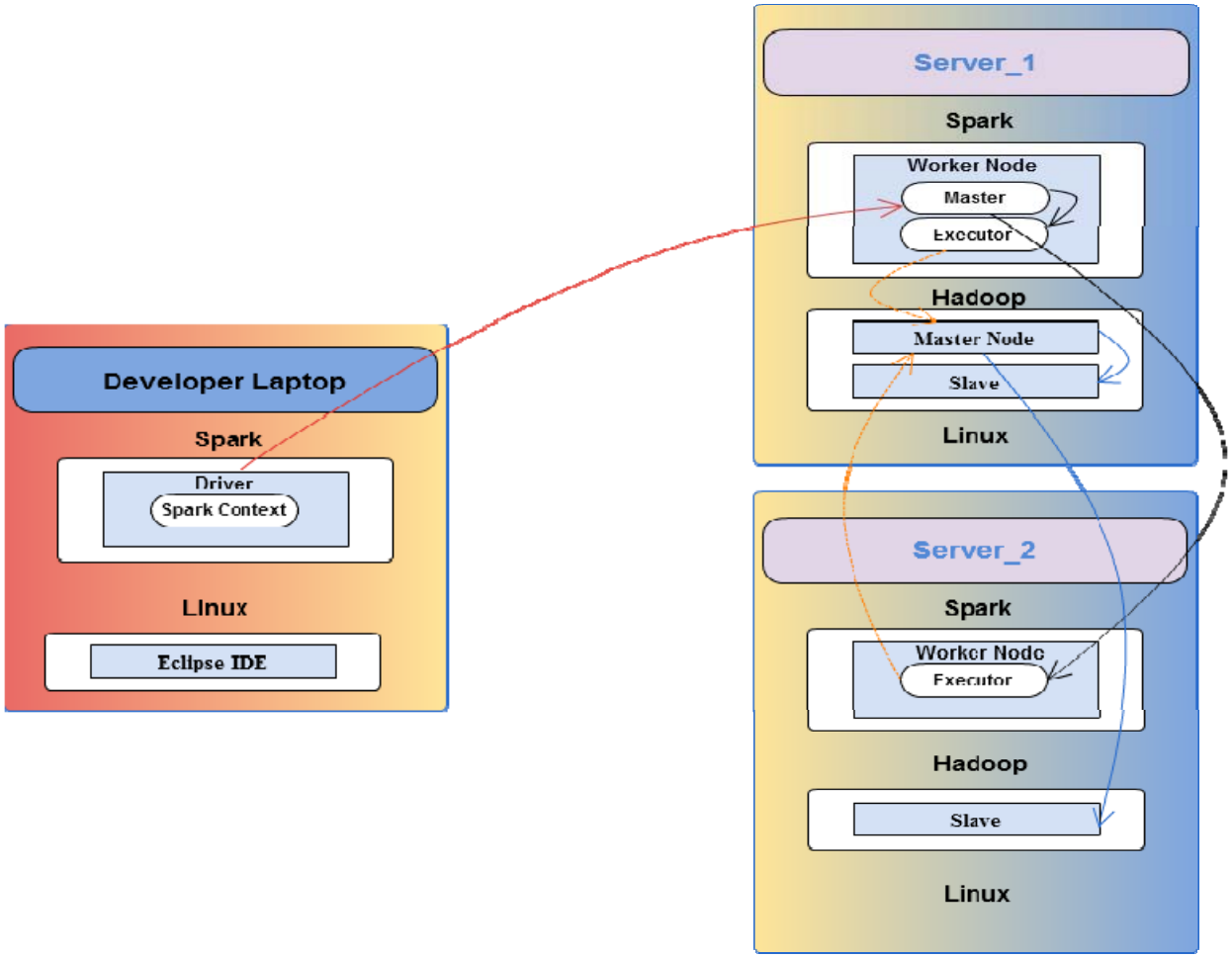
Server_1

- CPU: Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz
- bogomips: 4988.91
- Launch Date: Q1'11
- Cores: 2
- Threads: 4
- OS: Linux

Server_2

- CPU: Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz
- bogomips: 5320.23
- Launch Date: Q1'10
- Cores: 2
- Threads: 4
- OS: Linux

Είναι σημαντικό να έχουμε τον Driver σε ξεχωριστό υπολογιστή επειδή θα καταναλώνει μνήμη που μπορεί να χρησιμοποιηθεί από τους executors. Το Developer Laptop περιέχει το Πρόγραμμα οδήγησης, το οποίο αποτελεί το κύριο πρόγραμμα στο οποίο γίνεται η δήλωση του Spark Context και συντονίζει όλους τους Executors για την εκτέλεση της Spark εφαρμογής. Στο Developer Laptop επίσης εκτελούμε μέσω του Eclipse IDE τον κώδικα της εφαρμογής μας. Οι υπολογιστές Server_1 και Server_2 είναι πανομοιότυποι μεταξύ τους με την διαφορά ότι ο ένας από αυτούς αποτελεί τον Master Node τόσο του Spark framework όσο και του Hadoop.



Εικόνα 41: Το cluster των υπολογιστών που χρησιμοποιήσαμε για την εκτέλεση των σετ δεδομένων και η μεταξύ τους σύνδεση

3.5 Performance Counter Collection

Η προτεινόμενη προσέγγιση μας, εστιάζει σε memory-related θέματα απόδοσης και χρησιμοποιεί τους μετρητές απόδοσης του μεγέθους του σωρού (heap size performance counters) για τα σετ δεδομένων εκτέλεσης.

Σε γενικές γραμμές, υπάρχει ένας μεγάλος αριθμός από μετρητές απόδοσης τους οποίους μπορούμε να παρακολουθήσουμε σε επίπεδο συστήματος. Μερικές από τις κατηγορίες μετρητών απόδοσης που μπορούμε να παρακολουθήσουμε είναι οι memory-related (π.χ. λάθη σελίδων / sec, έγγραφες σελίδας / sec, χρήση μεγέθους σωρού), process-related (π.χ. χρόνος επεξεργασίας), network-related (π.χ. λαμβανόμενα bytes / sec).

Εκτός από μετρητές απόδοσης επιπέδου συστήματος, υπάρχουν επίσης μετρητές απόδοσης τους οποίους μπορούμε να παρακολουθήσουμε σε επίπεδο εφαρμογής (application level). Μηνύματα / sec, μέσο μήκος μηνύματος, μέσος χρόνος απόκρισης για κάθε λειτουργιά, είναι μερικοί από τους μετρητές απόδοσης τους οποίους μπορούμε να παρακολουθήσουμε σ' αυτό το επίπεδο. Σ' αυτό το σημείο πρέπει να σημειωθεί ότι σε περίπτωση που έχουμε μετρητές απόδοσης επιπέδου εφαρμογής και η γλώσσα προγραμματισμού που εφαρμόζεται στο περιβάλλον εκτέλεσης (runtime environment)

δε παρέχει δυνατότητες παρακολούθησης, τότε η δειγματοληψία (sampling) είναι αρκετά δύσκολη και απαιτεί κάποιες επιπλέον κινήσεις από τη πλευρά του χρήστη.

Κάποια από τα εργαλεία παρακολούθησης των μετρητών απόδοσης αποτελούν το Dtrace, sysdig και το JMX. Το DTrace ή Dynamic Tracing [87] αποτελεί ένα εργαλείο επίλυσης προβλημάτων και ανάλυσης απόδοσης το οποίο μπορεί να χρησιμοποιηθεί για να λύσει προβλήματα τόσο στον χώρο του χρηστή αλλά και του πυρήνα (kernel). Αποτελεί ένα ισχυρό διαγνωστικό εργαλείο το οποίο εφαρμόστηκε για πρώτη φορά στο λειτουργικό σύστημα Solaris 10 [73].

Το sysdig είναι ένα πρόγραμμα ανοιχτού κώδικα [88] το οποίο παρέχει τη δυνατότητα εξερεύνησης σε επίπεδο συστήματος: αιχμαλωτίζει την κατάσταση του συστήματος και τις δραστηριότητες μιας Linux οντότητας. Στη συνέχεια, αποθηκεύει, φιλτράρει και αναλύει την κατάσταση αυτή. Το sysdig έχει κατασκευαστεί ώστε να μας προσφέρει εύκολη πρόσβαση στην πραγματική συμπεριφορά των Linux συστημάτων και των δοχείων τους (containers). Εμείς για την παρακολούθηση των μετρητών απόδοσης χρησιμοποιούμε το JMX (java management extensions) καθώς έχουμε να κάνουμε με java εφαρμογές, και για αυτό το λόγο αποτελεί ένα κατάλληλο και χρήσιμο εργαλείο σε σχέση με τα άλλα δυο εργαλεία που αναφερθήκαν προηγουμένως.

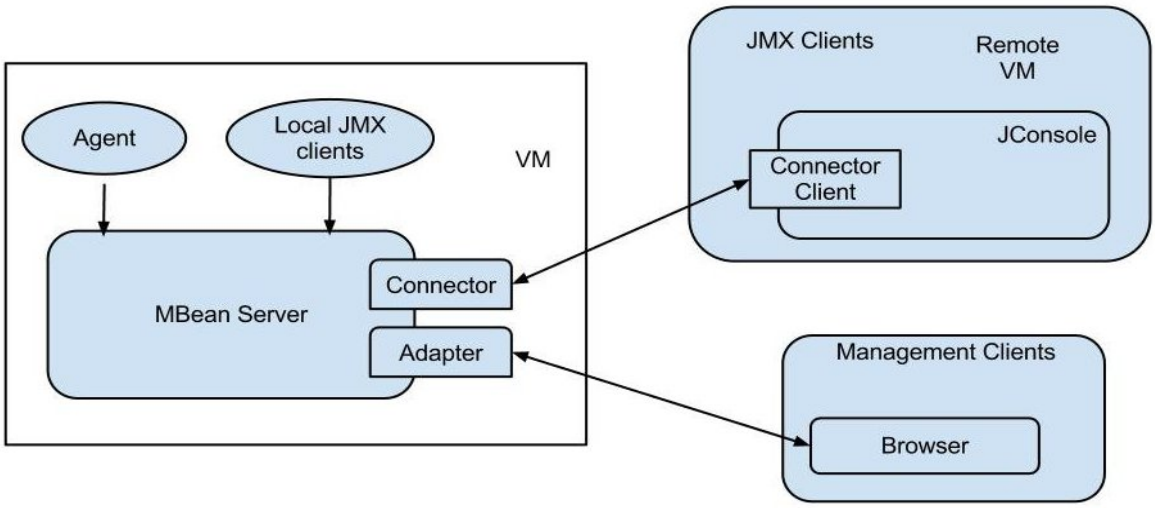
Java Management Extensions

Η τεχνολογία της διαχείρισης επεκτάσεων της Java (JMX) είναι ένα τυπικό μέρος της κανονικής έκδοσης της Java πλατφόρμας (Java SE) [89]. Η τεχνολογία JMX παρέχει έναν απλό, τυποποιημένο τρόπο διαχείρισης των πόρων, όπως εφαρμογές, συσκευές και υπηρεσίες. Χάρις τον δυναμικό χαρακτήρα που προσφέρει η JMX τεχνολογία, μπορούμε να την χρησιμοποιήσουμε για την παρακολούθηση και την διαχείριση των πόρων που έχουν δημιουργηθεί, εγκατασταθεί και εκτελούνται στην εφαρμογή μας. Μας δίνεται επίσης η δυνατότητα να χρησιμοποιήσουμε την τεχνολογία JMX για την παρακολούθηση και την διαχείριση της Java εικονικής μηχανής (JVM).

Οι προδιαγραφές των JMX ορίζουν την αρχιτεκτονική, τα σχεδιαστικά πρότυπα, τα APIs, και το σύνολο των υπηρεσιών στη Java γλώσσα προγραμματισμού για τη διαχείριση και την παρακολούθηση των εφαρμογών και δικτύων.

Χρησιμοποιώντας την τεχνολογία JMX, ένας δεδομένος πόρος εξοπλίζεται από ένα ή περισσότερα Java αντικείμενα γνωστά ως Managed Beans, ή MBeans [90]. Αυτά τα MBeans καταγράφονται σε ένα εξυπηρετητή γνωστό ως MBean server. Ο MBean διακομιστής ενεργεί ως ένας πράκτορας διαχείρισης και μπορεί να τρέξει στις περισσότερες συσκευές που επιτρέπουν την χρήση και εκτέλεση των λειτουργιών της Java γλώσσας προγραμματισμού.

Τα MBeans στοιχεία έχουν γνωρίσματα, τα οποία μπορούν να διαβαστούν και να γραφτούν καθως και λειτουργίες οι οποίες μπορούν να καλεστούν και παρέχουν διάφορες ειδοποιήσεις. Κάθε Managed Bean δημιουργείται και στην συνέχεια πραγματοποιείται η εγγραφή του με την χρήση ενός μοναδικού ονόματος αντικειμένου σε έναν MBean server. Οι MBean διακομηστές συμπεριφέρονται ως αποθηκευτικοί χώροι (repositories) για τα MBeans και παρέχουν πρόσβαση στην διαχείριση εφαρμογών τις οποίες θέλουν να χρησιμοποιούσουν.



Εικόνα 42: Τα κύρια χαρακτηριστικά των επεκτάσεων JMX [91]

Οι προδιαγραφές καθορίζουν τους JMX πράκτορες που χρησιμοποιούμε για τη διαχείριση τυχόν πόρων που έχουν ρυθμιστεί σωστά για διαχείριση. Ένας JMX πράκτορας αποτελείται από έναν MBean διακομιστή στον οποίο είναι εγγεγραμμένα τα Managed Beans και ένα σύνολο υπηρεσιών για τον χειρισμό τους. Με τον τρόπο αυτό, οι JMX πράκτορες ελέγχουν άμεσα τους πόρους και τους καθιστούν διαθεσίμους σε εφαρμογές απομακρυσμένης διαχείρισης. Ο τρόπος με τον οποίο οι πόροι εξοπλίζονται είναι εντελώς ανεξάρτητος από την υποδομή διαχείρισης. Πόροι μπορούν, επομένως, να διαχειριστούν ανεξάρτητα από το πώς υλοποιούνται οι εφαρμογές διαχείρισης τους.

Η τεχνολογία JMX ορίζει πρότυπα - συνδέσμους γνωστούς ως JMX συνδέσμους (connectors) που μας επιτρέπουν να έχουμε πρόσβαση στους JMX πράκτορες από εφαρμογές απομακρυσμένης διαχείρισης. Οι JMX υποδοχές χρησιμοποιώντας διαφορετικά πρωτόκολλα, παρέχουν την ίδια διεπαφή διαχείρισης. Κατά συνέπεια, μια εφαρμογή διαχείρισης μπορεί να διαχειριστεί τους πόρους με διαφάνεια, ανεξάρτητα από το πρωτόκολλο επικοινωνίας που χρησιμοποιείται. Οι JMX πράκτορες μπορούν επίσης να χρησιμοποιηθούν από τα συστήματα ή εφαρμογές που δεν είναι συμβατές με τις JMX προδιαγραφές, εφ 'όσον αυτά τα συστήματα ή εφαρμογές υποστηρίζουν JMX πράκτορες.

Η Java SE πλατφόρμα επίσης παρέχει ένα σετ από MXBeans πλατφόρμες [92]. Μια MXBean πλατφόρμα είναι ένα MBean για την παρακολούθηση και την διαχείριση μιας συγκεκριμένης πλευράς της Java εικονικής μηχανής. Την πρώτη φορά που κάποιος επιχειρεί να αποκτήσει πρόσβαση σ' έναν MBeans server, τότε δημιουργείται ο εξυπηρετητής και όλες οι MXBeans πλατφόρμες εγγράφονται χρησιμοποιώντας τα μοναδικά ονόματα αντικειμένων. Τα κυρία μέρη της JVM τα οποία διαχειρίζονται από την MXBeans πλατφόρμα είναι τα εξής:

- Class loading system
- Compilation system

- Garbage collector
- Logging System
- Memory
- Memory pool
- Memory system
- Runtime system
- Thread system
- Underlying operating system

Οι JMX επεκτάσεις [93] για τη διαχείριση και την παρακολούθηση των πόρων, αποτελούν μια προαιρετική επέκταση του JDK προτύπου και μπορούν να χρησιμοποιηθούν στη θέση του απλού πρωτοκόλλου διαχείρισης δικτύου (SNMP). Για την παράγωγη των μετρητών απόδοσης, χρησιμοποιήσαμε τις JMX επεκτάσεις καθώς επέτρεπαν μια καλύτερη διαχείριση της Java εικονικής μηχανής.

Memory Heap Usage

Έχοντας σαν κύριο στόχο την παρακολούθηση της συνολικής χρήσης της μνήμης του σωρού (memory heap) χρησιμοποιώντας τις JMX επεκτάσεις, αποκτάμε πρόσβαση στην MXBean πλατφόρμα με object name java.lang:type=Memory και παίρνουμε την τιμή της HeapMemoryUsage ιδιότητας. Συμφώνα με τις οδηγίες (documentation) της java.lang.management.MemoryUsage κλάσης, το επιστρεφόμενο αντικείμενο αναπαριστά την τωρινή χρήση της μνήμης του σωρού η οποία χρησιμοποιείται για την δέσμευση των αντικειμένων. Ο σωρός αποτελείται από μια η περισσότερες ομάδες μνήμης (pools) και το χρησιμοποιούμενο πεδίο του αντικειμένου αυτού είναι στην ουσία το ποσό της μνήμης που είναι δεσμευμένο από τα ζωντανά αντικείμενα (live objects) και τα αντικείμενα-σκουπίδια (garbage objects) τα οποία (αν υπάρχουν) δε έχουν συλλεχτεί ακόμα σε όλα αυτές τις ομάδες.

3.6 Εφαρμογή ενός μικρού σετ δεδομένων

Το σετ δεδομένων που εισάγεται σαν είσοδο στην εφαρμογή μας, ύστερα από τις φάσεις του log abstraction και time-slice profiling που ορίζουν οι Syer et al.,αποτελείται από 21 συνολικά time-slice προφίλ τα οποία φορτώνονται και αποθηκεύονται σε ένα JavaRDD από TimeSliceProfile αντικείμενα. Κάθε αντικείμενο της κλάσης αυτής έχει τις εξης ιδιοτητες (properties):

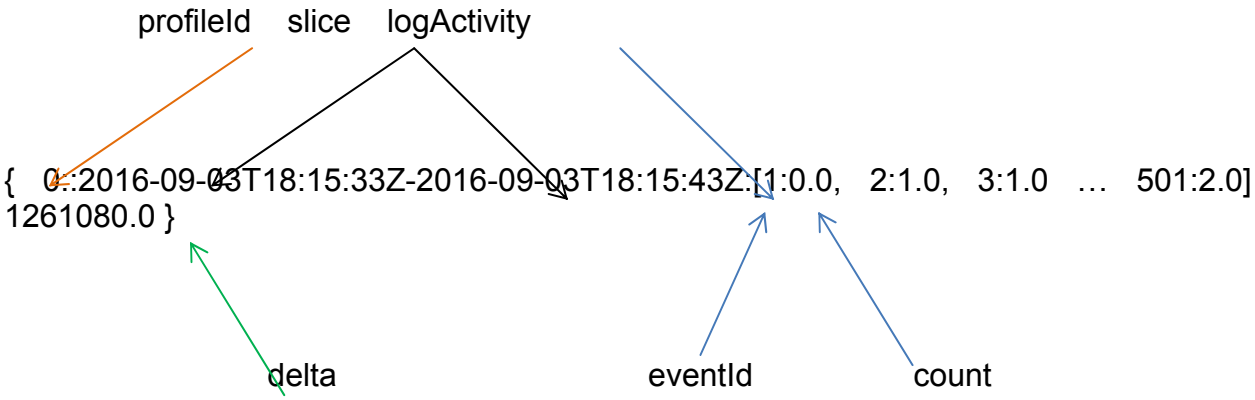
- Private final String profileId: Ο μοναδικός αριθμός Id του συγκεκριμένου προφίλ.

- Private final TimeSlice slice: η χρόνο-φέτα που αναπαριστά το προφίλ αυτό.

- Private final Map<String, Double> logActivity: η log δράση κατά την διάρκεια του χρονικού διαστήματος.

- Private final double delta: Οι αλλαγές που εντοπίζονται στην χρήση της μνήμης μέσω των μετρητών απόδοσης

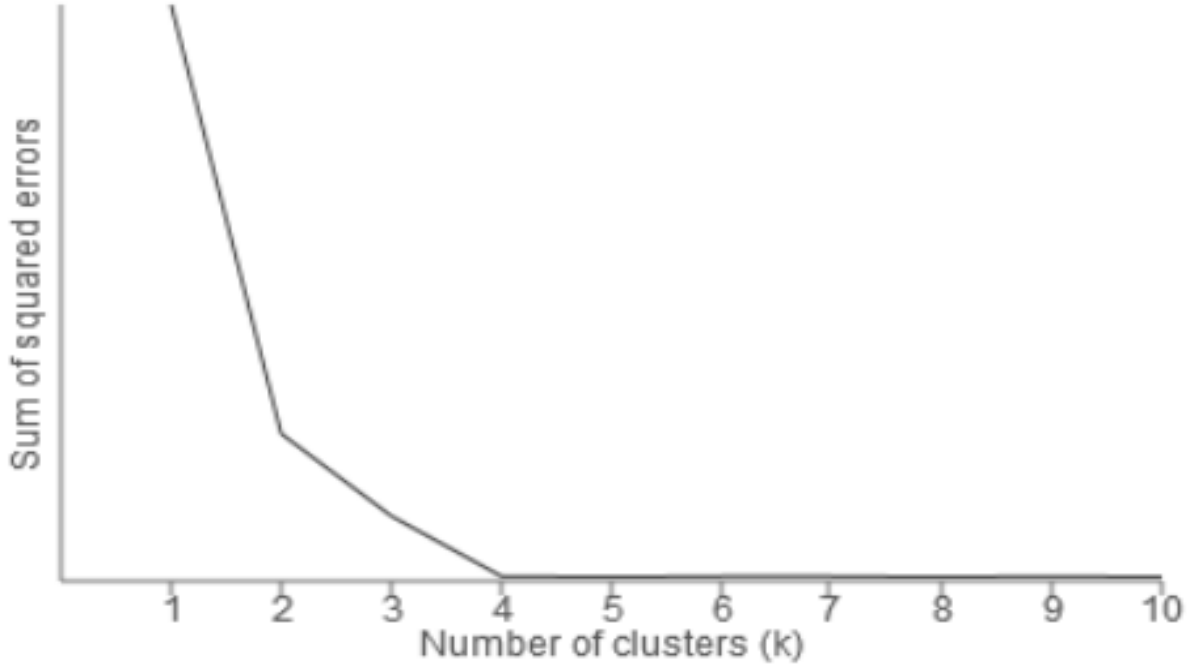
Κάθε timeSliceProfile το οποίο φορτώθηκε στην εφαρμογή έχει την εξής μορφή:



Αρχικά χρησιμοποιούμε την μέθοδο αγκώνα στην οποία τρέχουμε το JavaRDD<TimeSliceProfile> εφαρμόζοντας τον k-means αλγόριθμο για k = 2 έως k = 8 έτσι ώστε να βρούμε το κατάλληλο αριθμό k που θα εφαρμοστεί στην εκτέλεση του συγκεκριμένου σετ δεδομένων. Για κάθε k υπολογίζεται τόσο το άθροισμα τετραγώνων (WSSE) όσο και η διακύμανση και αποθηκεύονται σε έναν πίνακα τιμών διπλής ακρίβειας.

