ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

# Εφαρμογή για συλλογή μετρήσεων κατανάλωσης σε Data Center

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Δημήτριου Σ. Χίου**

Επιβλέπων: **Ευστάθιος Συκάς**
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

# Εφαρμογή για συλλογή μετρήσεων κατανάλωσης σε Data Center

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**Δημήτριου Σ. Χίου**

Επιβλέπων: **Ευστάθιος Συκάς**
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Φεβρουαρίου 2020.

| ..... | ..... | ..... |
|---|---|---|
| Ευστάθιος Συκάς<br>Καθηγητής Ε.Μ.Π. | Μιλτιάδης Αναγνώστου<br>Καθηγητής Ε.Μ.Π. | Ιωάννα Ρουσσάκη<br>Επίκ. Καθηγήτρια Ε.Μ.Π. |

Αθήνα, Φεβρουάριος 2020

.....

**ΔΗΜΗΤΡΙΟΣ ΧΙΟΣ**

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

# Περίληψη

Η ραγδαία ανάπτυξη τεχνολογιών NFV (Network FunctionsVirtualization) και η εισαγωγή του 5G, σε συνδυασμό με τη ζήτηση για παροχή υπολογιστικής ισχύος ως υπηρεσία μέσω του διαδικτύου (Cloud computing), αναμένεται να αυξήσουν το πλήθος και το μέγεθος των Data Center των παροχών υπηρεσιών. Παράλληλα, αναμένεται το σύνολο των ροών εργασίας παραδοσιακών υπηρεσιών πληροφορικής να εκτελούνται σε κεντρικές υποδομές Data Center. Ως αποτέλεσμα αναμένεται να αυξηθεί η συνολική κατανάλωση ενέργειας και κατ' επέκταση το συνολικό κόστος λειτουργίας. Καθώς, όμως, ο εξοπλισμός συνεχίζει να αυξάνεται σε μέγεθος και πολυπλοκότητα, αυξάνεται ο όγκος και η ταχύτητα των λειτουργικών δεδομένων που σχετίζονται με αυτόν. Κάτω από αυτές τις συνθήκες φαίνεται να είναι επιτακτική ανάγκη η δημιουργία νέων μηχανισμών και τεχνολογιών για τη συνεχή συγκέντρωση, αποθήκευση και ανάλυση δεδομένων που σχετίζονται με την ενεργειακή κατανάλωση και το ποσοστό χρησιμοποίησης των διαθέσιμων πόρων σε σχεδόν πραγματικό χρόνο από ετερόκλητες και κατανεμημένες πηγές χωρίς να επηρεάζονται οι κεντρικές λειτουργίες και η αξιοποίηση των πόρων των Data Center.

Στα πλαίσια της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε ο σχεδιασμός και η υλοποίηση ενός συστήματος συλλογής και διανομής των προαναφερθέντων μετρήσεων, αποθήκευσης τους σε βάση δεδομένων κατάλληλη για τη διαχείριση χρονοσειρών καθώς και απεικόνισης τους, τα όποια βασίζονται σε λογισμικό ανοιχτού κώδικα. Τα δεδομένα αυτά χρησιμοποιήθηκαν για την εφαρμογή και αξιολόγηση ενός μοντέλου βελτιστοποίησης ενεργειακής κατανάλωσης. Οι μετρήσεις προέρχονται από ένα πλήθος διαφόρων συσκευών (εξυπηρετητές, δικτυακές συσκευές, αισθητήρες και λοιπά) του εργαστηρίου Δικτύων Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, σε διάφορα σενάρια χρήσης.

**Λέξεις Κλειδιά**: Data Center, Big data, Time series, Data analysis, Network traffic, Monitoring, Prometheus, Influxdb, Kafka, Grafana, Virtual Machine, Energy consumption, Energy optimization, VM placement

# Abstract

The rapid development of NFV (Network FunctionsVirtualization) technologies and the introduction of 5G, coupled with the demand for cloud computing, are expected to increase the number and size of service providers' Data Centers. At the same time, it is expected that all traditional IT services workflows will be implemented in central Data Center infrastructures. As a result, it is expected that their total energy consumption and thus their overall operating costs will increase. However, as equipment continues to grow in size and complexity, the volume and speed of the operational data associated with it increases. Under these conditions, it seems imperative to create new mechanisms and technologies for the continuous collection, storage and analysis of energy consumption data and the rate of utilization of available resources in near real time by heterogeneous and distributed sources without affecting centralized operations and utilization of Data Center resources.

This thesis will design and implement a system for collecting and distributing the above measurements, storing them in a database suitable for time series management and displaying them, which will be based on open source software. This data will be used to implement and evaluate an energy consumption optimization model. The measurements will come from a multitude of different devices (servers, web devices, sensors, etc.) in the laboratory environment of the Computer Network Laboratory of the ECE School of the NTUA, in various usage scenarios.



**Keywords:** Data Center, Big data, Time series, Data analysis, Network traffic, Monitoring, Prometheus, Influxdb, Kafka, Grafana, Virtual Machine, Energy consumption, Energy optimization, VM placement

# Ευχαριστίες

Με τη σηματοδότηση της ολοκλήρωσης της διπλωματικής μου εργασίας στο Εργαστήριο Δικτύων Υπολογιστών θα ήθελα να ευχαριστώ ιδιαίτερα τον επιβλέποντα καθηγητή μου κ. Ευστάθιο Συκά που μου παρείχε αυτή τη δυνατότητα. Επίσης θα ήθελα να ευχαριστώ θερμά τον υποψήφιο Διδάκτορα Πάρι Χαραλάμπου για την άψογη συνεργασία που είχαμε καθ' όλη τη διάρκεια εκπόνησης της εργασίας, χωρίς την αρωγή του οποίου θα ήταν ένα δύσκολο έργο. Ο χρόνος που διέθεσε καθοδηγώντας με σε όλα τα στάδια της εργασίας και η διαθεσιμότητα του ανά πάσα στιγμή ήταν τα κύρια στοιχεία που οδήγησαν σε ένα άρτιο αποτέλεσμα.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου για την οικογένεια και τους φίλους μου, τόσο για την υπομονή όσο και τη συμπαράσταση που έδειξαν όλο αυτόν το καιρό. Το μεγαλύτερο ευχαριστώ, ωστόσο, πηγαίνει στους γονείς μου που πίστεψαν σε εμένα και χωρίς τους οποίους δεν θα τα είχα καταφέρει.

