ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός Ανωμαλιών Εκτέλεσης Με Χρήση Μεθόδων Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννη Ντάλλα

Επιβλέπων : Κοζύρης Νεκτάριος
Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εντοπισμός Ανωμαλιών Εκτέλεσης Με Χρήση Μεθόδων Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΙΩΑΝΝΗ ΝΤΑΛΛΑ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Απριλίου 2018.

.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Δημήτριος Τσουμάκος
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ι.Π.

Αθήνα, Απρίλιος 2018

.....

ΙΩΑΝΝΗΣ ΝΤΑΛΛΑΣ

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

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μεθοδολογίας για τον εντοπισμό ανωμαλιών στην εκτέλεση ενός τελεστή. Η μεθοδολογία ελέγχει τις χρονοσειρές μετρικών που περιγράφουν την κατάσταση των πόρων ενός συστήματος σε πραγματικό χρόνο κατά τη λειτουργία του τελεστή με στόχο την εύρεση μη-φυσιολογικών συμβάντων και συνεπώς την εύρεση προβλημάτων.

Συγκεκριμένα, υλοποιήθηκε σύστημα που συλλέγει μετρικές, που παρέχονται από το Ganglia Monitoring System, βάσει των οποίων κατασκευάζει μοντέλα πρόβλεψης του αποτυπώματος της εκτέλεσης του τελεστή επάνω στο υπολογιστικό σύστημα. Αυτά αποτελούν μέτρο σύγκρισης για τον εντοπισμό αποκλίσεων κατά την κανονική λειτουργία, στην οποία γίνεται κεντρικός έλεγχος του τελεστή. Στη συνέχεια έγινε πειραματική αξιολόγηση του συστήματος και των μοντέλων για διάφορους τύπους ανωμαλιών εκτέλεσης ,τελεστών και τοπολογιών του φυσικού επιπέδου.

Το σύστημα και η μεθοδολογία μπορούν να εντοπίσουν σε σύντομο χρόνο ανωμαλίες εκτέλεσης, είναι γενικευμένα και παραμετροποιημένα, ώστε να μπορούν να εφαρμοστούν σε οποιοδήποτε υπολογιστικό σύστημα, ελέγχοντας τελεστές ταυτόχρονα, δεχόμενα μετρικές από πολλές πηγές.

Λέξεις Κλειδιά: έλεγχος τελεστή, ανωμαλίες εκτέλεσης, εντοπισμός, μετρικές, ganglia.

Abstract

The scope of this thesis was the development of a methodology for the detection of anomalies in the execution of an operator. The methodology monitors timeseries of metrics describing the state of a system's resources during the execution in order to detect abnormal events and therefore problems.

Specifically, a system for collecting metrics that are provided by the Ganglia Monitoring System and using them to create models that predict the operator's footprint on the system has been implemented. Those models are the standard for detecting discrepancies during normal operation, which is centrally monitored. Subsequently the system and the models were experimentally evaluated for different types of execution anomalies, operators and topologies of the physical layer.

The system and the methodology can detect anomalies in a short time, are generalized and parameterized so that they can be applied on every computational system, can monitor many operators simultaneously while receiving data from various sources.

Keywords: operator monitoring, execution anomalies, detection, metrics, ganglia.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον κ. Νεκτάριο Κοζύρη και όλα τα μέλη του CSLab του ΕΜΠ για την υποστήριξη που μου παρείχαν.

Επίσης ευχαριστώ την Κατερίνα Δόκα, Phd για την πολύτιμη βοήθειά της και τις συμβουλές της καθ' όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας, η οποία δεν θα ήταν δυνατόν να ολοκληρωθεί χωρίς αυτήν, και τον υποψήφιο διδάκτορα Κωνσταντίνο Τσιτσεκλή για τις συμβουλές του σε τεχνικά θέματα.

Τέλος ευχαριστώ τους φίλους μου και την οικογένειά μου που με στήριξαν όλον αυτόν τον καιρό.

Πίνακας περιεχομένων

1 *Εισαγωγή*

Τα τελευταία χρόνια και ιδιαίτερα τη δεκαετία που διανύουμε, χάρη στην ανάπτυξη διαφόρων τομέων της εφαρμοσμένης πληροφορικής, όπως τα κοινωνικά δίκτυα και το Internet of Things (IoT), παρατηρείται μία τεράστια και διαρκώς αυξανόμενη συσσώρευση δεδομένων, προερχόμενων από ποικίλες πηγές όπως αισθητήρες, χρήστες διαδικτύου, πληροφοριακά συστήματα, ερευνητικά και επιστημονικά έργα κτλ, τα οποία αναφέρονται ως Μεγάλα Δεδομένα (Big Data). Αυτός ο μεγάλος όγκος δεδομένων περιέχει και χρήσιμη πληροφορία η οποία πρέπει να εξαχθεί από το σύνολο για να χρησιμοποιηθεί από τον εκάστοτε ενδιαφερόμενο, όπως μία επιχείρηση ή έναν οργανισμό.

Για παράδειγμα η αλματώδης ανάπτυξη των μέσων κοινωνικής δικτύωσης οδήγησε τις εταιρίες που τα διαχειρίζονται (Google, Facebook, Twitter και άλλες) να χρησιμοποιήσουν μεθόδους data mining για να εξάγουν τη χρήσιμη πληροφορία από τα δεδομένα των χρηστών τους με στόχο την περαιτέρω ανάπτυξη των πλατφορμών που παρέχουν ή την δημιουργία και βελτιστοποίηση recommender συστημάτων για την στοχευμένη προώθηση περιεχομένου σε αυτούς.

Η ανάλυση των Big Data είναι πλέον δομικό στοιχείο και των μεταφορών. Η εταιρία UPS με το project ORION (On-Road Integrated Optimization and Navigation) βασίζεται σε αυτή την ανάλυση επάνω σε συσσωρευμένα δεδομένα πλοήγησης για τον υπολογισμό των καλύτερων διαδρομών με στόχο την μείωση του κόστους [1].

Άλλες εταιρίες αντί να επεξεργάζονται οι ίδιες τα δεδομένα παρέχουν πλατφόρμες επεξεργασίας προς τρίτους για την εκτέλεση εφαρμογών big data analytics, όπως η Amazon με την AWS (Amazon Web Services).

Είναι ξεκάθαρο ότι σε όλους τους τομείς της τεχνολογίας υπάρχει η τάση επέκτασης της επεξεργασίας δεδομένων (data analytics) και μάλιστα αυτή η επεξεργασία γίνεται ολοένα συνθετότερη και χρονοβόρα λόγω του όγκου της επεξεργαζόμενης πληροφορίας αλλά και της πολυπλοκότητας των αλγορίθμων που εφαρμόζονται. Η επεξεργασία αυτή γίνεται πολλές φορές σε μεγάλα κέντρα δεδομένων ή σε clusters από υπολογιστές και γενικώς σε κατανεμημένα συστήματα, στα οποία εκτελείται πολύ μεγάλος αριθμός παράλληλων διεργασιών που έχουν μεγάλο χρόνο εκτέλεσης και είναι κρίσιμες για τη βιομηχανία.

Σε τέτοια σύνθετα συστήματα μπορούν να υπάρξουν προβληματικές καταστάσεις στην εκτέλεση των προγραμμάτων οι οποίες εμποδίζουν ή καθυστερούν τον επιτυχή τερματισμό τους. Ένα τέτοιο παράδειγμα είναι η μετάβαση ενός τμήματος υλικού σε fail-slow λειτουργία η οποία έχει τεράστιο αντίκτυπο στη σε μια διεργασία, χωρίς ωστόσο να την τερματίζει, πράγμα που δυσκολεύει τον εντοπισμό του προβλήματος [2]. Σε άλλες περιπτώσεις ένας στοχαστικός αλγόριθμος θα μπορούσε να εγκλωβιστεί σε τοπικό ελάχιστο προκαλώντας καθυστέρηση ή οι απαραίτητοι πόροι μιας διεργασίας να εξαντλούνται από άλλες.

Γι’ αυτό το λόγο είναι επιτακτική ανάγκη για έγκαιρη ανίχνευση σφαλμάτων στην εκτέλεση των προγραμμάτων και όχι ο έλεγχός τους μόνο από το τελικό αποτέλεσμα που παράγουν. Από την άλλη το μέγεθος των συστημάτων αυτών και ο αριθμός των διεργασιών καθιστούν αδύνατη την επιτήρησή τους από τους διαχειριστές. Επομένως θα πρέπει να γίνεται αυτόματα από ένα σύστημα επιτήρησης και εντοπισμού ανωμαλιών εκτέλεσης με στόχο τη βέλτιστη χρησιμοποίηση ενός συστήματος και τη μείωση του κόστους.

1.1 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική εργασία λοιπόν καταπιάνεται ακριβώς με αυτό, το πρόβλημα του ελέγχου της εκτέλεσης ενός προγράμματος, που στο εξής θα καλείται ο τελεστής (ή operator) που επιδρά επάνω σε ένα σύστημα. Ο έλεγχος της συμπεριφοράς του τελεστή θα γίνεται με στόχο τον εντοπισμό ανωμαλιών στην εκτέλεσή του, δηλαδή μη-κανονικών καταστάσεων που με τον ένα ή τον άλλο τρόπο καθυστερούν ή και εμποδίζουν πλήρως τον επιτυχή τερματισμό του, με στόχο να ενημερωθεί έγκαιρα ο διαχειριστής και να διορθώσει το πρόβλημα που υπάρχει.

Για την επίλυση του αυτού του προβλήματος η παρούσα εργασία κάνει μια συγκεκριμένη πρόταση από την σκοπιά του θεωρητικού σχεδιασμού και της αρχιτεκτονικής ενός συστήματος και στη συνέχεια παρουσιάζει μια υλοποίηση αυτής της πρότασης μαζί με μία επαρκή πειραματική αξιολόγηση. Η πρόταση αυτή είναι ένα ολοκληρωμένο πρόγραμμα που επιτηρεί την λειτουργία ενός ή περισσότερων υπολογιστικών συστημάτων βασιζόμενο σε μετρικές που περιγράφουν σε πραγματικό χρόνο την κατάσταση των πόρων τους. Χρησιμοποιώντας αυτές τις μετρικές ως δεδομένα εκπαίδευσης επιχειρεί με μεθόδους μηχανικής μάθησης να κατασκευάσει μοντέλα τα οποία θα περιγράφουν τη γενική συμπεριφορά ενός τελεστή, ώστε να μπορεί με μια σχετική ακρίβεια να την προβλέψει για κάθε δοθέν σύνολο εισόδου στο πεδίο του χρόνου. Έτσι η πρόβλεψη που βασίζεται σε μια σειρά από δοκιμαστικές και χωρίς λάθη εκτελέσεις αποτελεί το μέτρο σύγκρισης κατά την κανονική λειτουργία, στην οποία οι αποκλίσεις που παρατηρούνται στις μετρικές είναι η ένδειξη κάποιου πιθανού προβλήματος.

1.2 Συνεισφορά

Πιο συγκεκριμένα κατά τη διαδικασία εκπόνησης της εργασίας μελετήθηκαν διάφορες πτυχές του προβλήματος και προτάθηκαν λύσεις για τα υποπροβλήματά του τόσο σε θεωρητικό όσο και σε επίπεδο υλοποίησης . Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν στατιστικές μέθοδοι και τεχνικές μηχανικές μάθησης για την σύγκριση και πρόβλεψη χρονοσειρών.
2. Επιλέχθηκε η μηχανική μάθηση ως καλύτερη πρόταση μετά από αξιολόγηση.

3. Καταστρώθηκε αλγόριθμος και μεθοδολογία ελέγχου.
4. Υλοποιήθηκε ένα πλήρως επεκτάσιμο σύστημα βάσει αυτών.
5. Το σύστημα είναι ευέλικτο και αρκετά γενικευμένο ώστε να επιβλέπει τελείως διαφορετικά συστήματα ταυτόχρονα και να δέχεται δεδομένα από διαφορετικές πηγές.
6. Αξιολογήθηκε το αποτέλεσμα με διάφορους τελεστές πειραματικά.
7. Εξάχθηκαν συμπεράσματα σχετικά με την υλοποίηση και έγιναν συγκεκριμένες προτάσεις για επέκταση.

1.3 Οργάνωση κειμένου

Το παρόν κεφάλαιο είναι μια σύντομη εισαγωγή στο χώρο του προβλήματος και δίνει τον ορισμό του στα πλαίσια της εργασίας μαζί με μερικά στοιχεία γι’ αυτήν. Στη συνέχεια στο Κεφάλαιο 2 θα παρουσιαστούν κάποιες σχετικές εργασίες και επιστημονικά πεδία σχετικά με το πρόβλημα καθώς και τα θεωρητικά εργαλεία που εφαρμόζονται στην εργασία. Στο Κεφάλαιο 3 γίνεται αναλυτική παρουσίαση της αρχιτεκτονικής του συστήματος και των τμημάτων του. Το Κεφάλαιο 4 αφορά την υλοποίηση και τις τεχνικές λεπτομέρειες και το Κεφάλαιο 5 την πειραματική αξιολόγηση του συστήματος. Το Κεφάλαιο 6 συνοψίζει τα συμπεράσματα στα οποία κατέληξε η εργασία και το Κεφάλαιο 7 περιέχει την βιβλιογραφία.

2 *Θεωρητικό υπόβαθρο*

Σε αυτό το σημείο θα γίνει μια συνοπτική παρουσίαση των θεωρητικών περιοχών οι οποίες είναι συναφείς με το αντικείμενο της διπλωματικής και κάποιον σχετικών εργασιών. Η συλλογιστική με την οποία έγινε η έρευνα στο θεωρητικό σκέλος ήταν ο εντοπισμός ανωμαλιών στις μετρικές που παρέχονται από ένα σύστημα. Οι μετρικές αυτές σε κάθε περίπτωση έχουν τη μορφή χρονοσειρών, αφού προκύπτουν από τη δειγματοληψία σε τακτά χρονικά διαστήματα των μετρικών που περιγράφουν την κατάσταση των πόρων του συστήματος, και συνεπώς οι μέθοδοι που μελετήθηκαν αφορούν στην σύγκριση και πρόβλεψη χρονοσειρών. Μπορούν να χωριστούν στις εξής βασικές κατηγορίες.

2.1 Στατιστικές μέθοδοι

Οι στατιστικές μέθοδοι μπορούν να διακριθούν σε δύο είδη. Πρώτον στις μεθόδους πρόβλεψης (forecasting methods) οι οποίες χρησιμοποιώντας την χρονοσειρά προσπαθούν να προβλέψουν την επόμενη τιμή της (Theta, Arima, Linear Regression κτλ) και δεύτερον στις μεθόδους σύγκρισης χρονοσειρών (συγκεκριμένα Granger Causality Test). Η πρώτη κατηγορία δεν αφορά την παρούσα διπλωματική, αφού μας ενδιαφέρει η πρόβλεψη ολόκληρης της χρονοσειράς και όχι μόνο της επόμενης τιμής. Η δεύτερη μελετήθηκε και αξιολογήθηκε πειραματικά και κρίθηκε ανεπαρκής. Λεπτομέρειες δίνονται σε επόμενο κεφάλαιο.

2.2 Clustering

Η δεύτερη γενική κατηγορία αφορά μεθόδους με τις οποίες μπορούν να ομαδοποιηθούν χρονοσειρές ανάλογα με τα χαρακτηριστικά τους. Έτσι αν σε ένα σύνολο από δεδομένα που περιέχουν τις ορθές χρονοσειρές και κάποιες υπό εξέταση εφαρμοστεί ένας αλγόριθμος ομαδοποίησης, όπως για παράδειγμα ο k-means clustering, με κατάλληλο τρόπο, μπορούν να ομαδοποιηθούν μαζί με τις αξιωματικά ορθές όσες δεν περιέχουν ανωμαλίες και ξεχωριστά σε κάποιο άλλο σύνολο όσες περιέχουν.

Τέτοιες μέθοδοι αναλύονται στις εξής εργασίες:

- ***Finding Structural Similarity in Time Series Data UsingBag-of-Patterns Representation***, Jessica Lin and Yuan Li, Computer Science Department, George Mason University.
- ***Group Clustering Using Inter-Group Dissimilarities,*** Debessay Fesehaye, Lenin Singaravelu, Chien-Chia Chen, Xiaobo Huang, Amitabha Banerjee, Ruijin Zhou and Rajesh Somasundaran, VMWare IO Performance Group.

Αυτές οι μέθοδοι μπορούν να εντοπίσουν ανωμαλίες στις χρονοσειρές αλλά και αυτόματα να τις ομαδοποιήσουν ανά τύπο ανωμαλίας. Ωστόσο σημαντικό μειονέκτημα αυτών των μεθόδων είναι ότι χρησιμοποιούν αλγόριθμους ομαδοποίησης, οι οποίοι έχουν σχετικά μεγάλη πολυπλοκότητα. Επομένως δεν είναι αρκετά γρήγορες ώστε να απαντώνται σε σταθερό ή γραμμικό χρόνο ερωτήματα κατά τον έλεγχο ενός τελεστή σε πραγματικό χρόνο και γι’ αυτό το λόγο δεν χρησιμοποιήθηκαν στην εργασία. Επιπλέον καθώς συσσωρεύονται δεδομένα η ομαδοποίηση θα πρέπει να επαναλαμβάνεται από την αρχή για να ληφθούν και αυτά υπόψιν. Συνήθως χρησιμοποιείται για τη μελέτη συμβάντων σε μεταγενέστερο χρόνο και όχι σε online έλεγχο.

2.3 Μηχανική μάθηση

Η τρίτη πρόταση, που αποδείχθηκε η καλύτερη για τα δεδομένα τις εργασίας, είναι η κατασκευή ενός συνόλου δεδομένων εκπαίδευσης και ελέγχου με είσοδο τα χαρακτηριστικά της εκτέλεσης του τελεστή και το χρόνο και έξοδο την παρατηρούμενη τιμή της μετρικής σε κάθε στιγμή με σκοπό την κατασκευή και εκπαίδευση νευρωνικών δικτύων με κατάλληλες παραμέτρους που θα παράγουν την

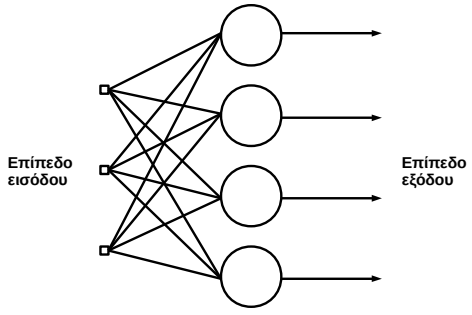
προβλεπόμενη συμπεριφορά για έναν τελεστή και συνεπώς θα μπορεί να γίνεται έλεγχος σε σταθερό χρόνο.

Το πλεονέκτημα αυτών των μεθόδων είναι ότι μπορούμε να μετακινήσουμε όλο το υπολογιστικό βάρος για την εκπαίδευση του δικτύου με ήδη αποθηκευμένες μετρικές αλλά και την παραγωγή της συνολικής πρόβλεψης για μία δεδομένη εκτέλεση στην αρχή της λειτουργίας, πριν καν ξεκινήσει ο τελεστής να λειτουργεί. Με αυτόν τον τρόπο έχουμε έτοιμο όταν ξεκινήσει ο έλεγχος ένα πολύ ελαφρύ και γρήγορο εργαλείο με ελάχιστο αποτύπωμα στους πόρους του συστήματος. Τέλος αξίζει να σημειωθεί ότι υπάρχει πληθώρα υλοποιημένων εργαλείων και βιβλιοθηκών για όλες τις γλώσσες προγραμματισμού που καθιστά αυτή την επιλογή την πιο εύκολη από τη σκοπιά της υλοποίησης.