Χρησιμοποιούμε το Octave [86] για υπολογισμό και παρουσίαση της γραφικής παράστασης που είναι πιθανό να μας δείξει τον κατάλληλο αριθμό k που πρέπει να εφαρμοστεί. Ο πινάκας ο οποίος παίρνουμε σαν αποτέλεσμα της elbow συνάρτησης έχει συνολικά 8 τιμές διπλής ακρίβειας.

Μετά την εφαρμογή του πίνακα αυτού στο Octave έχουμε την παρακάτω γραφική παράσταση:



Εικόνα 43: Η γραφική παράσταση του αριθμού k των συστάδων σε συνάρτηση με το άθροισμα των τετραγώνων

Παρατηρούμε πως σχηματίζεται μία μορφή αγκώνα για k = 4. Συνεπώς, θα δημιουργήσουμε συνολικά τέσσερα clusters (όσους δηλαδή είναι ο αριθμός k) για το συγκεκριμένο σετ δεδομένων που παίρνουμε σαν είσοδο.

Κάθε time-slice προφίλ περιέχει κάποιες πληροφορίες οι οποίες στην φάση αυτή δεν μας είναι χρήσιμες. Για αυτόν τον λόγο, στόχος μας είναι να απομονώσουμε τον πίνακα που περιέχει των αριθμό των φορών που εντοπίζεται ένα συγκεκριμένο event στην αντίστοιχη χρόνο-φέτα. Μαζεύουμε όλα αυτά τα δεδομένα, αφαιρούμε τα IDs των events και αναθέτουμε τις τιμές που μένουν σε ένα JavaRDD<Vector>. Δηλαδή:

[1:0.0, 2:1.0, 3:1.0 … 501:2.0] ⟶ [0.0, 1.0, 1.0 … 2.0]

Πλέον, τα δεδομένα μας έχουν την κατάλληλη μορφή ώστε να εφαρμοστεί σ' αυτά ο k-means αλγόριθμος που μας παρέχεται από την MLlib βιβλιοθήκη. Πιο συγκεκριμένα, αρχικά εκπαιδεύουμε τα δεδομένα μας χρησιμοποιώντας σαν ορίσματα το k = 4 που αντιπροσωπεύει τον αριθμό των συστάδων που θα παραχθούν και ορίζουμε την μεταβλητή numIterations ίσο με 50. Το numIterations είναι ένας ακέραιος αριθμός και αντιπροσωπεύει τον μέγιστο αριθμό των επαναλήψεων που θα εφαρμοστούν στα βήματα του k-means αλγορίθμου. Στην συνέχεια, υπολογίζονται τα κέντρα των τεσσάρων συστάδων που θα δημιουργηθούν. Έπειτα, παίρνουμε σαν έξοδο ένα JavaRDD από ακεραίους ο οποίος ονομάζεται Indices και μας παρέχει την πληροφορία για το που ακριβώς θα κατηγοριοποιηθεί το κάθε ένα από τα 21 διανύσματα που εισάγαμε. Η συλλογή αυτή έχει την μορφή:

Indices: [2 0 0 3 1 2 0 0 3 1 2 0 0 3 1 2 0 0 3 1 2]

Δηλαδή: το $1^{ο}$ διάνυσμα ⟶ Cluster 2

το $2^{ο}$ διάνυσμα ⟶ Cluster 0 κ.ο.κ.

Επιπλέον μπορούμε να βρούμε τον συνολικό αριθμό των vectors που κάθε συστάδα περιέχει:

Cluster 0 ⟶ 8 vectors

Cluster 1 ⟶ 4 vectors

Cluster 2 ⟶ 5 vectors

Cluster 3 ⟶ 4 vectors

Το κάθε διάνυσμα στην ουσία αντιστοιχεί σε ένα time-slice προφίλ. Επομένως, γνωρίζουμε ότι το $1^{ο}$ προφίλ θα πάει στο cluster 2, το $2^{ο}$ προφίλ στο cluster 0 κ.ο.κ. Σαν τελικό αποτέλεσμα παίρνουμε ένα JavaRDD<TimeSliceProfileCluster> το οποίο είναι μια συλλογή που αποτελεί τις τελικές μας συστάδες, συνολικά τέσσερα TimeSliceProfileCluster αντικείμενα. Το κάθε ένα από αυτά περιλαμβάνει τα time-slice προφίλ τα οποία ανατέθηκαν στις συστάδες σύμφωνα με τον k-means αλγόριθμο που εφαρμόστηκε προηγουμένως.

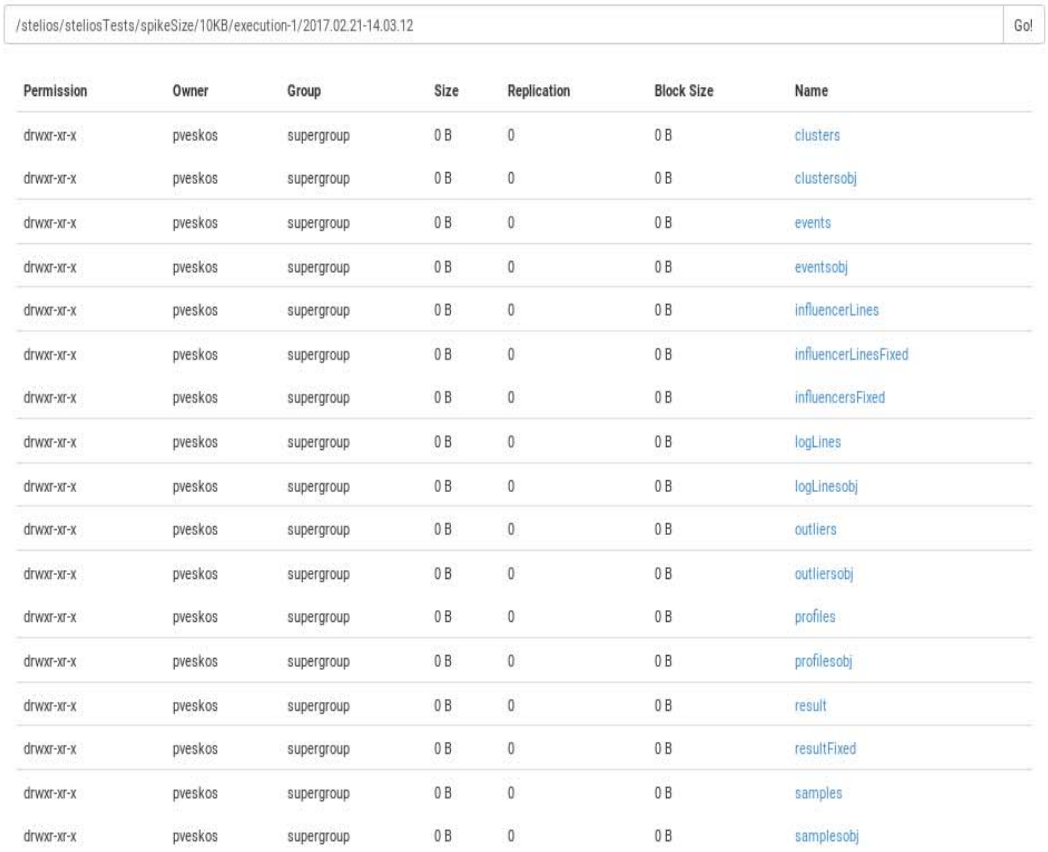
Το επόμενο μας βήμα είναι να εκτελέσουμε την συνάρτηση detect η οποία ανήκει στη κλάση του outlierDetector και δέχεται σαν είσοδο ένα JavaRDD<TimeSiceProfileCluster>, δηλαδή την συλλογή από τους clusters που σχηματιστήκαν στο προηγούμενο βήμα. Η detect συνάρτηση δίνει σαν έξοδο ένα JavaRDD<Outlier>, δηλαδή μια συλλογή από Outliers που βρεθήκαν στο σετ δεδομένων. Σαν τελευταίο στάδιο εκτέλεσης του σετ δεδομένων αυτού, είναι να εφαρμόσουμε την συνάρτηση analyse η οποία περιλαμβάνεται στην κλάση InfluenceAnalyzer, δέχεται είσοδο το JavaRDD<Outlier> το οποίο πήραμε από το προηγούμενο βήμα και δίνει έξοδο μια λίστα από <Tuple2<Outlier, List<Influencer>>>, δηλαδή μια λίστα από ζευγάρια ενός Outlier και μιας λίστας από influencer γραμμές που βρέθηκαν στο σετ δεδομένων που εκτελέσαμε. Οι Influencer γραμμές είναι αυτές οι οποίες μπορούν να περιγράφουν γεγονότα τα οποία είναι υπεύθυνα για την πρόκληση ενός προβλήματος που σχετίζεται με την μνήμη (πχ. ένα memory spike).

4. **ΑΠΟΤΕΛΕΣΜΑΤΑ**

Σαν αποτέλεσμα της εκτέλεσης ενός σετ δεδομένων, παρέχονται στο Hadoop distributed file system, φάκελοι οι οποίοι περιλαμβάνουν τα παρακάτω στοιχεία:

- Clusters: δηλαδή οι συστάδες που δημιουργήθηκαν (στην περίπτωση του k-means, ο αριθμός τους είναι ίσος με τον αριθμό k).
- Events: τα γεγονότα εκτέλεσης τα οποία περιλαμβάνουν δυναμική πληροφορία (π.χ. αντιγραφή ενός αριθμού από bytes).
- Log lines: όλες οι γραμμές καταγραφής του συγκεκριμένου σετ δεδομένων. Όλες οι γραμμές περιλαμβάνουν ένα timestamp περιγράφοντας τον ακριβή χρόνο εκτέλεσης τους.
- Profiles: όλα τα time-slice προφίλ τα οποία δημιουργούνται.
- Outliers: αποτελούν τις απομακρυσμένες συστάδες, οι οποίες περιλαμβάνουν τα time-slice προφίλ που είναι υπεύθυνα για προβλήματα διευθέτησης της μνήμης.
- Influencer lines: είναι όλες οι γραμμές οι οποίες αναγνωριστήκαν σαν γεγονότα τα οποία ασκούν επιρροή στο σετ δεδομένων που εκτελέστηκε. Οι influencer γραμμές είναι πιθανό οι ίδιες να αποτελούν κάποια προβλήματα τα οποία ανήκουν σε μια κατηγορία memory-related θεμάτων (π.χ. memory spikes).

Στόχος μας είναι να εντοπίσουμε τις γραμμές που επηρεάζουν την λειτουργία της εφαρμογής και προκαλούν κάποιο πρόβλημα όσον αφόρα την διευθέτηση της μνήμης. Οι γραμμές αυτές θα περιλαμβάνονται στον φάκελο των influencer lines. Σε περίπτωση που ο αλγόριθμος, είτε δώσει λανθασμένες influencer γραμμές (δηλαδή γραμμές που δε περιγράφουν κάποιο memory-related θέμα), είτε δε δώσει κάποιο αποτέλεσμα (δηλαδή ο φάκελος των influencer lines θα είναι κενός), μπορούμε να ελέγξουμε τον φάκελο με τις απομακρυσμένες συστάδες και να κάνουμε μία αναζήτηση των time-slice προφίλ που ανήκουν στις συστάδες αυτές.



/stelios/steliosTests/spikeSize/10KB/execution-1/2017.02.21-14.03.12 Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	clusters
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	clustersobj
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	events
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	eventsobj
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	influencerLines
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	influencerLinesFixed
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	influencersFixed
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	logLines
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	logLinesobj
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	outliers
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	outliersobj
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	profiles
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	profilesobj
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	result
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	resultFixed
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	samples
drwxr-xr-x	pveskos	supergroup	0 B	0	0 B	samplesobj

Εικόνα 44: Ο οθόνη του HDFS περιλαμβάνοντας όλα τα στοιχεία εξόδου της εφαρμογής μας

Αρχικά, πρέπει να αναφέρουμε δυο έννοιες τις οποίες χρησιμοποιούμε για την αξιολόγηση των αποτελεσμάτων μας, τα μεγέθη precision (ακρίβεια) και recall. Υπάρχουν δυο περιπτώσεις χρήσης τους. Η πρώτη αφορά την εύρεση των γεγονότων τα οποία ευθύνονται για τη πρόκληση ενός memory-related θέματος και η δεύτερη περίπτωση αφορά το επίπεδο των συστάδων που δημιουργούνται σαν αποτέλεσμα. Οι έννοιες αυτές αναλύονται εκτενεστέρα παρακάτω:

- Precision: σε επίπεδο των influencer γραμμών, αποτελεί το πηλίκο των συνολικών memory-related θεμάτων που βρέθηκαν ως προς την ακρίβεια τους. Το precision είναι ένα μέγεθος το οποίο αξιολογεί τα αποτελέσματα και πιο συγκεκριμένα την αναλογία μεταξύ των γραμμών που αντιστοιχούν σε memory-related θέματα και σ' αυτές οι οποίες είναι «φυσιολογικές» χωρίς να επηρεάζουν την εκτέλεση του σετ δεδομένων. Αντίστοιχα, στο επίπεδο των outliers, το precision υπολογίζεται ως το πηλίκο των απομακρυσμένων συστάδων που βρέθηκαν ως προς τον συνολικό αριθμό των σωστών απομακρυσμένων συστάδων που βρέθηκαν μετά την εκτέλεση του σετ δεδομένων.

$$Precision_{mri} = \frac{Total\ Correct\ Memory{-}Related\ Issues}{Total\ Influencer\ Lines} \quad (4.1)$$

όπου αφορά το επίπεδο των influencer γραμμών που εντοπίζονται.

$$Precision_{outlier} = \frac{Total\ Outliers\ Detected}{Total\ Correct\ Outliers\ Detected} \quad (4.2)$$

όπου αφορά το επίπεδο των outliers που εντοπίζονται.

- Recall (μνήμη): αποτελεί το πηλίκο των συνολικών influencer γραμμών που βρεθήκαν οι οποίες αντιστοιχούν σε γεγονότα που αποτελούν memory-related θέματα, ως προς τα συνολικά memory-related θέματα που περιέχονται στο αρχείο δεδομένων. Αντίστοιχα, στο επίπεδο των outliers, αποτελεί το πηλίκο των σωστών απομακρυσμένων συστάδων που βρέθηκαν ως προς τα συνολικά θεωρητικά outliers. Ένα cluster αναφέρεται ως «θεωρητικό» outlier σε περίπτωση που παρόλο έχει διαγνωστεί από τον αλγόριθμο ως φυσιολογικό cluster, περιέχει time-slice προφίλ τα οποία αντιστοιχούν σε events που σχετίζονται με memory-related θέματα. Ουσιαστικά, το recall είναι ένα σημαντικό ποσοτικό μέγεθος καθώς χαρακτηρίζει την επιτυχία του αλγορίθμου ως προς την διάγνωση όλων των προβλημάτων που σχετίζονται με την διευθέτηση της μνήμης. Το recall δίνεται από τον παρακάτω τύπο:

$$Recall_{\ mri} = \frac{Total\ Memory{-}Related\ Issues\ Detected}{Total\ Memory{-}Related\ Issues} \quad (4.3)$$

όπου αφορά το επίπεδο των influencer γραμμών που εντοπίζονται.

$$Recall_{outlier} = \frac{Correct\ Outliers}{Total\ Theoritical\ Outliers} \quad (4.4)$$

όπου αφορά το επίπεδο των outliers που εντοπίζονται.

Τα precision και recall μεγέθη έχουν σαν μέγιστη τιμή την μονάδα. Ο κύριος σκοπός μας δεν είναι να βρούμε όλα τα προβλήματα που σχετίζονται με κάποιο πρόβλημα διευθέτησης της μνήμης, αλλά να βελτιώσουμε όσο μπορούμε περισσότερο τον τρόπο εύρεσης των outliers. Αυτό επιτυγχάνουν και τα μεγέθη precision και recall στο επίπεδο των outliers.

Αρχικά εφαρμόζουμε τους δυο αλγορίθμους σε μικρά σετ δεδομένων ώστε να δοκιμάσουμε την λειτουργία τους ως προς τα διάφορες ιδιότητες (π.χ. χρόνοι δειγματοληψίας, αριθμό των συστάδων, χρήση ευριστικών μεθόδων κ.α.) και κάτω από διαφορετικές συνθήκες που μπορούν να έχουν. Στη συνέχεια θα εφαρμόσουμε τους δυο αλγορίθμους σε επίπεδο Big Data (δηλαδή μεγαλύτερης κλίμακας δεδομένων) ώστε να ελέγξουμε πως λειτουργούν οι δυο μέθοδοι όταν εισάγουμε περισσότερα δεδομένα. Για την αξιολόγηση των δυο μεθόδων συσταδοποίησης, απαιτείται η κατασκευή μιας ρεαλιστικής προσομοίωσης δεδομένων.

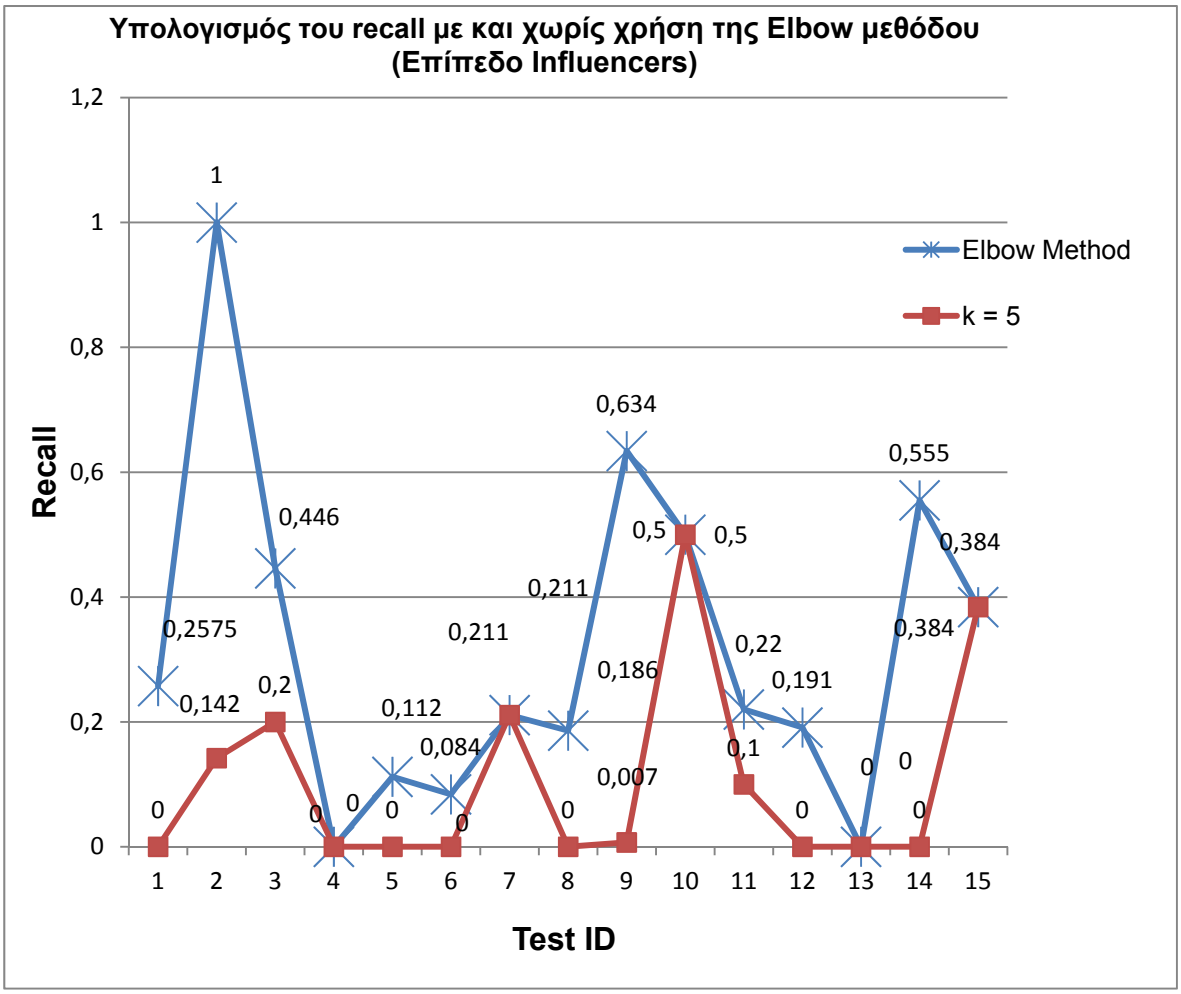
Για την δημιουργία των σετ δεδομένων, χρησιμοποιούμε ένα πρόγραμμα παραγωγής συνθετικών δεδομένων υλοποιημένο από τη Software Competitiveness International. Η

εφαρμογή αυτή είναι γραμμένη σε γλώσσα Java και προσομοιώνει μια Java εφαρμογή πραγματικού κόσμου που προκαλεί διαμορφώσιμες (configurable) αιχμές μνήμης και παράγει συγχρόνως τα κατάλληλα αρχεία καταγραφής εκτέλεσης. Με λίγα λόγια, το πρόγραμμα αυτό κατασκευάζει P διαδικασίες οι οποίες είναι ουσιαστικά ένα προκαθορισμένο σύνολο γεγονότων που επαναλαμβάνεται επ 'αόριστον. Κάθε διαδικασία εκτελείται από ένα ξεχωριστό νήμα. Η κάθε P διαδικασία, μόλις ξεκινήσει, εκτελούνται τα γεγονότα με την σειρά, ένα, ένα σύμφωνα με την συχνότητα της. Η εκτέλεση ενός φυσιολογικού γεγονότος προκαλεί την εκτύπωση μόνο μιας γραμμής η οποία περιγράφει το event που πραγματοποιήθηκε. Αντίθετα, η εκτέλεση ενός αφύσικου (abnormal) γεγονότος δημιουργεί ένα memory spike μεγέθους S πριν την εκτύπωση της συγκεκριμένης γραμμής. Το μέγεθος S καθορίζεται από εμάς.