Δημήτρης

# Πίνακας Περιεχομένων

# Ευρετήριο Εικόνων

# Ευρετήριο Πινάκων

# 1 Εισαγωγή

## 1.1 Data Center

Ένα κέντρο δεδομένων μπορεί να οριστεί ως μια εγκατάσταση με όλους εκείνους τους πόρους που απαιτούνται για την αποθήκευση και την επεξεργασία ψηφιακής πληροφορίας και τους τομείς υποστήριξής του. Τα κέντρα δεδομένων περιλαμβάνουν τόσο την απαιτούμενη υποδομή (π.χ. διανομή ισχύος, συστήματα περιβαλλοντικού ελέγχου, τηλεπικοινωνίας, ασφάλειας, πυροπροστασίας, συστήματα αυτοματισμού) όσο και τον κατάλληλο τεχνολογικό εξοπλισμό (π.χ. διακομιστές για επεξεργασία, αποθήκευση δεδομένων, εξοπλισμό δικτύου/ επικοινωνίας) καθώς και τις διασυνδέσεις με το διαδίκτυο.

Τα κέντρα δεδομένων διαδραματίζουν σημαντικό ρόλο ειδικά σήμερα λόγω του μεγάλου όγκου ψηφιακής πληροφορίας που αποθηκεύουν. Σύμφωνα με τα πρότυπα και τις απαιτήσεις των χρηστών, η υποδομή τους πρέπει να πληροί αυστηρές τεχνικές απαιτήσεις προκειμένου να διασφαλιστεί η αξιοπιστία, η διαθεσιμότητα και η ασφάλεια, καθώς αυτές έχουν άμεσο αντίκτυπο στο κόστος και στην αποτελεσματικότητα. Μια υποδομή με υψηλή αξιοπιστία και διαθεσιμότητα πρέπει να έχει πλεόνασμα σε συστήματα και, για το λόγο αυτό, θα έχει μεγαλύτερο κόστος. Κάθε επιχείρηση ανάλογα με τις απαιτήσεις έχει να επιλέξει ανάμεσα σε διάφορες κατηγορίες Data Center [7]:

**Κατηγορία 1:** Ένα κέντρο δεδομένων σε αυτό το επίπεδο είναι απλό στη δομή διότι έχει μόνο μία πηγή διακομιστών, συνδέσμους δικτύου και άλλα στοιχεία. Η εφεδρεία (redundancy) και τα αντίγραφα ασφαλείας σε αυτή τη βαθμίδα είναι ελάχιστα ή ανύπαρκτα. Αυτό περιλαμβάνει ελαχιστοποίηση ισχύος και αποθήκευσης. Ως εκ τούτου, οι προδιαγραφές για ένα κέντρο δεδομένων για αυτή τη βαθμίδα δεν προκαλούν δέος. Εάν μια διακοπή ρεύματος επρόκειτο να συμβεί, το σύστημα θα πήγαινε εκτός σύνδεσης, καθώς δεν υπάρχουν δικλείδες ασφαλείας για να το επαναφέρουν σε λειτουργία.

Οι προδιαγραφές ενός κέντρου δεδομένων πρώτης βαθμίδας επιτρέπουν διαθεσιμότητα (uptime) περίπου 99,671% (28,8 ώρες χρόνο διακοπής ετησίως) [8]. Η έλλειψη εφεδρικών μηχανισμών τα καθιστά ριψοκίνδυνα για πολλές επιχειρήσεις, αλλά μπορούν να λειτουργήσουν για μικρές εταιρίες με βάση το διαδίκτυο χωρίς υποστήριξη πελατών σε πραγματικό χρόνο. Ωστόσο, για εταιρείες με μεγάλη εξάρτηση από τα δεδομένα τους, ένα κέντρο δεδομένων πρώτης βαθμίδας δεν θα ήταν πρακτικό. Ένα από τα πλεονεκτήματα των κέντρων δεδομένων πρώτης βαθμίδας είναι ότι παρέχουν την πιο φθηνή προσφορά υπηρεσιών για επιχειρήσεις με περιορισμένο προϋπολογισμό. Ωστόσο, η έλλειψη πλεονασμού σημαίνει ότι ο χρόνος λειτουργίας των διακομιστών είναι σημαντικά χαμηλότερος από τη δεύτερη, την τρίτη και την τέταρτη και ότι η συντήρηση της εγκατάστασης θα απαιτήσει κλείσιμο ολόκληρης της εγκατάστασης, συνεπώς περισσότερο χρόνο αναμονής.

**Κατηγορία 2:** Η δεύτερη βαθμίδα περιλαμβάνει περισσότερες υποδομές και μέτρα για να εξασφαλιστεί η μικρότερη ευπάθεια σε απρόβλεπτες διακοπές. Οι απαιτήσεις για αυτό το επίπεδο κέντρου δεδομένων περιλαμβάνουν όλες εκείνες της πρώτης βαθμίδας, αλλά με κάποιο πλεόνασμα. Συνήθως διαθέτουν ένα μόνο μονοπάτι για την παροχή ρεύματος και ψύξης. Ωστόσο, διαθέτουν μια εφεδρική γεννήτρια και ένα εφεδρικό σύστημα ψύξης για να διατηρούν το περιβάλλον του κέντρου δεδομένων βέλτιστο.