2.4 Στοιχεία θεωρίας που εφαρμόστηκαν

Στόχος μας είναι να προβλέψουμε τη συμπεριφορά ενός τελεστή για δεδομένες παραμέτρους εισόδου έχοντας στη διάθεσή μας προηγούμενες δοκιμαστικές εκτελέσεις της λειτουργίας του. Επομένως μπορούμε να ορίσουμε και να εκπαιδεύσουμε κατάλληλα νευρωνικά δίκτυα και στη συνέχεια να υποβάλουμε ερωτήματα σε αυτά για την τιμή των μετρικών που μας ενδιαφέρουν. Παρακάτω γίνεται μια σύντομη ανάλυση των τύπων νευρωνικών δικτύων που χρησιμοποιούνται από το σύστημα καθώς και των επιπλέον μαθηματικών μεθόδων που χρησιμοποιούνται για την επεξεργασία των δεδομένων και τον εντοπισμό ανωμαλιών.

2.4.1 Perceptron ενός επιπέδου



Σχήμα 2.1 Ένα perceptron ενός επιπέδου με τέσσερεις νευρώνες.

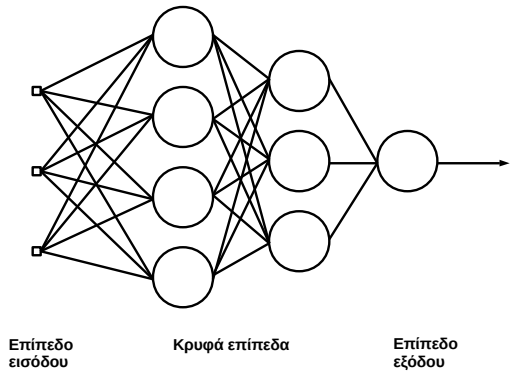
Το δομικό στοιχείο αυτού του μοντέλου είναι το απλό perceptron. Το perceptron είναι ένας αλγόριθμος εκμάθησης ενός δυαδικού ταξινομητή, δηλαδή μιας συνάρτησης που αντιστοιχίζει ένα διάνυσμα πραγματικών τιμών σε μια δυαδική συνάρτηση εξόδου:

$$f(x)=\begin{cases}1, \alpha\nu\, w\cdot x+b>0\\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma\end{cases}$$

Ο νευρώνας αποτελείται από μια σειρά εισόδους με βάρη, μια πόλωση, μια συνάρτηση άθροισης και μία συνάρτηση ενεργοποίησης.

Στο perceptron ενός επιπέδου έχουμε ένα σύνολο από νευρώνες που βρίσκονται στο ίδιο επίπεδο και υλοποιούν το παραπάνω μοντέλο και κάθε ένας από αυτούς είναι συνδεδεμένος με όλες τις εισόδους.

2.4.2 Πολυεπίπεδο Perceptron (Regressor)



Σχήμα 2.2 Ένα perceptron με δύο κρυφά επίπεδα με τέσσερεις και τρεις νευρώνες αντίστοιχα.

Το perceptron πολλών επιπέδων (Multi-Layer Perceptron, στο εξής MLP), όπως υποδηλώνει και το όνομα, αποτελείται από περισσότερα του ενός επίπεδα νευρώνων συνδεδεμένα σε σειρά. Το MLP είναι ένας αλγόριθμος επιβλεπόμενης μάθησης μιας συνάρτησης μέσω της εκπαίδευσης του νευρωνικού δικτύου με ένα σύνολο δεδομένων εισόδου και την επιθυμητή έξοδό τους. Δεδομένου μιας εισόδου τιμών $X=x_1, x_2, ..., x_m$ και της αντίστοιχης εξόδου y μπορεί να εκπαιδευτεί στην προσέγγιση μιας μη-γραμμικής συνάρτησης. Όπως φαίνεται και στο σχήμα υπάρχουν τριών ειδών επίπεδα, τα επίπεδα εισόδου, εξόδου και ένας αριθμός από εσωτερικά, κρυφά επίπεδα. Στο επίπεδο εισόδου ο αριθμός των νευρώνων καθορίζεται από το μήκος του διανύσματος εισόδου, ενώ το επίπεδο εξόδου έχει ένα στοιχείο που μας

δίνει την τιμή της συνάρτησης. Το μέγεθος και το πλήθος των κρυφών επιπέδων και οι παράμετροί τους μπορούν να καθοριστούν ανάλογα με το προς επίλυση πρόβλημα.

Κάθε νευρώνας του κρυφού επιπέδου μετατρέπει τις τιμές του προηγούμενου επιπέδου που δέχεται ως είσοδο σύμφωνα με το γραμμικό άθροισμα με βάρη $w_1 x_1 + w_2 x_2 + ... + w_m x_m + b$ και μετά με μία μη-γραμμική συνάρτηση ενεργοποίησης $g(\cdot): R \rightarrow R$, όπως για παράδειγμα η συνάρτηση της υπερβολικής εφαπτομένης, καθορίζει την ενεργοποίησή του. Στη συνέχεια οι τιμές που προκύπτουν από τα κρυφά επίπεδα οδηγούνται στο επίπεδο εξόδου που τις μετατρέπει στην τελική τιμή.

Ανάλογα με το είδος της εξόδου τα πολυεπίπεδα perceptron διακρίνονται σε classifiers και regressors. Στην πρώτη η έξοδος του δικτύου δίνει συγκεκριμένες τιμές που ορίζουν κλάσεις δεδομένων (class labels) ενώ στη δεύτερη δίνει συνεχείς τιμές. Γι' αυτό το λόγο μας ενδιαφέρει η δεύτερη κατηγορία, αφού θέλουμε να προβλέψουμε τις τιμές των μετρικών ενός συστήματος.

Ο MLP Regressor υλοποιεί ένα MLP όπως ακριβώς ορίστηκε παραπάνω και χρησιμοποιεί τον αλγόριθμο backpropagation (BK) για την εκπαίδευση των νευρώνων, ωστόσο δεν έχει συνάρτηση ενεργοποίησης στον επίπεδο εξόδου (χρησιμοποιεί την ταυτοτική συνάρτηση).

Κατά την εκπαίδευση η ενημέρωση των βαρών γίνεται με την μέθοδο BK που εκτελεί στοχαστική βαθμωτή κατάβαση (Stochastic Gradient Decent - SGD) στο χώρο των βαρών σύμφωνα με τον τύπο

$$w \leftarrow w - \eta \left(a \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w} \right)$$

όπου *w* είναι το βάρος, *η* ο ρυθμός μάθησης, *α* ο συντελεστής κανονικοποίησης και *Loss* είναι η συνάρτηση απωλειών που χρησιμοποιείται από το δίκτυο.

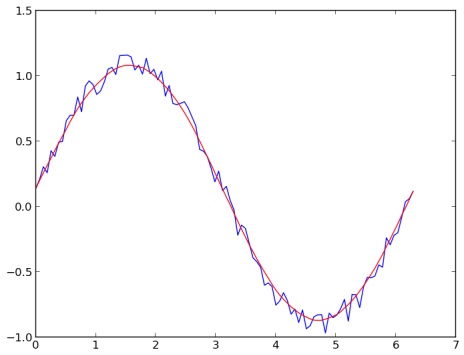
Η πολυπλοκότητα της διαδικασίας είναι $O(n \cdot m \cdot h^k \cdot o \cdot i)$, όπου *n* είναι τα στοιχεία του συνόλου εκπαίδευσης, *m* είναι το μήκος κάθε στοιχείου, *k* είναι το πλήθος των κρυφών επιπέδων, *h* ο αριθμός των νευρώνων σε κάθε ένα από αυτά και *o* ο αριθμός των νευρώνων εξόδου.

2.4.3 Εξομάλυνση κινητού μέσου όρου

Έστω μία χρονοσειρά ομοιόμορφα κατανεμημένων τιμών $x_i \equiv x(t_i)$, όπου $t_i \equiv t_0 + i\Delta$ για κάποιο σταθερό διάστημα δειγματοληψίας *Δ*. Μπορούμε τότε να

αντικαταστήσουμε κάθε τιμή x_i της χρονοσειράς με μία νέα $y_i = \frac{1}{n} \sum_{k=i-\frac{n-1}{2}}^{i+\frac{n-1}{2}} (x_k)$, όπου $n = 2k+1, \kappa \in N$ το μήκος του κινητού παραθύρου στο οποίο υπολογίζουμε το μέσο όρο. Προφανώς λαμβάνουμε υπόψιν ότι για τις ακριανές τιμές της χρονοσειράς το παράθυρο θα μεταβληθεί κατάλληλα ώστε να μην έχουμε δείκτες τιμών που δεν ορίζονται.

2.4.4 Φίλτρο Savitzky – Golay



Σχήμα 2.3 Μια χρονοσειρά (μπλε) και η εξομάλυνσή της (πράσινο).

Επειδή οι μετρικές που συλλέγονται από το σύστημα είναι χρονοσειρές που περιέχουν θόρυβο και επηρεάζονται από πολλούς διαφορετικούς παράγοντες αναπόφευκτα θα περιέχουν και διάφορα ανεπιθύμητα στοιχεία, που ονομάζονται ειδικά γεγονότα (special events), όπως για παράδειγμα απότομες διακυμάνσεις (overshoots), που δεν σχετίζονται με τους τελεστές που παρακολουθούμε και δυσκολεύουν την εκπαίδευση των νευρωνικών δικτύων.

Γι’ αυτό το λόγο γίνεται χρήση ενός φίλτρου εξομάλυνσης (smoothing filter) ώστε να απλοποιηθούν οι χρονοσειρές χωρίς όμως να αλλοιωθούν τα ποιοτικά τους χαρακτηριστικά αλλά και για να ενισχυθεί θετικά ο σηματοθορυβικός λόγος. Το φίλτρο που χρησιμοποιήθηκε είναι το ψηφιακό φίλτρο Savitzky-Golay.

Το φίλτρο αυτό λειτουργεί με τον τρόπο που περιγράψαμε για τον κινητό μέσο όρο, ωστόσο για να διατηρήσει τη μορφή της χρονοσειράς με καλύτερο τρόπο δεν χρησιμοποιεί έναν απλό μέσο όρο ή έναν μέσο όρο με βάρη αλλά υπολογίζει τη συνέλιξη της χρονοσειράς με ένα πολυώνυμο που υπολογίζεται με τη γραμμική μέθοδο ελαχίστων τετραγώνων μέσα σε ένα κινητό παράθυρο. Επειδή τα σημεία είναι

ισοκατανεμημένα λόγω του σταθερού ρυθμού δειγματοληψίας υπάρχει αναλυτική έκφραση του αποτελέσματος των ελαχίστων τετραγώνων με τη μορφή συντελεστών C_i , επομένως μπορούν να εφαρμοστούν επάνω στη χρονοσειρά για να δώσουν μια εκτίμηση για το κεντρικό σημείο του παραθύρου. Για μία χρονοσειρά με *n* σημεία έχουμε:

$$y_i = \sum_{i=-\frac{m-1}{2}}^{i=\frac{m-1}{2}} C_i x_{j+i} \quad , \quad \frac{m-1}{2} \le j \le n - \frac{m-1}{2} \quad ,$$

όπου *x* η παρατηρούμενη μεταβλητή, *m* το μήκος του παραθύρου, *C* οι συντελεστές και *Y* το εξομαλυνθέν σημείο.

2.4.5 Μέσο τετραγωνικό σφάλμα

Το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error, MSE) ενός διανύσματος εκτίμησης Y^{pred} και ενός διανύσματος παρατηρούμενων τιμών Y^{real} που αντιστοιχούν στις ίδιες εισόδους για τις οποίες προέκυψε η εκτίμηση ορίζεται ως εξής.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i^{pred} - Y_i^{real} \right)^2$$

Αντίστοιχα μπορούμε να ορίζουμε την ποσότητα

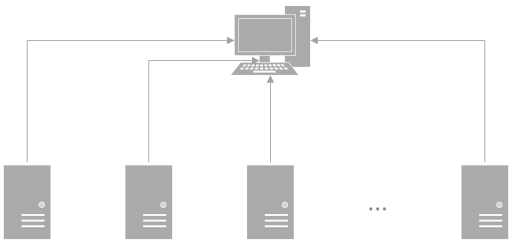
$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(Y_i^{pred} - Y_i^{real} \right)^2}$$

Που είναι η μη διορθωμένη τυπική απόκλιση ενός δείγματος.

3 Αρχιτεκτονική

Σε αυτό το κεφάλαιο περιγράφεται η αρχιτεκτονική του συστήματος και γίνεται αναλυτική παρουσίαση του σχεδιασμού του. Στο πρώτο κομμάτι του κεφαλαίου παρουσιάζεται η σχεδίασή του από μία πιο αφηρημένη σκοπιά, δηλαδή τα βασικά υποσυστήματα από τα οποία αποτελείται και οι σχέσεις μεταξύ αυτών.

3.1 Το Σύστημα



Σχήμα 3.1 Ο master κόμβος και ένας αριθμός slave κόμβοι συνδεδεμένοι.

Σε φυσικό επίπεδο το σύστημα αποτελείται από από ένα σύνολο slave κόμβων – υπολογιστών οι οποίοι επικοινωνούν με ένα master κόμβο – υπολογιστή που βρίσκεται στο ίδιο τοπικό δίκτυο. Οι slave κόμβοι είναι τα μηχανήματα που φέρουν την ευθύνη της εκτέλεσης ενός ή περισσότερων τελεστών καθώς και της αποστολής των μετρικών του συστήματος στον master κόμβο ο οποίος με τη σειρά του εκτελεί

τις λειτουργίες της συλλογής των δεδομένων, της μοντελοποίησής τους και βάσει αυτών του ελέγχου της λειτουργίας των τελεστών με στόχο τον εντοπισμό ανωμαλιών σε αυτήν.

Σε επίπεδο λογισμικού σε κάθε slave κόμβο εγκαθίσταται το Περιβάλλον Εκτέλεσης του Τελεστή (Execution Environment) το οποίο επικοινωνεί με τον master κόμβο ώστε να συγχρονίσει κατάλληλα την εκτέλεση του τελεστή. Επιπλέον εγκαθίσταται και λειτουργεί συνεχώς το πρόγραμμα παροχής μετρικών – στην παρούσα εργασία το Ganglia Monitoring System (GMS) – το οποίο συλλέγει και αποστέλλει τα δεδομένα.

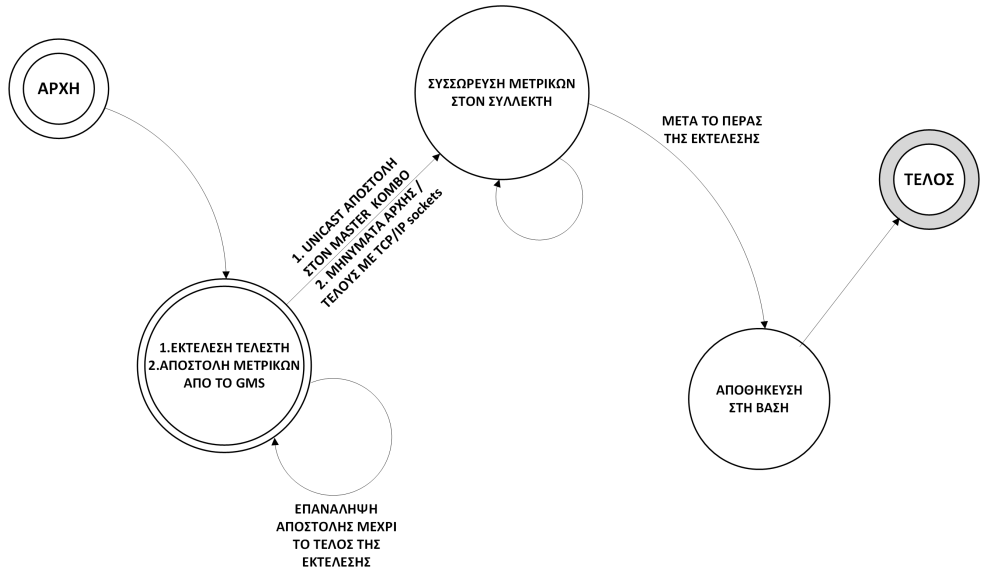
Ο master κόμβος του συστήματος έχει σαφώς περισσότερες αρμοδιότητες από τους υπόλοιπους. Τα υποσυστήματα που λειτουργούν σε αυτόν είναι ο Συλλέκτης Μετρικών (Mertics Collector) που επικοινωνεί με τούς slave κόμβους και συλλέγει τα δεδομένα που παράγονται κατά την εκτέλεση του τελεστή, ο Επιτηρητής Τελεστή (Operator Monitor) που κατά την online λειτουργία του συστήματος ελέγχει τη συμπεριφορά του τελεστή, ο Εκπαιδευτής Μοντέλων (Model Trainer) που βάσει τον μετρικών που έχουν συλλεχθεί κατασκευάζει και εκπαιδεύει μοντέλα που προβλέπουν τη συμπεριφορά του συστήματος που ελέγχεται και το Ganglia Monitoring System (GMS). Τέλος διατηρείται μια μη – σχεσιακή βάση δεδομένων (NoSQL) στην οποία αποθηκεύονται οι μετρικές, ώστε να είναι εύκολα προσβάσιμες και να μην χρειάζεται η επανάληψη της συλλογής τους που είναι ιδιαιτέρως χρονοβόρα.

3.2 Περιγραφή Λειτουργιών

3.2.1 Διαγράμματα Ροής Δεδομένων

Γενικά η λειτουργία του συστήματος μπορεί να χωριστεί σε δύο διαφορετικές φάσεις, τη διαδικασία συλλογής δεδομένων με στόχο την εκπαίδευση των μοντέλων (offline / δοκιμαστική λειτουργία) και την διαδικασία του ελέγχου του τελεστή σε πραγματικό χρόνο (online / κανονική λειτουργία), σύμφωνα με τα δεδομένα που έχουμε από την πρώτη, τα οποία θεωρούμε ότι περιγράφουν ορθά την κανονική λειτουργία του τελεστή και συνεπώς αποτελούν το μέτρο σύγκρισης για τον εντοπισμό ανωμαλιών.

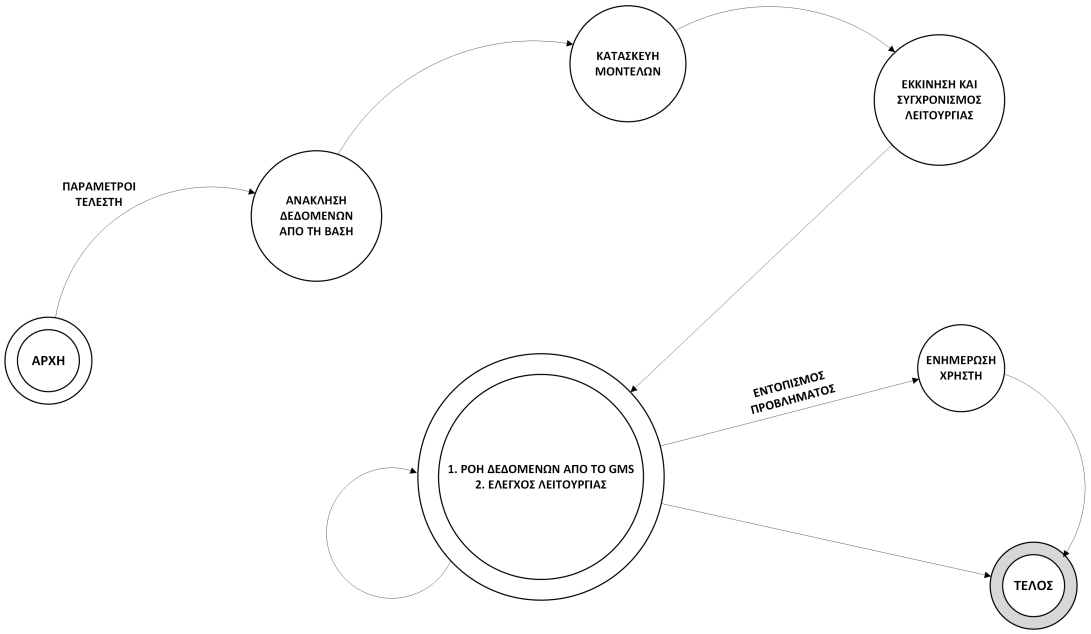
Το διάγραμμα ροής δεδομένων της πρώτης φαίνεται παρακάτω.