4.1 Χρήση της Elbow Μεθόδου

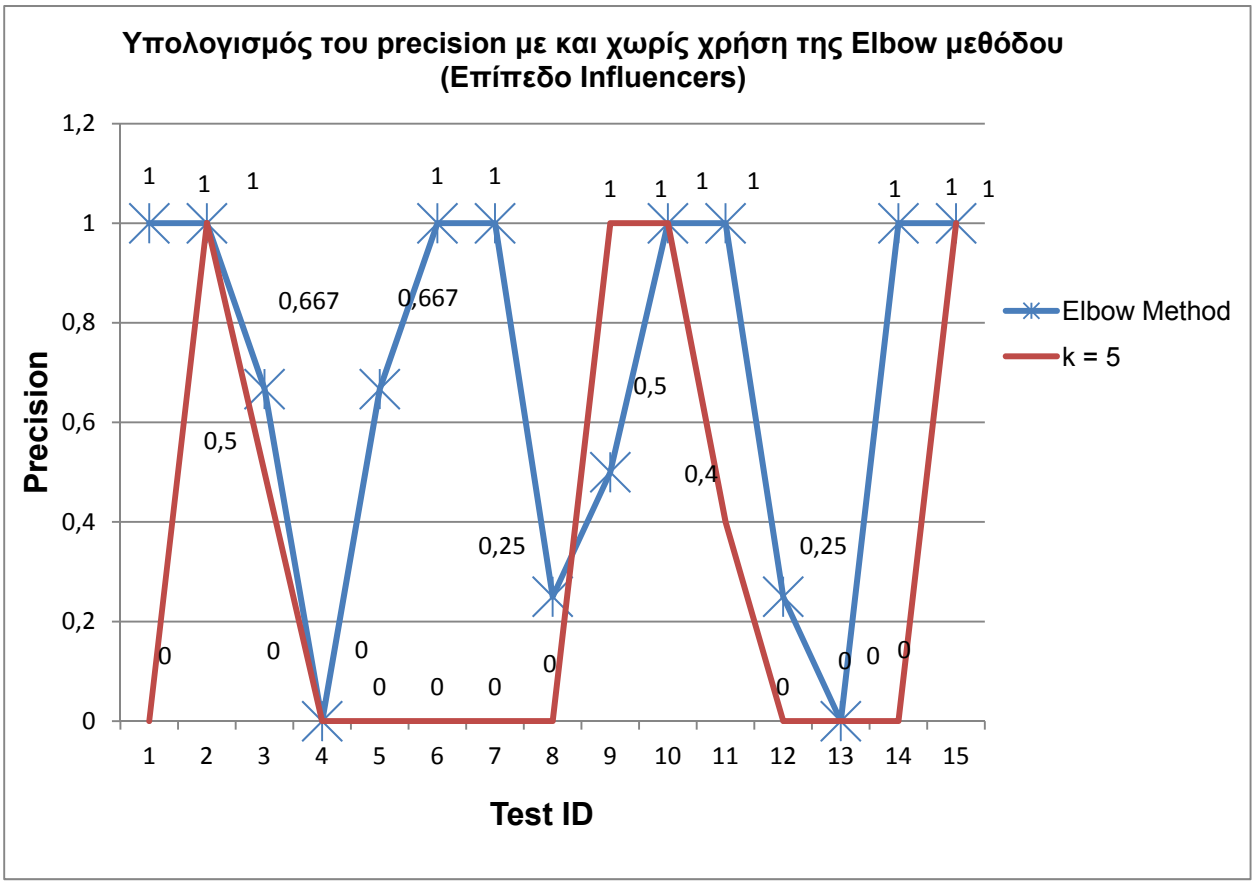
Όσον αφορά τον αριθμό των συστάδων που θα εφαρμόσουμε στην εκτέλεση του k-means αλγορίθμου, η χρήση της μεθόδου αγκώνα προσφέρει καλύτερα αποτελέσματα σε σχέση με τη χρήση ενός τυχαίου αριθμού k. Μερικές φορές βέβαια, η μέθοδος δεν σχηματίζει κάποια μορφή «αγκώνα» αναγκάζοντας μας να πραγματοποιήσουμε την εκτέλεση των δεδομένων εφαρμόζοντας στην θέση του k, μια ποικιλία αριθμών (συνήθως $3 < k < 9$).

Για να βρούμε την απόδοση που έχει η elbow μέθοδος στην εκτέλεση των δεδομένων μας, εξετάζουμε την χρήση της σε κάποια δοκιμαστικά συνθετικά τεστ παρουσιάζοντας τα precision και recall μεγέθη που βρήκαμε. Πιο συγκεκριμένα, εκτελέσαμε 15 διαφορετικά σετ δεδομένων εφαρμόζοντας τον αλγόριθμο μας με σταθερό $k = 5$ και με χρήση του k το οποίο βρίσκεται χάρις τη χρήση της elbow μεθόδου. Ο στόχος μας είναι να βρούμε τα memory spikes που περιέχουν τα σετ δεδομένων. Η αξιολόγηση των αποτελεσμάτων παρουσιάζεται παρακάτω:



Σχήμα 1: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means με και χωρίς χρήση της elbow μεθόδου

Παρατηρούμε πως σε όλα τα δοκιμαστικά τεστ, η χρήση της μεθόδου «αγκώνα» για την εύρεση του αριθμού k, παρουσιάζει ίσα ή καλύτερα αποτελέσματα σε σχέση με την εκτέλεση του k-means αλγορίθμου με ένα τυχαίο σταθερό k (π.χ. k = 5 στην περίπτωση μας). Γι' αυτόν τον λόγο, πριν τρέξουμε ένα νέο σετ δεδομένων, εφαρμόζουμε πρώτα την elbow μέθοδο για να πάρουμε τον πίνακα με τα αθροίσματα τετραγώνων. Η γραφική παράσταση των SSE θα μας δείξει ποιο ακριβώς k να εφαρμόσουμε στον αλγόριθμο μας εξοικονομώντας μας μ' αυτόν τον τρόπο πολύτιμο χρόνο και κόπο.



Σχήμα 2: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means με και χωρίς χρήση της elbow μεθόδου

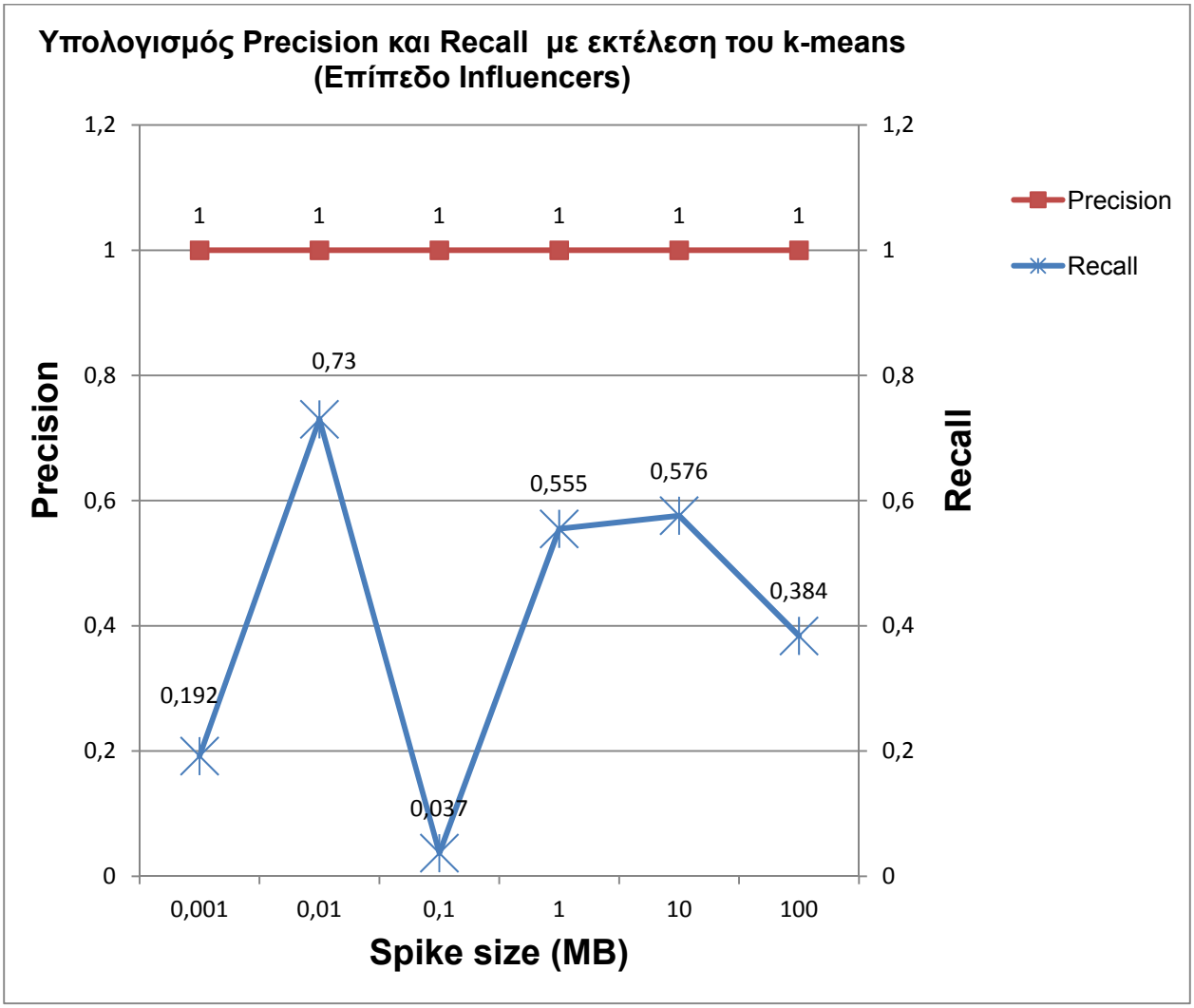
Ομοίως, στην περίπτωση του precision, παρατηρούμε πως σε όλα τα δοκιμαστικά τεστ, η χρήση της μεθόδου «αγκώνα» για την εύρεση του αριθμού k, παρουσιάζει καλύτερα αποτελέσματα σε σχέση με την εκτέλεση του k-means αλγορίθμου με ένα τυχαίο σταθερό k (π.χ. k = 5 στην περίπτωση μας). Γι' αυτόν τον λόγο, στα διάφορα σετ δεδομένων, εφαρμόζουμε τον αριθμό k τον οποίο βρίσκουμε μετά την εκτέλεση της elbow μεθόδου.

4.2 Memory Spike Size Case Study

Εφαρμόζουμε τον k-means αλγόριθμο και την ιεραρχική ομαδοποίηση σε έξι (6) διαφορετικές περιπτώσεις σετ δεδομένων τα οποία περιλαμβάνουν memory spikes. Τα σετ δεδομένων αυτά, περιλαμβάνουν συνολικά 26 memory spikes τα οποία διαφέρουν ως προς το μέγεθος τους. Πιο συγκεκριμένα έχουμε τις εξής περιπτώσεις σετ δεδομένων:

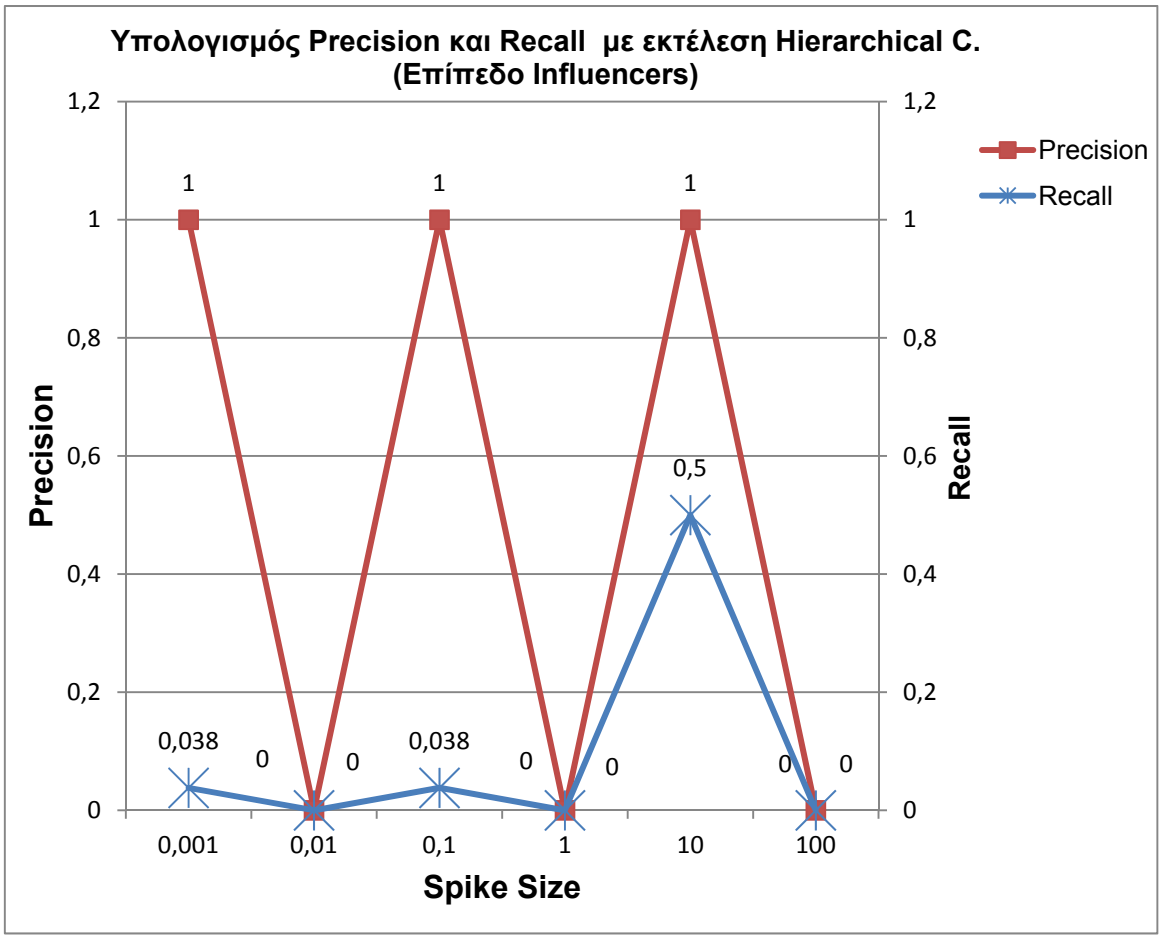
- 1KB Memory spike data set
- 10KB Memory spike data set
- 100KB Memory spike data set
- 1MB Memory spike data set
- 10MB Memory spike data set
- 100MB Memory spike data set

Τρέχουμε όλα αυτά τα σετ δεδομένων εφαρμόζοντας τον k-means αλγόριθμο και την ιεραρχική ομαδοποίηση με χρόνο δειγματοληψίας ίσο με δέκα (10) δευτερόλεπτα. Τα αποτελέσματα των data sets αυτών παρατίθενται στη συνέχεια:



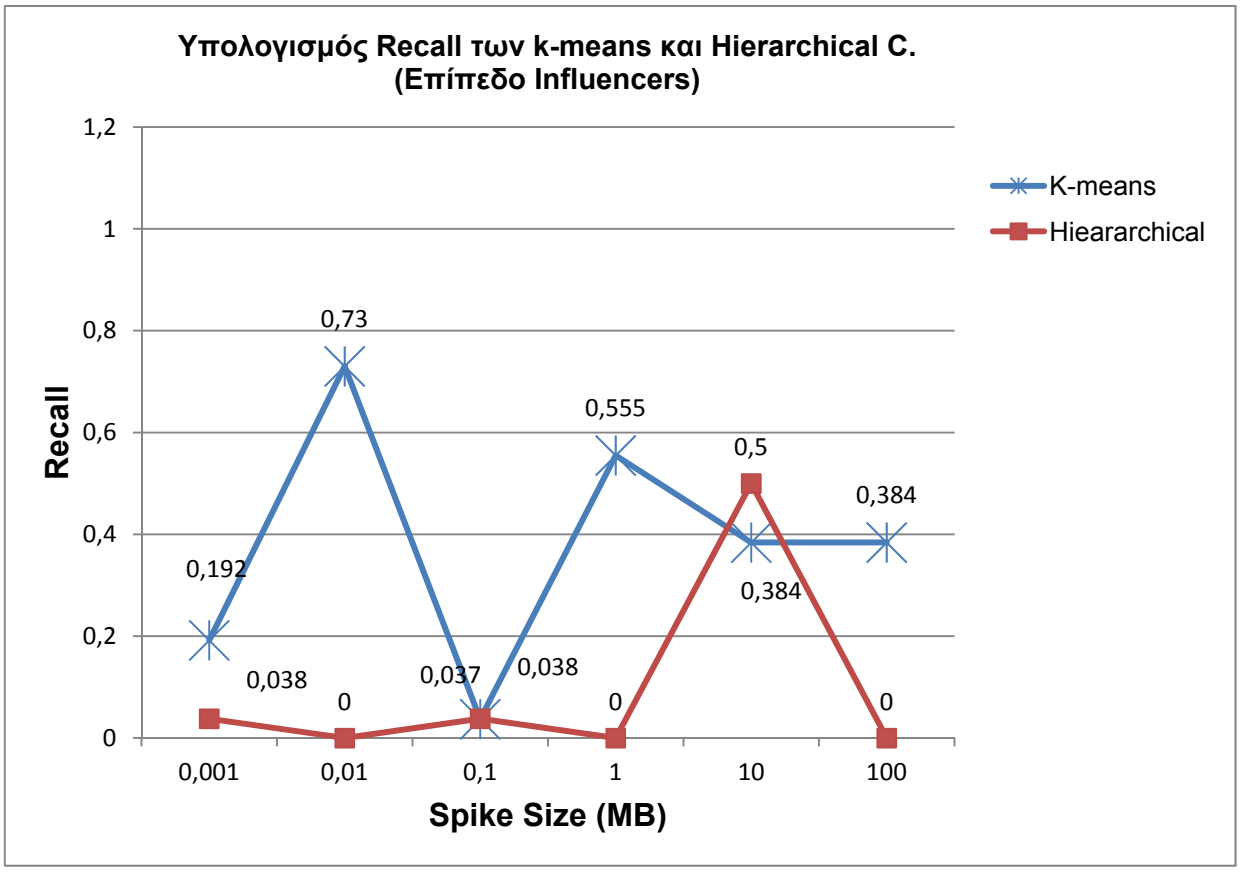
Σχήμα 3: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο

Παρατηρούμε πως σε όλες τις περιπτώσεις, η ακρίβεια είναι ίση με μονάδα, που σημαίνει ότι οποιαδήποτε influencer γραμμή βρίσκουμε, αντιστοιχεί σε περίπτωση ενός memory spike. Το recall μέγεθος δείχνει το ποσοστό των συνολικών memory spikes που βρέθηκαν μετά το πέρας της εκτέλεσης του αλγορίθμου.