Οι προδιαγραφές ενός κέντρου δεδομένων για τη δεύτερη βαθμίδα επιτρέπουν υψηλότερο χρόνο λειτουργίας σε σύγκριση με τα κέντρα δεδομένων πρώτου επιπέδου που είναι περίπου 99,741% (22 ώρες χρόνο διακοπής ετησίως) [8].

**Κατηγορία 3:** Οι απαιτήσεις τρίτης βαθμίδας για τα κέντρα δεδομένων περιλαμβάνουν όλα τα δεδομένα της πρώτης βαθμίδας, αλλά έχουν μια πιο εξελιγμένη υποδομή. Αυτό επιτρέπει πλεονασμό και δημιουργία αντιγράφων ασφαλείας σε περίπτωση απρόβλεπτων συμβάντων που μπορεί να προκαλέσουν διακοπές. Όλος ο εξοπλισμός διακομιστή έχει πολλές πηγές ενέργειας και διαδρομές διανομής ψύξης. Σε περίπτωση αποτυχίας οποιουδήποτε από τα μονοπάτια διανομής, ένας άλλος αναλαμβάνει τη διασφάλιση ότι το σύστημα παραμένει συνδεδεμένο. Τα κέντρα δεδομένων κατηγορίας 3 πρέπει να έχουν πολλαπλές συνδέσεις ανερχόμενης ζεύξης και πρέπει να είναι διπλής τροφοδοσίας. Ορισμένες διαδικασίες εφαρμόζονται προκειμένου να εξασφαλιστεί ότι η συντήρηση μπορεί να γίνει χωρίς διακοπή λειτουργίας. Τα κέντρα δεδομένων τρίτης βαθμίδας είναι η οικονομικότερη λύση για την πλειοψηφία των επιχειρήσεων. Έχει ένα αναμενόμενο χρόνο λειτουργίας 99,982% (1,6 ώρες χρόνο διακοπής ετησίως) [8].

**Κατηγορία 4:** Η 4η βαθμίδα είναι το υψηλότερο επίπεδο όσον αφορά τις κατηγορίες κέντρων δεδομένων. Ένα κέντρο δεδομένων της κατηγορίας 4 είναι πιο εξελιγμένο όσον αφορά την υποδομή του, καθώς διαθέτει την πλήρη χωρητικότητα, υποστήριξη και διαδικασίες που εφαρμόζονται για να εξασφαλίσουν τα μέγιστα και βέλτιστα επίπεδα χρόνου λειτουργίας (uptime). Το κέντρο δεδομένων Tier 4 πληροί πλήρως όλες τις προδιαγραφές των άλλων τριών κατηγοριών. Είναι ανεκτικό σε σφάλματα, καθώς μπορεί να λειτουργήσει κανονικά ακόμη και όταν υπάρχει περίπτωση αποτυχίας εξοπλισμού υποδομής.

Ένα κέντρο δεδομένων Tier 4 είναι πλήρως εξοπλισμένο με πολλαπλά συστήματα ψύξης, πηγές ενέργειας και γεννήτριες για να το υποστηρίξουν. Είναι κατασκευασμένο για να είναι εντελώς ανεκτικό σε σφάλματα και έχει πλεονασμό για κάθε στοιχείο. Έχει αναμενόμενο ποσοστό διαθεσιμότητας 99,995% (26,3 λεπτά χρόνο διακοπής ετησίως) [8].

Οι εγκαταστάσεις των κέντρων δεδομένων έχουν μεγάλο βαθμό δυναμικότητας καθώς ο εξοπλισμός μπορεί να αναβαθμίζεται συχνά, είτε για να προστεθεί νέος, είτε να αφαιρείται ο απαρχαιωμένος εξοπλισμός. Σε ορισμένες περιπτώσεις, παλιά και νέα συστήματα χρησιμοποιούνται ταυτόχρονα για σημαντική χρονική περίοδο. Σε ένα τέτοιο περιβάλλον, μπορούν να προκληθούν πολλά ανεπιθύμητα προβλήματα από πληθώρα διαφορετικών

παραγόντων. Η συνολική απόδοση της εγκατάστασης μπορεί να επηρεαστεί (α) από μεταβολές ισχύος, (β) συνθήκες περιβάλλοντος και (γ) ανθρώπινο σφάλμα . Επομένως, ο εξοπλισμός χρειάζεται μόνιμη παρακολούθηση και συντήρηση για να διασφαλιστεί η σωστή και αποτελεσματική λειτουργία του.

Η λήψη δεδομένων σε σχεδόν πραγματικό χρόνο δεν είναι ένα εύκολο έργο χωρίς κατάλληλα μέσα για τη συλλογή τους, διότι πρόκειται για μετρήσεις που προέρχονται από ετερόκλητες πηγές και η κάθε μια απαιτεί την εφαρμογή διαφορετικού πρωτοκόλλου συλλογής τους. Τα δεδομένα αυτά συνεισφέρουν ενεργά σε αποφάσεις που αφορούν την αποφυγή ή αναβολή μελλοντικών επεκτάσεων ή ακόμα και μετεγκατάστασης.