Σχήμα 3.2 Διάγραμμα ροής δεδομένων της διαδικασίας συλλογής μετρικών.

Όπως βλέπουμε στην αρχή έχουμε την εκτέλεση του τελεστή στον slave κόμβο και την ταυτόχρονη unicast αποστολή μετρικών από το GMS του slave στο GMS του master. Ο Συλλέκτης Μετρικών που τρέχει κεντρικά συσσωρεύει για όλη τη διάρκεια της εκτέλεσης τα δεδομένα και μετά το πέρας αυτής τα αποθηκεύει στη βάση δεδομένων.

Τώρα για την δεύτερη διαδικασία έχουμε το παρακάτω σχήμα.



Σχήμα 3.3 Διάγραμμα ροής δεδομένων για την διαδικασία επιτήρησης.

Εδώ βλέπουμε ότι προηγείται η διαδικασία κατασκευής των αντίστοιχων μοντέλων. Μετά τον συγχρονισμό της εκτέλεσης έχουμε την συνεχή επιτήρηση μέχρι να εντοπιστεί κάποια ανωμαλία ή να τερματίσει ο τελεστής.

3.2.2 Ανάλυση Υποσυστημάτων

Σε αυτό το σημείο δίνεται μια σύντομη ανάλυση του σκοπού των επιμέρους υποσυστημάτων που αναφέρθηκαν στις προηγούμενες παραγράφους.

3.2.2.1 Περιβάλλον Εκτέλεσης του Τελεστή (Execution Environment)

Το Περιβάλλον Εκτέλεσης του Operator (Execution Environment) δημιουργήθηκε με στόχο την επίλυση του προβλήματος του συγχρονισμού της εκτέλεσής στον slave κόμβο και της συλλογής μετρικών ή του ελέγχου της ομαλής λειτουργίας από τον master κόμβο.

Το περιβάλλον εκτέλεσης ενεργοποιείται από τον χρήστη σε κάθε slave κόμβο και αναμένει εντολή από τον master κόμβο που περιέχει την εντολή με τις πληροφορίες για το ποιόν τελεστή και με ποίες παραμέτρους λειτουργίας θα τον εκτελέσει. Όταν λάβει την εντολή ξεκινάει την εκτέλεση ενημερώνοντας ταυτόχρονα τον master κόμβο ώστε ο δεύτερος να ξεκινήσει δική του λειτουργία και να επιτευχθεί ο

συγχρονισμός αυτών. Τέλος ενημερώνει τον master κόμβο για τον τερματισμό της διαδικασίας.

3.2.2.2 Συλλέκτης Μετρικών (Metrics Collector)

Ο Συλλέκτης Μετρικών είναι το υποσύστημα που βρίσκεται σε λειτουργία παράλληλα με την εκτέλεση του τελεστή στο περιβάλλον που περιγράφεται πιο πάνω. Αφού συγχρονίσει τη λειτουργία του στη συνέχεια επικοινωνεί με το πρόγραμμα παροχής μετρικών (GMS) και συλλέγει της μετρικές που αφορούν τον εκάστοτε κόμβο και τελεστή και στην συνέχεια συνδέεται με τη βάση δεδομένων όπου και τις αποθηκεύει. Η μορφή της βάσης που δεν ακολουθεί το σχεσιακό μοντέλο ορίζεται σε επόμενη παράγραφο.

3.2.2.3 Επιτηρητής Τελεστή (Operator Monitor)

Ο Επιτηρητής Τελεστή επιτελεί την βασική λειτουργία του συστήματος, δηλαδή τον έλεγχο σε πραγματικό χρόνο της λειτουργίας ενός τελεστή. Οι ενέργειες με τις οποίες επιτυγχάνεται αυτό μπορούν να συνοψιστούν στα εξής:

- Ενεργοποίηση του Εκπαιδευτή Μοντέλων δίνοντας ως παράμετρο τον τελεστή και το μηχάνημα στο οποίο θα τρέξει.
- Επικοινωνία με το πρόγραμμα παροχής μετρικών συστήματος.
- Σύγκριση των πραγματικών τιμών των μετρικών και του χρόνου λειτουργίας με τις προβλεπόμενες από τα μοντέλα.
- Ενημέρωση του χρήστη για παρατηρούμενες ανωμαλίες στην εκτέλεση και καταγραφή στοιχείων που τις περιγράφουν ώστε να παρουσιαστούν αν χρειαστεί.

3.2.2.4 Εκπαιδευτής Μοντέλων (Model Trainer)

Αυτό το τμήμα του προγράμματος είναι υπεύθυνο για την δημιουργία των μοντέλων που περιγράφουν τη συμπεριφορά ενός τελεστή. Συνεπώς ανάλογα με τον τύπο της μετρικής που καλείται να μοντελοποιήσει επιλέγει και την κατάλληλη μέθοδο γι’ αυτό το σκοπό, η οποία έχει προκύψει από την πειραματική αξιολόγηση του συστήματος στη φάση της σχεδίασής του.

3.2.2.5 Πρόγραμμα παροχής μετρικών συστήματος

Το πρόγραμμα που μας παρέχει μετρικές για την κατάσταση των υπολογιστικών πόρων του κάθε κόμβου παρέχεται από τον χρήστη. Στην παρούσα διπλωματική εργασία χρησιμοποιούμε μια third - party εφαρμογή, το Ganglia Monitoring System (GMS). Οι λειτουργίες του προγράμματος που χρησιμοποιούνται από το σύστημά μας είναι:

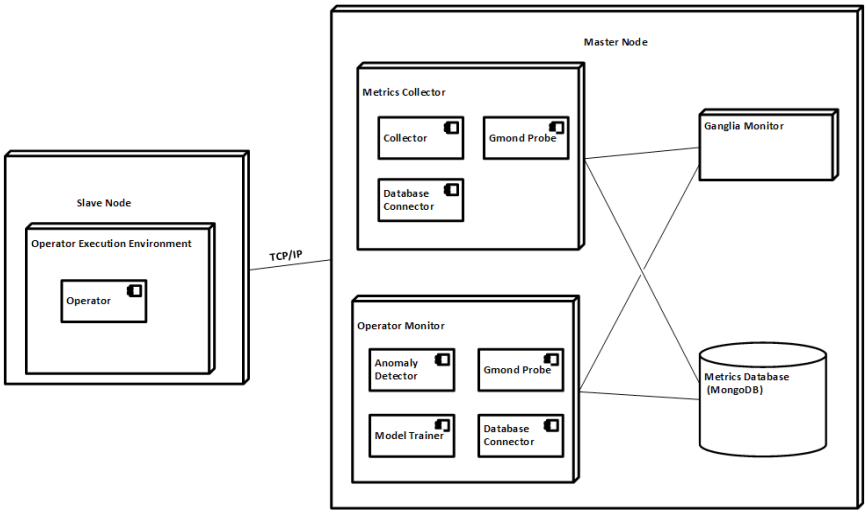
- Η μέτρηση της χρήσης υπολογιστικών πόρων σε κάθε μηχάνημα.
- Η αποστολή των μετρικών στο instance του Ganglia που τρέχει στον master κόμβο του συστήματος ώστε να καταστούν διαθέσιμες στον Συλλέκτη Μετρικών.

3.2.2.6 Βάση Δεδομένων για τις μετρικές

Εφόσον το σύστημα συλλέγει μετρικές στην πρώτη φάση της λειτουργίας του που θα χρησιμοποιηθούν αρκετές φορές στο μέλλον είναι απαραίτητη μια βάση δεδομένων για την αποθήκευσή τους και την εύκολη πρόσβαση σε αυτές με ερωτήματα. Επίσης μπορούμε να εξάγουμε τη βάση σε αρχείο backup για χρήση από νέο master κόμβο ή για την ανάκτησή της σε περίπτωση απώλειας του συστήματος.

3.3 Αρχιτεκτονική Λογισμικού

Στην προηγούμενη παράγραφο έγινε μια αφηρημένη – υψηλού επιπέδου ανάλυση των τμημάτων του συστήματος. Τώρα θα παρουσιαστούν οι συνιστώσες αυτών, ο τρόπος διασύνδεσής τους και διάφορες σχεδιαστικές λεπτομέρειες. Αρχικά παρατίθεται το αναπτυξιακό διάγραμμα του συστήματος.



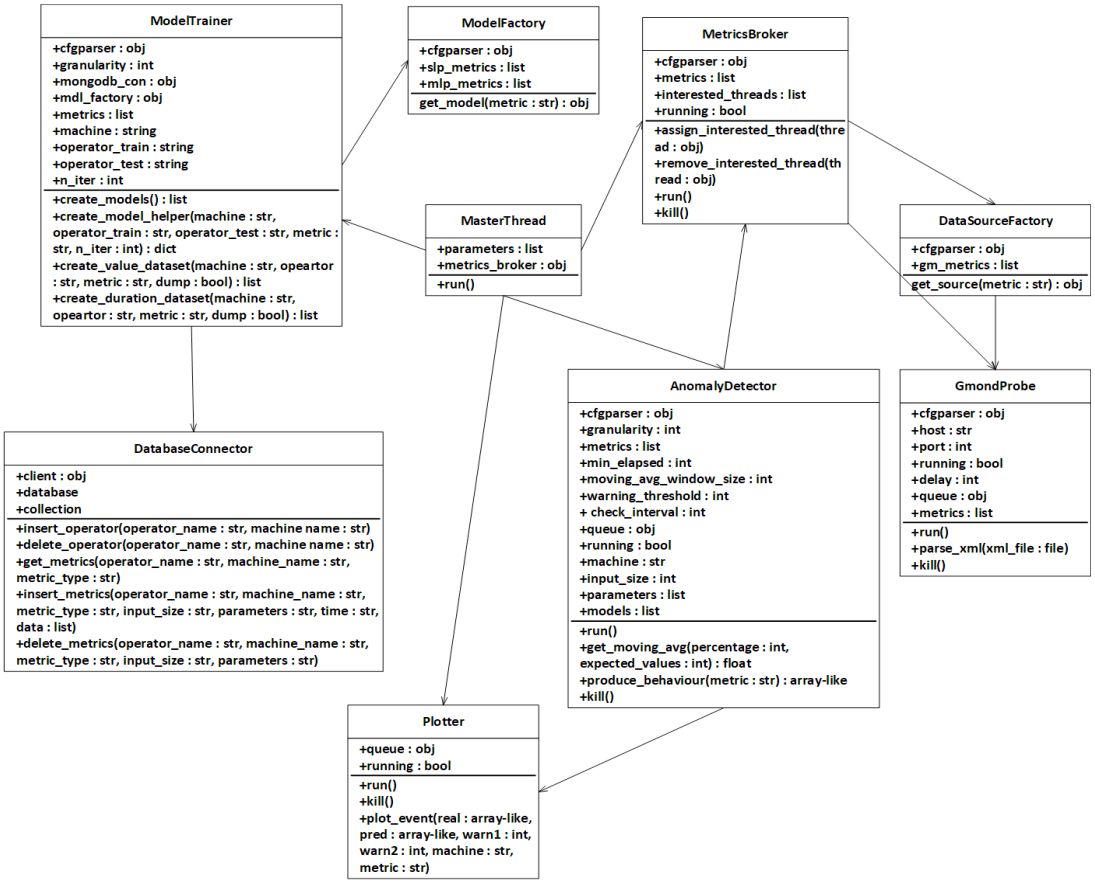
Σχήμα 3.4 Αναπτυξιακό διάγραμμα του συστήματος.

Όπως φαίνεται από το διάγραμμα, στον slave κόμβο υπάρχει ένα στοιχείο, το περιβάλλον εκτέλεσης του τελεστή και αυτή είναι η μορφή που έχουν όλοι οι υπολογιστές που βρίσκονται υπό επιτήρηση. Στον master κόμβο έχουμε τα module του Συλλέκτη Μετρικών, ο οποίος χρησιμοποιεί τις βασικές κλάσεις Collector, GmondProbe και DatabaseConnector, του Επιτηρητή του Τελεστή που περιέχει τις βασικές κλάσεις AnomalyDetector, GmondProbe, ModelTrainer και DatabaseConnector, το πρόγραμμα παροχής μετρικών GMS και η βάση δεδομένων MongoDB με τις μετρικές. Από αυτά διαπιστώνεται ότι η επιτήρηση γίνεται κεντρικά, από ένα μηχάνημα το οποίο περιέχει τα αντίστοιχα προγράμματα και τη βάση δεδομένων.

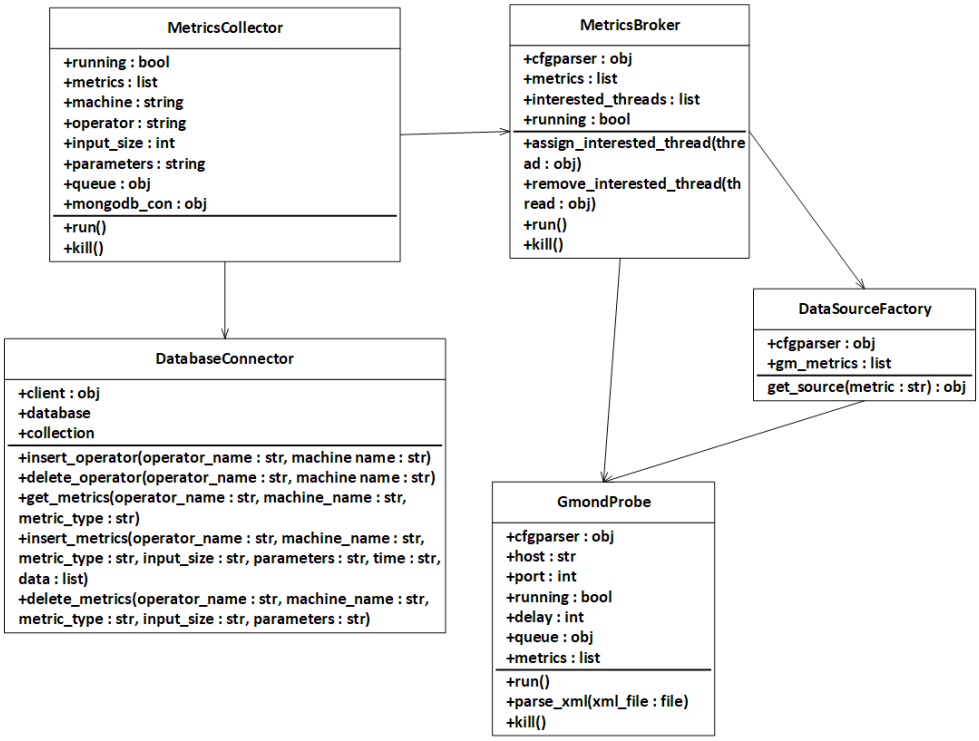
Σημειώνεται ότι στα modules που παρουσιάζονται αναφέρονται τα βασικά λειτουργικά στοιχεία και όχι όλα τα υποπρογράμματα και οι βιβλιοθήκες του συστήματος. Γι’ αυτά θα γίνει εκτενής παρουσίαση της υλοποίησης στο Κεφάλαιο 6.

3.4 Περιγραφή Κλάσεων

Ακολουθούν τα διαγράμματα κλάσεων της υλοποίησης του συστήματος και στη συνέχεια γίνεται μια ανάλυση των κλάσεων που κατασκευάστηκαν επεξηγώντας τα πεδία και τις συναρτήσεις που περιέχουν.



Σχήμα 3.5 Διάγραμμα κλάσεων του υποσυστήματος που κατασκευάζει τα μοντέλα πρόβλεψης και ελέγχει σε πραγματικό χρόνο τη λειτουργία του τελεστή (κανονική λειτουργία).



Σχήμα 3.6 Διάγραμμα κλάσεων του υποσυστήματος που συλλέγει μετρικές και τις αποθηκεύει στη βάση δεδομένων (δοκιμαστική λειτουργία).

3.4.1 Model Trainer

Η κλάση ModelTrainer εκπαιδεύει τα νευρωνικά δίκτυα που μοντελοποιούν τη συμπεριφορά του τελεστή. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
granularity	Καθορίζει τη λεπτομέρεια της δειγματοληψίας των μετρικών.
mongodb_con	Παρέχει μια διαπροσωπεία για την επικοινωνία με τη βάση δεδομένων.
mdl_factory	Αντικείμενο που επιλέγει το κατάλληλο μοντέλο ανά μετρική.
metrics	Λίστα που περιέχει τους τύπους μετρικής που θα μοντελοποιηθούν.
machine	Όνομα του υπολογιστή όπου εκτελείται ο τελεστής.
operator_train	Όνομα του τελεστή για τη δημιουργία του συνόλου δεδομένων εκπαίδευσης
operator_test	Όνομα του τελεστή για τη δημιουργία του συνόλου δεδομένων ελέγχου.

n_iter	Αριθμός επαναλήψεων της διαδικασίας.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

create_models()	Ο κύριος επαναληπτικός βρόχος που δημιουργεί το μοντέλο κάθε μετρικής.
create_model_helper()	Βοηθητική συνάρτηση δημιουργίας μοντέλων.
create_value_dataset()	Κατασκευάζει το σύνολο δεδομένων εκπαίδευσης και ελέγχου για μοντέλα πρόβλεψης μετρικών.
create_duration_dataset()	Κατασκευάζει το σύνολο δεδομένων εκπαίδευσης και ελέγχου για την πρόβλεψη της διάρκειας λειτουργίας του τελεστή.

3.4.2 Model Factory

Κλάση που υλοποιεί το factory σχεδιαστικό μοτίβο με στόχο την επιλογή του κατάλληλου μοντέλου πρόβλεψης για κάθε μετρική. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
slp_metrics	Λίστα με τις μετρικές που μοντελοποιούνται από perceptron ενός επιπέδου.
mlp_metrics	Λίστα με τις μετρικές που μοντελοποιούνται από πολυεπίπεδο perceptron.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

get_model()	Επιστρέφει το instance του μοντέλου για την αντίστοιχη μετρική.

3.4.3 Metrics Broker

Η κλάση Metrics Broker διαμοιράζει τις μετρικές πολλών slave κόμβων στο κατάλληλο νήμα που τους επιβλέπει υλοποιώντας το broker προγραμματιστικό μοτίβο. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
metrics	Λίστα που περιέχει τους τύπους μετρικής που θα μοντελοποιηθούν.
interested_threads	Περιέχει μια λίστα με αναφορές σε ανεξάρτητα νήματα που παρακολουθούν διαφορετικούς υπολογιστές και τελεστές.
running	Bool μεταβλητή που καθορίζει τη λειτουργία του νήματος.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

assign_interested_thread()	Προσθέτει ένα καινούριο νήμα στη λίστα.
remove_interested_thread()	Διαγράφει ένα νήμα από τη λίστα.
run()	Ξεκινά τη λειτουργία της κλάσης.
kill()	Τερματίζει τη λειτουργία της κλάσης.