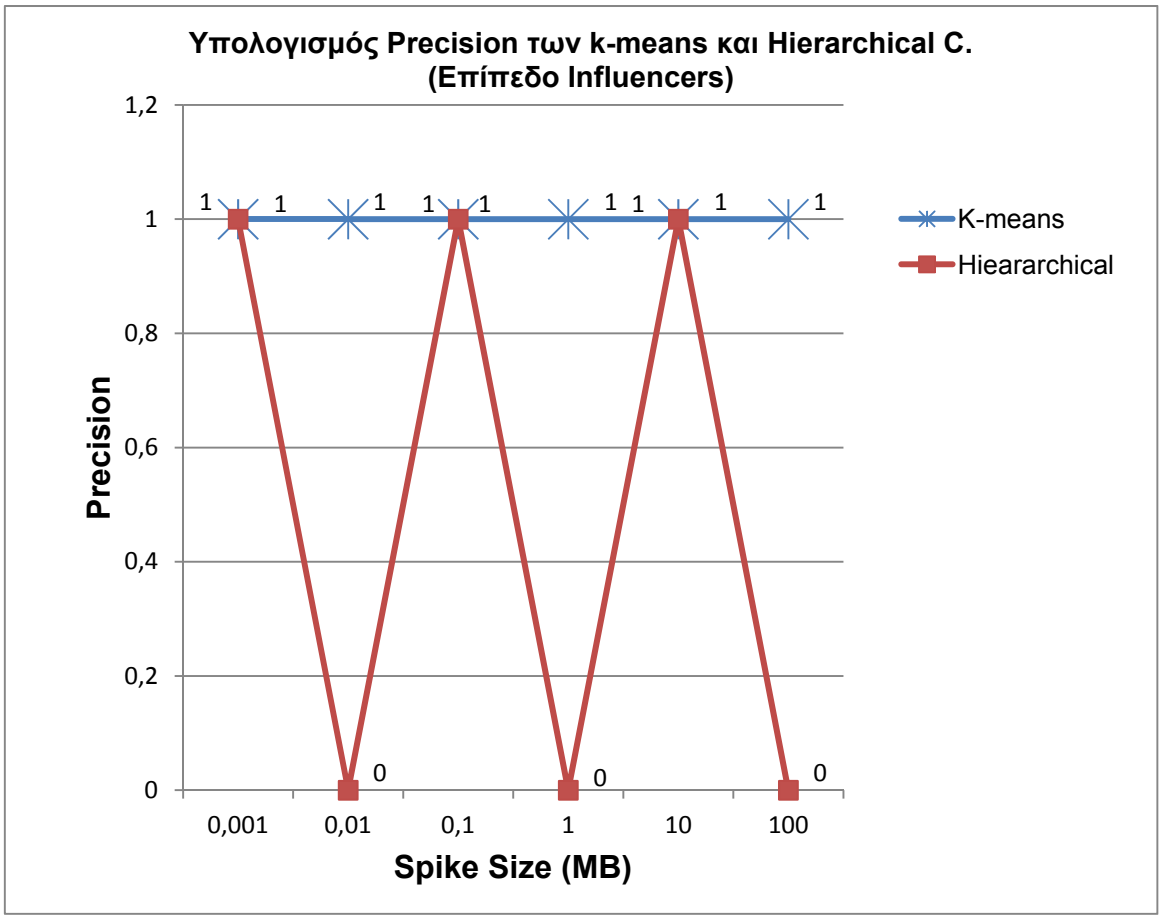
Σχήμα 4: Υπολογισμός των precision και recall εκτελώντας την ιεραρχική ομαδοποίηση

Παρατηρούμε πως σε τρεις από τις συνολικές έξι περιπτώσεις, η ακρίβεια είναι ίση με μονάδα. Στις υπόλοιπες περιπτώσεις, το precision είναι ίσο με το πηλίκο $\frac{0}{0}$ καθώς ο αριθμός των influencer γραμμών είναι ίσος με μηδέν, συνεπώς ο αριθμός των γραμμών που αντιστοιχούν σε memory spikes είναι επίσης ίσος με μηδέν. Για λόγους απλούστευσης του σχεδιασμού της γραφικής παράστασης του precision, θα θεωρούμε μια τέτοια περίπτωση να έχει μηδενικό αποτέλεσμα.



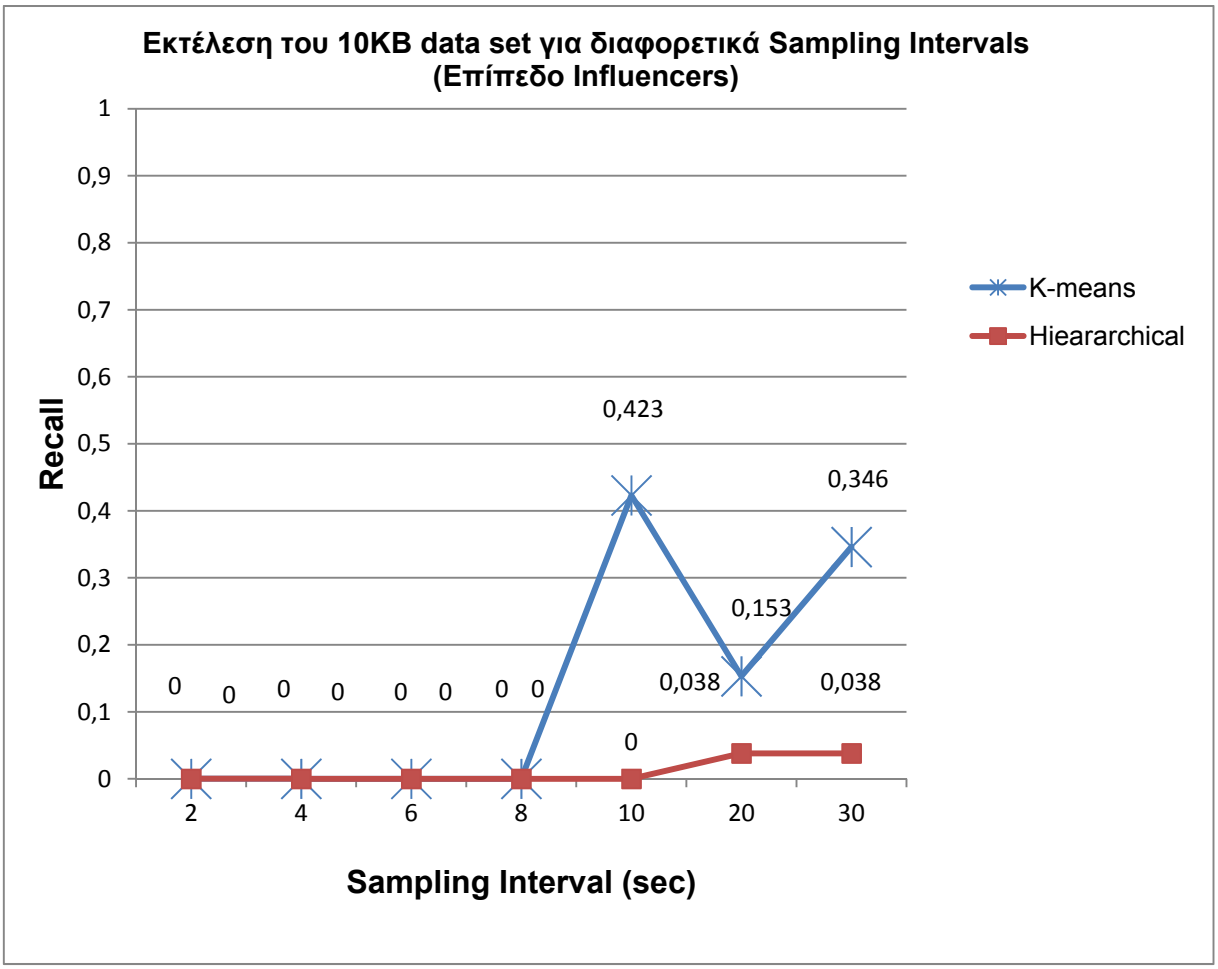
Σχήμα 5: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως στις τέσσερις από τις έξι περιπτώσεις, ο k-means αλγόριθμος παρουσιάζει καλυτέρα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση. Ο συνολικός αριθμός των memory spikes που βρίσκονται μετά το πέρας της εκτέλεσης του k-means αλγορίθμου, είναι μεγαλύτερος (εκτός των 100KB και 10MB data set) σε σχέση μ' αυτόν της ιεραρχικής ομαδοποίησης.



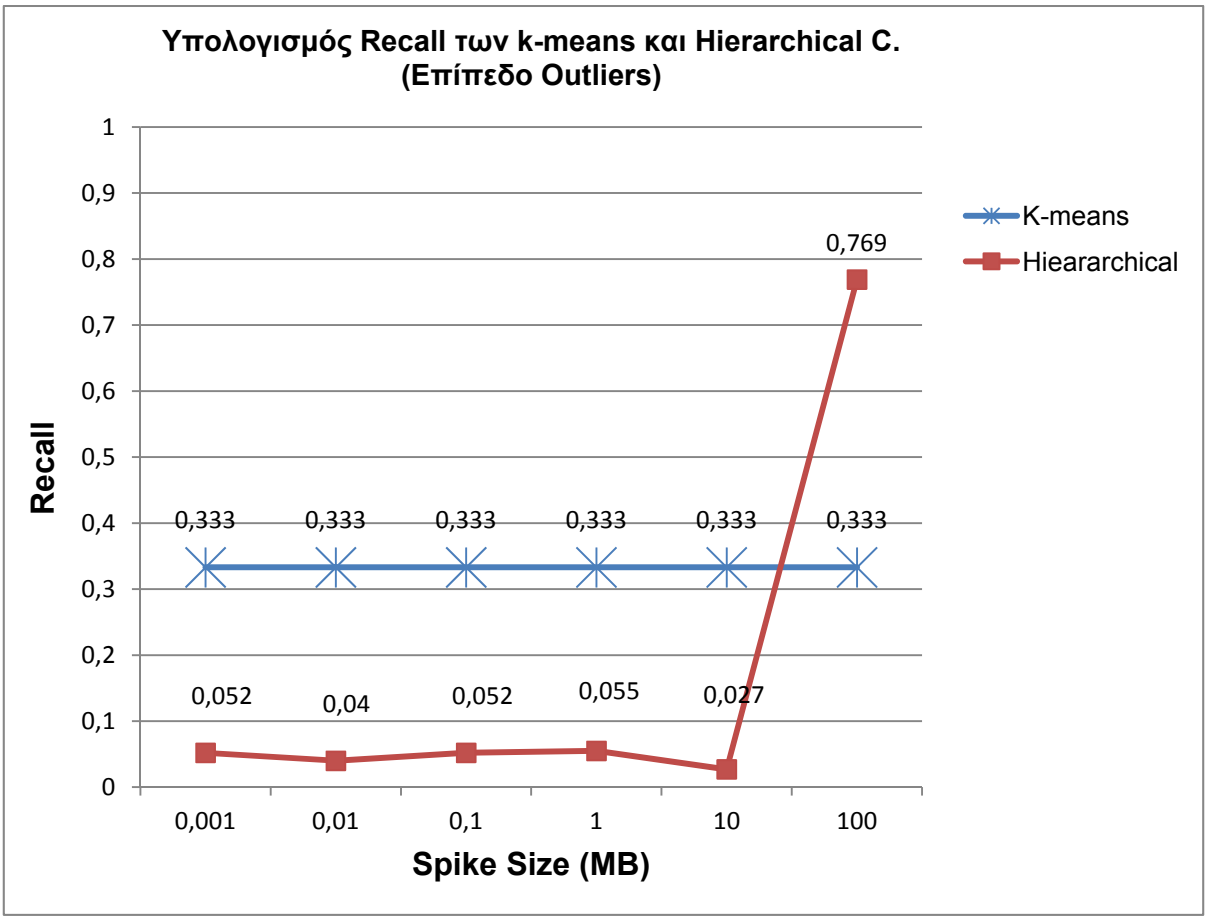
Σχήμα 6: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως σε όλες τις περιπτώσεις, ο k-means αλγόριθμος έχει precision ίσο με τη μονάδα που σημαίνει ότι είναι πιο αξιόπιστος όσον αφορά την εύρεση των influencer γραμμών. Αντίθετα, η ιεραρχική ομαδοποίηση έχει απόδοση ίση με την μονάδα στο 50% των σετ δεδομένων εκτέλεσης.



Σχήμα 7: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση για το 10KB data set με εφαρμογή διαφορετικών χρόνων δειγματοληψίας

Εκτελούμε το 10KB σετ δεδομένων εφαρμόζοντας κάθε φορά διαφορετικό χρόνο δειγματοληψίας (2, 4, 6, 8, 10, 20 και 30 δευτερόλεπτα) για τις περιπτώσεις του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης. Στους χρόνους 2 – 8 sec, καμία από τις δυο μεθόδους δε βρίσκει κάποιο memory spike. Αντιθέτως στις υπόλοιπες τρεις εκτελέσεις, το recall που παρουσιάζει ο k-means είναι καλύτερο σε σχέση με το αντίστοιχο της ιεραρχικής ομαδοποίησης.



Σχήμα 8: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως στις πέντε από τις έξι περιπτώσεις, ο k-means αλγόριθμος παρουσιάζει καλυτέρα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση όσον αφορά το recall μέγεθος. Αυτό είναι λογικό, επειδή ο αριθμός των συστάδων που δημιουργούνται στη εφαρμογή του k-means αλγορίθμου (στην εκτέλεση αυτών των σετ δεδομένων, k = 3) είναι αρκετά μικρότερος σε σχέση με τον αντίστοιχο αριθμό συστάδων της ιεραρχικής ομαδοποίησης με αποτέλεσμα να παρουσιάζει καλύτερα αποτελέσματα σε επίπεδο outlier.

4.3 Memory Spike Weight Case Study

Εφαρμόζουμε τον k-means αλγόριθμο και την ιεραρχική ομαδοποίηση σε έξι (6) διαφορετικές περιπτώσεις σετ δεδομένων τα οποία περιλαμβάνουν memory spikes. Κάθε σετ δεδομένων περιλαμβάνει ένα διαφορετικό αριθμό από memory spikes τα των οποίων η συχνότητα εξαρτάται από μια παράμετρο W. Η παράμετρος W (weight) στα τεστ δεδομένων, υποδεικνύει το βάρος των κανονικών τύπων εκδήλωσης: η εκδήλωση ενός προβλήματος που σχετίζεται με την μνήμη, εμφανίζεται μία φορά για κάθε W φορές που συμβαίνει ένα φυσιολογικό γεγονός. Πιο συγκεκριμένα έχουμε τις εξής περιπτώσεις σετ δεδομένων:

- 1W Memory spike data set
- 2W Memory spike data set

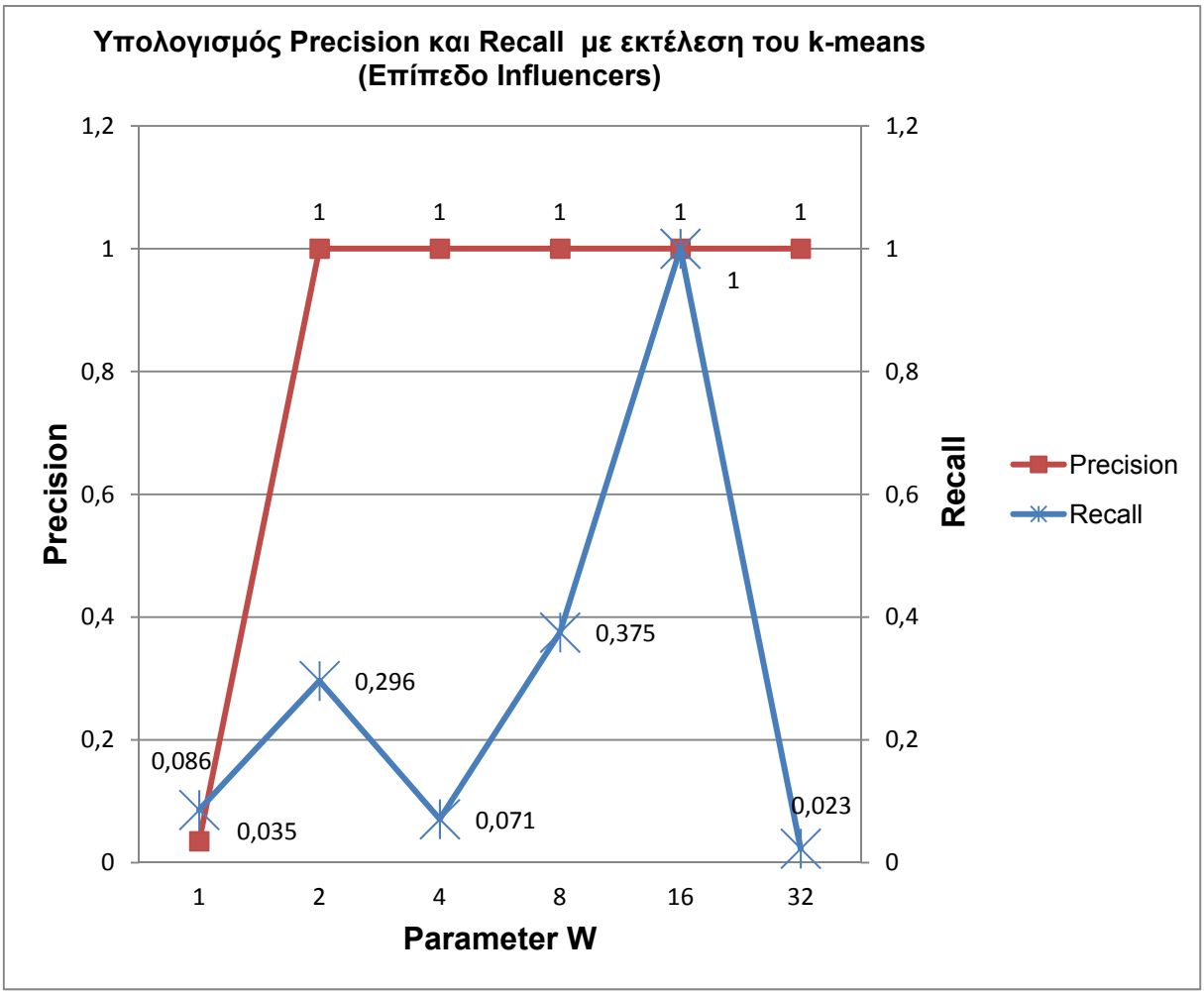
- 4W Memory spike data set
- 8W Memory spike data set
- 16W Memory spike data set
- 32W Memory spike data set

Η παράμετρος W στα τεστ δεδομένων υποδεικνύει το βάρος των φυσιολογικών γεγονότων: η εκδήλωση ενός προβλήματος που σχετίζεται με την μνήμη, εμφανίζεται μία φορά για κάθε W φορές που συμβαίνει ένα φυσιολογικό γεγονός (δηλαδή ένα γεγονός το οποίο δεν αποτελεί κάποιο memory-related θέμα).

Παρακάτω παρουσιάζεται ο πινάκας με τα memory spikes που έχουμε εισάγει σε κάθε ένα από τα έξι σετ δεδομένων:

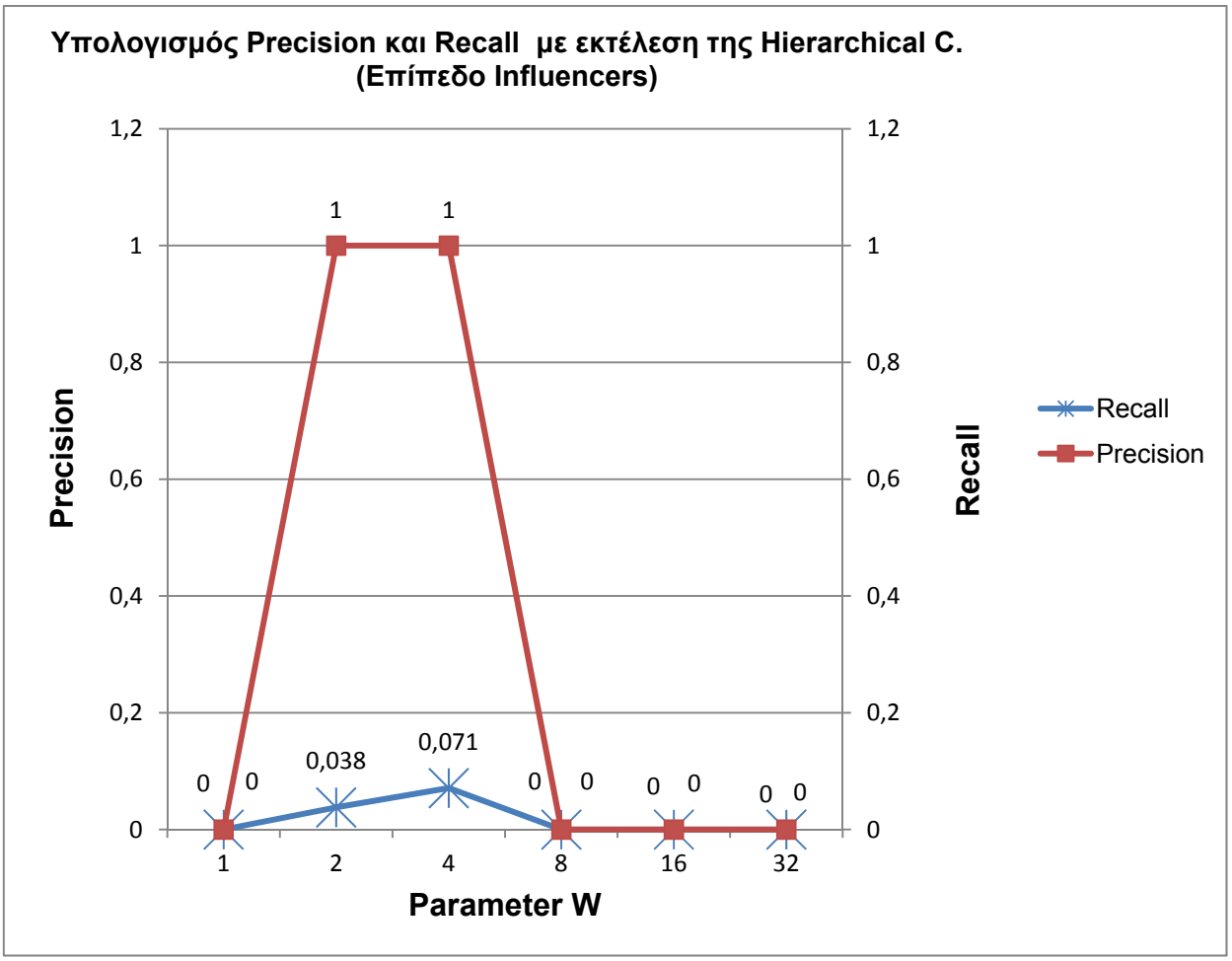
Πίνακας 2: Αριθμός των memory spikes στα W τεστ δεδομένων

Data set ID	**1W**	**2W**	**4W**	**8W**	**16W**	**32W**
Number of memory spikes	46	27	14	8	3	2