Πρόσφατες έρευνες έδειξαν ότι τα κέντρα δεδομένων απαιτούν σχεδόν 40 φορές περισσότερη ενέργεια για να λειτουργούν σε σύγκριση με τα συμβατικά κτίρια [1]. Αν συμπεριληφθεί η ραγδαία ανάπτυξη των τεχνολογιών NFV (Network FunctionsVirtualization), η ζήτηση για παροχή υπολογιστικής ισχύος ως υπηρεσία μέσω του διαδικτύου (Cloud computing as a service), η άνοδος της τεχνητής νοημοσύνης (AI), η μηχανική μάθηση (machine learning), τα μεγάλα δεδομένα (Big Data), το διαδίκτυο των πραγμάτων (Internet of things-IoT) καθώς και η εισαγωγή του 5G, εύκολα αντιλαμβάνεται κάποιος ότι μελλοντικά το ποσοστό αυτό θα αυξηθεί. Το γεγονός αυτό έχει προκαλέσει σοβαρές ανησυχίες στους ερευνητές που αναμένουν ότι η ενεργειακή κατανάλωση των κέντρων δεδομένων θα έχει τριπλασιαστεί στο τέλος του 2020 και θα φτάσει στο 13% της παγκόσμιας ζήτησης ηλεκτρικής ενέργειας μέχρι το 2030 [11], συγκριτικά με το 2012 που ήταν 300-400 Twh, περίπου το 2-3% της παγκόσμιας καταναλισκόμενης ισχύος [9].

Αναπόφευκτα διαφαίνεται η αναγκαιότητα για τη δημιουργία ενός ευέλικτου συστήματος εποπτείας καθώς η συλλογή δεδομένων σε πραγματικό χρόνο έχει κρίσιμο ρόλο σε διάφορους τομείς όπως είναι η παρακολούθηση, ο έλεγχος και η διαχείριση της υποδομής μεγιστοποιώντας τη διαθεσιμότητα, την αξιοπιστία, τις στρατηγικές βελτιστοποίησης των μετρήσεων και την εφαρμογή μοντέλων για τη μείωση της ενεργειακής κατανάλωσης και κατ’ επέκταση των συνολικών δαπανών . Η σχεδίαση ενός τέτοιου συστήματος πρέπει να ακολουθεί κάποιες απαιτήσεις όπως είναι η ορθή μελέτη και κατανόηση της υπό εξέταση υποδομής, η αξιολόγηση και ο προσδιορισμός των προβλημάτων με τις ήδη υπάρχουσες λύσεις, η χρήση εργαλείων ανοικτού κώδικα που αφήνει περιθώρια για πρωτότυπες υλοποιήσεις και η κλιμάκωση σε σενάρια επέκτασης [2].

Σκοπός της διπλωματικής εργασίας είναι η υλοποίηση μιας εφαρμογής για την παρακολούθηση σε πραγματικό χρόνο δεδομένων που αφορούν την ενεργειακή κατανάλωση και τη χρησιμοποίηση των διαθέσιμων πόρων εντός Data Center. Οι μετρήσεις λαμβάνονται από ένα σύνολο ετερόκλητων συσκευών με χρήση τυπικών πρωτοκόλλων. Η αποθήκευση των δεδομένων γίνεται σε βάση που μπορεί να διαχειριστεί χρονοσειρές και χρησιμοποιείται λογισμικό κατάλληλο για την απεικόνισή τους. Το σύνολο του εξοπλισμού είναι στο χώρο

του εργαστηρίου Δικτύου Υπολογιστών. Οι δυνατότητες που παρέχονται από την ύπαρξη της παραπάνω λύσης επιδεικνύονται (α) με τη μελέτη και ανάπτυξη μοντέλου κατανάλωσης εξυπηρετητών και (β) με την εφαρμογή ενός μοντέλου βελτιστοποίησης της ενεργειακής κατανάλωσης χωρίς την ύπαρξη επιπτώσεων στην ποιότητα λειτουργίας του εργαστηρίου.

## 1.2 Οργάνωση κειμένου

Η παρούσα διπλωματική εργασία αποτελείται από έξι (6) κεφάλαια. Στο παρόν κεφάλαιο, παρουσιάσθηκαν αναλυτικά οι λόγοι και οι ανάγκες που οδήγησαν στη δημιουργία του συγκεκριμένου θέματος.

Στο Κεφάλαιο 2 παρουσιάζονται εκτενώς οι απαραίτητες έννοιες προκειμένου ο αναγνώστης να αποκτήσει το κατάλληλο θεωρητικό υπόβαθρο για την ευκολότερη κατανόηση του αντικειμένου. Συγκεκριμένα αναλύονται οι έννοιες της ενεργειακής κατανάλωσης των Data center, Big Data και Time Series Data.

Στο Κεφάλαιο 3 παρουσιάζεται θεωρητικά η αρχιτεκτονική του δικτύου που σχεδιάστηκε στα πλαίσια της εργασίας. Γίνεται παρουσίαση των διαθέσιμων εργαλείων ανοικτού κώδικα που χρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής. Μερικά εξ αυτών είναι το prometheus για τη συλλογή δεδομένων, το influxdb ως backend time series storage, το grafana ως εργαλείο απεικόνισης γραφημάτων και το kafka για τη διανομή των δεδομένων σε άλλους φορείς.

Στο Κεφάλαιο 4 πραγματοποιείται διεξοδική ανάλυση της διαδικασίας που ακολουθήθηκε για την υλοποίηση της εφαρμογής που απαιτούσε το θέμα της συγκεκριμένης διπλωματικής εργασίας. Παρουσιάζονται τα βήματα που ακολουθήθηκαν μαζί με τις απαραίτητες επεξηγήσεις για τη δημιουργία ενός εργαλείου εποπτείας Data Center.