3.4.4 Data Source Factory

Το σύστημα είναι σχεδιασμένο ώστε να είναι επεκτάσιμο όσον αφορά στις πηγές δεδομένων που διαθέτει. Συνεπώς υπάρχει μία κλάση που υλοποιεί το προγραμματιστικό μοτίβο factory που ανάλογα το είδος της μετρικής επιλέγει και την πηγή της. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
gm_metrics	Λίστα που περιέχει τους τύπους μετρικής που παρέχει το Ganglia Monitoring System.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

get_source()	Επιστρέφει την πηγή της δοθείσας μετρικής.

3.4.5 Gmond Probe

Όπως αναφέρθηκε προηγουμένως, η κλάση DataSourceFactory επιστρέφει την πηγή μιας μετρικής. Για τις μετρικές που παρέχονται από το Ganglia Monitoring System η πηγή είναι η κλάση GmondProbe. Η κλάση αυτή επικοινωνεί με το σύστημα και φιλτράρει τις μετρικές που μας ενδιαφέρουν. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
host	Η διεύθυνση IP του ganglia monitoring daemon (gmond).
port	Η θύρα στην οποία ανοίγει ένα TCP socket για την επικοινωνία.
running	Bool μεταβλητή που καθορίζει τη λειτουργία του νήματος.
delay	Ρυθμός δειγματοληψίας.
queue	Ουρά (FIFO) στην οποία τοποθετούνται οι μετρικές.
metrics	Λίστα που περιέχει τους τύπους μετρικής που θα μοντελοποιηθούν.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

run()	Ξεκινά τη λειτουργία της κλάσης.
parse_xml()	Διατρέχει το αρχείο XML που παρέχει ο gmond και κρατά τις μετρικές που χρειάζονται.
kill()	Τερματίζει τη λειτουργία της κλάσης.

3.4.6 Master Thread

Η κλάση MasterThread τρέχει σε ένα ξεχωριστό νήμα και είναι υπεύθυνη για τον συγχρονισμό των διαφόρων λειτουργιών του ελέγχου του τελεστή. Σε αυτήν κατασκευάζονται τα instances των υπολοίπων κλάσεων και συγχρονίζεται η επικοινωνία μεταξύ master και slave κόμβων. Περιέχει τα εξής πεδία.

parameters	Λίστα με τις παραμέτρους εκτέλεσης του τελεστή.
metrics_broker	Instance της κλάσης MetricsBroker.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

run()	Ξεκινά τη λειτουργία της κλάσης.

3.4.7 Anomaly Detector

Αυτή είναι η κλάση που επιτελεί την ουσιαστική διαδικασία του ελέγχου της συμπεριφοράς του τελεστή βάσει των μοντέλων. Επίσης ενημερώνει το χρήστη για ό,τι παρατηρείται. Περιέχει τα εξής πεδία.

cfgparser	Parser του αρχείου παραμέτρων του προγράμματος.
granularity	Καθορίζει τη λεπτομέρεια της δειγματοληψίας των μετρικών.
metrics	Λίστα που περιέχει τους τύπους μετρικής που θα μοντελοποιηθούν.
min_elapsed	Ελάχιστος χρόνος που απαιτείται να εχει λειτουργήσει ο τελεστής πριν ξεκινήσει ο έλεγχος.
moving_avg_window_size	Μέγεθος του παραθύρου κινητού μέσου όρου.
warning_threshold	Κατώφλι συνεχόμενων σφαλμάτων μέχρι να εκδοθεί προειδοποίηση.
check_interval	Χρονικό διάστημα μεταξύ δύο διαδοχικών ελέγχων.
queue	Ουρά (FIFO) στην οποία τοποθετούνται οι μετρικές.
running	Bool μεταβλητή που καθορίζει τη λειτουργία του νήματος.
machine	Όνομα του υπολογιστή που τρέχει ο τελεστής.
input_size	Μέγεθος της εισόδου του τελεστή.
parameters	Παράμετροι λειτουργίας του τελεστή.
models	Λίστα με τα μοντέλα που έχουν κατασκευαστεί.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

run()	Ξεκινά τη λειτουργία της κλάσης.
get_moving_avg()	Επιστρέφει το αποτέλεσμα της εφαρμογής κινητού μέσου όρου επάνω στις προβλεπόμενες τιμές.
produce_behaviour()	Παράγει τη συνολική προβλεπόμενη συμπεριφορά κάνοντας ερωτήματα στο αντίστοιχο μοντέλο και σύμφωνα με την προβλεπόμενη διάρκεια του τελεστή.
kill()	Τερματίζει τη λειτουργία της κλάσης.

3.4.8 Database Connector

Αυτή η κλάση περιέχει τις βασικές συναρτήσεις για την διαχείριση της βάσης δεδομένων και την επικοινωνία με αυτή. Περιέχει τα εξής πεδία.

client	Instance του client για σύνδεση στη MongoDB.
database	Αναφορά στη βάση δεδομένων.
collection	Αναφορά στην συγκεκριμένη συλλογή αρχείων.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

insert_operator()	Κατασκευάζει ένα νέο αρχείο για τον τελεστή.
delete_operator()	Διαγράφει το αρχείο του τελεστή.
get_metrics()	Επιστρέφει τις μετρικές.
insert_metrics()	Εισάγει καινούριες μετρικές.
delete_metrics()	Διαγράφει τις μετρικές.

3.4.9 Metrics Collector

Αυτή η κλάση τρέχει σε ξεχωριστό νήμα και συλλέγει τις μετρικές κατά τη δοκιμαστική λειτουργία του τελεστή και τις αποθηκεύει στη βάση δεδομένων. Περιέχει τα εξής πεδία.

running	Bool μεταβλητή που καθορίζει τη λειτουργία του νήματος.
metrics	Λίστα που περιέχει τους τύπους μετρικής που θα μοντελοποιηθούν.
machine	Όνομα του υπολογιστή που τρέχει ο τελεστής.
operator	Το όνομα του τελεστή.
input_size	Μέγεθος της εισόδου του τελεστή.
parameters	Παράμετροι λειτουργίας του τελεστή.
mongodb_con	Παρέχει μια διαπροσωπεία για την επικοινωνία με τη βάση δεδομένων.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

run()	Ξεκινά τη λειτουργία της κλάσης.
kill()	Τερματίζει τη λειτουργία της κλάσης.

3.4.10 Plotter

Η κλάση αυτή τρέχει σε ξεχωριστό νήμα και δέχεται μέσω μίας ουράς αιτήματα για δημιουργία γραφικών απεικονίσεων των ανωμαλιών που εντοπίστηκαν.

Περιέχει τα εξής πεδία.

queue	Ουρά (FIFO) στην οποία τοποθετούνται οι πληροφορίες για την δημιουργία της κάθε γραφικής.
running	Bool μεταβλητή που καθορίζει τη λειτουργία του νήματος.

Επιπλέον περιέχει τις ακόλουθες συναρτήσεις.

run()	Ξεκινά τη λειτουργία της κλάσης.
kill()	Τερματίζει τη λειτουργία της κλάσης.
create_2d_plot()	Παράγει και αποθηκεύει τις γραφικές παραστάσεις.

3.5 Βάση Δεδομένων

Όπως αναφέρθηκε στην προηγούμενη παράγραφο το σύστημά μας χρησιμοποιεί μια μη-σχεσιακή βάση δεδομένων. Επιλέχθηκε αυτός ο τύπος βάσης διότι τα δεδομένα που έχουμε είναι απλές αριθμητικές τιμές που δεν μπορούν να χωριστούν σε οντότητες με σχέσεις μεταξύ τους. Έτσι για λόγους απλότητας και ευελιξίας χρησιμοποιείται μία NoSQL βάση, συγκεκριμένα η MongoDB, η οποία αποθηκεύει τις πληροφορίες σε JSON - τύπου αρχεία (BSON – Binary JSON) που ομαδοποιούνται σε συλλογές (Collections). Στην εφαρμογή χρησιμοποιείται μία τέτοια συλλογή αρχείων και το σχήμα αρχείου που ορίστηκε έχει ως εξής:

```
{
    id: <document_id>
    operator: <operator_name>
    machine: <machine_name>
    metrics:<metric_type>
    {
        <metric_type>:
```

```
        {
            size: <input_size>
            parameters: <execution_parameters>
            time: <execution_duration>
            data: <collected_metrics>
        }
    }
}
```

Κάθε καταχωρημένο αρχείο στη βάση έχει ένα id, το όνομα του τελεστή, το όνομα του υπολογιστή στον οποίο γίνονται οι μετρήσεις και στη συνέχεια ένα υποαρχείο για κάθε τύπο μετρικής που ελέγχουμε. Στο υποαρχείο αποθηκεύονται το μέγεθος της εισόδου του τελεστή, οι παράμετροι εκτέλεσής του, η διάρκεια της λειτουργίας και τα δεδομένα που συλλέγονται σε μορφή μονοδιάστατου πίνακα.

3.6 Αρχείο παραμέτρων

Το σύστημα διαθέτει επίσης ένα αρχείο παραμέτρων config.ini που περιέχει τις τιμές που μπορεί να αλλάξει ο χρήστης του προγράμματος. Η κωδικοποίηση του αρχείου έχει ως εξής.

```
; This is a comment.
[Section]
ParameterA: value
ParameterB: value1, value2, value3
...
```

Οι παράμετροι ομαδοποιούνται σε τομείς – Sections ανάλογα με το υποσύστημα που αφορούν, για παράδειγμα ο τομέας [Gmond] περιέχει πληροφορίες σύνδεσης στον gmond του GMS.

3.7 Αρχεία εισόδου

Για την λειτουργία τόσο του Συλλέκτη Μετρικών όσο και του Επιτηρητή του Τελεστή χρειάζεται να δοθούν ως είσοδος αρχεία με τις παραμέτρους εκτέλεσης του τελεστή. Τα αρχεία σε κάθε γραμμή περιέχουν και από μία εκτέλεση του τελεστή.

Για τον Συλλέκτη Μετρικών η μορφή είναι η εξής.

```
machine_name operator_name input_size <parameters 1 .. N> <terminal command>
...
```

Ενώ για τον Επιτηρητή του Τελεστή.

```
IP port machine_name trainset testset inputsize <parameters 1 .. N> \
<terminal command>
...
```

4 *Υλοποίηση*

Ως τώρα έχει γίνει εκτενής ανάλυση της διάρθρωσης του συστήματος και σε αυτό το κεφάλαιο γίνεται η παρουσίαση των λεπτομερειών της υλοποίησης όλων των στοιχείων του και των τεχνολογιών που χρησιμοποιούνται.

4.1 Λεπτομέρειες υλοποίησης

Η υλοποίηση του προγράμματος έγινε σε γλώσσα Python λόγω των βιβλιοθηκών που υπάρχουν και εξυπηρετούν τις ανάγκες του συστήματος, την απλότητα του κώδικα και την μεταφερσιμότητα και επεκτασιμότητα του τελικού αποτελέσματος. Ο πηγαίος κώδικας έχει διαμοιραστεί στα εξής python modules.

data_source_factory.py, ***gmond_probe.py,*** ***metrics_collector.py,*** ***model_factory.py,*** ***model_trainer.py,*** ***mongodb_connector.py,*** ***operator_exec.py,*** ***operator_monitor.py***	Modules που περιέχουν τις υλοποιήσεις των κλάσεων του προγράμματος, όπως αυτές έχουν περιγραφεί στο Κεφάλαιο 5 και υλοποιούν τα υποσυστήματα της αρχιτεκτονικής που αναλύεται στο Κεφάλαιο 4.
evaluation_utils.py	Βιβλιοθήκη συναρτήσεων για την αξιολόγηση της ακρίβειας του συστήματος.
machine_learning.py	Βιβλιοθήκη με τους ορισμούς των νευρωνικών δικτύων

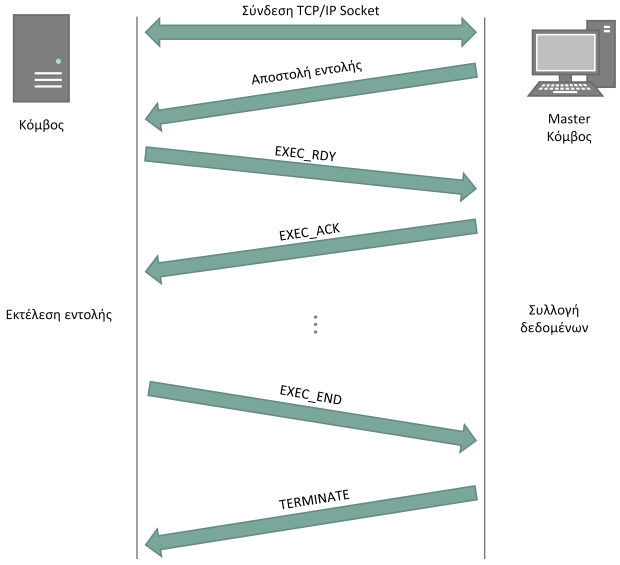
	και της διεπαφής τους.
model_utils.py	Βιβλιοθήκη με βοηθητικές συναρτήσεις για την κατασκευή των μοντέλων του συστήματος.

Τα προγράμματα του περιβάλλοντος εκτέλεσης του τελεστή και του εντοπισμού ανωμαλιών βρίσκονται αντίστοιχα στα αρχεία operator_exec.py και operator_monitor.py και αυτά τα δύο είναι που χρησιμοποιεί μέσω κονσόλας ο χρήστης.

Στη συνέχεια ακολουθεί ανάλυση διαφόρων σχεδιαστικών θεμάτων του προγράμματος.

4.1.1 Επικοινωνία master και slave κόμβων

Οι κόμβοι βρίσκονται στο ίδιο φυσικό δίκτυο και η επικοινωνία τους γίνεται προγραμματιστικά με TCP sockets. Ακολουθείται μια διαδικασία ανταλλαγής μηνυμάτων η οποία φαίνεται στο παρακάτω σχήμα με τα πιο πρόσφατα γεγονότα να βρίσκονται πιο πάνω από τα υπόλοιπα.



Σχήμα 4.1 Επικοινωνία μεταξύ του master κόμβου και ενός slave κόμβου.

Αρχικά ο slave κόμβος βρίσκεται σε κατάσταση αναμονής, δηλαδή έχει ξεκινήσει να τρέχει το Περιβάλλον Εκτέλεσης του Τελεστή και περιμένει για αίτημα σύνδεσης TCP socket στη θύρα 8888 από τον master κόμβο. Όταν συμβεί αυτό, ο master

κόμβος αποστέλλει την πλήρη εντολή εκτέλεσης του τελεστή σε ένα string, για παράδειγμα:

```
python kmeans.py input1.csv 16
```

Στη συνέχεια ο slave αποστέλλει ένα μήνυμα `EXEC_RDY` και ο master απαντά με `EXEC_ACK` και ξεκινά την λειτουργία του Συλλέκτη Μετρικών. Ταυτόχρονα, ο slave μόλις λάβει την απάντηση δημιουργεί μια νέα διεργασία και εκτελεί την εντολή που του δόθηκε. Ο λόγος που γίνεται η ανταλλαγή των μηνυμάτων `EXEC_RDY` και `EXEC_ACK` είναι ώστε να γίνει ένας απλός έλεγχος της σύνδεσης προτού ξεκινήσει η εκτέλεση και να τυπωθούν κατάλληλα μηνύματα ανάλογα με τον κωδικό επιστροφής των συναρτήσεων αποστολής.

Αφού ολοκληρωθεί η διαδικασία ο slave κόμβος αποστέλλει ένα μήνυμα `EXEC_END` ώστε να ενημερώσει τον master και εκείνος με τη σειρά του να τερματίσει τον Συλλέκτη Μετρικών και να αποθηκεύσει τα δεδομένα στη βάση.

Εάν δεν υπάρχουν άλλες εντολές προς εκτέλεση στο αρχείο που παρέχει ο διαχειριστής τότε ο master κόμβος στέλνει μήνυμα `TERMINATE` στον slave και μετά τα δύο προγράμματα τερματίζουν τη λειτουργία τους.

4.1.2 Η μορφή των δεδομένων

Η εκπαίδευση και αξιολόγηση των νευρωνικών δικτύων απαιτεί σύνολα δεδομένων την μορφής (X, Y) δηλαδή ζευγαριών παραμέτρων εισόδου του τελεστή και τιμής της αντίστοιχης μετρικής. Για τις ανάγκες του προγράμματος κατασκευάζονται δύο τέτοια σύνολα, ένα σύνολο εκπαίδευσης και ένα αξιολόγησης του νευρωνικού δικτύου. Αυτά τα ζευγάρια προκύπτουν από τις μετρικές που είναι αποθηκευμένες στη βάση δεδομένων και η ποσότητα τους εξαρτάται από το ρυθμό δειγματοληψίας του προγράμματος παροχής μετρικών.

Το σύστημα είναι σχεδιασμένο ώστε να κατασκευάζει σύνολα τα οποία έχουν σταθερό μέγεθος που ορίζεται από τη μεταβλητή granularity στο αρχείο παραμέτρων και ουσιαστικά αποτελεί τον μέγιστο ρυθμό με τον οποίο μπορει να δέχεται ερωτήματα το αντίστοιχο μοντέλο (και να μην επαναλαμβάνει την ίδια τιμή). Για παράδειγμα με τιμή 100 το σύστημα θα κάνει ερωτήματα στο νευρωνικό δίκτυο κάθε 1% της προβλεπόμενης διάρκειας του τελεστή και το σύνολο δεδομένων θα έχει 100 τιμές για κάθε εκτέλεση. Ωστόσο ο χρήστης μπορεί να επιλέξει όποιο ρυθμό ελέγχου

θέλει, απλώς αν είναι πολύ μικρός θα χρησιμοποιεί τις τιμές περισσότερες από μία φορά.

4.1.3 Νευρωνικά δίκτυα και διεπαφή που υλοποιούν

Τα νευρωνικά δίκτυα που χρησιμοποιούνται παρέχονται από το πακέτο λογισμικού SciKit Learn. Πιο συγκεκριμένα χρησιμοποιείται το Perceptron ενός επιπέδου της βιβλιοθήκης linear_model και το πολυεπίπεδο Perceptron (Regressor) της βιβλιοθήκης neural_network. Για κάθε νευρωνικό που έχει σχεδιαστεί ή που επιθυμεί να προσθέσει κανείς κατασκευάζεται μια νέα κλάση η οποία υποχρεωτικά υλοποιεί μια συγκεκριμένη διεπαφή[1], ώστε να μπορεί να αναγνωριστεί από το υπόλοιπο πρόγραμμα. Η κλάση που ακολουθεί το πρότυπο αυτό φαίνεται στον παρακάτω ψευδοκώδικα.

```
class NeuralModel()
     private model
     public train(x, y):
            self.model = SomeNeuralModel(params...)
     public predict(x):
            return self.model.SomePredictionMethod(x)
```

Όπως βλέπουμε πρέπει να παρέχονται δύο συναρτήσεις στο χρήστη. Η πρώτη είναι η `train(x, y)` η οποία κρύβει τη διαφορετική υλοποίηση και εκπαίδευση κάθε νευρωνικού δικτύου πίσω από το ίδιο όνομα και η δεύτερη είναι η `predict(x)` που επιστρέφει την πρόβλεψη για την είσοδο που παρέχεται. Επίσης υπάρχει η μεταβλητή `model` που αποθηκεύει για κάθε στιγμιότυπο της κλάσης το αποτέλεσμα της διαδικασίας εκπαίδευσης.