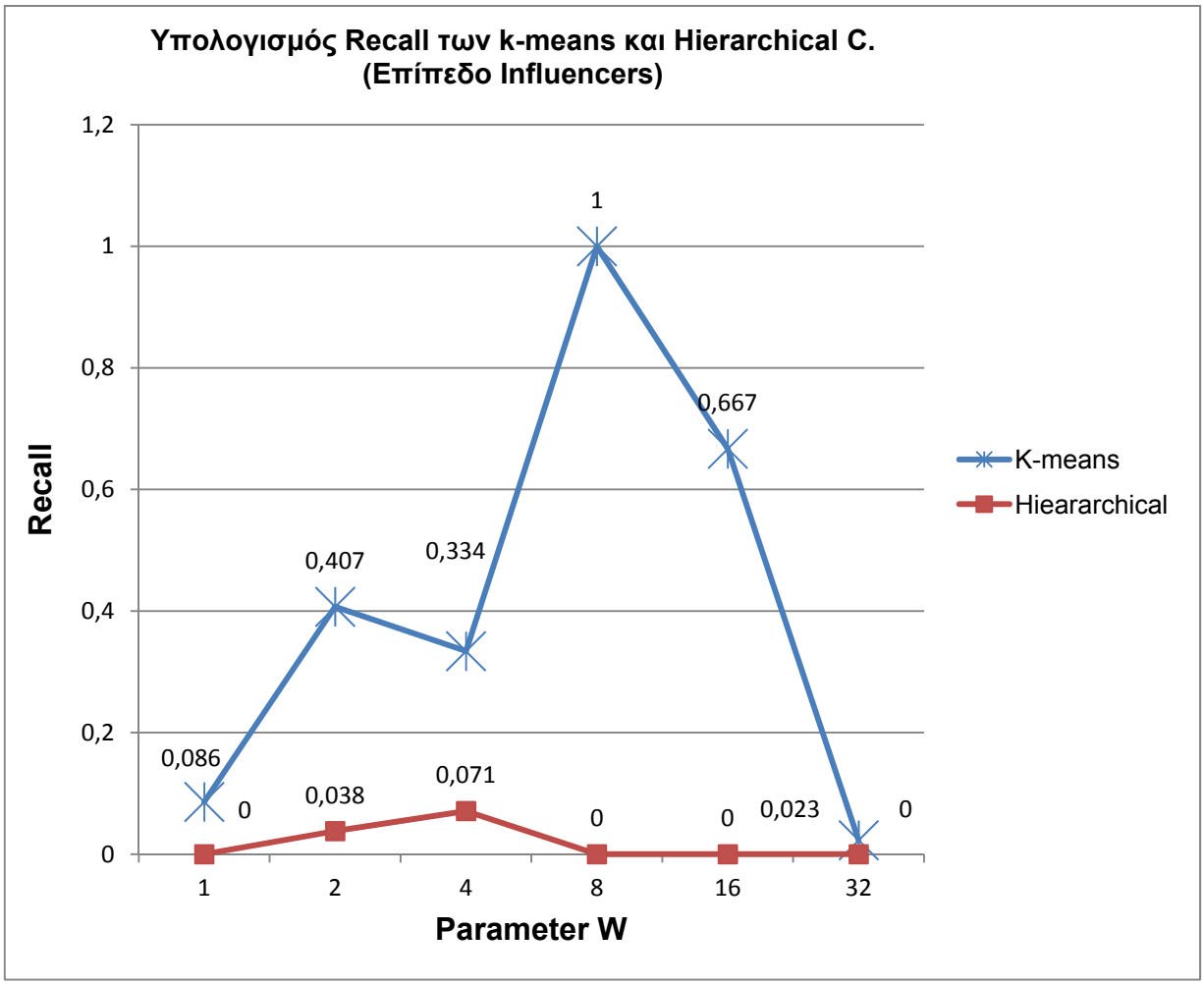
Σχήμα 9: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο

Παρατηρούμε πως σε όλες τις περιπτώσεις (εκτός για W = 1), η ακρίβεια είναι ίση με μονάδα, που σημαίνει ότι οποιαδήποτε influencer γραμμή βρίσκουμε, αντιστοιχεί σε περίπτωση ενός memory spike. Το recall μέγεθος δείχνει τον ποσοστό των συνολικών memory spikes που βρέθηκαν μετά το πέρας της εκτέλεσης του αλγορίθμου.



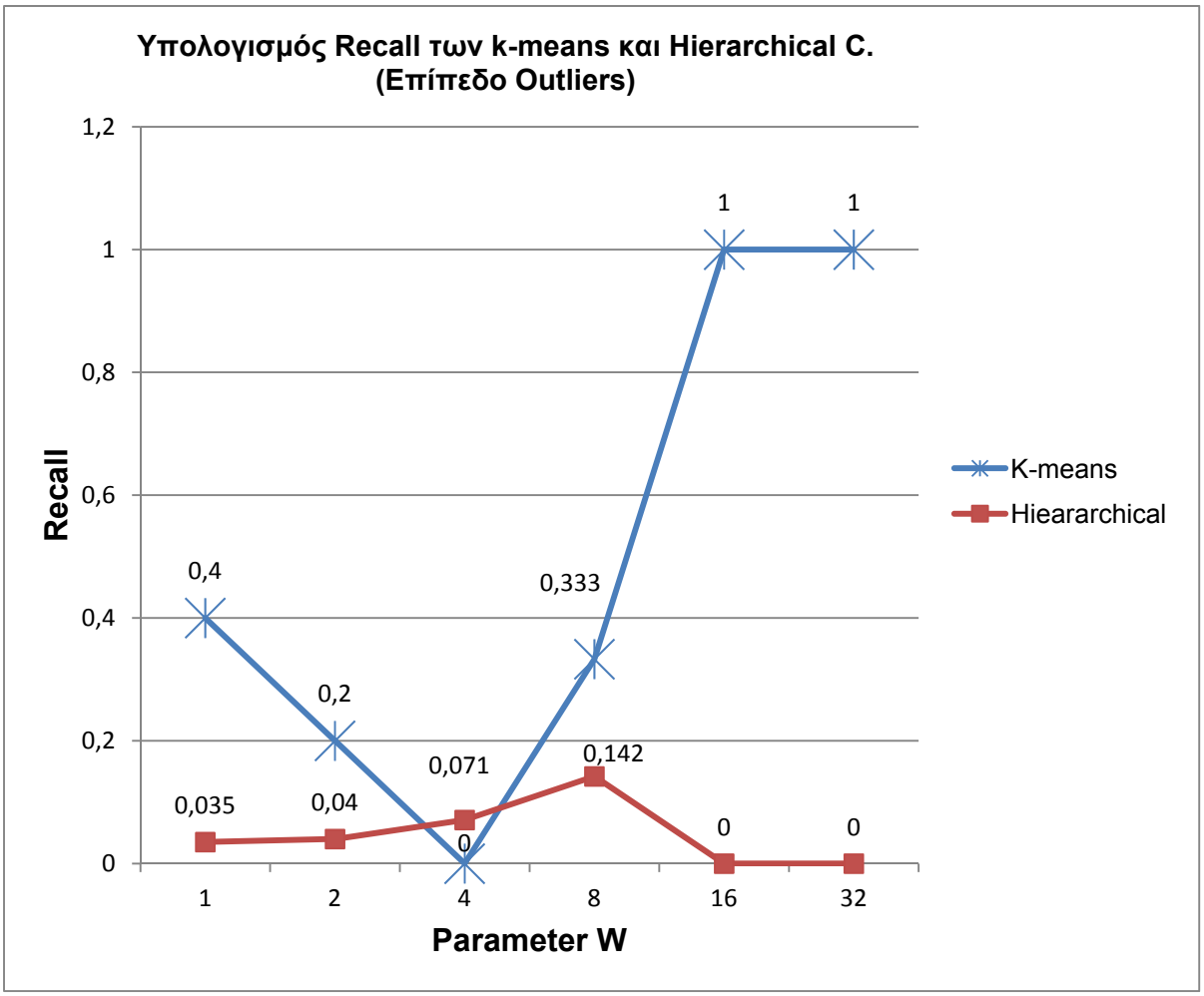
Σχήμα 10: Υπολογισμός των precision και recall εκτελώντας την ιεραρχική ομαδοποίηση

Παρατηρούμε πως μόλις σε δύο από τις συνολικές έξι περιπτώσεις, η ακρίβεια είναι ίση με μονάδα. Στις υπόλοιπες περιπτώσεις, το precision είναι ίσο με το πηλίκο $\frac{0}{0}$ καθώς ο αριθμός των influencer γραμμών είναι ίσος με μηδέν, συνεπώς ο αριθμός των γραμμών που αντιστοιχεί σε spikes είναι επίσης ίσος με μηδέν. Για λόγους απλούστευσης του σχεδιασμού της γραφικής παράστασης του precision, θα θεωρούμε μια τέτοια περίπτωση να έχει μηδενικό αποτέλεσμα.



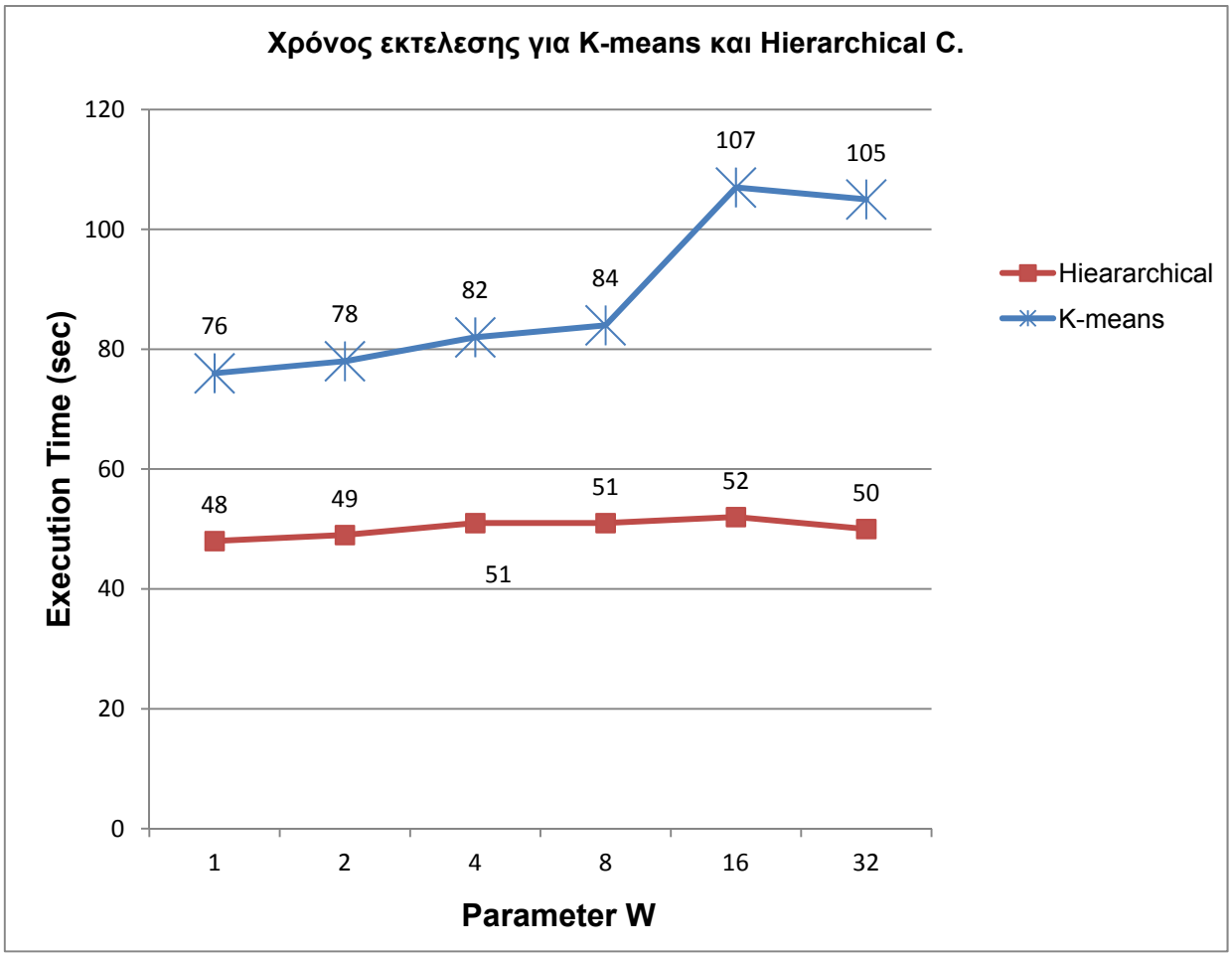
Σχήμα 11: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως σε όλες τις περιπτώσεις, ο k-means αλγόριθμος παρουσιάζει καλυτέρα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση. Ο συνολικός αριθμός των memory spikes που βρίσκονται μετά το πέρας της εκτέλεσης του k-means αλγορίθμου, είναι μεγαλύτερος σε σχέση μ' αυτόν της ιεραρχικής ομαδοποίησης.



Σχήμα 12: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως στις πέντε από τις έξι περιπτώσεις, ο k-means αλγόριθμος παρουσιάζει καλύτερα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση όσον αφορά το recall μέγεθος. Αυτό είναι λογικό, επειδή ο αριθμός των συστάδων που δημιουργούνται στη εφαρμογή του k-means αλγορίθμου (στην εκτέλεση αυτών των σετ δεδομένων, k = 3 έως 5) είναι αρκετά μικρότερος σε σχέση με τον αντίστοιχο αριθμό συστάδων της ιεραρχικής ομαδοποίησης με αποτέλεσμα να παρουσιάζει καλυτέρα αποτελέσματα σε επίπεδο outlier.



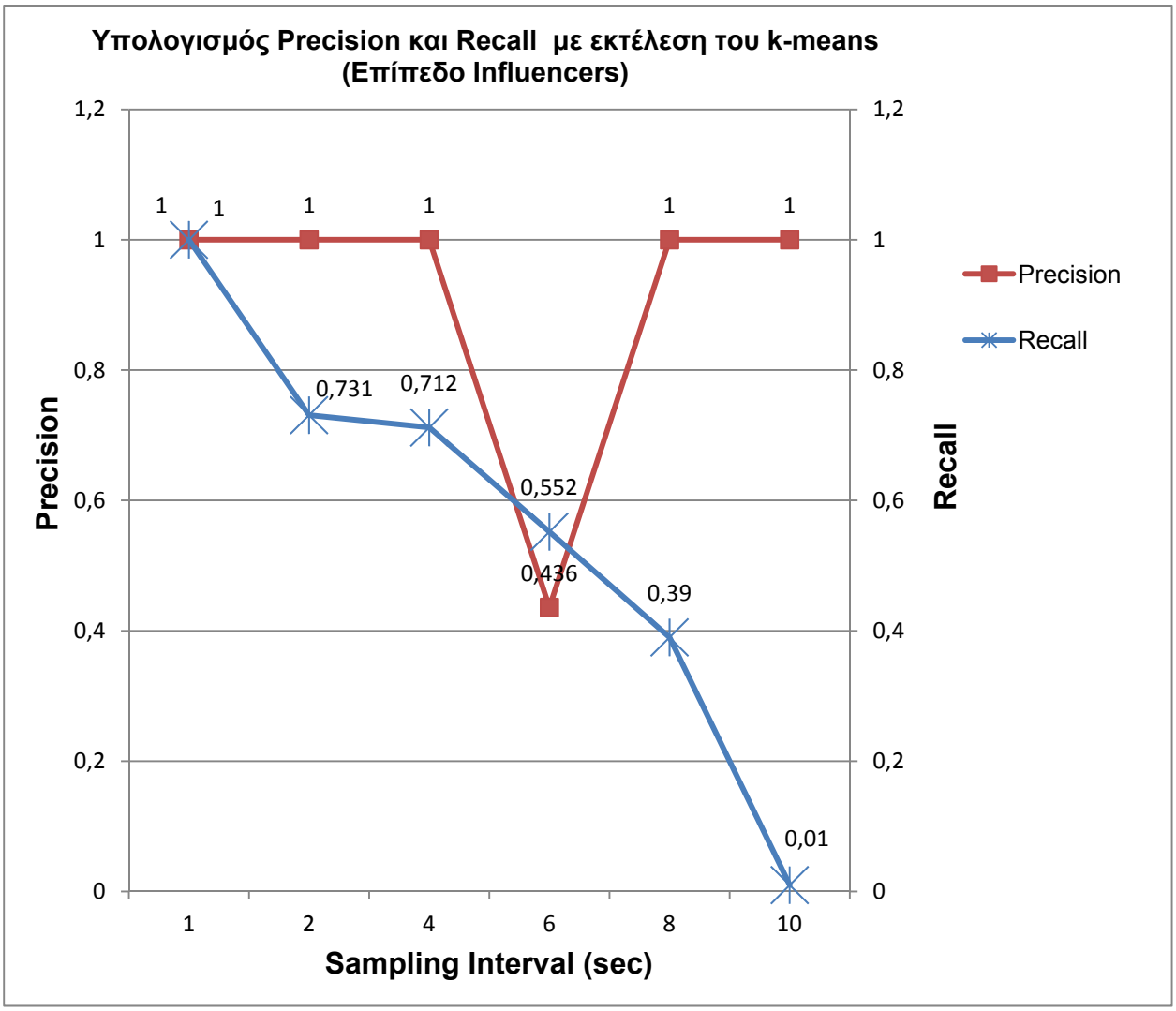
Σχήμα 13: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως σε όλα τα σετ δεδομένων αυτά, ο χρόνος εκτέλεσης του k-means αλγόριθμου είναι σχετικά πιο μεγάλος από τον αντίστοιχο χρόνο εκτέλεσης της ιεραρχικής ομαδοποίησης ο οποίος είναι σταθερός (48 – 52 δευτερόλεπτα). Τα σετ δεδομένων αυτά είναι μικρά σε μέγεθος. Στόχος μας είναι να ελέγξουμε τους χρόνους εκτέλεσης ενός μεγαλύτερου σετ δεδομένων ώστε να σχηματίσουμε μια σφαιρική άποψη για τους συνολικούς χρόνους εκτέλεσης των δυο αλγορίθμων συσταδοποίησης που εφαρμόζουμε.

4.4 Stress Test Case Study

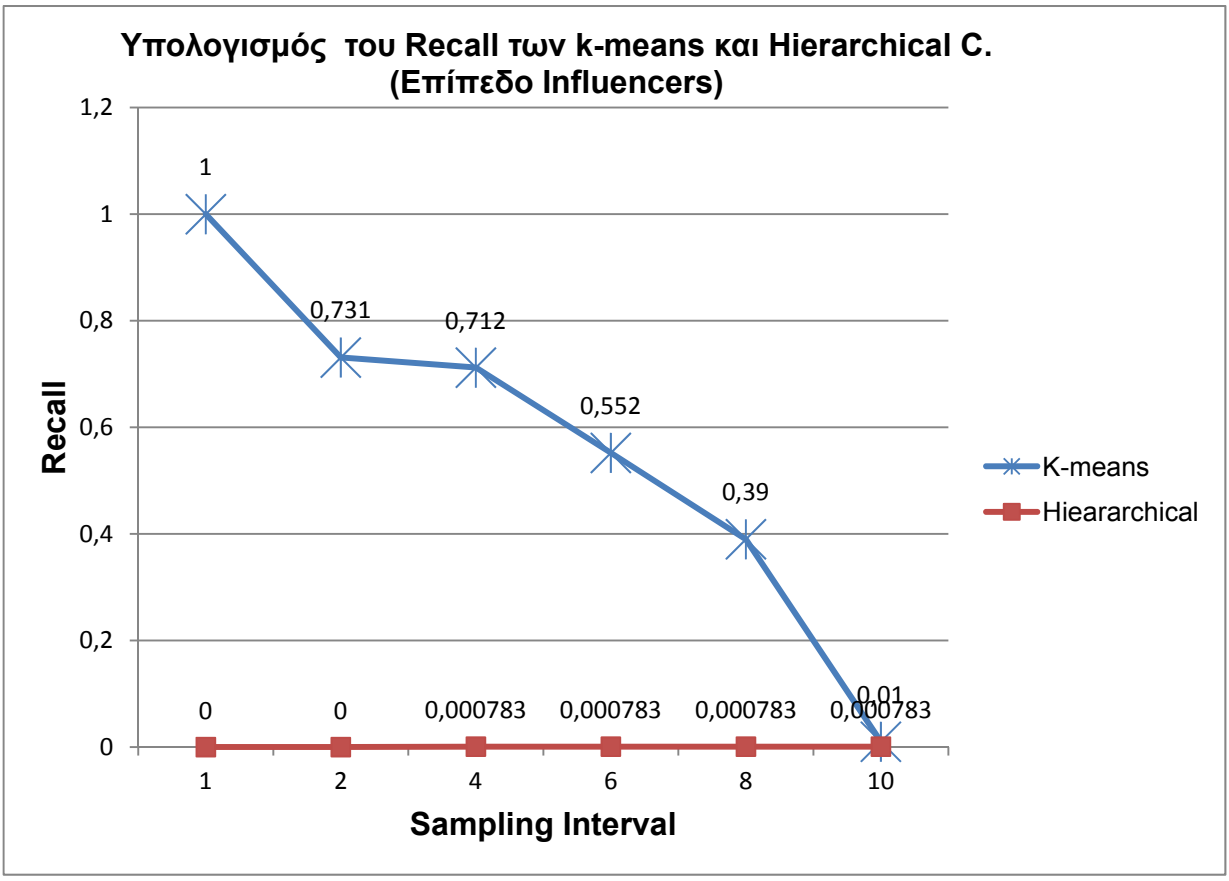
Στην ανάλυση των αποτελεσμάτων του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης, πραγματοποιούμε stress testing σε ένα σετ δεδομένων επίπεδου Big Data με σκοπό να βρούμε κάποια λειτουργικά στοιχεία (π.χ. απόδοση, χρόνος εκτέλεσης κ.α.) των δυο μεθόδων σε μεγαλύτερα δεδομένα. Κύριος στόχος μας είναι να ελέγξουμε την λειτουργία του αλγορίθμου στις περιπτώσεις (δηλαδή εφαρμόζοντας τα σετ δεδομένων) που η ιεραρχική ομαδοποίηση αποτυγχάνει λόγω μεγάλης πολυπλοκότητας αδυνατώντας να δώσει κάποιο αποτέλεσμα.