Στο Κεφάλαιο 5 γίνεται προσπάθεια αξιολόγησης της υλοποίησης με παρουσίαση ενός σεναρίου χρήσης των δεδομένων που συλλέχθηκαν στην εφαρμογή ενός μοντέλου ενεργειακής βελτιστοποίησης και των αποτελεσμάτων που προέκυψαν και τη διενέργεια του.

Στο Κεφάλαιο 6 συνοψίζονται τα συμπεράσματα, παρουσιάζονται οι προκλήσεις που πρέπει να αντιμετωπιστούν και οι πιθανές μελλοντικές προεκτάσεις μετά την ολοκλήρωσή της.

Στο τέλος παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε κατά τη συγγραφή της παρούσας διπλωματικής εργασίας και το παράρτημα, το οποίο περιέχει τον κώδικα που αναπτύχθηκε για το παρόν θέμα.

# 2 Θεωρητικό Υπόβαθρο

## 2.1 Ενεργειακή κατανάλωση σε Data Center

Η ενέργεια που καταναλώνεται από ένα κέντρο δεδομένων μπορεί να κατηγοριοποιηθεί σε τρία μέρη [3]: χρήση ενέργειας (α) από εξοπλισμό πληροφορικής, (β) δικτυακό εξοπλισμό και (γ) από τις εγκαταστάσεις υποδομής (π.χ. συστήματα ψύξης και κλιματισμού). Η ποσότητα ενέργειας που καταναλώνεται από αυτές τις τρεις συνιστώσες εξαρτάται από το σχεδιασμό του κέντρου δεδομένων καθώς και από την αποτελεσματικότητα του εξοπλισμού. Για παράδειγμα, σύμφωνα με τις στατιστικές που δημοσιεύονται από την ομάδα Infotech, ο μεγαλύτερος καταναλωτής ενέργειας σε ένα τυπικό κέντρο δεδομένων είναι η υποδομή ψύξης (50%), ενώ οι διακομιστές και οι συσκευές αποθήκευσης (26%) κατέχουν τη δεύτερη θέση στην ιεραρχία κατανάλωσης ενέργειας.

Μια γενική προσέγγιση για τη διαχείριση της κατανάλωσης ενέργειας στα κέντρα αποτελείται από τέσσερα κύρια βήματα: εξαγωγή χαρακτηριστικών, κατασκευή μοντέλου, επικύρωση του μοντέλου και εφαρμογή του μοντέλο σε μια εργασία όπως η πρόβλεψη.

**Εξαγωγή χαρακτηριστικών:** Προκειμένου να μειωθεί η κατανάλωση ενέργειας ενός κέντρου δεδομένων, πρέπει πρώτα να μετρήσουμε την κατανάλωση ενέργειας των συστατικών στοιχείων της και να αποφανθούμε που καταναλώνεται το μεγαλύτερο μέρος της ενέργειας.

**Κατασκευή μοντέλου:** Δεύτερον, τα επιλεγμένα χαρακτηριστικά εισόδου χρησιμοποιούνται για την κατασκευή ενός μοντέλου κατανάλωσης ενέργειας χρησιμοποιώντας τεχνικές ανάλυσης όπως η παλινδρόμηση, η μηχανική μάθηση, κλπ. Ένα από τα βασικά προβλήματα που αντιμετωπίζουμε σε αυτό το βήμα είναι ότι ορισμένες σημαντικές παράμετροι του συστήματος, όπως η καταναλισκόμενη ισχύς ενός συγκεκριμένου στοιχείου σε ένα κέντρο δεδομένων δεν μπορεί να μετρηθεί άμεσα. Οι μέθοδοι μηχανικής μάθησης και μέθοδοι βαθιάς εκμάθησης συχνά αναφέρθηκαν ως η βασική λύση για όλα τα προβλήματα μοντελοποίησης. Τα αποτελέσματα πρόσφατων ερευνών, ωστόσο, έδειξαν ότι σε περιπτώσεις που χρησιμοποιούνται χρονοσειρές, κλασσικές μέθοδοι (linear regression, exponential smoothing, κλπ) υπερτερούν από σύνθετες και εξελιγμένες μεθόδους (decision trees, Multilayer Perceptrons (MLP), Long Short-Term Memory (LSTM) network models κλπ) [10]. Το αποτέλεσμα αυτού του βήματος είναι ένα ενεργειακό μοντέλο.

**Επικύρωση μοντέλου:** Στη συνέχεια, το μοντέλο πρέπει να επικυρωθεί για την καταλληλότητα του για τους επιδιωκόμενους σκοπούς.

**Χρήση μοντέλου:** Τέλος, μπορεί να χρησιμοποιηθεί το μοντέλο ως βάση για την πρόβλεψη κατανάλωση ενέργειας του στοιχείου ή του συστήματος. Τέτοιες προβλέψεις μπορούν στη συνέχεια να χρησιμοποιηθούν για τη βελτίωση της ενεργειακής απόδοσης του κέντρου δεδομένων. Επομένως, έχουμε τη δυνατότητα βελτίωσης των αλγορίθμων που

χρησιμοποιούνται από εφαρμογές, τη μετάβαση σε καταστάσεις χαμηλής ισχύος, το κλείδωμα ισχύος ή ακόμα και την πλήρη απενεργοποίηση αχρησιμοποίητων εξυπηρετητών κ.λπ., καθιστώντας τα κέντρα δεδομένων ενεργειακά αποδοτικότερα. Ωστόσο, σημειώνουμε ότι δεν είναι πάντοτε απαραίτητο να μπορούμε να προβλέψουμε την κατανάλωση ενέργειας.