4.1.4 Διαδικασία εκπαίδευσης νευρωνικών δικτύων

Η διαδικασία εκπαίδευσης των νευρωνικών δικτύων δεν παράγει πάντοτε το ίδιο αποτέλεσμα, δηλαδή κάποια νευρωνικά δίκτυα αποδίδουν καλύτερα κατά την αξιολόγησή τους με το σύνολο δεδομένων ελέγχου σε σχέση με άλλα. Αυτό συμβαίνει διότι η μέθοδος εκπαίδευσης είναι στοχαστική (Stochastic Gradient

[1]Στην υλοποίηση που συνοδεύει την εργασία δεν υπάρχει κάποιο interface που υλοποιούν οι κλάσεις αφού αυτό δεν υποστηρίζεται από την Python αλλά είναι ευθύνη του προγραμματιστή να το τηρεί. Αναφέρεται όμως ως διεπαφή αφού είναι ένας αυστηρός κανόνας για τις συγκεκριμένες κλάσεις.

Decent). Γι’ αυτό το λόγο η διαδικασία εκπαίδευσης έχει τοποθετηθεί εντός επαναληπτικού βρόχου με παράμετρο n_iter και από αυτήν κρατείται το καλύτερο αποτέλεσμα από τις n_iter επαναλήψεις που γίνονται. Το κριτήριο βάσει του οποίου αξιολογείται ένα αποτέλεσμα είναι το ποσοστό των προβλέψεων του νευρωνικού δικτύου που βρίσκονται εντός μιας εμπειρικά επιλεγμένης απόστασης από τις πραγματικές τιμές, δηλαδή

$$\frac{S:|y_{pred}-y_{real}|<M}{S+S'}\cdot 100\,\%$$

όπου *S* τα στοιχεία για τα οποία ισχύει η συνθήκη και *S΄* τα υπόλοιπα.

4.1.5 Λειτουργικές δυνατότητες του προγράμματος ελέγχου

Έχει ειπωθεί στο πρώτο κεφάλαιο τις εργασίας ότι το σύστημα είναι επεκτάσιμο και ευέλικτο ως προς το τι επιτηρεί και από που δέχεται δεδομένα. Σε αυτό το σημείο θα δοθεί μια σύντομη εξήγηση για το πως επιτυγχάνεται αυτό.

Όπως έχει εξηγηθεί και προηγουμένως το πρόγραμμα αρχικά εκκινεί νήματα που τρέχουν την κλάση `MasterThread`. Ο αριθμός αυτών των νημάτων εξαρτάται από το αρχείο εισόδου με τις κλίσεις που επιθυμεί ο χρήστης να γίνουν. Κάθε τέτοιο νήμα συγχρονίζεται με την εκτέλεση ενός τελεστή σε ένα συγκεκριμένο υπολογιστή και με συγκεκριμένες παραμέτρους εισόδου. Συνεπώς μπορούμε να έχουμε οποιονδήποτε αριθμό τέτοιων νημάτων, φυσικά εντός των δυνατοτήτων του υπολογιστή που είναι ο master κόμβος, και να ελέγχουμε διαφορετικούς τελεστές ταυτόχρονα.

Για κάθε ένα νήμα εκτελείται στη συνέχεια και ένα νέο νήμα με την κλάση `AnomalyDetector` που χρησιμοποιεί τα μοντέλα που κατασκευάζονται ειδικά για τον συγκεκριμένο τελεστή, μηχάνημα και παραμέτρους εισόδου που δόθηκαν στο αντίστοιχο `MasterThread`. Το νήμα της κλάσης `AnomalyDetector` επικοινωνεί και με ένα νήμα `DataProvider` που ανάλογα με τις μετρικές που παρακολουθούνται μπορεί να συνδυάζει έναν αριθμό από διαφορετικές πηγές δεδομένων και να τις παρέχει συγκεντρωμένες στην διαδικασία ελέγχου.

4.1.6 Ο Αλγόριθμος Ελέγχου

Η μέθοδος με την οποία γίνεται ο έλεγχος της λειτουργίας είναι η ανά τακτά χρονικά διαστήματα σύγκριση των τιμών των μετρικών που έρχονται από τα προγράμματα παροχής μετρικών, δηλαδή οι πραγματικές τιμές τους, με τις προβλεπόμενες από τα

μοντέλα τιμές. Ο αλγόριθμος σε ψευδοκώδικα που ακολουθεί επιτελεί αυτή την εργασία. Το τμήμα αυτό διαθέτει από προηγούμενο σημείο του προγράμματος την προβλεπόμενη διάρκεια καθώς και την συνολική προβλεπόμενη συμπεριφορά του τελεστή με μορφή χρονοσειράς που έχει υποστεί δειγματοληψία σύμφωνα με την παράμετρο `granularity`.

```
// Check behaviour for anomalies.
while True:
    data = queue.get()
    elapsed = elapsed + 1
    if (elapsed > min_elapsed) and (elapsed % check_interval == 0):
        for metric in self.metrics:
            // Get real data and predicted data.
            real = get_real_data()
            pred = get_moving_avg(min(round(100 * elapsed / expected_duration)),
                100), expected_values[metric])

            // Compare the absolute differece between real and predicted values and
            // issue warnings if necessary.
            if abs(real - pred) > RMSE:
                warning[metric] += 1
            else:
                warning[metric] = 0

            // Second type of warning is issued if the problem persists.
            // Then plot the metric that caused the problem.
            if warning[metric] > 2 * warning_threshold:
                // Issue second warning.

            // First type of warning is issued after real values diverge from
            // predicted ones for more than the warning_threshold in seconds.
            else if warning[metric] > warning_threshold:
                // Issue first warning.


    # Check if duration is normal.
    if elapsed > (expected_duration + duration_error / 2):
        // Issue duration warning.
```

Αρχικά δίνεται ένα χρονικό διάστημα στο το οποίο ο τελεστής δεν ελέγχεται, ώστε να συσσωρευθεί ένας ικανός αριθμός μετρικών για να εκτελεστούν οι μέθοδοι ελέγχου. Αυτό ορίζεται από την μεταβλητή `min_elapsed`. Στη συνέχεια ξεκινά η διαδικασία που επαναλαμβάνεται σε διαστήματα διάρκειας που ορίζεται από το `check_interval`. Αυτές οι μεταβλητές δίνονται από το χρήστη στο αρχείο παραμέτρων.

Αφού παρέλθει αυτό το χρονικό διάστημα αρχίζει η σύγκριση της πραγματικής με την προβλεπόμενη τιμή. Ωστόσο επειδή η εκτέλεση του προγράμματος δεν έχει ακριβώς την ίδια διάρκεια με την προβλεπόμενη λόγω σφάλματος στην πρόβλεψη που παράγει το μοντέλο η σύγκριση δεν γίνεται μόνο με την τιμή που αντιστοιχεί στην κάθε δεδομένη χρονική στιγμή αλλά με τον μέσο όρο ενός κινητού παραθύρου (`get_moving_avg`) επάνω στην συνολική προβλεπόμενη συμπεριφορά.

Έτσι λοιπόν κάθε στιγμή που γίνεται έλεγχος υπολογίζεται το ποσοστό διάρκειας στο οποίο αντιστοιχεί αυτή, εντοπίζεται η κεντρική τιμή στο αντίστοιχο ποσοστό ολοκλήρωσης της εκτέλεσης (operator execution progress percentage) και γύρω από αυτήν υπολογίζεται ο μέσος όρος, το εύρος του οποίου καθορίζεται από την παράμετρο `moving_avg_window_size`.

Αυτός ο μέσος όρος συγκρίνεται για κάθε μετρική με την πραγματική τιμή και αν η διαφορά τους κατ’ απόλυτη τιμή ξεπερνά το προβλεπόμενο σφάλμα (RMSE) τότε έχει εντοπιστεί μια ανωμαλία στην συμπεριφορά του τελεστή. Αν αυτό επαναληφθεί συνεχόμενα για έναν αριθμό ελέγχων που ορίζεται από το `warning_threshold` τότε δίνεται μια πρώτη προειδοποίηση στον χρήστη. Αν συνεχιστεί και για άλλο τόσο δίνεται η δεύτερη προειδοποίηση και εκτυπώνεται μια γραφική παράσταση που παρουσιάζει την απόκλιση στην συμπεριφορά της αντίστοιχης μετρικής.

4.2 Προγραμματιστικά εργαλεία και εγκατάσταση

Η υλοποίηση έγινε με Python 2.7 σε περιβάλλον GNU/Linux (Ubuntu 16.04/17.10) με το εργαλείο Visual Studio Code που περιέχει ενσωματωμένα εργαλεία για την γλώσσα. Τα πακέτα που απαιτούνται για την εγκατάσταση και τη λειτουργία της εφαρμογής είναι τα εξής:

- ***Python*** (>=2.7)
- ***Numpy*** (>=1.8.2)
- ***Scipy*** (>=0.13.3)
- ***SciKit Learn*** (>=0.19.1)
- ***Matplotlib***
- ***PyMongo***(>=3.5.1)
- ***Ganglia Monitoring System*** (gmond, gmetad, web-frontend)

Επιπλέον απαιτούνται από το φυσικό τμήμα του συστήματος τα εξής:

- ***2 ή περισσότερα φυσικά μηχανήματα***
- ***Ένα τοπικό δίκτυο (LAN) στο οποίο βρίσκονται***

Σε αυτό το σημείο θα γίνει αναλυτική περιγραφή της διαδικασίας εγκατάστασης του συστήματος. Αρχικά ο χρήστης θα πρέπει να εγκαταστήσει από το repository της διανομής Linux που χρησιμοποιεί το Ganglia Monitoring System. Οι οδηγίες θα δοθούν για διανομές Ubuntu/Debian, ωστόσο η διαδικασία είναι ανάλογη και για τις υπόλοιπες.

Αρχικά εγκαθιστούμε στον master κόμβο την εφαρμογή.

```
sudo apt-get install -y ganglia-monitor rrdtool gmetad ganglia-webfrontend
```

Κατά την εγκατάσταση θα ζητηθεί από το χρήστη η επανεκκίνηση του Apache. Για όλες τις φορές που συμβεί αυτό πρέπει να επιλεχθεί yes. Στη συνέχεια ετοιμάζεται το γραφικό περιβάλλον αντιγράφοντας το αρχείο παραμέτρων του web-frontend του Ganglia στον φάκελο sites-enabled του Apache.

```
sudo cp /etc/ganglia-webfrontend/apache.conf /etc/apache2/sites-enabled/ganglia.conf
```

Στην συνέχεια ανοίγουμε με δικαιώματα υπερχρήστη το αρχείο /etc/ganglia/gmond.conf και ορίζουμε όνομα για το cluster υπολογιστών που θα επιβλέπει η εφαρμογή.

```
data_source "my cluster" localhost
```

Αποθηκεύουμε τις αλλαγές και ανοίγουμε το αρχείο /etc/ganglia/gmond.conf. Στο σημείο που ορίζεται το `cluster` εισάγουμε το ίδιο όνομα με αυτό που δώσαμε στο προηγούμενο αρχείο.

```
[...]
cluster {
  name = "my cluster" ## use the name from gmetad.conf
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
}
[...]
```

Στη συνέχεια στο σημείο `udp_send_channel` εισάγουμε μια γραμμή `host` και θέτουμε τη τιμή `localhost`. Επίσης διαγράφουμε τη γραμμή `mcast_join`.

```
[...]
udp_send_channel   {
  #mcast_join = 239.2.11.71 ## comment out
  host = localhost
  port = 8649
  ttl = 1
}
[...]
```

Στο σημείο `udp_recv_channel` αφαιρούμε τις γραμμές `mcast_join` και `bind`.

```
[...]
udp_recv_channel {
  #mcast_join = 239.2.11.71 ## comment out
  port = 8649
  #bind = 239.2.11.71 ## comment out
}

/* You can specify as many tcp_accept_channels as you like to share
```

```
   an xml description of the state of the cluster */
tcp_accept_channel {
  port = 8649
}
[...]
```

Τέλος επανεκκινούμε τα services των Ganglia-Monitor, Gmetad και Apache.

```
sudo service ganglia-monitor restart && sudo service gmetad restart
&& sudo service apache2 restart
```

Τώρα θα ασχοληθούμε με τους slave κόμβους του συστήματος. Η διαδικασία είναι η ίδια για όλους. Αρχικά εγκαθιστούμε το ganglia-monitor.

```
sudo apt-get install -y ganglia-monitor
```

Ύστερα ανοίγουμε το αρχείο /etc/ganglia/gmond.conf και ορίζουμε όνομα `cluster` όπως και πριν.

```
[...]
cluster {
  name = "my cluster"     ## Cluster name
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
[...]
```

Στο πεδίο `host` του `udp_send_channel` δίνουμε την διεύθυνση IP του master κόμβου. Επίσης αφαιρούμε το `mcast_join`.

```
[...]
udp_send_channel {
  #mcast_join = 239.2.11.71   ## Comment
  host = 1.1.1.1   ## IP address of master node
  port = 8649
  ttl = 1
}
[...]
```

Το τμήμα `udp_recv_channel` το αφαιρούμε ολόκληρο αφού οι slave κόμβοι δεν χρειάζεται να δέχονται μετρικές από άλλους κόμβους.

```
[...]
/* You can specify as many udp_recv_channels as you like as well.
udp_recv_channel {
  mcast_join = 239.2.11.71
  port = 8649
  bind = 239.2.11.71
}
*/
[...]
```

Επίσης θα πρέπει οπωσδήποτε η μεταβλητή `deaf` να τεθεί `true` γιατί σε αντίθετη περίπτωση το πρόγραμμα θα έχει δραματική επίπτωση στο ποσοστό χρήσης του επεξεργαστή καθιστώντας τον κόμβο αναξιόπιστο.

Τέλος επανεκκινούμε το service του Ganglia-Monitor.

```
sudo service ganglia-monitor restart
```

Μετά από μερικά δευτερόλεπτα το σύστημα είναι λειτουργικό και μπορούμε να δούμε τους υπολογιστές που παρακολουθούμε καθώς και γραφικές των μετρικών αν με τον φυλλομετρητή του master κόμβου ανοίξουμε τη διεύθυνση 127.0.0.1/ganglia.

Αυτό που έγινε στις ρυθμίσεις των slave κόμβων είναι η ρύθμιση του Ganglia σε unicast λειτουργία αποστολής δεδομένων, αφού το σύστημα εφαρμόζει κεντρικό έλεγχο των τελεστών.

Το επόμενο βήμα είναι η εγκατάσταση όλων των πακέτων που αναφέρονται στην αρχή της παραγράφου. Για τα πακέτα που αποτελούν βιβλιοθήκες της Python η εγκατάσταση μπορεί να γίνει πολύ εύκολα μέσω του pip, ωστόσο χρειάζεται προσοχή να γίνουν με δικαιώματα ***απλού χρήστη*** αλλιώς θα υπάρχουν προβλήματα με τα δικαιώματα του προγράμματος.

Ο χρήστης θα πρέπει στη συνέχεια να κατασκευάσει τη βάση δεδομένων για τις μετρικές. Η βάσει θα πρέπει να ονομαστεί gmdb και η συλλογή των αρχείων gmcol. Αυτό γίνεται από την κονσόλα του Linux. Αρχικά εκκινούμε το service της MongoDB.

```
sudo service mongod start
```

και εισερχόμαστε στο κέλυφος εντολών με την εντολή `mongo`.

Εκεί κατασκευάζουμε τη βάση και τη συλλογή ως εξής.

```
use gmdb
```

```
db.createCollection(“gmcol”)
```

και εξερχόμαστε από το κέλυφος με την εντολή `exit`.

Αφού εγκαταστήσουμε τα εργαλεία δημιουργούμε στον master κόμβο έναν φάκελο όπου επιθυμούμε και εισάγουμε όλα τα αρχεία του προγράμματος.

Στους slave κόμβους δημιουργούμε έναν φάκελο και αποθηκεύουμε τους τελεστές, τα αρχεία εισόδου τους και το πρόγραμμα του Περιβάλλοντος Εκτέλεσης του Τελεστή (operator_exec.py). Με δεδομένο ότι η σύνδεση μεταξύ των υπολογιστών είναι εφικτή και ότι το Ganglia και η MongoDB είναι λειτουργικά το πρόγραμμα είναι έτοιμο για χρήση.

5 *Έλεγχος*

Το σημαντικότερο σκέλος της παρούσας εργασίας είναι η πειραματική αξιολόγηση του συστήματος που προτάθηκε. Αυτό γίνεται με τη λειτουργική δοκιμή της εφαρμογής σε διάφορες συνθήκες εκτέλεσης τελεστών, με τη μελέτη της συμπεριφοράς του συστήματος όταν προκαλούνται εσκεμμένα ανωμαλίες με στόχο τον εντοπισμό αυτών και με την αξιολόγηση των επιμέρους υποσυστημάτων.

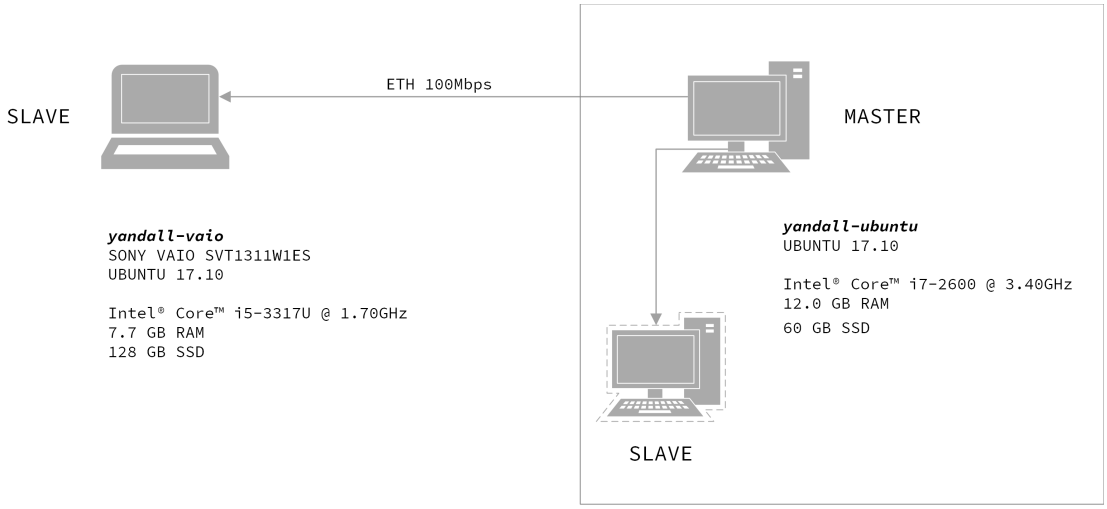
5.1 Μεθοδολογία ελέγχου

Η μεθοδολογία αποτελείται από την εκτέλεση μιας σειράς πειραμάτων με διάφορες παραμέτρους εισόδου, τελεστές και τύπους ανωμαλιών που επιθυμούμε να εντοπίσουμε. Η επιτυχία ή μη της λειτουργίας του συστήματος θα κριθεί από

- Την ακρίβεια πρόβλεψης των νευρωνικών δικτύων,
- την δυνατότητα του συστήματος να εντοπίσει τις ανωμαλίες που θα προκαλέσουμε στα πειράματα και
- την ικανότητα για γενίκευση σε διάφορους τελεστές και συστήματα.

Στα πειράματα που θα εκτελεστούν θα έχουμε 2 φυσικά μηχανήματα εκ των οποίων το ένα θα είναι ο master κόμβος που εφαρμόζει τον κεντρικό έλεγχο των εκτελέσεων και το δεύτερο είναι ο slave κόμβος όπου τρέχουν τους τελεστές.

Η διασύνδεση των υπολογιστών φαίνεται στο παρακάτω σχήμα.



Σχήμα 5.1 Διάταξη πειράματος με 2 υπολογιστές. Ο ένας είναι slave κομβος ενώ ο δεύτερος είναι slave και master.