Το load test αυτό περιλαμβάνει συνολικά 1277 events τα οποία σχετίζονται με την πρόκληση ενός memory-related θέματος (πιο συγκεκριμένα ένα memory spike). Εκτελούμε το σετ δεδομένων για διαφορετικούς χρόνους δειγματοληψίας (1, 2, 4, 6, 8, 10 δευτερόλεπτα) με στόχο να ελέγξουμε την λειτουργία των δυο μεθόδων κάτω από μεγαλύτερη πίεση. Όσο μειώνεται ο χρόνος δειγματοληψίας, παράγονται από τον αλγόριθμο περισσότερα time-slice προφίλ με αποτέλεσμα την αύξηση της πολυπλοκότητας. Τα αποτελέσματα των δυο αλγορίθμων που εφαρμόστηκαν στο σετ δεδομένων αυτό παρατίθενται παρακάτω:



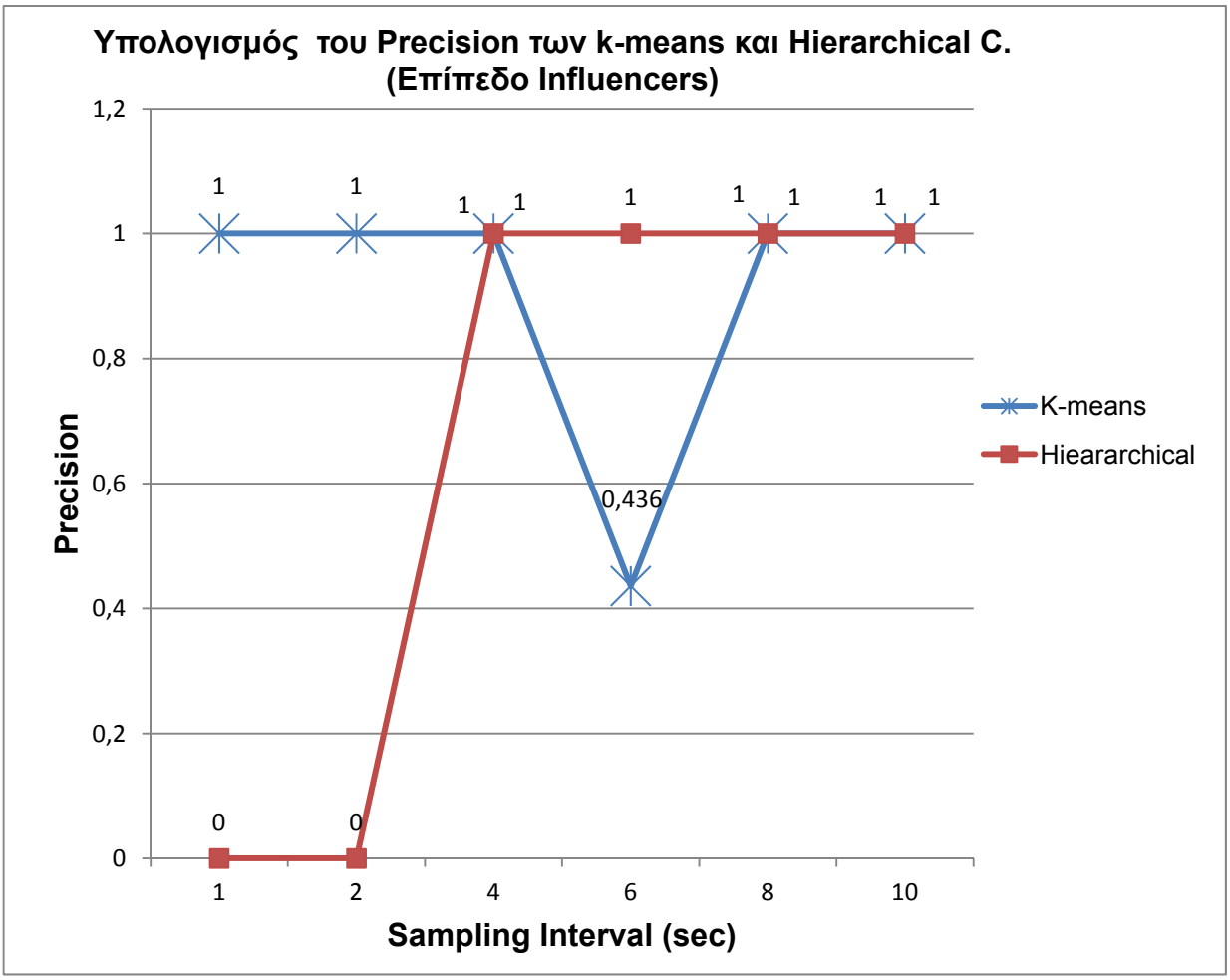
Σχήμα 14: Υπολογισμός των precision και recall εκτελώντας τον k-means αλγόριθμο

Παρατηρούμε πως σε όλες τι δειγματοληψίες εκτός μιας περίπτωσης, η ακρίβεια είναι ίση με μονάδα, που σημαίνει ότι οποιαδήποτε influencer γραμμή βρίσκουμε, αντιστοιχεί σε περίπτωση ενός memory spike. Το recall μέγεθος δείχνει το ποσοστό των συνολικών memory spikes που βρέθηκαν μετά το πέρας της εκτέλεσης του αλγορίθμου. Μια γενική παρατήρηση είναι ότι όσο μεγαλώνει ο χρόνος δειγματοληψίας, τόσα λιγότερα memory spikes ανιχνεύει ο k-means αλγόριθμος.



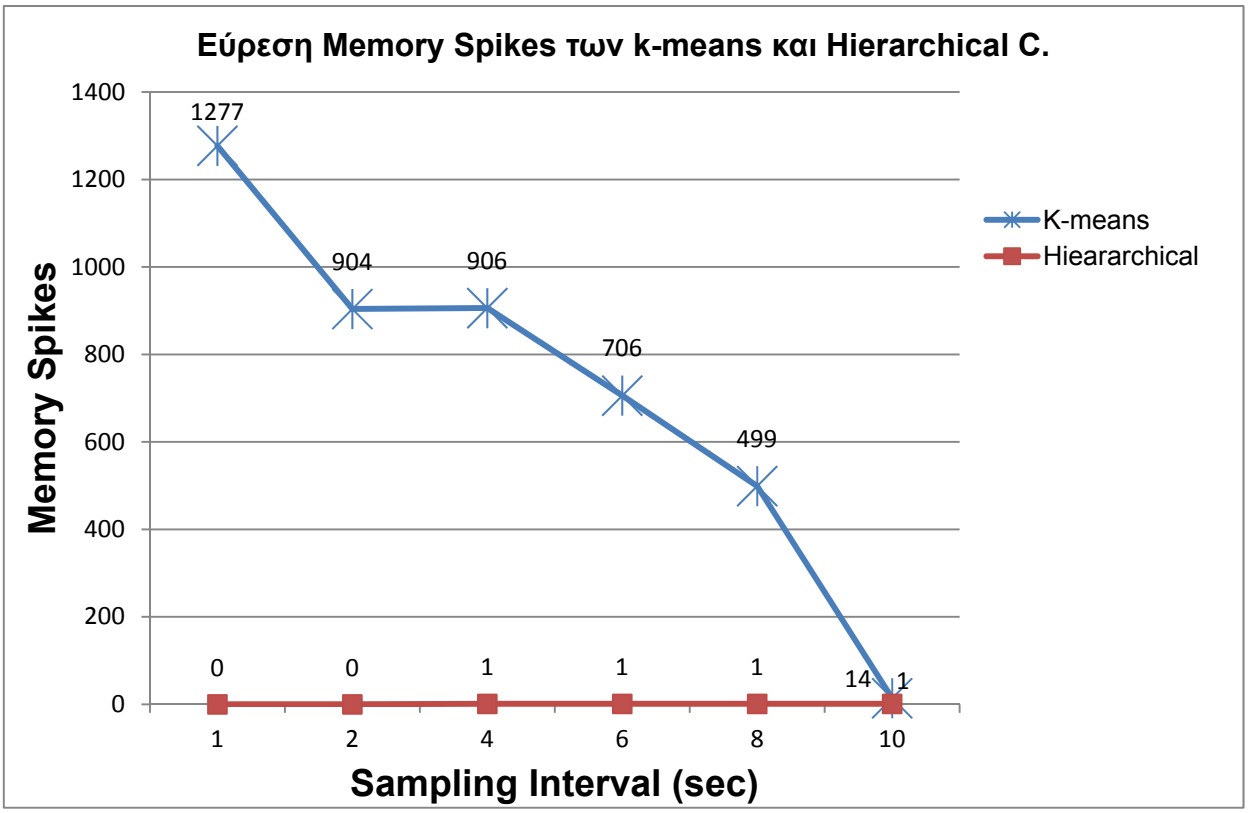
Σχήμα 15: Υπολογισμός του recall εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως σε όλες τις περιπτώσεις εκτέλεσης, ο k-means αλγόριθμος παρουσιάζει καλυτέρα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση. Για ένα και δύο δευτερόλεπτα χρόνους δειγματοληψίας, η ιεραρχική μέθοδος αποτυγχάνει, συνεπώς το recall είναι ίσο με τη μονάδα. Αντίθετα, ο k-means αποδίδει πολύ καλά έχοντας στους περισσότερους χρόνους δειγματοληψίας απόδοση μεγαλύτερη του 50%.



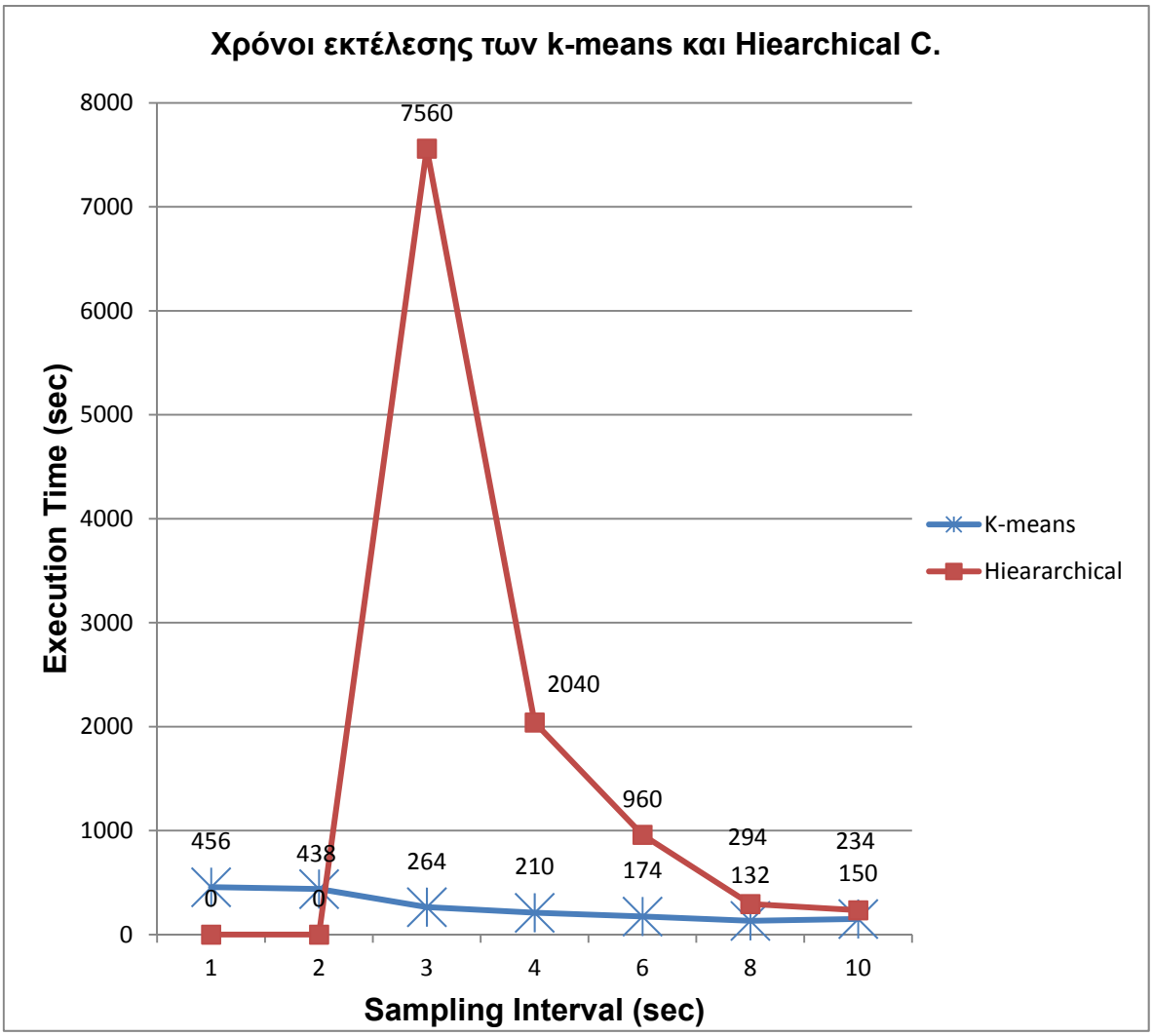
Σχήμα 16: Υπολογισμός του precision εκτελώντας τον αλγόριθμο k-means και την ιεραρχική ομαδοποίηση

Παρατηρούμε πως στις περισσότερες περιπτώσεις εκτέλεσης, ο k-means αλγόριθμος παρουσιάζει καλυτέρα αποτελέσματα σε σχέση με την ιεραρχική ομαδοποίηση. Για ένα και δύο δευτερόλεπτα χρόνους δειγματοληψίας, η ιεραρχική μέθοδος αποτυγχάνει, συνεπώς το precision όπως και τα υπόλοιπα στοιχεία εξόδου που δίνει ο αλγόριθμος δεν είναι δυνατό να βρεθούν. Αντίθετα, ο k-means αποδίδει πολύ καλά σ' αυτές τις δυο περιπτώσεις εκτέλεσης έχοντας ακρίβεια ίση με τη μονάδα.



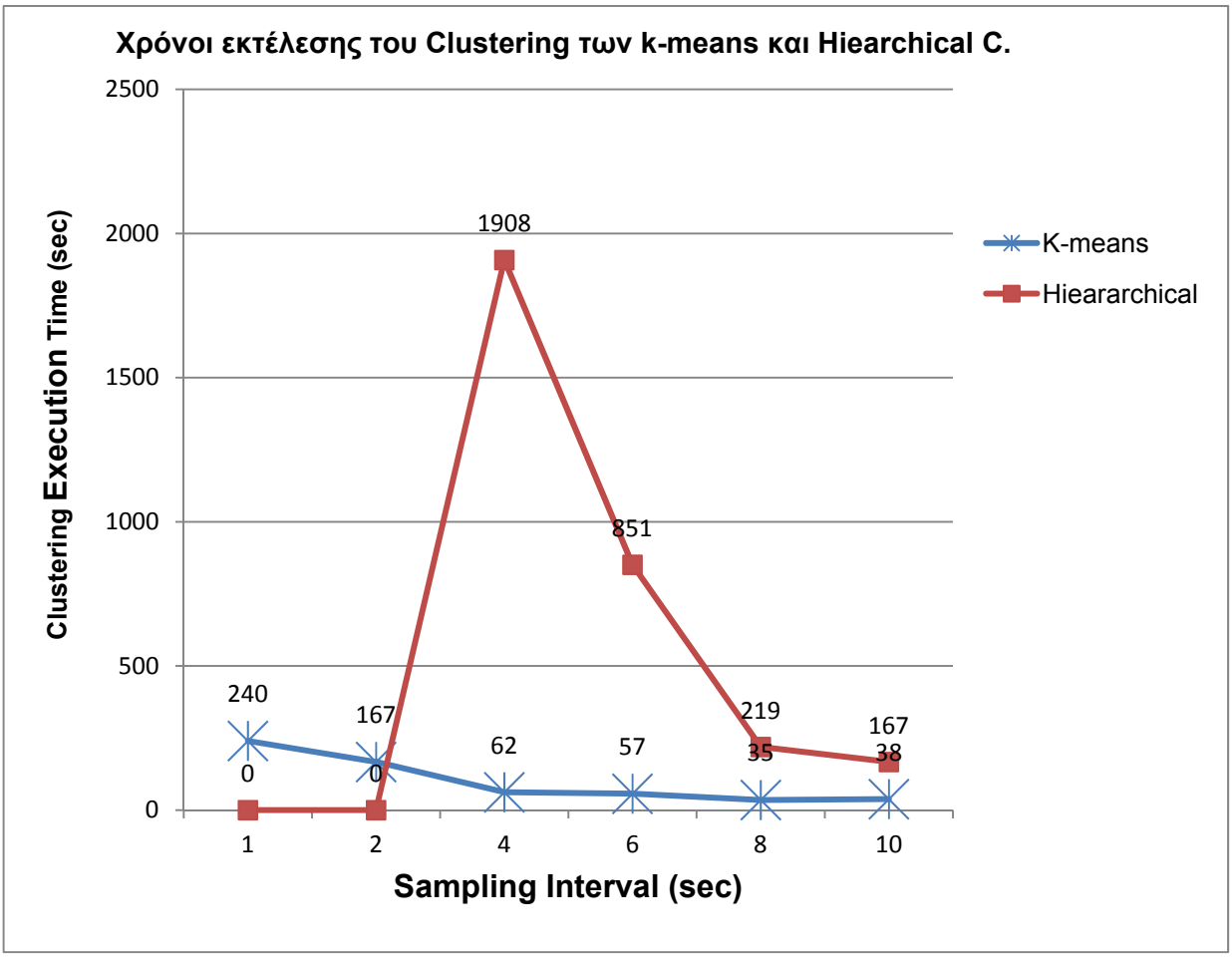
Σχήμα 17: Εύρεση των memory spikes του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης

Παρατηρούμε πως σε όλες τι δειγματοληψίες ο k-means αλγόριθμος αποδίδει πολύ καλυτέρα σε σχέση με την ιεραρχική ομαδοποίηση η οποία εντοπίζει μόνο ένα memory spike στους μεγαλύτερους χρόνους δειγματοληψίας. Αντίθετα, ο k-means παρόλο που έχει μια καθοδική πορεία όσο μεγαλώνει το διάστημα δειγματοληψίας, παρουσιάζει μια εξαιρετική διάγνωση: στην περίπτωση που το sampling interval ισούται με ένα (1) δευτερόλεπτο, εντοπίζονται και τα 1277 memory spikes που περιέχονται στο σετ δεδομένων που εκτελείται.



Σχήμα 18: Χρόνοι εκτέλεσης του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης

Παρατηρούμε πως σε όλες τις δειγματοληψίες ο χρόνος εκτέλεσης του k-means αλγορίθμου είναι πολύ μικρότερος σε σχέση με τον αντίστοιχο χρόνο εκτέλεσης της ιεραρχικής ομαδοποίησης. Παρόλο που ο k-means σε μικρότερα σετ δεδομένων (όπως το προηγούμενο τεστ εκτέλεσης) είναι ελάχιστα πιο αργός, σε περιπτώσεις μεγαλυτέρων σετ δεδομένων, ο χρόνος εκτέλεσης του k-means διαφέρει σε μεγάλο βαθμό ως προς τον αντίστοιχο της ιεραρχικής ομαδοποίησης. Αυτό δικαιολογείται διότι για κατασκευή των συστάδων (στο στάδιο του time-slice profiling), η ιεραρχική μέθοδος απαιτεί τον υπολογισμό του πίνακα αποστάσεων ο οποίος έχει πολυπλοκότητα $O(n^2)$. Συνεπώς, όσο μικραίνει το διάστημα δειγματοληψίας, μεγαλώνει ο αριθμός και ο χρόνος του υπολογισμού της μεταξύ απόστασης των time-slice προφίλ που θα δημιουργηθούν και εξαιτίας αυτού, αυξάνεται ο χρόνος εκτέλεσης του αλγορίθμου.



Σχήμα 19: Χρόνοι εκτέλεσης της φάσης συσταδοποίησης του k-means αλγορίθμου και της ιεραρχικής ομαδοποίησης

Παρατηρούμε πως σε όλες τι δειγματοληψίες ο χρόνος εκτέλεσης του σταδίου της συσταδοποίησης του k-means αλγορίθμου είναι μικρότερος σε σχέση με τον αντίστοιχο χρόνο εκτέλεσης της ιεραρχικής ομαδοποίησης. Αυτό δικαιολογείται διότι για κατασκευή των συστάδων (στο στάδιο του time-slice profiling), η ιεραρχική μέθοδος απαιτεί τον υπολογισμό του πίνακα αποστάσεων ο οποίος έχει πολυπλοκότητα $O(n^2)$. Συνεπώς, όσο μικραίνει το διάστημα δειγματοληψίας, μεγαλώνει ο αριθμός και ο χρόνος του υπολογισμού της μεταξύ απόστασης των time-slice προφίλ που θα δημιουργηθούν και εξαιτίας αυτού, αυξάνεται ο χρόνος εκτέλεσης του clustering. Αντίθετα, ο k-means δε απαιτεί τον υπολογισμό ενός πίνακα αποστάσεων των time-slice προφίλ για να κατασκευάσει τα clusters. Μια άλλη αξιοσημείωτη διαφορά είναι, ότι στην ιεραρχική ομαδοποίηση, κατασκευάζεται ένας μεγάλος αριθμός από clusters σε αντίθεση με την εκτέλεση του k-means αλγορίθμου που ο αριθμός των συστάδων είναι αρκετά μικρότερος.

4.5 Apache Tomcat Real Case Study

Η προσέγγιση μας, χρησιμοποιεί μετρητές απόδοσης και αρχεία εκτέλεσης από μία Apache Tomcat εφαρμογή. Το Apache Tomcat project αποτελεί μια υλοποίηση ανοιχτού κώδικα του Java Servlet, των JavaServer Pages (JSP), Java Expression

Language και μιας προδιαγραφής του Java Websocket και παρουσιάζεται στο GitHub (https://github.com/apache/tomcat). Χρησιμοποιούμε τον trunk branch.

Τα WebSockets είναι αμφίδρομες, επίμονες συνδέσεις προερχόμενες από ένα πρόγραμμα περιήγησης στο Web σε ένα διακομιστή [97]. Μόλις καθιερωθεί μια WebSocket σύνδεση, η σύνδεση παραμένει ανοιχτή μέχρι ο πελάτης ή ο διακομιστής αποφασίσει να τη κλείσει. Με αυτή την ανοικτή σύνδεση, ο πελάτης ή ο διακομιστής μπορεί να στείλει ένα μήνυμα κάθε δεδομένη στιγμή στον άλλον. Μια ενιαία αίτηση διακομιστή η οποία τρέχει, γνωρίζει όλες τις συνδέσεις, επιτρέποντάς του να επικοινωνεί με οποιοδήποτε αριθμό ανοιχτών συνδέσεων σε κάθε δεδομένη στιγμή.