Τα ενεργειακά μοντέλα είναι χρήσιμα σε πληθώρα εφαρμογών συμπεριλαμβανομένης της σχεδίασης των συστημάτων κέντρων δεδομένων, της πρόβλεψης ενεργειακών τάσεων και της βελτιστοποίησης κατανάλωσης ενέργειας. Η μοντελοποίηση της ενέργειας είναι ένας ενεργός τομέας έρευνας, μελετώντας γραμμικούς και μη γραμμικούς συσχετισμούς μεταξύ της χρήσης του συστήματος και της ενέργειας. Ωστόσο, μοντελοποίηση της ακριβούς συμπεριφοράς κατανάλωσης ενέργειας ενός κέντρου δεδομένων, είτε σε ολόκληρο το σύστημα είτε σε κάποιο επίπεδο του, δεν είναι απλή. Συγκεκριμένα, τα διάφορα ενεργειακά μοτίβα εξαρτώνται από πολλούς παράγοντες όπως προδιαγραφές υλικού, φόρτο εργασίας, απαιτήσεις ψύξης, τύπους εφαρμογών κ.λπ., οι οποίες δεν μπορούν να μετρηθούν εύκολα. Η ισχύς που καταναλώνεται από το υλικό, το λογισμικό που εκτελείται στο υλικό, την υποδομή ψύξης και τροφοδοσίας της υποδομής όπου βρίσκονται τα συστήματα των κέντρων δεδομένων είναι όλα στενά συζευγμένα. Επιπλέον, δεν είναι πρακτικό να εκτελεσθούν λεπτομερείς μετρήσεις της κατανάλωσης ενέργειας των εξαρτημάτων όλων των κατώτερων επιπέδων, καθώς εισάγει επιβάρυνση στο σύστημα. Λόγω αυτών έχουν αναπτυχθεί τεχνικές πρόβλεψης που μπορούν να κάνουν σχεδόν ρεαλιστική εκτίμηση του επιπέδου ενέργειας που καταναλώνεται από ένα σύστημα για δεδομένο φόρτο εργασίας. Οι τεχνικές πρόβλεψης κατανάλωσης ενέργειας μπορούν επίσης να χρησιμοποιηθούν για την πρόβλεψη της ενεργειακής αξιοποίησης ενός κέντρου το οποίο λειτουργεί σε ένα συγκεκριμένο πλαίσιο.

Επίτευξη της λειτουργικής και ενεργειακής αποδοτικότητας με όσα αναφέρθηκαν πρωτύτερα σε ένα περιβάλλον που διαρκώς αυξάνεται σε μέγεθος και πολυπλοκότητα απαιτεί τη συλλογή πληροφοριών από όλα τα συστήματα και τις πηγές που υποστηρίζονται, την ανάλυση και ανταπόκριση σε γεγονότα κοντά σε πραγματικό χρόνο, όταν είναι απαραίτητο. Ωστόσο, η φύση αυτών των δεδομένων είναι ετερογενής, προέρχονται από διάφορες πηγές που βρίσκονται σε όλη την έκταση της εγκατάστασης, το κεντρικό κτίριο ή τον εξωτερικό χώρο και σε διαφορετικές μορφές. Η κλίμακα των πόρων σημαίνει επίσης ότι η ποσότητα των δεδομένων που πρέπει να συλλεχθούν είναι αναλογικά τεράστια. Επιπλέον, κάποια από τα δεδομένα είτε συμβαίνουν σε δευτερόλεπτα και άλλα σε μεγαλύτερα χρονικά διαστήματα. Κάποια από αυτά μας ενδιαφέρουν σε πραγματικό χρόνο και απαιτείται άμεση έκθεση τους ενώ για άλλα μια μέση τιμή είναι αρκετή. Επιπρόσθετα, τα αρχειοθετημένα δεδομένα λειτουργούν ως σημαντικές πληροφορίες που μπορούν να βοηθήσουν στην ενημέρωση των αποφάσεων σχετικά με την ενεργειακή απόδοση, τις μελλοντικές προμήθειες, την προληπτική συντήρηση κ.λπ.