Για την πειραματική μελέτη της ταυτόχρονης παρακολούθησης δύο συστημάτων ο master κόμβος χρησιμοποιήθηκε και ως slave.

Στην πρώτη φάση των πειραμάτων έχει γίνει συλλογή μετρικών για έναν αριθμό από εισόδους και παραμέτρους σε κάθε τελεστή. Οι τιμές αυτών αναφέρονται με λεπτομέρεια στην παράγραφο του κάθε πειράματος. Οι παράμετροι εκτέλεσης που είναι και οι τιμές του πίνακα εισόδου του νευρωνικού δικτύου καλύπτουν ένα ικανοποιητικό εύρος ώστε να είναι καλύτερα γενικευμένο το μοντέλο που προκύπτει.

Στη συνέχεια έγινε ο έλεγχος της εκπαίδευσης των μοντέλων με παραμέτρους παρεμφερείς αλλά όχι ίδιες με αυτές που χρησιμοποιήθηκαν για την συλλογή μετρικών και ελέγχθηκε η ακρίβεια των νευρωνικών δικτύων κατά την παραγωγή της συνολικής πρόβλεψης σύμφωνα με το ποσοστό accuracy, όπως ορίζεται στην παράγραφο 4.1.4. Η μετρικές και οι τιμές ακρίβειας που χρησιμοποιήθηκαν φαίνονται στον παρακάτω πίνακα.

Μετρική	***Accuracy***
cpu_user	3 %
mem_free	100000 KB
duration	50 sec

Ύστερα έγινε η εκτέλεση του τελεστή κατά τη διάρκεια της οποίας προκλήθηκαν τεχνητές ανωμαλίες. Όπως είδαμε ο αλγόριθμος ελέγχου χρησιμοποιεί τη ρίζα του μέσου τετραγωνικού σφάλματος (RMSE, παράγραφος 2.4.5) μεταξύ δεδομένων

εκπαίδευσης και ελέγχου ως την μέγιστη επιτρεπτή διαφορά που μπορεί να παρατηρηθεί. Σφάλματα μεγαλύτερα από αυτήν δύνανται να αποτελέσουν ανωμαλίες εκτέλεσης. Επειδή σε κάθε εκπαίδευση το σφάλμα έχει και διαφορετική τιμή, οι τιμές αναγράφονται σε κάθε πείραμα ξεχωριστά.

Οι μέθοδοι - μοντέλα που μελετήθηκαν είναι οι εξής.

Μέθοδος	***Πακέτο Λογισμικού***
Granger Causality Test	StatsModels
Perceptron	SciKit Learn (linear model)
MLP Regressor	SciKit Learn (neural networks)

Στα πειράματα έγινε συλλογή μετρικών για τους εξής δύο τελεστές.

Τελεστής	***Περιγραφή***
K-Means Clustering	Βασίζεται στην υλοποίηση του K-Means Clustering στο πακέτο SciKit Learn. Χρησιμοποιεί σύνολα σημείων αποθηκευμένα σε csv αρχεία.
Geographical Routing	Άπληστος αλγόριθμος δρομολόγησης που υλοποιήθηκε για τις ανάγκες της εργασίας. Ως είσοδο δέχεται τα αρχεία του προηγούμενου τελεστή τα οποία τα τροποποιεί ώστε τα σημεία που περιέχουν να συνδέονται μεταξύ τους.

5.2 Αναλυτική παρουσίαση ελέγχου

Αρχικά παρατίθεται η πειραματική αξιολόγηση των μοντέλων και στη συνέχεια τα πειράματα εντοπισμού ανωμαλιών.

5.2.1 Αξιολόγηση Μοντέλων

5.2.1.1 Στατιστική μέθοδος Granger Causality Test.

Κατά την εκπόνηση της εργασίας μελετήθηκε η στατιστική μέθοδος αιτιότητας Granger (Granger Causality Test) η οποία κρίθηκε ανεπαρκής όπως αναφέρθηκε και στην εισαγωγή και δίνεται μια σύντομη εξήγηση γιατί συνέβη αυτό.

Δοθέντων δύο χρονοσειρών η μέθοδος Granger Causality Test είναι μία στατιστική μέθοδος η οποία υπολογίζει την πιθανότητα η μία χρονοσειρά να μπορεί να προβλέψει σωστά την άλλη και μάλιστα γίνεται κύλιση της χρονοσειράς προς τα πίσω μέσα σε ένα ορισμένο διάστημα ώστε να βρεθεί το σημείο στο οποίο μεγιστοποιείται η πιθανότητα, επομένως μπορεί αν εφαρμοστεί και σε χρονοσειρές που έχουν μια χρονική διαφορά μεταξύ τους.

Πειραματικά έγινε έλεγχος της υλοποίησης της μεθόδου στη βιβλιοθήκη StatsModels 0.8.0 για τη γλώσσα Python. Οι λόγοι για τους οποίους δεν χρησιμοποιήθηκε η μέθοδος τελικά είναι οι εξής.

- Για να προκύψει μεγάλη πιθανότητα ταύτισης θα πρέπει οι χρονοσειρές να έχουν ελάχιστες διαφορές, συνεπώς ο θόρυβος των μετρικών προκαλούσε μη ακριβή αποτελέσματα.
- Σε κάθε έλεγχο θα πρέπει να ελέγχεται όλο το μήκος των χρονοσειρών και μάλιστα δύο φορές αφού η κύλιση της μιας χρονοσειράς γίνεται μόνο προς μια κατεύθυνση. Συνεπώς θα μπορούσαν δυο χρονοσειρές (X, Y) να είναι ίδιες αλλά η σύγκριση X με Y να δίνει άλλο αποτέλεσμα από την Y με τη X.
- Υπήρχε αρκετή τυχαιότητα στα αποτελέσματα της μεθόδου καθιστώντας την αναξιόπιστη. Επομένως δεν έδινε καλύτερα αποτελέσματα από μόνη της ή σε συνδυασμό με ένα νευρωνικό δίκτυο απ’ ότι ένα νευρωνικό δίκτυο μόνο του. Η χρήση της θα προκαλούσε απλώς συμφόρηση του κώδικα και γι’ αυτό τελικά δε χρησιμοποιήθηκε.

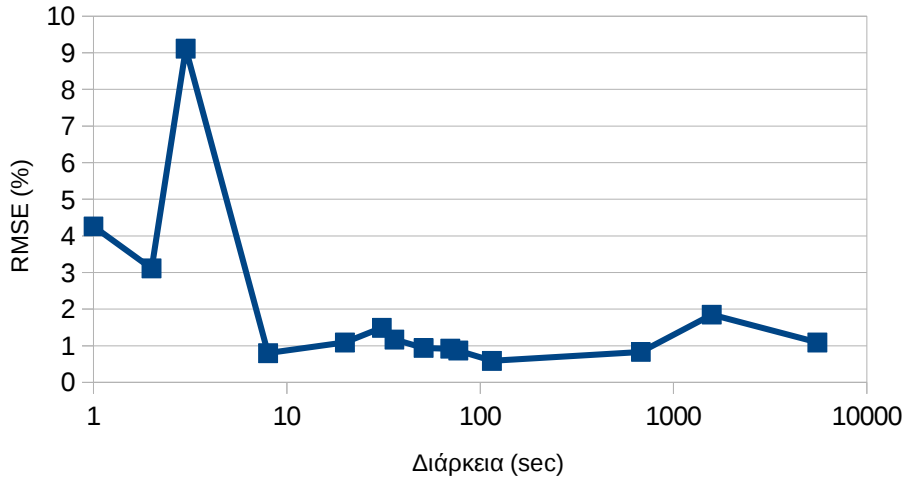
5.2.1.2 Νευρωνικά δίκτυα

Η αξιολόγηση των νευρωνικών δικτύων έγινε με μετρικές εκτέλεσης του τελεστή kmeans clustering από το πακέτο SciKit Learn για Python. Οι παράμετροι που χρησιμοποιήθηκαν για την κατασκευή των συνόλων δεδομένων εκπαίδευσης και ελέγχου φαίνονται στον παρακάτω πίνακα.

	Εκπαίδευση	***Έλεγχος***
Μέγεθος εισόδου	14 csv αρχεία με μέγεθος 4.1 kB ~ 200.3 MB	16 csv αρχεία με μέγεθος 7.2 kB ~ 313.0 MB
Αριθμός clusters	2, 4, 8, 16, 32	3, 6, 10, 18, 24
Δειγματοληψία	1 sec	1 sec

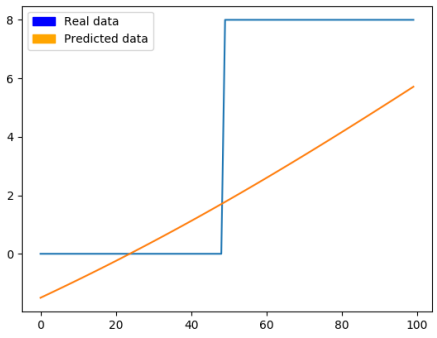
Η εκπαίδευση έγινε με παράμετρο επανάληψης `n_iter = 10` και κρατήθηκε η καλύτερη εξ' αυτών.

Η ακρίβεια του συστήματος βασίζεται στο RMSE, αφού αυτό αποτελεί το κριτήριο ελέγχου του τελεστή. Ανάλογα με το μέγεθος της εισόδου μεταβάλλεται και το RMSE. Η παρακάτω καμπύλη δείχνει αυτή ακριβώς τη μεταβολή η οποία προέκυψε από την εκπαίδευση των νευρωνικών δικτύων με τις παραμέτρους που μόλις αναφέρθηκαν για τη μετρική cpu_user.

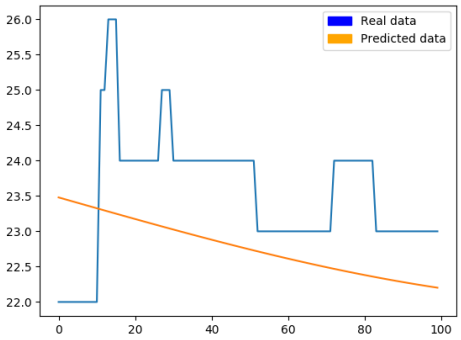


Σχήμα 5.2 Η μεταβολή του RMSE σε σχέση με το μέγεθος εισόδου του τελεστή (πρόβλεψη για cpu_user).

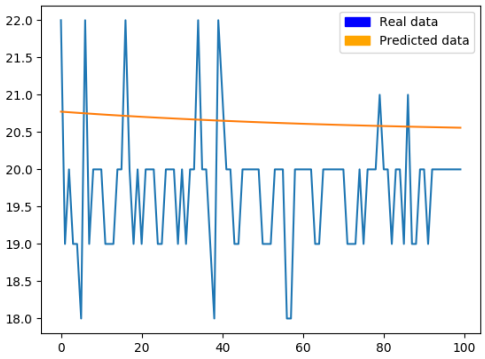
Όπως βλέπουμε το σφάλμα είναι μεγαλύτερο για μικρά μεγέθη εισόδου. Το αποτέλεσμα που έχει το γεγονός αυτό φαίνεται στις παρακάτω τρεις προβλέψεις για την μετρική cpu_user.



Σχήμα 5.3 (α) Πρόβλεψη και πραγματικές τιμές για εκτέλεση διάρκειας 2 sec, RMSE=3.25%.



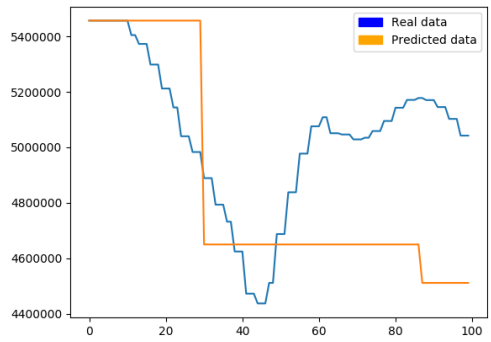
Σχήμα 5.3 (β) Εκτέλεση διάρκειας 36 sec, RMSE=1.17%.



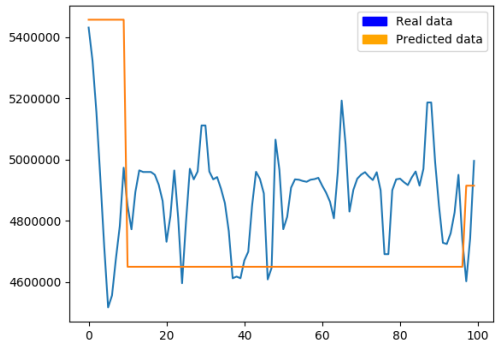
Σχήμα 5.3 (γ) Εκτέλεση διάρκειας 5549 sec, RMSE=0.98%.

Παρατηρούμε ότι η επίδοση του νευρωνικού είναι πολύ καλύτερη για μεγαλύτερα μεγέθη εισόδου, γεγονός που αναμενόταν αφού ο τελεστής γι' αυτές τις εισόδους είχε μεγαλύτερη διάρκεια και συνεπώς υπήρχε περισσότερη πληροφορία εκπαίδευσης. Για πολύ μικρό μέγεθος εισόδου η διάρκεια ήταν λίγα δευτερόλεπτα (<10) και πολλές τιμές επηρεάζονταν από θόρυβο στο σύστημα. Η περιοχή όμως που ενδιαφέρει έδωσε αποτελέσματα με RMSE της τάξης του 1%-2% της χρήσης του επεξεργαστή που είναι αποδεκτό για την εφαρμογή.

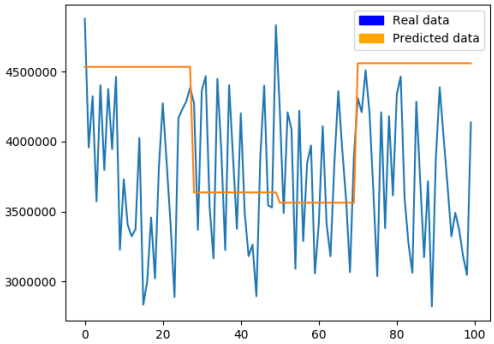
Η αντίστοιχη γραφική για την ελεύθερη μνήμη του συστήματος που δίνεται από τη μετρική mem_free είναι η εξής.



Σχήμα 5.4 (α) Πρόβλεψη και πραγματικές τιμές για εκτέλεση διάρκειας 36 sec, RMSE=361MB



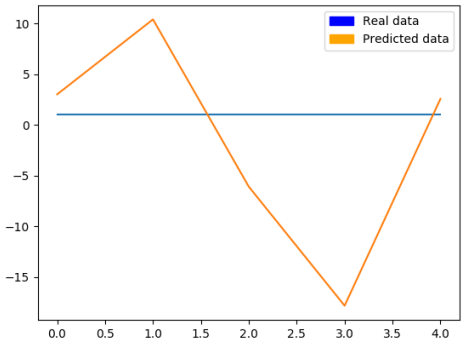
Σχήμα 5.4 (β) Εκτέλεση διάρκειας 264 sec, RMSE=321MB.



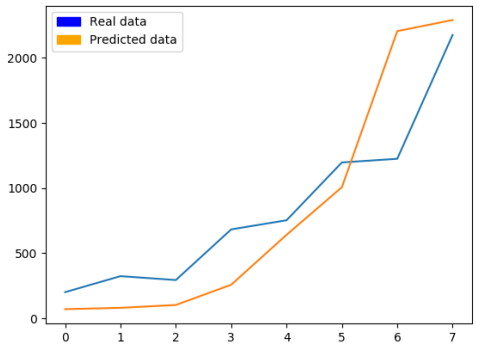
Σχήμα 5.4 (γ) Εκτέλεση διάρκειας 1196 sec, RMSE=783MB.

Όπως βλέπουμε η μετρική mem_free δεν έχει την ίδια συμπεριφορά ως προς το RMSE, ωστόσο μπορούμε να τη χρησιμοποιήσουμε για να παρακολουθήσουμε την μνήμη του συστήματος και να εντοπίσουμε ανωμαλίες που την αφορούν (memory leaks).

Όσα αναφέρθηκαν αφορούν σε προσπάθειες πρόβλεψης των μετρικών ενός συστήματος. Η δεύτερη εξίσου σημαντική πρόβλεψη που πρέπει να γίνει αφορά στην διάρκεια του τελεστή. Το αντίστοιχο νευρωνικό δίκτυο για παραμέτρους εκπαίδευσης ίδιες με πριν δίνει το ακόλουθο αποτέλεσμα.



Σχήμα 5.5 (α) Πρόβλεψη και πραγματικές τιμές διάρκειας τελεστή k-means για 5 εκτελέσεις με cluster=3,5,9,18,24 μεγέθους εισόδου 7.2KB. RMSE = 10 sec.



Σχήμα 5.5 (β) Η ίδιες προβλέψεις για 24 clusters και μεγέθη εισόδου 28, 41, 55, 91, 120, 155, 221 και 263 MB. RMSE = 406 sec.

Όπως βλέπουμε το νευρωνικό δίκτυο δίνει καλύτερα αποτελέσματα για εκτελέσεις με μεγάλη διάρκεια ενώ για αυτές που διαρκούν λίγα δευτερόλεπτα δίνει απροσδιόριστες τιμές. Αυτές όμως είναι τόσο σύντομες ώστε να μην δικαιολογείται η παρακολούθησή τους για ανωμαλίες. Επίσης παρατηρείται ένα σχετικά μεγάλο σφάλμα, γι’ αυτό το λόγο δίνεται αρκετό περιθώριο στο σφάλμα του ελέγχου της διάρκειας εκτέλεσης ώστε να μην γίνεται ενημέρωση του χρήστη για ανωμαλίες που δεν υφίστανται.

5.2.2 Αξιολόγηση Συστήματος

Η αξιολόγηση του συστήματος γίνεται με μία σειρά από εκτελέσεις τελεστών στις οποίες εντοπίζονται ανωμαλίες. Για καθένα από αυτά παρουσιάζεται η δομή του συστήματος, τα χαρακτηριστικά των υπολογιστών που συμμετέχουν, οι παράμετροι των τελεστών και του προγράμματος ελέγχου και οι γραφικές παραστάσεις της συμπεριφοράς τους.

Επίσης το πρώτο πείραμα αποτελεί και ένα εγχειρίδιο χρήσης αφού περιγράφονται και οι ενέργειες του χρήστη για την εκτέλεσή του.

5.2.2.1 Ανωμαλίες στη λειτουργία του επεξεργαστή

Σε αυτή την κατηγορία πειραμάτων το σύστημα αποτελείται από δύο μηχανήματα που επικοινωνούν μέσω δικτύου Ethernet. Ο ένας υπολογιστής εκτελεί τον τελεστή και ο δεύτερος παρακολουθεί την λειτουργία του.

Ο τελεστής του πειράματος είναι ο αλγόριθμος kmeans clustering από το πακέτο SciKit Learn. Η είσοδος του αλγορίθμου είναι αρχεία CSV που περιέχουν σύνολα τυχαίων σημείων του καρτεσιανού επιπέδου στα οποία γίνεται ομαδοποίηση. Οι παράμετροι εκπαίδευσης των νευρωνικών δικτύων είναι οι εξής.

	Εκπαίδευση	***Έλεγχος***
Μέγεθος εισόδου	14 csv αρχεία με μέγεθος 4.1 kB ~ 200.3 MB	16 csv αρχεία με μέγεθος 7.2 kB ~ 313.0 MB
Αριθμός clusters	2, 4, 8, 16, 32	3, 5, 9, 18, 24
Δειγματοληψία GMS	1 sec	1 sec
Granularity	100	100

Ενώ για τον έλεγχο του τελεστή έχουμε τις ακόλουθες ρυθμίσεις.