4.5.1 Διαδικασία εκχώρησης ενός Memory Spike

Αρχικά, ενσωματώνουμε ένα memory spike συνολικού μεγέθους 100 MB μέσα στον πηγαίο κώδικα (source code), ο οποίος εκτελείται κάθε φορά που ένας χρήστης εγκαθιδρύει μια νέα websocket συνέδρια (session). Ειδικότερα, εντοπίζουμε μέσα στον πηγαίο κώδικα τη δήλωση εξόδου (output statement) η οποία εκτυπώνει την γραμμή που εμφανίζεται στο log εκτέλεσης κατά τη διάρκεια δημιουργίας μιας websocket συνεδρίας.

Πιο συγκεκριμένα, στον φάκελο tomcat/java/org/apache/tomcat/websocket, βρίσκουμε το αρχείο WsSession.java και στην συνάρτηση public getContainer() η οποία επιστρέφει έναν νέο WebSocketContainer, προσθέτουμε την έξης δήλωση κάτω από αυτό:

final byte[] spike = new byte[1000000000]; που αποτελεί ένα memory spike συνολικού μεγέθους 100 MB.

Κατασκευάζουμε το Apache Tomcat [94], καθορίζουμε το log level σε FINE και εκκινούμε την εκτέλεση του. Επιπλέον, τρέχουμε την οθόνη παρακολούθησης μετρητών απόδοσης (performance counter monitor) χρησιμοποιώντας τον JMX πάροχο στην διαδικασία του Tomcat με σκοπό να πάρουμε δείγματα του μεγέθους σωρού εφαρμόζοντας ένα διάστημα δειγματοληψίας (sampling interval) δύο συνολικά δευτερόλεπτων.

Χρησιμοποιούμε το Apache JMeter [95] μαζί μ' ένα plugin για το websocket πρωτόκολλο με σκοπό να δημιουργήσουμε φόρτο για την τρεχούμενη διαδικασία του Tomcat. Η εφαρμογή Apache JMeter [96] είναι ένα λογισμικό ανοικτού κώδικα. Αποτελεί μια καθαρή εφαρμογή Java και έχει σχεδιαστεί για να φορτώνει δοκιμές λειτουργικής συμπεριφοράς (load tests) και να μέτρα την απόδοση. Αρχικά είχε σχεδιαστεί για τη δοκιμή εφαρμογών διαδικτύου (web applications), αλλά έκτοτε έχει επεκταθεί και σε άλλες λειτουργίες ελέγχου. Τα JMeter test plan στέλνουν αιτήσεις στις εφαρμογές / παραδείγματα οι οποίες αποστέλλονται με το Apache Tomcat.

Σε κάποιο σημείο μέσα στις δόκιμες του φόρτου, παράγουμε ένα memory spike. Σταματάμε την εκτέλεση όλων (δηλαδή του load test, του monitor και του Apache Tomcat) μετά από περίπου 10 λεπτά. Κρατάμε μόνο έναν από τους φακέλους εκτέλεσης που περιέχει το Apache Tomcat, τον catalina.out log file.

Το επόμενο μας στάδιο είναι να ελέγξουμε αν τελικά εμφανίζεται κανονικά το memory spike που εισάγαμε. Στον φάκελο catalina.out, ελέγχουμε αν η παρακάτω γραμμή εκτυπώθηκε:

14-Feb-2017 10:43:23.836 FINE [http-nio-8080-exec-4]

Org.apache.tomcat.websocket.WsSession.<init> Created WebSocket session [0]

Η πλειοψηφία των γραμμών καταγραφής σ' αυτόν τον φάκελο μοιάζουν με την παρακάτω γραμμή:

14-Feb-2017 10:40:52.263 FINE[Attach Listener] sun.rmi.transport.tcp.TCPEndpoint.<clinit> Attach Listener: localHostKnown = true, localhost = 127.0.1.1

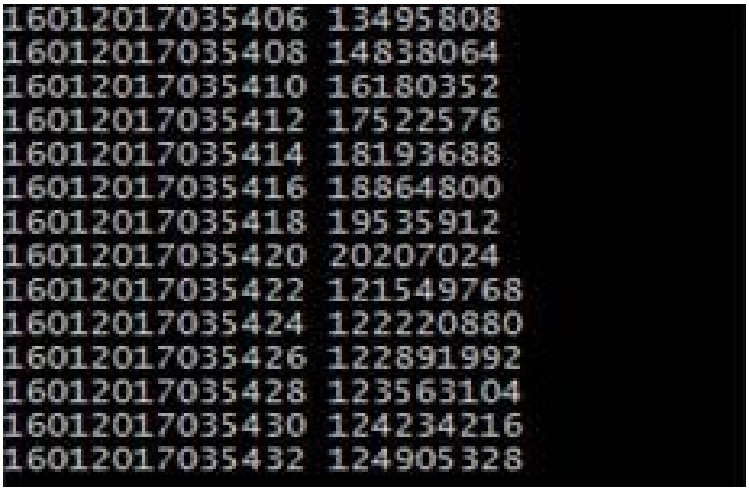
Και περιλαμβάνει τα εξής πεδία:

- ένα timestamp της μορφής dd-MMM-yyyy HH:mm:ss.SSS (*14-Feb-2017 10:40:52.263*)
- ένα log level (FINE στην περίπτωση μας)
- ένα όνομα νήματος κλεισμένο σε αγκύλες (π.χ. [Attach Listener])
- ένα πλήρως αναγνωρισμένο όνομα κλάσης ακολουθημένο από το όνομα μιας μεθόδου και χωρισμένο με τελεία
- ενα μήνυμα (Attach Listener: localHostKnown = true, localhost = 127.0.1.1)

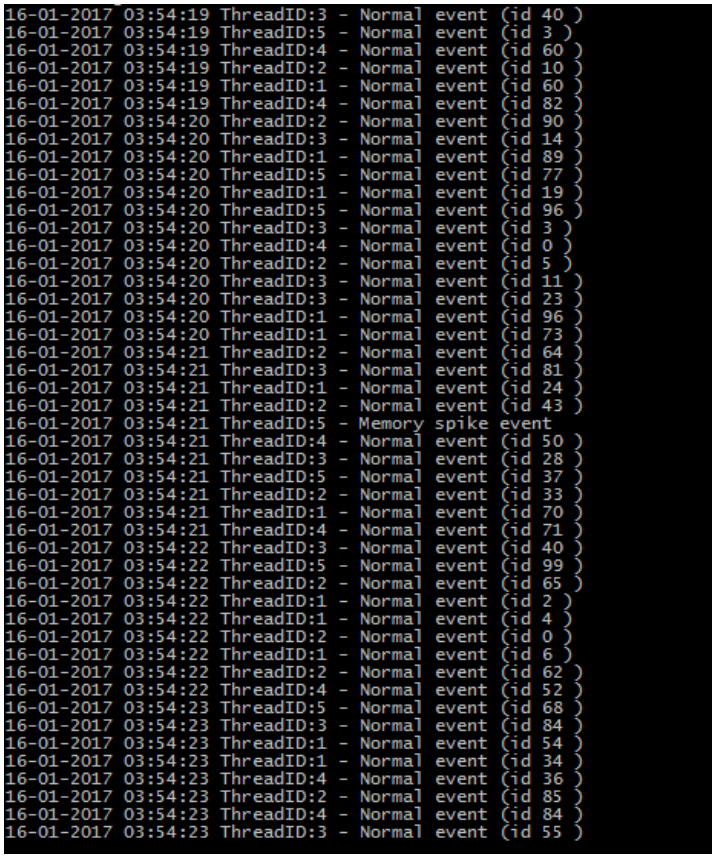
Τα παραπάνω πεδία είναι χωρισμένα με κενά, αλλά μερικά από αυτά μπορούν να περιέχουν τα ίδια κάποιο κενό (π.χ. το όνομα νήματος).

4.5.2 Αποτελέσματα της εκτέλεσης

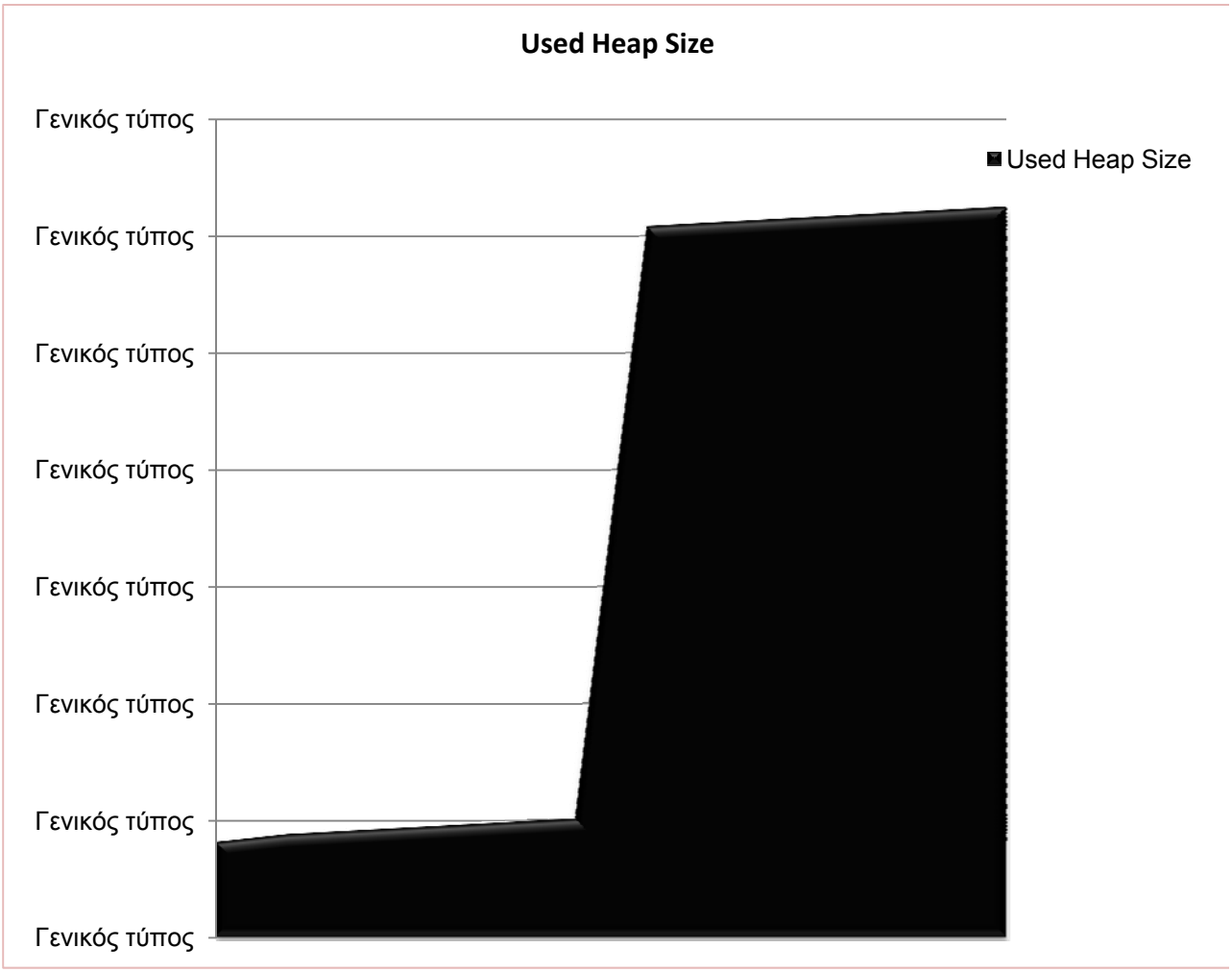
Εκτελούμε τον k-means αλγόριθμο στα αρχεία καταγραφής και τους μετρητές απόδοσης της Apache Tomcat εφαρμογής με χρόνο δειγματοληψίας ίσο με δυο (2) δευτερόλεπτα. Το αρχείο εξόδου που δεχόμαστε, περιλαμβάνει κάποιες influencer lines οι οποίες όμως δεν αντιστοιχούν σε memory spikes. Παρόλα αυτά όμως, ελέγχουμε τον μοναδικό outlier που εντοπίζει ο performance counter issue diagnoser . Ο outlier αυτός περιέχει συνολικά τρία (3) time-slice προφίλ. Τα ένα από αυτά τα προφίλ (συνολικής διάρκειας δυο δευτερόλεπτων) περιέχει το memory spike συνολικού μεγέθους 100MB που εισάγαμε προηγουμένως. Με άλλα λόγια, όλες οι φάσεις συμπεριλαμβανομένης της φάσης εντοπισμού του outlier, δείχνουν να δουλεύουν ομαλά.



Εικόνα 45: Οι μετρήσεις του used heap size που λάβαμε χάρις τους μετρητές απόδοσης



Εικόνα 46: Οι γραμμές καταγραφής αρχείου μαζί με το timestamp εκτέλεσης τους



Σχήμα 20: Το χρησιμοποιημένο μέγεθος του σωρού και το memory spike που διαγνώστηκε

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Για την αξιολόγηση χρησιμοποιήσαμε συνολικά, τρία συνθετικά σετ δεδομένων από ένα πρόγραμμα υλοποίησης της Software Competitiveness International αλλά και ένα σετ δεδομένων από την εφαρμογή του Apache Tomcat, έχοντας εισάγει ένα memory spike. Τα δυο πρώτα συνθετικά σετ δεδομένων σχετίζονται με το μέγεθος των memory spike και με την συχνότητα εμφάνισης τους αντίστοιχα. Το τρίτο σετ δεδομένων (stress test case study) που εκτελέσαμε, περιλαμβάνει έναν μεγάλο αριθμό από log lines οι οποίες αντιστοιχούν σε memory spikes. Στην εκτέλεση του, εφαρμόσαμε διαφορετικούς χρόνους δειγματοληψίας με σκοπό να ανακαλύψουμε τα όρια του κάθε αλγορίθμου.

Όσον αφορά περιπτώσεις εκτέλεσης μικρών σετ δεδομένων, η χρήση του Spark framework δε μας παρέχει κάποιο πλεονέκτημα ως προς τον χρόνο εκτέλεσης διότι παρουσιάζει μεγάλο overhead και δεν επιτυγχάνεται παραλληλισμός των διεργασιών. Συνεπώς, όταν τα δεδομένα δεν είναι αρκετά μεγάλα, μη κατανεμημένες λύσεις είναι συχνά πιο γρήγορες σε σχέση μ' αυτές του Spark. Αντίθετα, όταν έχουμε να κάνουμε με μεγάλο όγκο δεδομένων, η χρήση του Spark framework αποτελεί μία ιδανική λύση καθώς μας επιτρέπει παράλληλη επεξεργασία των διεργασιών και την δυνατότητα αποθήκευσης μεγάλων σετ δεδομένων στην μνήμη.

Η αξιολόγηση των αποτελεσμάτων αυτών δείχνει ότι ο k-means αλγόριθμος αποδίδει πολύ καλυτέρα σε σχέση με την ιεραρχική ομαδοποίηση έχοντας διαγνώσει αρκετές φόρες όλα τα προβλήματα διευθέτησης της μνήμης επιτυγχάνοντας ακρίβεια ίση με τη μονάδα. Τις περισσότερες φορές, ο k-means μπορεί να δώσει ένα ικανοποιητικό αποτέλεσμα εύρεσης των outliers το οποίο θα βοηθήσει τους προγραμματιστές να ελαχιστοποιήσουν τον χρόνο αναζήτησης ενός προβλήματος διευθέτησης μνήμης.

Παρόλα αυτά, ο k-means αποτελεί έναν ασταθή αλγόριθμο εκτέλεσης. Η ιεραρχική ομαδοποίηση σε κάθε εκτέλεση του ίδιου σετ δεδομένων, θα δώσει ακριβώς τα ίδια αποτελέσματα, ενώ η k-means μέθοδος αδυνατεί να επιτύχει κάτι τέτοιο δίνοντας μας τις περισσότερες φόρες (αν όχι όλες) διαφορετικά αποτελέσματα. Σε κάθε εκτέλεση, ορίζονται διαφορετικά αρχικά κέντρα συστάδων με αποτέλεσμα τη δημιουργία διαφορετικών αρχικών συστάδων. Συνεπώς, πραγματοποιείται μία διαφορετική διάγνωση των προβλημάτων που σχετίζονται με τη μνήμη. Αυτό είναι ένα σημαντικό θέμα του k-means, καθώς στις βέλτιστες εκτελέσεις ενός σετ δεδομένων, μπορεί να εντοπίσει όλο το σύνολο των memory-related θεμάτων και να έχει ακρίβεια ίση με τη μονάδα, αλλά σε άλλες του ίδιου σετ δεδομένων, μπορεί να παύσει να έχει θετικά αποτελέσματα.

Εμείς εκτελέσαμε τον αλγόριθμο εφαρμόζοντας σαν k τον αριθμό που παίρνουμε χάρις την χρήση της elbow μεθόδου (σε όσες περιπτώσεις ήταν εμφανής η επιλογή) και παρουσιάσαμε τις καλύτερες εκτελέσεις του k-means αλγορίθμου. Η εφαρμογή της elbow μεθόδου σαν κριτήριο επιλογής του αριθμού k, μας προσέφερε καλύτερα αποτελέσματα σε σχέση με την χρήση ενός σταθερού αριθμού k στην εκτέλεση των σετ δεδομένων μας.

Η ιεραρχική ομαδοποίηση είναι σχετικά πιο γρήγορη από τον k-means αλγόριθμο όσον αφορά μικρά σετ δεδομένων Όμως, όσο μεγαλώνει το μέγεθος των δεδομένων, αυξάνεται εκθετικά η πολυπλοκότητα της, σε αντίθεση με τον k-means, ο οποίος

αυξάνεται γραμμικά. Στις περιπτώσεις εκτέλεσης μεγάλων σετ δεδομένων, έχοντας λάβει μικρό χρόνο δειγματοληψίας (συνήθως 1-3 δευτερόλεπτα), η ιεραρχική ομαδοποίηση αδυνατεί να τρέξει και να φέρει αποτελέσματα. Αντιθέτως ο μεγαλύτερος χρόνος εκτέλεσης της k-means μεθόδου είναι μόλις 9 λεπτά που αποδεικνύει ότι αποτελεί έναν γρήγορο αλγόριθμο εκτέλεσης.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος Όρος	**Ελληνικός Όρος**
Clustering	Συσταδοποίηση
Server	Εξυπηρετητής
Agent	Πράκτορας
Generator	Γεννήτρια
Implementation	Υλοποίηση
Mapping	Χαρτογράφηση
Open source	Ανοιχτού κώδικα
Distributed systems	Κατανεμημένα συστήματα
Variance	Διακύμανση
Partition	Διαμέριση
Time-slice	Χρόνο-φέτα
Agglomerative	Αθροιστικός
Divisive	Διαιρετικός
Scalability	Κλιμάκωση
Overhead	Επιβάρυνση
A priori	Εκ των προτέρων
Optimization	Βελτιστοποίηση
Cross validation	Διασταυρούμενη επικύρωση
Heuristic	Ευρετική
Kernel	Πυρήνας
Hierarchical clustering	Ιεραρχική ομαδοποίηση
Performance counter	Μετρητής απόδοσης
Unsupervised learning	Εκμάθηση χωρίς επίβλεψη
Artificial intelligence	Τεχνητή νοημοσύνη
Machine learning	Εκμάθηση μηχανής
Thread	Νήμα
Log	Κούτσουρο
Garbage collector	Συλλέκτης σκουπιδιών
Documentation	Οδηγίες
Heap	Σωρός
Repository	Αποθήκη

Clause	Ρήτρα
Workload	Φόρτος εργασίας
Peer-to-peer systems	Ομότιμα συστήματα
Transducer	Μετατροπέας
Execution Logs	Αρχεία εκτέλεσης
Debugging	Αποσφαλμάτωση
Developer	Προγραμματιστής
Memory leak	Διαρροή μνήμης
Standard deviation	Τυπική απόκλιση
Persistent	Επίμονος
Fault injection	Εισαγωγή σφαλμάτων
Shared data	Διαμοιραζόμενα δεδομένα
Unit testing	Δοκιμές μονάδας
Patch	Ενημέρωση
Software regression	Παλινδρόμηση λογισμικού
Query	Ερώτημα
Communication bottleneck	Συμφόρηση επικοινωνίας
Monitoring	Παρακολούθηση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ULS	Ultra Large Scale
SPE	Software Performance Engineering
CPU	Central Processing Unit
DB	Database
UI	User Interface
IRC	Internet Relay Chat
HTML	HyperText Markup Language
SQL	Structured Query Language
ID	Identification
MLlib	Machine Learning Library
AI	Artificial Intelligence
OS	Operating System
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
SSE	Error Sum of Squares
HDFS	Hadoop Distributed File System
UC	University of California
RDD	Resilient Distributed Data sets
HBase	Hadoop Database
JDK	Java Development Kit
API	Application Programming Interface
IT	Information Technology
POSIX	Portable Operating System Interface for Unix
YARN	Yet Another Resource Negotiator
JMX	Java Management Extensions
I/O	Input / Output
MBeans	Managed Beans
SNMP	Simple Network Management Protocol
JVM	Java Virtual Machine
Java SE	Java Standard Edition
P2P	Peer-to-peer
JSP	JavaServer Pages

ΠΑΡΑΡΤΗΜΑ Ι

Κώδικας εκτέλεσης του k-means αλγορίθμου χρησιμοποιώντας την MLlib βιβλιοθήκη του Spark framework.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.commons.lang3.ArrayUtils;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.clustering.KMeans;
import org.apache.spark.mllib.clustering.KMeansModel;
import org.apache.spark.mllib.linalg.Vector;
import org.apache.spark.mllib.linalg.Vectors;

import kmeans.correlation.common.TimeSliceProfile;
import kmeans.correlation.common.TimeSliceProfileCluster;

public class KMeansClustering implements Serializable
{
   public static class Parameters implements Serializable
   {
// number iterations used for the algorithm
      private final int numberOfIterations;
   // number of clusters used using the elbow method
      private final int numberOfClusters;


      public Parameters(int numberOfIterations, int numberOfClusters)
      {
         this.numberOfIterations = numberOfIterations;
         this.numberOfClusters = numberOfClusters;
      }

      public int getNumberOfIterations()
      {
         return numberOfIterations;
      }

      public int getNumberOfClusters()
```

```
        {
            return numberOfClusters;
        }

    }


    private final Parameters parameters;


    public KMeansClustering(Parameters parameters)
    {
        this.parameters = parameters;
    }


    public JavaRDD<TimeSliceProfileCluster> cluster(JavaRDD<TimeSliceProfile>
profiles)
    {
        final JavaRDD<Map<String, Double>> logActivities = profiles.map(prof ->
prof.getLogActivity());
        final JavaRDD<Collection<Double>> dataCollections = logActivities.map(activity ->
activity.values());
        final JavaRDD<Double[]> dataArrays = dataCollections.map(collection ->
collection.toArray(new Double[0]));
        final JavaRDD<Vector> dataVectors = dataArrays.map(array ->
Vectors.dense(ArrayUtils.toPrimitive(array)));

        final int numberOfIterations = parameters.getNumberOfIterations();
        final int numberOfClusters = parameters.getNumberOfClusters();

        // k-means training
        KMeansModel clusteringModel = KMeans.train(dataVectors.rdd(),
numberOfClusters, numberOfIterations);

        // vector placement - the cluster in which every vector is placed
        final JavaRDD<String> clusterIndices =
clusteringModel.predict(dataVectors).map(i -> i.toString());

        // connection into pairs of indices and time-slice-profile-clusters
        final JavaPairRDD<String, TimeSliceProfile> indexedProfiles =
clusterIndices.zip(profiles);

       // creation of the clusters containing the time-slice profiles
        final JavaPairRDD<String, List<TimeSliceProfile>> indexedProfileGroups =
indexedProfiles
                .combineByKey(profile -> new
ArrayList<TimeSliceProfile>(Arrays.asList(profile)),
                        (profileList, profile) -> {
```

```
                        profileList.add(profile);
                        return profileList;
                    } ,
                    (profileList1, profileList2) -> {
                        profileList1.addAll(profileList2);
                        return profileList1;
                    });

        final JavaRDD<TimeSliceProfileCluster> clusters = indexedProfileGroups
                .map(profileList -> new TimeSliceProfileCluster(profileList._1, profileList._2));

        return clusters;
    }

}
```

ΠΑΡΑΡΤΗΜΑ ΙΙ

Κώδικας εκτέλεσης της Elbow μεθόδου για τον εντοπισμό του αριθμού k χρησιμοποιώντας την MLlib βιβλιοθήκη του Spark framework.