## 2.2 Μεγάλα δεδομένα και χρονοσειρές

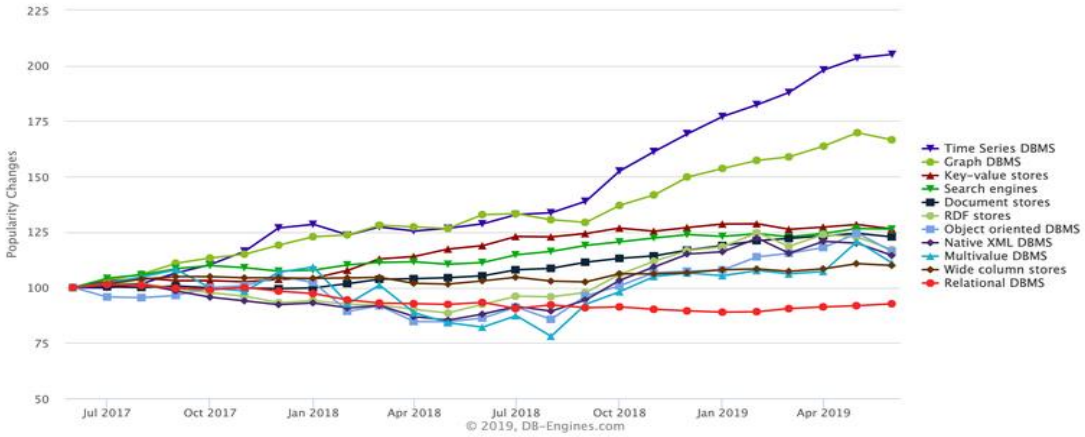
### 2.2.1 Ορισμός χρονοσειρών

Στην προσπάθεια κατασκευής ενεργειακών μοντέλων, σημαντικό ρόλο διαδραματίζουν οι χρονοσειρές. Η χρονοσειρά είναι μια ακολουθία δεδομένων, που μετρούν την ίδια μεταβλητή με την πάροδο του χρόνου και αποθηκεύονται με χρονολογική σειρά. Η δομή της, στην πιο απλή μορφή, αποτελείται από μία σειρά αριθμητικών τιμών (value), κάθε μία από τις οποίες συνδυάζεται με μία χρονική σήμανση (timestamp) και ορίζεται από ένα όνομα (name) και ένα σύνολο ετικετών (tags).[4] Για την καλύτερη κατανόηση παρουσιάζεται το ακόλουθο παράδειγμα. Έστω ότι υπάρχουν αισθητήρες που συλλέγουν στοιχεία από τρία περιβάλλοντα: μία πόλη, ένα αγρόκτημα και ένα εργοστάσιο. Κάθε μία από αυτές τις πηγές στέλνει περιοδικά νέες αναγνώσεις, δημιουργώντας μια σειρά μετρήσεων που συλλέχθηκαν με την πάροδο του χρόνου. Τα σύνολα δεδομένων έχουν κατά κύριο λόγο 3 κοινά πράγματα:

- Τα δεδομένα που φθάνουν καταγράφονται σχεδόν πάντα ως νέα εγγραφή
- Τα δεδομένα συνήθως φτάνουν με χρονοδιάγραμμα
- Ο χρόνος είναι ένας κύριος άξονας (τα χρονικά διαστήματα μπορεί να είναι είτε κανονικά είτε ακανόνιστα)

### 2.2.2 Αποθήκευση χρονοσειρών

Γενικά τα σύνολα χρονοσειρών παρακολουθούν τις αλλαγές του συνολικού συστήματος ως νέες εισαγωγές και όχι ως ενημερώσεις. Η πρακτική καταγραφής κάθε αλλαγής στο σύστημα ως μια νέα, διαφορετική σειρά είναι αυτό που καθιστά τα δεδομένα της χρονοσειράς τόσο ισχυρά. Επιτρέπει να αναλυθεί το πώς άλλαξε κάτι στο παρελθόν, να παρακολουθηθεί πώς αλλάζει κάτι στο παρόν και να προβλεφθεί το πώς μπορεί να αλλάξει στο μέλλον. Φυσικά, η αποθήκευση δεδομένων σε αυτά τα πλαίσια έρχεται με ένα προφανές πρόβλημα: τα δεδομένα χρονοσειρών συσσωρεύονται πολύ γρήγορα. Αυτό έχει αντίκτυπο όχι μόνο στην καταγραφή τους αλλά και την ανάκληση τους, γεγονός που έστρεψε τους διαχειριστές στη δημιουργία ειδικών βάσεων δεδομένων χρονοσειρών καθώς οι σχεσιακές δεν κάλυπταν επαρκώς τις ανάγκες τους. Όπως φαίνεται ξεκάθαρα από την Εικόνα 1 οι βάσεις δεδομένων χρονοσειρών (Time Series Data Bases – TSDB) γνωρίζουν ιδιαίτερη άνθιση τα τελευταία χρόνια [5] και αποτελούν πόλο έλξης για διεξοδική μελέτη και επέκταση από αρκετούς ερευνητές, με την ταυτόχρονη ανάπτυξη του διαδικτύου των πραγμάτων (internet of things), για 2 κύριους λόγους: (1) κλιμάκωση και (2) χρηστικότητα.




*Εικόνα 1. Τάσεις σε σύγχρονες Βάσεις Δεδομένων*

**Κλιμάκωση:** Με το πέρασμα του χρόνου τα δεδομένα χρονοσειρών συσσωρεύονται με ταχείς ρυθμούς, δημιουργώντας έναν τεράστιο όγκο πληροφορίας που πρέπει να αποθηκευτεί [4]. Για παράδειγμα, ένα μόνο συνδεδεμένο αυτοκίνητο συλλέγει 4.000 GB δεδομένων ανά ημέρα. Οι παραδοσιακές βάσεις δεδομένων δεν έχουν σχεδιαστεί για να χειρίζονται αυτήν την κλίμακα πληροφορίας. Οι σχεσιακές βάσεις δεδομένων αποδίδουν ελάχιστα με πολύ μεγάλα σύνολα δεδομένων. Αντίστοιχα, αν και οι βάσεις δεδομένων NoSQL ταιριάζουν καλύτερα ως προς την κλιμάκωση, έχουνε μεγάλα περιθώρια βελτιστοποίησης σχετικά με την ακρίβεια για δεδομένα χρονοσειρών. Αντίθετα, οι βάσεις δεδομένων χρονοσειρών (που μπορούν να βασίζονται σε σχεσιακές ή NoSQL βάσεις δεδομένων) χειρίζονται την κλίμακα με τρομερή αποτελεσματικότητα που είναι εφικτό μόνο όταν αντιμετωπίζετε ο χρόνος ως πολίτης πρώτης τάξης. Αυτές οι βελτιώσεις αποδόσεων έχουν ως αποτέλεσμα υψηλότερα ποσοστά ανάγνωσης δεδομένων, ταχύτερα ερωτήματα και καλύτερη συμπίεση δεδομένων.

**Χρηστικότητα:** Οι βάσεις δεδομένων χρονοσειρών περιλαμβάνουν συνήθως λειτουργίες και συναρτήσεις που είναι χρήσιμες για την ανάλυση δεδομένων χρονοσειρών, όπως πολιτικές διατήρησης δεδομένων, συνεχείς ερωτήσεις, ευέλικτα χρονικά σύνολα κ.λπ. Ακόμη και αν η κλίμακα δεν αποτελεί ανησυχία αυτή τη στιγμή (π.χ. αν είναι η αρχή συλλογής δεδομένων), αυτά τα χαρακτηριστικά μπορούν να παρέχουν ακόμα καλύτερη εμπειρία χρήστη και να διευκολύνουν την όλη διαδικασία.