Παράμετρος	***Τιμή***
Παράθυρο μέσου όρου	5 μετρήσεις
Ρυθμός ελέγχου	2 sec
Κατώφλι σφάλματος	5 έλεγχοι
RMSE	9.54%

Ο χρήστης εκκινεί το Περιβάλλον Εκτέλεσης στον slave κομβο με την εντολή:

```
python operator_exec.py
```

Οπότε το πρόγραμμα ανταποκρίνεται ως εξής.

```
Creating server socket.
Waiting for connections...
```

Στη συνέχεια εκτελείται στον master κόμβο η εντολή:

```
python operator_monitor.py kmeans_demo.txt
```

Όπου το αρχείο που δίνεται ως παράμετρος περιέχει όλες τις πληροφορίες για την εκτέλεση σε μια γραμμή:

```
192.168.1.14 8888 yandall-vaio kmeans_train kmeans_test 120326525
18 \ python kmeans.py 12t.csv 18
```

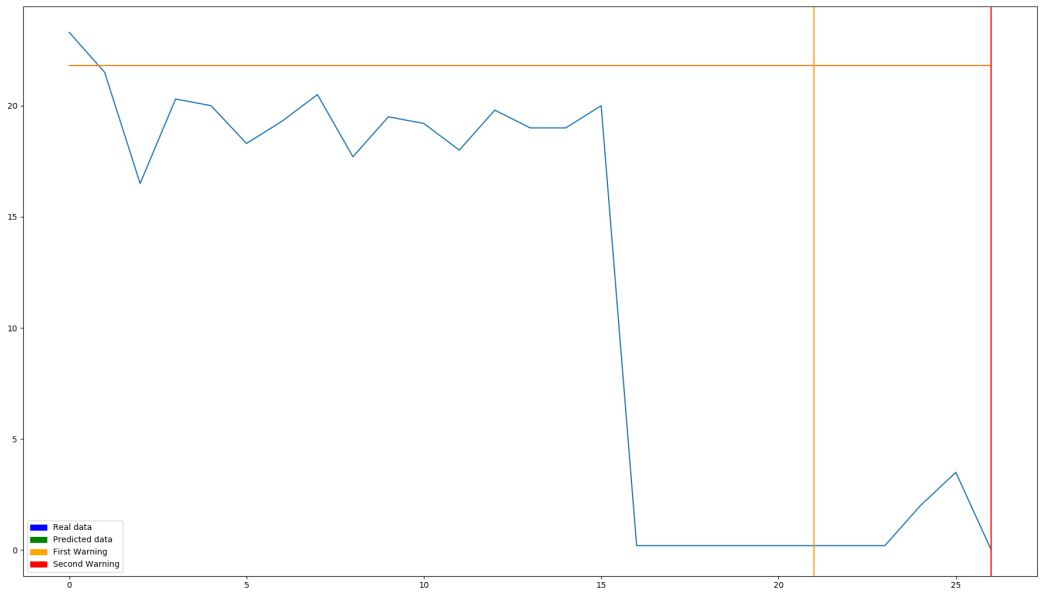
Αρχικά ξεκινά η εκτέλεση του τελεστή ενώ το πρόγραμμα ελέγχου δίνει μια περίοδο χάριτος 10 δευτερολέπτων ώστε να συσσωρευθούν αρκετές μετρικές και να ξεκινήσει ο εντοπισμός ανωμαλιών. Στο πρώτο πείραμα ελέγχονται μετρήσεις του ποσοστού χρήσης του επεξεργαστή σε user space (cpu_user) και ακολουθούνται δύο βασικά σενάρια τα οποία εφαρμόζονται με τις παραπάνω παραμέτρους στην ίδια διάταξη.

A. ***Εντοπισμός χαμηλού επιπέδου χρήσης επεξεργαστή*** το οποίο προκαλείται τεχνητά με την μη-κανονική παύση της λειτουργίας του τελεστή. Ο στόχος είναι να ελέγξουμε αν το πρόγραμμα μπορεί να εντοπίσει καταστάσεις στις οποίες ο τελεστής είναι είτε idle είτε έχει σταματήσει να λειτουργεί χωρίς να έχει ενημερωθεί με κάποιο τρόπο ο διαχειριστής του συστήματος.

Η εκτέλεση που ελέγχθηκε είναι η εξής.

Τελεστής	***Μέγεθος Εισόδου***	***Clusters***
kmeans.py	120 MBytes	18

Ο προβλεπόμενος χρόνος λειτουργίας για τον συγκεκριμένο υπολογιστή είναι 868 sec. Μετά από μερικά δευτερόλεπτα λειτουργίας τερματίστηκε απότομα η λειτουργία του τελεστή χωρίς να ενημερωθεί γι' αυτό ο master κόμβος. Η συμπεριφορά που παρατηρήθηκε είναι η εξής.



Σχήμα 5.6 Εκτέλεση τελεστή με απότομη διακοπή λειτουργίας. Στον κάθετο άξονα έχουμε ποσοστό χρήσης επεξεργαστή σε user space (cpu_user) και στον οριζόντιο ο αριθμός των ελέγχων από την έναρξή του.

Στην αρχή του ελέγχου οι γραφικές των πραγματικών και προβλεπόμενων μετρικών βρίσκονται σχετικά κοντά και συνεπώς οι απόσταση των σημείων δεν ξεπερνά το RMSE. Στη συνέχεια τερματίζεται η λειτουργία και το ποσοστό χρήσης του επεξεργαστή πέφτει κατακόρυφα προκαλώντας μία προειδοποίηση μετά από 5 δευτερόλεπτα και μια δεύτερη μετά από άλλα 5. Το συγκεκριμένο πρόγραμμα προσομοιώνει μια fail-slow[2] κατάσταση ενός τμήματος υλικού.

Τα μηνύματα που εμφανίζονται στην κονσόλα έχουν την ακόλουθη μορφή.

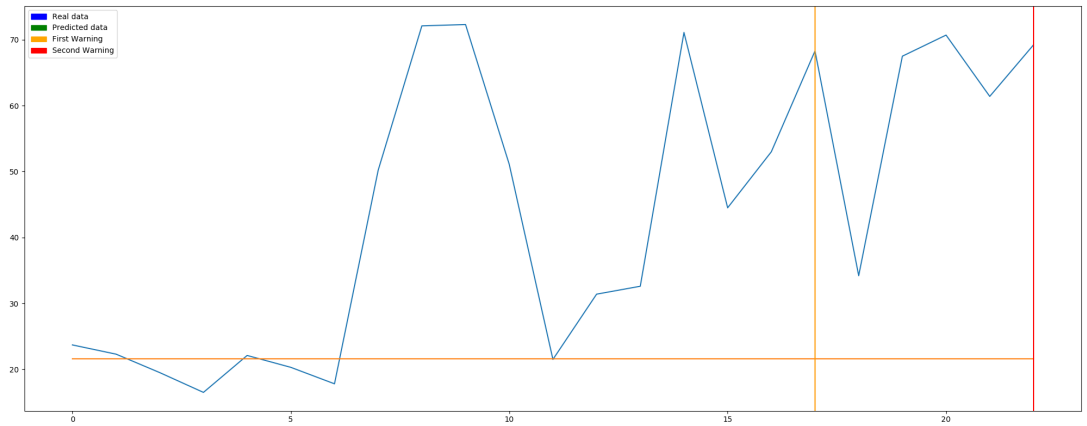
```
Real: 23.5, Predicted: 21.7.
Real: 21.5, Predicted: 21.7.
Real: 19.0, Predicted: 21.7.
Real: 0.5, Predicted: 21.7.
Real: 0.5, Predicted: 21.7.
```

[2] Κατάσταση στην οποία το υλικό υπολειτουργεί σε σημείο που να εμποδίζει τη σωστή λειτουργία του λογισμικού ωστόσο δεν εντοπίζεται εύκολα αφού δεν έχει διακοπεί πλήρως η λειτουργία του.

```
 Real: 0.5, Predicted: 21.9.
 Real: 0.5, Predicted: 21.9.
 Real: 0.5, Predicted: 21.9.
 Real: 0.5, Predicted: 21.9.
[WARNING] Possible operator issue according to cpu_user metrics.
 Real: 0.5, Predicted: 21.9.
[WARNING] Possible operator issue according to cpu_user metrics.
 Real: 0.5, Predicted: 22.1.
[WARNING] Possible operator issue according to cpu_user metrics.
 Real: 0.5, Predicted: 22.1.
[WARNING] Possible operator issue according to cpu_user metrics.
 Real: 0.5, Predicted: 22.1.
[WARNING] Possible operator issue according to cpu_user metrics.
 Real: 0.5, Predicted: 22.1.
[FAILURE] cpu_user metric constantly out of normal levels.
```

Β. ***Εντοπισμός υπερβολικής χρήσης του επεξεργαστή*** από τρίτο πρόγραμμα το οποίο προκαλεί απώλεια πόρων για τον τελεστή καθυστερώντας τη λειτουργία του.

Σε αυτό το σενάριο χρησιμοποιούμε τον ίδιο τελεστή και παραμέτρους με το προηγούμενο. Αρχικά επιτρέπουμε για κάποια δευτερόλεπτα την κανονική λειτουργία και μετά αυξάνουμε τεχνητά τη χρήση του επεξεργαστή με τυχαίο τρόπο. Η γραφική της συμπεριφοράς του συστήματος είναι η εξής.



Σχήμα 5.7 Εκτέλεση τελεστή με αύξηση της χρήσης του επεξεργαστή. Στον κάθετο άξονα έχουμε ποσοστό χρήσης επεξεργαστή σε user space (cpu_user) και στον οριζόντιο ο αριθμός των ελέγχων από την έναρξή του.

Παρατηρούμε ότι η έκδοση της πρώτης προειδοποίησης καθυστέρησε λίγο σε σχέση με πριν αφού για λίγο επανήλθε το σύστημα στα φυσιολογικά επίπεδα άλλα στη

συνέχεια συνέχισε να δίνει μετρικές που ξεπερνούν κατά πολύ το RMSE, γι’ αυτό και το σύστημα έδωσε τις δύο προειδοποιήσεις.

5.2.2.2 Δέσμευση μνήμης του συστήματος από άλλο πρόγραμμα

Σε αυτό το πείραμα παρακολουθείται η ελεύθερη μνήμη του συστήματος (mem_free) κατά την εκτέλεση του τελεστή kmeans clustering στην ίδια διάταξη με αυτή της παραγράφου 6.2.2.1. Οι μετρικές που έχουν συλλεχθεί προϋποθέτουν ότι ο τελεστής είναι το μοναδικό πρόγραμμα που δεσμεύει μνήμη στο σύστημα. Ο στόχος του πειράματος αυτού είναι να εντοπιστεί μια σταδιακή απώλεια της μνήμης του συστήματος που ενδέχεται να καθυστερήσει ή εμποδίσει τον επιτυχή τερματισμό του προγράμματος.

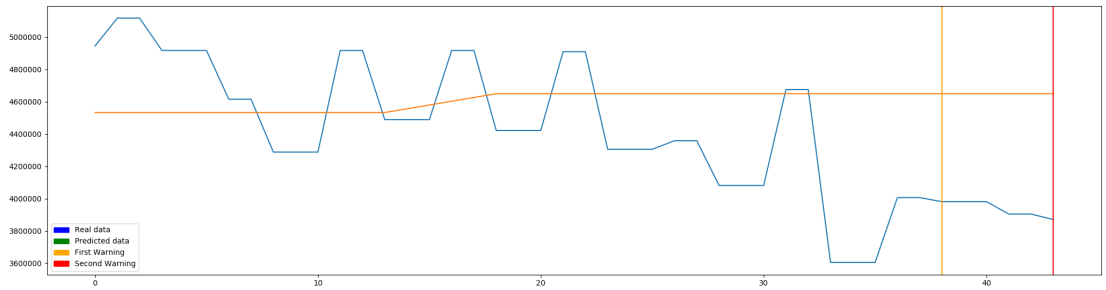
Οι παράμετροι εκπαίδευσης των νευρωνικών δικτύων είναι οι εξής.

	Εκπαίδευση	***Έλεγχος***
Μέγεθος εισόδου	14 csv αρχεία με μέγεθος 4.1 kB ~ 200.3 MB	16 csv αρχεία με μέγεθος 7.2 kB ~ 313.0 MB
Αριθμός clusters	2, 4, 8, 16, 32	3, 5, 9, 18, 24
Δειγματοληψία GMS	1 sec	1 sec
Granularity	100	100

Ενώ για τον έλεγχο του τελεστή έχουμε τα εξής.

Παράμετρος	***Τιμή***
Παράθυρο μέσου όρου	5 μετρήσεις
Ρυθμός ελέγχου	2 sec
Κατώφλι σφάλματος	5 έλεγχοι
RMSE	398 MB

Αρχικά αφήνεται ο τελεστής να λειτουργήσει χωρίς προβλήματα για 25 δευτερόλεπτα και στη συνέχεια γίνεται μια σταδιακή δέσμευση της μνήμης του συστήματος έως ότου εντοπιστεί η ανωμαλία.



Σχήμα 5.8 Παρακολούθηση της μετρικής mem_free. Στον κάθετο άξονα είναι η ελεύθερη μνήμη του συστήματος και στον οριζόντιο ο αριθμός των ελέγχων από την έναρξή του. Το πρόβλημα εισάγεται στο 26ο δευτερόλεπτο.

Όπως παρατηρούμε και σε αυτή την περίπτωση ήταν επιτυχής ο εντοπισμός του προβλήματος, ωστόσο χρειάστηκαν περισσότερα δευτερόλεπτα αφού η απόκλιση έγινε με πιο αργό ρυθμό.

5.2.2.3 Εγκλωβισμός τελεστή σε τοπικό ελάχιστο

Μία ειδική περίπτωση προβλήματος τελεστή είναι ο εγκλωβισμός του σε κάποια κατάσταση που εμποδίζει τον τερματισμό του. Η δοκιμή του προγράμματος σε μία τέτοια περίπτωση μπορεί να γίνει με έναν αλγόριθμο που μπορεί να εγκλωβιστεί σε κάποιο τοπικό ελάχιστο κατά τη διάρκεια εύρεσης κάποιας προσεγγιστικής λύσης σε ένα πρόβλημα.

Ο αλγόριθμος που ελέγχεται σε αυτό το πείραμα είναι η εύρεση ενός μονοπατιού σε γράφο με κριτήριο την ευκλείδεια απόσταση του εκάστοτε κόμβου από τον στόχο (geographical routing). Ένα μεγάλο ποσοστό προσπαθειών μπορεί να οδηγήσει σε ατέρμονα βρόχο αν δεν δίνεται η δυνατότητα να γίνει κάποια “κακή” επιλογή στην περίπτωση κύκλου (annealing). Ο τελεστής γράφτηκε ειδικά γι’ αυτό το πείραμα (δεν βρίσκεται σε κάποια βιβλιοθήκη ή πακέτο).

Σε αυτό το πείραμα έχουμε πάλι τη διάταξη της παραγράφου 6.2.2.1 και χρησιμοποιούνται μικρότερα και πιο απλά σύνολα εκπαίδευσης και ελέγχου που έχουν τις εξής παραμέτρους.

	Εκπαίδευση	***Έλεγχος***
Είσοδος	Γράφος μεγέθους 120.3 kB	Γράφος μεγέθους 120.3 kB
Αριθμός ζητούμενων μονοπατιών	5, 10, 15, 20	6, 9, 14, 21
Δειγματοληψία GMS	1 sec	1 sec
Granularity	100	100

Οι παράμετροι ελέγχου για το πείραμα είναι οι εξής.

Παράμετρος	***Τιμή***
Παράθυρο μέσου όρου	3 μετρήσεις
Ρυθμός ελέγχου	5 sec
Κατώφλι σφάλματος	5 έλεγχοι
RMSE διάρκειας	34 sec
RMSE cpu_user	2.63%
RMSE mem_free	20MB

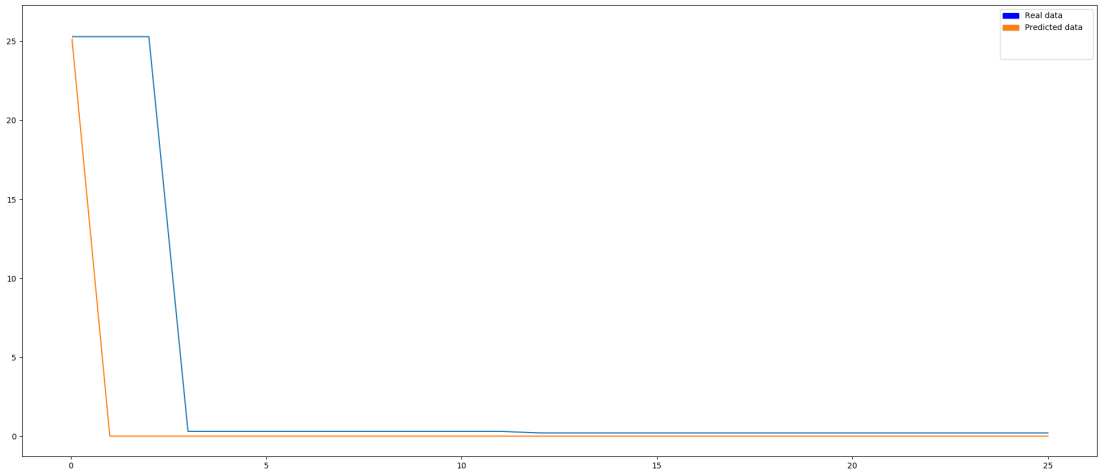
Η εκτέλεση που ελέγχθηκε είναι η εξής (με εισαγωγή μη επιλύσιμου μονοπατιού).

Τελεστής	***Μέγεθος Εισόδου***	***Paths***
geo_routing.py	120 KBytes	5

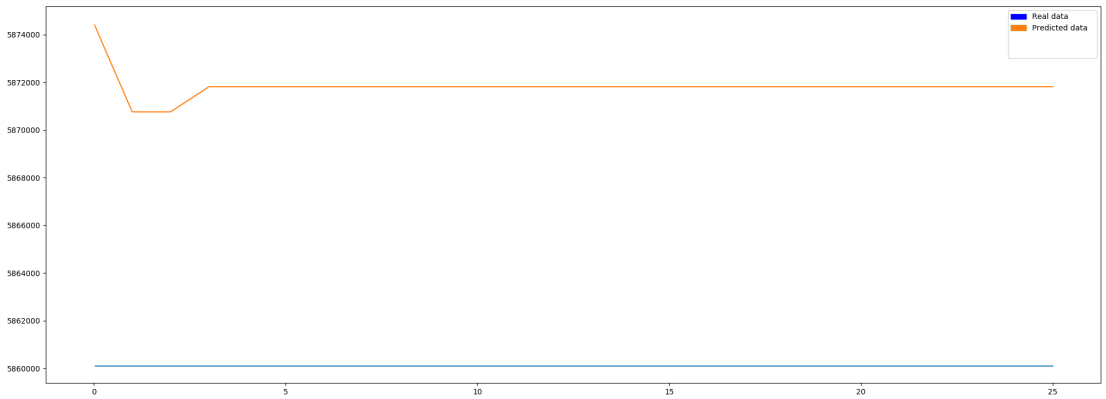
Στη συνέχεια γίνεται η εκτέλεση του προγράμματος ωστόσο ζητείται η εύρεση ενός μονοπατιού που οδηγεί σε εγκλωβισμό σε τοπικό ελάχιστο. Έτσι ο τελεστής αμφιταλαντεύεται μεταξύ δυο κόμβων σε έναν ατέρμονα βρόχο και συνεπώς προκαλείται εντοπισμός ανωμαλίας στη διάρκεια του τελεστή.