```
// taking a JavaRDD of a time-slice profile values as an input
static public void getElbowMethod(JavaRDD<Vector> vectors)
{

// maximum number of clusters
      int numClusters = 10;
      Map<Integer, Double> varianceMap = new HashMap<>();

      for(int i = 2; i < numClusters; i++)
{
// finding the SSE and variance of kmeans clustering for every k
         KMeansModel clusters = KMeans.train(vectors.rdd(), i, numIterations);
         double WSSSE = clusters.computeCost(vectors.rdd());
         double variance = WSSSE / (numClusters - 1);
         varianceMap.put(i, variance);
       }

// printing of the total values
      for(int key: varianceMap.keySet())
{
         if(key != numClusters - 1)
{
            System.out.print(varianceMap.get(key) + ", ");
         }
         else
{
            System.out.print(varianceMap.get(key));
         }
       }
}
```

ΑΝΑΦΟΡΕΣ

[1] S. Institute, «Ultra-Large-Scale Systems: The Sofware Challenge of the Future,» *Carnegie Mellon Univercity,* 2006.

[2] A. E. Hassan, Z. M. Jiang, G. Hamann και P. Flora, «Automated performance analysis of load tests,» *Proceedings of the International Conference on Software Maintenance,* September 2009.

[3] E. Weyuker και F. Vokolos, «Experience with performance testing of software systems: issues, an approach, and case study,» *Transactions on Software Engineering, vol 26,* December 2000.

[4] «Rsystems,» [Ηλεκτρονικό]. Available: http://www.rsystems.com/CommonResource/KnowledgeRepository/How-Performance-Testing-Impacts-Customers-Business.pdf. [Πρόσβαση 15 February 2017].

[5] «Abstracta,» [Ηλεκτρονικό]. Available: http://www.abstracta.us/2015/10/12/why-performance-testing-is-necessary/. [Πρόσβαση 15 February 2017].

[6] H. Malik, «A methodology to support load test analysis,» *Proceedings of the International Conference on Software Enginnering,* May 2010.

[7] J. M. Jiang, A. E. Hassan, G. Hamann και P. Flora, «Automatic identification of load testing problems,» *Proceedings of the International Conference on Software Maintenance,* October 2008.

[8] M. D. Syer, Z. M. Jiang, M. Nagappan, A. E. Hassan, M. Nasser και P. Flora, «Leveraging Performance Counters and Execution Logs to Diagnose Memory-Related Peformance Issues,» *Software Maintenance (ICSM), 2013 29th IEEE International Conference,* 22 September 2013.

[9] «Perfeng,» [Ηλεκτρονικό]. Available: http://www.perfeng.com/speis.htm. [Πρόσβαση 15 February 2017].

[10] Z. M. Jiang και A. E. Hassan, «A Survey on Load Testing of Large-Scale Software Systems,» *IEEE Transactions on Software Engineering vol. 41, no. 11,* November 2015.

[11] «Terikhelper,» [Ηλεκτρονικό]. Available: https://telerikhelper.net/tag/performance-testing/. [Πρόσβαση 15 February 2017].

[12] «Techtarget,» [Ηλεκτρονικό]. Available: http://searchdatacenter.techtarget.com/definition/workload. [Πρόσβαση 15 February 2017].

[13] Ragunath, «Geekswithblogs,» 14 November 2011. [Ηλεκτρονικό]. Available: http://geekswithblogs.net/TarunArora/archive/2011/11/14/load-and-web-performance-testing-using-visual-studio-ultimate-2010-part.aspx. [Πρόσβαση 15 February 2017].

[14] «Smartbear,» [Ηλεκτρονικό]. Available: https://smartbear.com/learn/performance-testing/what-is-load-testing/. [Πρόσβαση 15 February 2017].

[15] «Osfi-bsif,» 24 September 2014. [Ηλεκτρονικό]. Available: http://www.osfi-bsif.gc.ca/Eng/fi-if/rg-ro/gdn-ort/gl-ld/Pages/e18.aspx. [Πρόσβαση 15 February 2017].

[16] A. Macedo, T. B. Ferreira και R. Matias, «The mechanics of memory-related software aging,» *Proceedings of the International Workshop on Sofware Aging and Rejuvenation,* November 2010.

[17] R. Matias, B. Evangelista και A. Macedo, «Monitoring memory-related software aging: An exploratory study,» *Proceedings of the International Symposium of Software Reliability Engineering Workshops,* November 2012.

[18] «Microsoft,» Microsoft, 2017. [Ηλεκτρονικό]. Available: https://msdn.microsoft.com/en-us/library/ms859408.aspx. [Πρόσβαση 15 February 2017].

[19] «Github,» Github, [Ηλεκτρονικό]. Available:

https://github.com/CoolElvis/sidekiq_memory_bloat. [Πρόσβαση 15 February 2017].

[20] «Scoutapp,» [Ηλεκτρονικό]. Available: http://book.scoutapp.com/memory-bloat.html. [Πρόσβαση 15 February 2017].

[21] «Davidklee,» 28 January 2014. [Ηλεκτρονικό]. Available: http://www.davidklee.net/2014/01/28/virtualization-resource-consumption-counters-lie-to-you/. [Πρόσβαση 15 February 2017].

[22] «Slideshare,» 18 February 2015. [Ηλεκτρονικό]. Available: http://www.slideshare.net/databricks/sparkcamp-strata-ca-intro-to-apache-spark-with-handson-tutorials. [Πρόσβαση 15 February 2017].

[23] «Techtarget,» [Ηλεκτρονικό]. Available: http://whatis.techtarget.com/definition/log-log-file. [Πρόσβαση 15 February 2017].

[24] «Brickmarketing,» [Ηλεκτρονικό]. Available: http://www.brickmarketing.com/define-log-file.htm. [Πρόσβαση 15 February 2017].

[25] T. Peters, «The history and development of transaction log analysis,» *Library Hi Tech,* 1993.

[26] «Sonus,» [Ηλεκτρονικό]. Available: https://support.sonus.net/display/UXDOC22/Viewing+and+Exporting+Windows+Event+Logs+on+the+ASM. [Πρόσβαση 15 February 2017].

[27] «Amihalj,» 29 January 2013. [Ηλεκτρονικό]. Available: https://amihalj.wordpress.com/2013/01/29/tracking-transaction-log-records-with-sql-server-2012-extended-events/. [Πρόσβαση 15 February 2017].

[28] «Dell,» Dell, 17 September 2016. [Ηλεκτρονικό]. Available: https://documents.software.dell.com/sonicwall-sma-1000-series/11.4/administration-guide/administration/system-administration/system-logging-and-monitoring/log-files/system-message-log?ParentProduct=858. [Πρόσβαση 15 February 2017].

[29] «Explorefinancialservices,» [Ηλεκτρονικό]. Available: https://www.explorefinancialservices.com/Careers/152. [Πρόσβαση 15 February 2017].

[30] J. M. Jiang, A. E. Hassan, G. Hamann και P. Flora, «An automated approach for abstracting execution logs to execution events,» *Journal of Software Maintenance and Evolution, vol. 20, no. 4,* July 2008.

[31] «Webopedia,» [Ηλεκτρονικό]. Available: http://www.webopedia.com/TERM/C/clustering.html. [Πρόσβαση 15 February 2017].

[32] «Sourceforge,» [Ηλεκτρονικό]. Available: http://pypr.sourceforge.net/kmeans.html. [Πρόσβαση 15 February 2017].

[33] «Saedsayad,» [Ηλεκτρονικό]. Available: http://www.saedsayad.com/clustering_hierarchical.htm. [Πρόσβαση 15 February 2017].

[34] D. M. Blei, «Princeton,» Princeton, [Ηλεκτρονικό]. Available: http://www.cs.princeton.edu/courses/archive/spr08/cos424/slides/clustering-2.pdf. [Πρόσβαση 15 February 2017].

[35] A. Huang, «Similarity measures for text document clustering,» *Proceedings of the New Zealand Computer Science Research Student Conference,* April 2008.

[36] N. Sandbya και A. Govardhan, «Analysis of similarity measures with wordnet based text document clustering,» *Proceedings of the international conference on Information Systems Design and Intelligent Applications,* January 2012.

[37] «Improvedoutcomes,» [Ηλεκτρονικό]. Available: http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Pearson_Correlation_and_Pearson_Squared_Distance_Metric.htm. [Πρόσβαση 15 February 2017].

[38] P. N. Tan, N. Steinbach και V. Kumar, Cluster Analysis: Basic Concepts and Algorithms, 1st ed., Addison-Wesley Longman Publishing Co., 2005.

[39] I. Frades και R. Matthiesen, «Overview on techniques in cluster analysis,» *Bioinformatics Methods In Clinical Research, vol. 593,* March 2009.

[40] «Solver,» [Ηλεκτρονικό]. Available: http://www.solver.com/xlminer/help/hierarchical-clustering-intro. [Πρόσβαση 15 February 2017].

[41] C. Jin, R. Liu, W. Hendrix, C. Zhengzhang, A. Agrawal, W.-k. Liao και A. Cloudhary, «A Scalable Hierarchical Clustering Algorithm Using,» 2015.

[42] J. H. Ward, «Hierarchical grouping to optimize an objective function,» *Journal of Amer,* 1963.

[43] G. H. Lance και W. T. Williams, «A general theory of classificatory sorting strategies,» *The Computer Journal,* 1966.

[44] «Sthda,» [Ηλεκτρονικό]. Available: http://www.sthda.com/english/wiki/print.php?id=237. [Πρόσβαση 15 February 2017].

[45] T. Calinski και J. Harabasz, «A dendrite method for cluster analysis,» *Communications in Statistics, vol. 3,* January 1874.

[46] «Stattrek,» [Ηλεκτρονικό]. Available: http://stattrek.com/statistics/dictionary.aspx?definition=Influential%20point. [Πρόσβαση 15 February 2017].

[47] «Stackexchange,» 2016. [Ηλεκτρονικό]. Available: http://stats.stackexchange.com/questions/157569/what-is-the-problem-with-statistical-outlier-detection-approaches-if-we-have-dis. [Πρόσβαση 15 February 2017].

[48] L. Xu, W. Y. Poon και S. Y. Lee, «Influence analysis for the factor analysis model with ranking data,» May 2008.

[49] «Onlinestatbook,» [Ηλεκτρονικό]. Available: http://onlinestatbook.com/2/regression/influential.html. [Πρόσβαση 15 February 2017].

[50] «Mspguide,» [Ηλεκτρονικό]. Available: http://www.mspguide.org/tool/stakeholder-analysis-importanceinfluence-matrix. [Πρόσβαση 15 February 2017].

[51] «Codeproject,» [Ηλεκτρονικό]. Available: https://www.codeproject.com/Articles/35671/Distributed-and-Parallel-Processing-using-WCF. [Πρόσβαση 15 February 2017].

[52] «Techopedia,» [Ηλεκτρονικό]. Available: https://www.techopedia.com/definition/7/distributed-computing-system. [Πρόσβαση 15 February 2017].

[53] «Webopedia,» [Ηλεκτρονικό]. Available: http://www.webopedia.com/TERM/D/distributed_computing.html. [Πρόσβαση 15 February 2017].

[54] «Hadoop,» [Ηλεκτρονικό]. Available: www.hadoop.apache.org/. [Πρόσβαση 15 February 2017].

[55] J. Dean και S. Ghemawat, «MapReduce: simplified data processing on large clusters,» *Communications of the ACM, vol. 51, no. 1,* January 2008.

[56] T. d. s. Morais, «Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm,» *Proceedings of the 10th Doctoral Symposium in Informatics Engineering,* 2015.

[57] «Webdam,» [Ηλεκτρονικό]. Available: http://webdam.inria.fr/Jorge/html/wdmch17.html. [Πρόσβαση 15 February 2017].

[58] «IBM,» IBM, [Ηλεκτρονικό]. Available: https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/. [Πρόσβαση 15 February 2017].

[59] «Tutorialspoint,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm. [Πρόσβαση 15 February 2017].

[60] «Searchcloudcomputing,» [Ηλεκτρονικό]. Available: http://searchcloudcomputing.techtarget.com/definition/MapReduce. [Πρόσβαση 15 February

2017].

[61] S. P. Bappalige, «Opensource,» 26 August 2014. [Ηλεκτρονικό]. Available: https://opensource.com/life/14/8/intro-apache-hadoop-big-data. [Πρόσβαση 15 February 2017].

[62] «Hadoop,» Apache, 26 January 2016. [Ηλεκτρονικό]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#HDFS_Architecture. [Πρόσβαση 15 February 2017].

[63] S. Penchikala, «Infoq,» 30 January 2015. [Ηλεκτρονικό]. Available: https://www.infoq.com/articles/apache-spark-introduction. [Πρόσβαση 15 February 2017].

[64] «Spark,» Apache, [Ηλεκτρονικό]. Available: http://spark.apache.org/docs/latest/cluster-overview.html. [Πρόσβαση 15 February 2017].

[65] I. Szegedi, «Dzone,» 7 April 2014. [Ηλεκτρονικό]. Available: https://dzone.com/articles/apache-spark-fast-big-data. [Πρόσβαση 15 February 2017].

[66] «Uncle-ba,» [Ηλεκτρονικό]. Available: https://uncle-bae.blogspot.gr/2016/05/apache-spark.html. [Πρόσβαση 15 February 2017].

[67] «Spark,» Apache, [Ηλεκτρονικό]. Available: http://spark.apache.org/docs/latest/programming-guide.html. [Πρόσβαση 15 February 2017].

[68] S. P. Singh, «Tothenew,» 13 February 2015. [Ηλεκτρονικό]. Available: http://www.tothenew.com/blog/spark-1o3-spark-internals/. [Πρόσβαση 15 February 2017].

[69] «Srinivastata,» 7 April 2015. [Ηλεκτρονικό]. Available: http://www.srinivastata.com/tez/directed-acyclic-graph-dag-strength-of-spark-and-tez/. [Πρόσβαση 15 February 2017].

[70] A. Kharbanda, «Sigmoid,» [Ηλεκτρονικό]. Available: https://www.sigmoid.com/apache-spark-internals/. [Πρόσβαση 15 February 2017].

[71] «Jaceklaskowski,» [Ηλεκτρονικό]. Available: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-broadcast.html. [Πρόσβαση 15 February 2017].

[72] «Spark,» Apache, [Ηλεκτρονικό]. Available: https://spark.apache.org/docs/2.0.2/mllib-clustering.html. [Πρόσβαση 15 February 2017].

[73] «Oracle,» Oracle, [Ηλεκτρονικό]. Available: http://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html. [Πρόσβαση 15 February 2017].

[74] A. Kumaraswamipillai, «Java-success,» 24 February 2015. [Ηλεκτρονικό]. Available: http://www.java-success.com/java-8-streams-lambdas-intermediate-vs-terminal-ops-lazy-loading-simple-examples/. [Πρόσβαση 15 February 2017].

[75] «Docs.Oracle,» Oracle, [Ηλεκτρονικό]. Available: http://docs.oracle.com/javase/tutorial/jmx/overview/index.html. [Πρόσβαση 15 February 2017].

[76] «Javarevisited,» [Ηλεκτρονικό]. Available: http://javarevisited.blogspot.gr/2015/01/how-to-use-lambda-expression-in-place-anonymous-class-java8.html. [Πρόσβαση 15 February 2017].

[77] P. Bholowalia και A. Kumar, «EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN,» *International Journal of Computer Applications vol. 105, no. 9,* November 2014.

[78] «Bl.ocks,» 3 December 2015. [Ηλεκτρονικό]. Available: https://bl.ocks.org/rpgove/0060ff3b656618e9136b. [Πρόσβαση 15 February 2017].

[79] G. W. Milligan και M. C. Cooper, «An examination of procedures for determing the number of clusters in a data set,» *Phychometrika, vol. 50, no. 2,* June 1985.

[80] «Aegean,» Aegean Univercity, [Ηλεκτρονικό]. Available: http://www.icsd.aegean.gr/lecturers/kavallieratou/PattRec_files/pr_10.pdf. [Πρόσβαση 15 February 2017].

[81] «Statistics4u,» [Ηλεκτρονικό]. Available: http://www.statistics4u.com/fundstat_eng/cc_cross_validation.html. [Πρόσβαση 15 February 2017].

[82] P. J. Rousseeuw, «Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,» *Journal of Computational and Applied Mathematics, vol. 20, no. 1,* November 1987.

[83] R. Rleti, M. C. Ortiz, L. A. Sarabia και M. S. Sanchez, «Selecting Variables for k-Means Cluster Analysis by Using a Genetic Algorithm that Optimises the Silhouettes,» 5 July 2004.

[84] «Sthda,» [Ηλεκτρονικό]. Available: http://www.sthda.com/english/wiki/determining-the-optimal-number-of-clusters-3-must-known-methods-unsupervised-machine-learning. [Πρόσβαση 15 February 2017].

[85] «Stackexchange,» [Ηλεκτρονικό]. Available: http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means. [Πρόσβαση 15 February 2017].

[86] «GNU,» GNU, [Ηλεκτρονικό]. Available: https://www.gnu.org/software/octave/doc/octave-4.0.0.pdf. [Πρόσβαση February 15 2017].

[87] «Dtrace,» Dtrace, [Ηλεκτρονικό]. Available: Source: http://dtrace.org/blogs/about/ . [Πρόσβαση 15 February 2017].

[88] L. Degioanni, «Sysdig,» Sysdig, 10 April 2014. [Ηλεκτρονικό]. Available: https://sysdig.com/blog/sysdig-vs-dtrace-vs-strace-a-technical-discussion/. [Πρόσβαση 15 February 2017].

[89] S. P. Reiss, «Efficient monitoring and display of thread state in java,» *Proceedings of the International Workshop on Program Comprehension,* May 2005.

[90] «Docs.Oracle,» Oracle, [Ηλεκτρονικό]. Available: https://docs.oracle.com/javase/tutorial/jmx/mbeans/mxbeans.html. [Πρόσβαση 15 February 2017].

[91] B. Chamith, «Chamibuddhika,» 24 June 2012. [Ηλεκτρονικό]. Available: https://chamibuddhika.wordpress.com/2012/06/24/jmx-some-introductory-notes/. [Πρόσβαση 15 February 2017].

[92] «Docs.Oracle,» Oracle, [Ηλεκτρονικό]. Available: https://docs.oracle.com/javase/8/docs/api/java/lang/management/MemoryUsage.html. [Πρόσβαση 15 February 2017].

[93] «Theserverside,» [Ηλεκτρονικό]. Available: http://www.theserverside.com/definition/JMX-Java-Management-Extensions. [Πρόσβαση 15 February 2017].

[94] «Tomcat,» Apache, [Ηλεκτρονικό]. Available: http://tomcat.apache.org/. [Πρόσβαση 15 February 2017].

[95] «Jmeter,» Apache, [Ηλεκτρονικό]. Available: http://jmeter.apache.org/. [Πρόσβαση 15 February 2017].

[96] «Jmeter,» Apache, [Ηλεκτρονικό]. Available: http://jmeter.apache.org/. [Πρόσβαση 15 February 2017].

[97] «Socketo,» [Ηλεκτρονικό]. Available: http://socketo.me/docs/. [Πρόσβαση 15 February 2017].