Στις παρακάτω γραφικές φαίνεται η συμπεριφορά για τις μετρικές ποσοστού χρήσης επεξεργαστή (cpu_user) και ελεύθερης μνήμης συστήματος (mem_free). Το πρόγραμμα δεν τοποθετεί τα σημεία έκδοσης προειδοποίησης αφού είναι μια διαρκής

κατάσταση . Αντιθέτως ενημερώνει τη χρήστη ότι έχει ξεπεραστεί ο προβλεπόμενος χρόνος εκτέλεσης μόνο μέσω της κονσόλας εντολών.



Σχήμα 5.9 Ποσοστό χρήσης επεξεργαστή (cpu_user). Στον οριζόντιο άξονα ο αριθμός των ελέγχων.



Σχήμα 5.10 Ελεύθερη μνήμη συστήματος (mem_free). Η απόσταση μεταξύ των δύο γραμμών είναι μικρή (~10-12 MB). Στον οριζόντιο άξονα ο αριθμός των ελέγχων.

Σε αυτές τις γραφικές παραστάσεις δεν έχουμε κάποια απόκλιση στη συμπεριφορά αλλά όπως αναφέρθηκε η διάρκεια ήταν μεγαλύτερη από τη προβλεπόμενη.

Η προβλεπόμενη διάρκεια ήταν 60 δευτερόλεπτα, ενώ η πραγματική τη στιγμή που εκτυπώθηκαν οι γραφικές παραστάσεις.

$$InitialTime + Checks \times CheckInterval = 12 + 25 \times 5 = 137\ sec$$

που είναι κατά πολύ μεγαλύτερη. Ο επιπλέον χρόνος που θα δοθεί σε σχέση με την προβλεπόμενη διάρκεια μέχρι να γίνει η επισήμανση του προβλήματος μπορεί να καθοριστεί στο πρόγραμμα.

5.2.2.4 Παρακολούθηση δύο συστημάτων ταυτόχρονα.

Σε αυτό το πείραμα έχουμε δύο slave κόμβους από τους οποίους ο ένας υλοποιείται στο ίδιο μηχάνημα μαζί με τον master κόμβο (Σχήμα 5.1). Η παρακολούθησή τους γίνεται ταυτόχρονα για δύο διαφορετικές εκτελέσεις του τελεστή k-means clustering. Στο ακόλουθο σχήμα φαίνεται η διάταξη του πειράματος.

Στον slave κόμβο yandall-vaio εκπαιδεύεται το σύστημα με τα δεδομένα που χρησιμοποιήθηκαν και στο πείραμα 5.2.2.1, ενώ για την εκτέλεση του τελεστή στον slave κόμβο localhost κατασκευάστηκε ένα νέο σύνολο δεδομένων με μικρότερο μέγεθος, δηλαδή είναι στοχευμένο σε μία μικρότερο εύρος εισόδου για να διαρκέσει λιγότερο η συλλογή μετρικών αλλά και για να γίνει μια σύγκριση της απόδοσης των νευρωνικών δικτύων επάνω στο ίδιο πρόβλημα με διαφορετικές παραμέτρους.

Αρχικά παρατίθενται οι παράμετροι εκπαίδευσης.

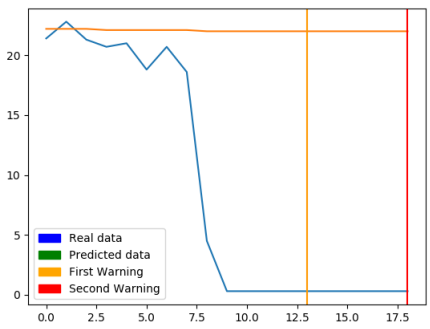
yandall-vaio		
	Εκπαίδευση	***Έλεγχος***
Μέγεθος εισόδου	14 csv αρχεία με μέγεθος 4.1 kB ~ 200.3 MB	16 csv αρχεία με μέγεθος 7.2 kB ~ 313.0 MB
Αριθμός clusters	2, 4, 8, 16, 32	3, 5, 9, 18, 24
Δειγματοληψία GMS	1 sec	1 sec
Granularity	100	100

localhost		
	Εκπαίδευση	***Έλεγχος***
Μέγεθος εισόδου	7 csv αρχεία με μέγεθος 28.1 kB ~ 200.3 MB	6 csv αρχεία με μέγεθος 28.2 kB ~ 154.6 MB
Αριθμός clusters	2, 4, 8, 16, 32	3, 5, 9, 18, 24
Δειγματοληψία GMS	1 sec	1 sec
Granularity	100	100

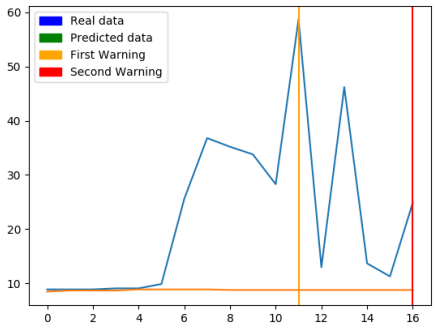
Οι παράμετροι ελέγχου είναι οι εξής.

	yandall-vaio	***localhost***
Παράμετρος	***Τιμή***	
Παράθυρο μέσου όρου	5 μετρήσεις	
Ρυθμός ελέγχου	2 sec	
Κατώφλι σφάλματος	5 έλεγχοι	
RMSE	9.54%	2.15%

Η εκπαίδευση έγινε για την παρακολούθηση στης μετρικής cpu_user. Κάθε τελεστής παρακολουθούνταν από ένα ξεχωριστό νήμα στο πρόγραμμα όπως έχει περιγραφεί σε προηγούμενο κεφάλαιο. Μετά από μερικά δευτερόλεπτα εκτέλεσης τερματίστηκε απότομα η λειτουργία του τελεστή στο μηχάνημα yandall-vaio και στη συνέχεια δημιουργήθηκε τεχνητά επιπλέον χρήση του επεξεργαστή στο μηχάνημα localhost. Το σύστημα παρήγαγε τις εξής γραφικές.



Σχήμα 5.11 Εντοπισμένη ανωμαλία στη μετρική cpu_user για το μηχάνημα yandall-vaio.



Σχήμα 5.12 Το αντίστοιχο πρόβλημα για τη ίδια μετρική στο μηχάνημα localhost.

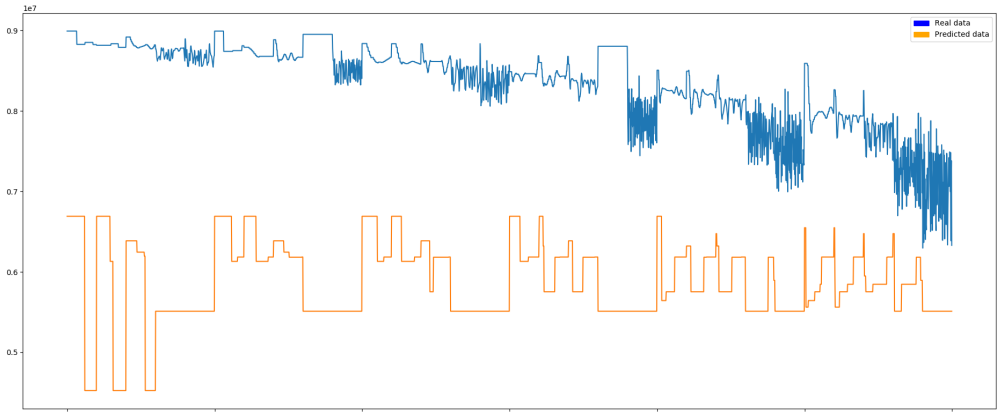
Παρατηρούμε ότι το σύστημα εντόπισε επιτυχώς και τα δύο προβλήματα και ενημέρωσε τον διαχειριστή.

5.2.3 Παρατηρήσεις

Από τα παραπάνω μπορούμε να επισημάνουμε κάποιες σύντομες αλλά ενδιαφέρουσες λεπτομέρειες που μπορούν να οδηγήσουν σε καλύτερες μελλοντικές υλοποιήσεις ή αρτιότερη στρατηγική στην επιλογή των συνόλων δεδομένων εκπαίδευσης.

- Κατά τη διάρκεια της οποιασδήποτε χρήσης ενός υπολογιστικού συστήματος ενδέχεται να απομείνουν δεδομένα – σκουπίδια στη μνήμη του για πολλούς λόγους, όπως για παράδειγμα κακός προγραμματισμός, χρήση μεταγλωττιστών χωρίς συλλέκτη σκουπιδιών και άλλα. Αν κάτι τέτοιο συμβεί κατά την λειτουργία συλλογής μετρικών ή κατά τον έλεγχο του τελεστή θα προκύψει ένα νευρωνικό δίκτυο για μετρικές όπως οι mem_free και mem_shared που είναι ασύμβατο με την πραγματικότητα του συστήματος εκείνη τη στιγμή. Με άλλα λόγια η κατάσταση (state) του πόρου που περιγράφει μία μετρική πρέπει να είναι ίδια και στις δύο φάσεις λειτουργίας αλλιώς είναι αδύνατον να μοντελοποιηθεί και να ελεγχθεί.

  Για να γίνει ακόμα πιο ξεκάθαρο αυτό παρουσιάζεται η περίπτωση της μετρικής mem_free, όταν συνέβη αυτό το πρόβλημα κατά τη φάση της εκτέλεσης πειραμάτων της παρούσας εργασίας.



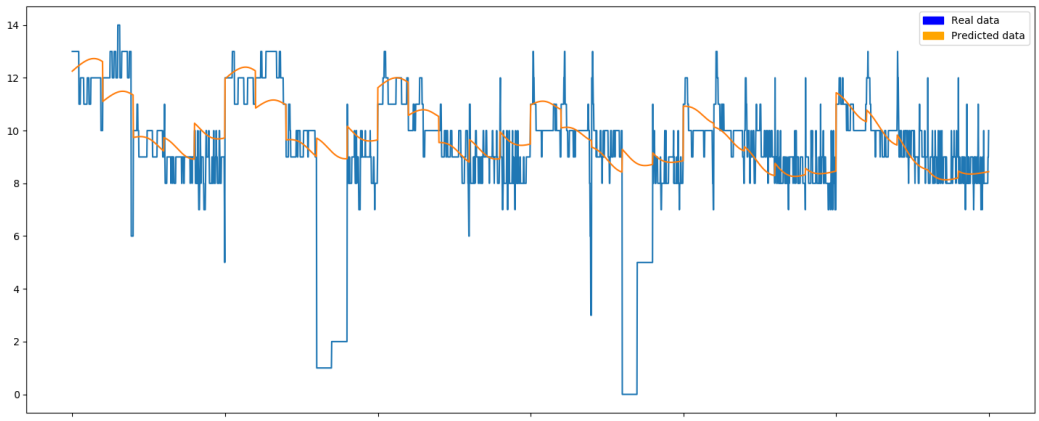
Σχήμα 5.13 Οι γραφικές παραστάσεις των δεδομένων ελέγχου (real data) και του αποτελέσματος του νευρωνικού δικτύου για τα ίδια ερωτήματα (predicted data)

Όπως βλέπουμε οι προβλέψεις απέχουν πολύ από την πραγματικότητα. Στην συγκεκριμένη περίπτωση κατά τη διάρκεια της συλλογής των δεδομένων

εκπαίδευσης υπήρχαν σκουπίδια στη μνήμη του μηχανήματος. Επομένως οι προβλέψεις για την ελεύθερη μνήμη δείχνουν πολύ χαμηλότερες τιμές αφού βασίζονται σε αυτά τα δεδομένα.

- Ένα δεύτερο ζήτημα είναι η επιλογή των συνόλων εκπαίδευσης/ελέγχου των νευρωνικών δικτύων. Το βασικότερο σημείο σε αυτήν είναι η κατάλληλη επιλογή του εύρους του μεγέθους των εισόδων (π.χ. τα αρχεία csv για τον τελεστή kmeans clustering). Από τα πειράματα που εκτελέστηκαν παρατηρήθηκε ότι ένα σύνολο εισόδων στοχευμένο σε μία περιοχή με ομοιόμορφα κατανεμημένα δείγματα και άκρα που βρίσκονται στην ίδια περίπου τάξη μεγέθους δίνει τα καλύτερα αποτελέσματα.

  Στον αλγόριθμο kmeans clustering τα σύνολα δεδομένων του πειράματος 6.2.2.4 έδωσαν ποιοτικότερα αποτελέσματα με μικρότερο σφάλμα απ’ ότι αυτά του πειράματος 6.2.2.1. Στην πρώτη περίπτωση το σφάλμα ήταν περίπου 9% στη χρήση του επεξεργαστή ενώ στη δεύτερη περίπου 2%. Επίσης μπορεί κανείς να συγκρίνει το Σχήμα 6.1 που είναι τα αποτελέσματα του νευρωνικού δικτύου για το πρώτο πείραμα με την εξής γραφική παράσταση.



Σχήμα 5.14 Αποτέλεσμα της εκπαίδευσης για τη μετρική cpu_user.

Όπως βλέπουμε τη πρόβλεψη ακολουθεί καλύτερα τα δεδομένα ελέγχου σε σχέση με το πρώτο σχήμα.

- Ανάμεσα στις μετρικές που μελετήθηκαν υπάρχει και μια κατηγορία αυτών που αποτελούν μέσους όρους της κατάστασης του συστήματος (load_one, load_five, load_fifteen). Σε αυτές τις περιπτώσεις εμφανίζεται ένα πρόβλημα παρόμοιο με αυτό που περιγράφηκε προηγουμένος. Κατά την εκκίνηση του

ελέγχου η μετρικές αυτές κρατούν πληροφορία από την προηγούμενη κατάσταση του συστήματος η οποία μπορεί να είναι διαφορετική από αυτή που προηγείτο της συλλογής μετρικών. Μία απλή λύση γι’ αυτό αν επιθυμεί ο χρήστης να τις χρησιμοποιήσει είναι να θέτει το χρόνο έναρξης του ελέγχου `min_elapsed_time` σε τιμή ίση ή μεγαλύτερη από το μέγεθος του μέσου όρου, δηλαδή για παράδειγμα στη μετρική load_one τιμή ίση ή μεγαλύτερη του 60.

- Τέλος μετρικές που εμφανίζουν λίγες μεμονωμένες μη-μηδενικές τιμές κατά τη διάρκεια της εκτέλεσης και τον υπόλοιπο χρόνο είχαν τιμή μηδέν ήταν αδύνατον να μοντελοποιηθούν ως έχουν από τα νευρωνικά δίκτυα.

6 *Επίλογος*

Έχοντας ολοκληρώσει τον πειραματικό έλεγχο του συστήματος μπορούμε να συνοψίσουμε το περιεχόμενο και το αποτέλεσμα της υλοποίησης της παρούσας διπλωματικής εργασίας.

6.1 Σύνοψη και συμπεράσματα

Η εργασία καταπιάστηκε με το πρόβλημα του ελέγχου της εκτέλεσης ενός προγράμματος (τελεστή – operator) επάνω σε ένα υπολογιστικό σύστημα και του εντοπισμού προβλημάτων – ανωμαλιών σε αυτήν. Αυτό το πέτυχε με την συλλογή μετρικών που περιγράφουν την κατάσταση των πόρων του συστήματος, την κατασκευή μοντέλων που τις προβλέπουν και τη χρήση τους σε συνθήκες πραγματικής λειτουργίας.

Μετά την εκτέλεση των πειραμάτων μπορεί κανείς να συμπεράνει ότι:

- Το σύστημα έχει μια ικανοποιητική ακρίβεια στον εντοπισμό βασικών προβλημάτων στις εκτελέσεις τελεστών,
- η απόκρισή του όταν εισάγεται ένα τεχνητό πρόβλημα είναι αρκετά γρήγορη
- και η ακρίβειά του μπορεί να είναι καλή δεδομένου ότι έχει γίνει σωστή επιλογή δεδομένων εκπαίδευσης.

- Επιπλέον είναι επεκτάσιμο και γενικευμένο ώστε να παρακολουθεί πολλές εκτελέσεις και μετρικές ταυτόχρονα και
- να μπορεί να εφαρμοστεί στον εντοπισμό ανωμαλιών σε μια χρονοσειρά ανεξάρτητα από το τι αυτή περιγράφει.

6.2 Μελλοντικές επεκτάσεις

Φυσικά το σύστημα και η υλοποίησή του είναι μια πρώτη απόπειρα εφαρμογής της μεθοδολογίας που διατυπώθηκε στην εργασία και συνεπώς υπάρχει περιθώριο για αρκετές και διαφορετικές μελλοντικές επεκτάσεις και βελτιώσεις.

Μερικές ενδιαφέρουσες προτάσεις για την περαιτέρω βελτίωση του είναι:

- Ο κατακερματισμός του συνόλου δεδομένων σε μικρότερα και η δημιουργία ξεχωριστών νευρωνικών δικτύων για κάθε ένα από αυτά με στόχο την βελτίωση της ακρίβειας τους. Όπως είδαμε στην παράγραφο 6.2.3 ένα σύνολο εκπαίδευσης που τα στοιχεία του είναι όλα ή σχεδόν όλα ίδιας τάξης μεγέθους δίνουν καλύτερα αποτελέσματα για εκτελέσεις που ανήκουν σε αυτό απ’ ότι ένα μεγάλο σύνολο που καλύπτει όλο το πιθανό εύρος του μεγέθους εισόδου του τελεστή.
- Σε συνδυασμό με το παραπάνω η χρήση καλύτερων βιβλιοθηκών (π.χ. TensorFlow, Keras) και παραμέτρων εκπαίδευσης των νευρωνικών δικτύων αλλά και η προσθήκη καινούριων μεθόδων για τον έλεγχο του τελεστή.
- Η χρήση της στατιστικής μεθόδου Granger Causality Test όχι για τον έλεγχο της χρονοσειράς αλλά για την εύρεση της χρονικής μετατόπισης μεταξύ πραγματικών και προβλεπόμενων μετρικών, αφού η μέθοδος αυτή εκτελείται μετακινώντας την μια χρονοσειρά ως προς την άλλη και το σημείο όπου μεγιστοποιείται η πιθανότητα η μία να προβλέπει την άλλη είναι και το σημείο όπου είναι πιθανότερο οι χρονοσειρές να είναι σωστά διατεταγμένες.

7 *Βιβλιογραφία*

1. Thomas H. Davenport, Jill Dyché. *Big Data in Big Companies.* May 2013, International Institue for Analytics.

2. Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Gary Grider, Parks M. Fields, Kevin Harms, Robert B. Ross, Andree Jacobson, Robert Ricci, Kirk Webb, Peter Alvaro, H. Birali Runesha, Mingzhe Hao, and Huaicheng Li. *Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems.* 16th USENIX Conference on File and Storage Technologies.

3. Debessay Fesehaye, Lenin Singaravelu, Chien-Chia Chen, Xiaobo Huang, Amitabha Banerjee. *Group Clustering Using Inter-Group Dissimilarities.* 2017 IEEE 37th International Conference on Distributed Computing Systems.

4. Jessica Lin and Yuan Li. *Finding Structural Similarity in Time Series Data Using Bag-of-Patterns Representation.* M. Winslett (Ed.): SSDBM 2009,

LNCS 5566, pp. 461–477, 2009.

5. William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. *Numerical Recipes in Fortran 77: The Art of Scientific Computing.* 1992 Cambridge University Press.

6. Ronald W. Schafer. *What Is a Savitzky-Golay Filter? (lecture notes).*

7. Simon Haykin. *Νευρωνικά Δίκτυα και Μηχανική Μάθηση (3η έκδοση).* Εκδόσεις Παπασωτηρίου, Αθήνα 2010.

8. Matt Massie, Bernard Li, Brad Nicholes, Vladimir Vuksan. *Monitoring With Ganglia.* O'Reilly 2013.