Αυτός είναι ο λόγος για τον οποίο οι προγραμματιστές υιοθετούν όλο και περισσότερο βάσεις δεδομένων χρονολογικών σειρών και τις χρησιμοποιούν για διάφορες περιπτώσεις εφαρμογών, όπως:

- Παρακολούθηση συστημάτων λογισμικού: Εικονικές μηχανές, containers, υπηρεσίες, εφαρμογές.
- Παρακολούθηση φυσικών συστημάτων: Εξοπλισμός, μηχανήματα, συνδεδεμένες συσκευές, περιβάλλον, σπίτια, σώμα.

- Παρακολούθηση περιουσιακών στοιχείων: Οχήματα, φορτηγά, φυσικά εμπορευματοκιβώτια, παλέτες.
- Συστήματα χρηματοοικονομικών συναλλαγών: Εκτίμηση τιμών κρυπτονομισμάτων, έλεγχος εύρεσης ανωμαλιών που οδηγούν σε κενά ασφαλείας.
- Εφαρμογές συμβάντων: Παρακολούθηση δεδομένων αλληλεπίδρασης χρήστη / πελάτη.
- Εργαλεία επιχειρηματικής ευφυΐας: Παρακολούθηση των βασικών μετρήσεων και της συνολικής υγείας της επιχείρησης.

### 2.2.3 Ανάλυση χρονοσειρών

Η ανάλυση δεδομένων είναι μια διαδικασία επιθεώρησης, καθαρισμού, μετασχηματισμού και μοντελοποίησης δεδομένων με στόχο την ανακάλυψη χρήσιμων πληροφοριών, τη διαμόρφωση τελικών συμπερασμάτων και την υποστήριξη λήψης αποφάσεων. Η ανάλυση δεδομένων έχει πολλαπλές πτυχές και προσεγγίσεις, που περιλαμβάνουν πληθώρα τεχνικών με ποικίλα ονόματα και χρησιμοποιείται σε διαφορετικούς τομείς των επιχειρήσεων, της επιστήμης και της κοινωνικής επιστήμης. Η ανάλυση των χρονοσειρών μας βοηθάει να κατανοήσουμε ποιες είναι οι υποκείμενες δυνάμεις που οδηγούν σε μια συγκεκριμένη τάση σε διάφορα σημεία και μας βοηθά στην πρόβλεψη και την παρακολούθηση αυτών με την τοποθέτηση των κατάλληλων μοντέλων [6]. Οι διάφοροι τύποι ανάλυσης χρονοσειρών περιλαμβάνουν:

**Περιγραφική:** Για να προσδιοριστεί η τάση ή το πρότυπο σε μια χρονοσειρά χρησιμοποιώντας γραφήματα ή άλλα εργαλεία. Αυτό μας βοηθά να προσδιορίσουμε τα κυκλικά πρότυπα, τις γενικές τάσεις και τα σημεία καμπής.

**Εκπαιδευτική:** Μελετά τη διασταυρούμενη συσχέτιση μεταξύ δύο χρονικών σειρών και την εξάρτηση μεταξύ τους.

**Παρεμβατική:** Χρησιμοποιείται για να προσδιοριστεί εάν ένα γεγονός μπορεί να οδηγήσει σε αλλαγή στις χρονοσειρές.

**Πρόβλεψη:** Χρησιμοποιείται εκτενώς στην επιχειρηματική πρόβλεψη, προϋπολογισμό κ.λπ. με βάση τις ιστορικές τάσεις.

**Φασματική:** Αναφέρεται επίσης ως τομέας συχνοτήτων και αποσκοπεί στο διαχωρισμό περιοδικών ή κυκλικών συνιστωσών σε μια χρονοσειρά.

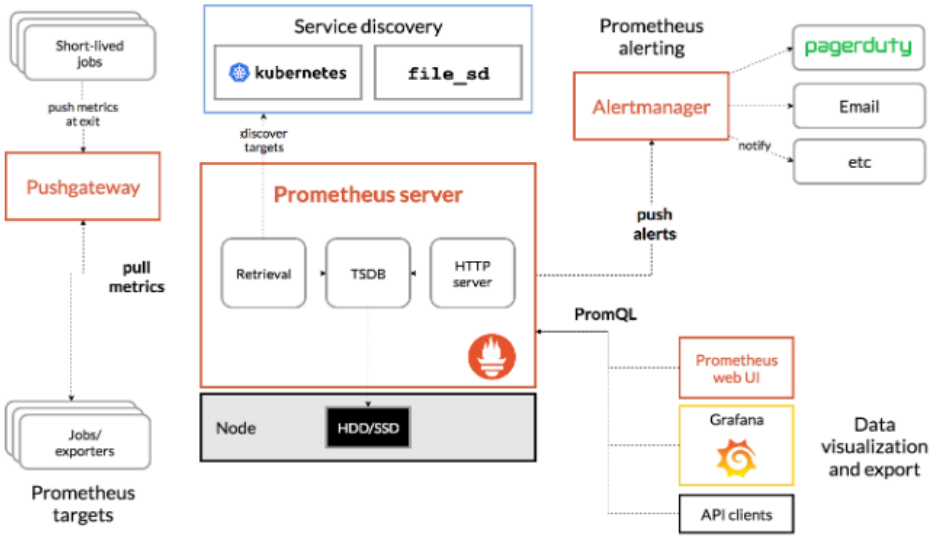
# 3 Αρχιτεκτονική Εφαρμογής

## 3.1 Prometheus

### 3.1.1 Δομή συστήματος

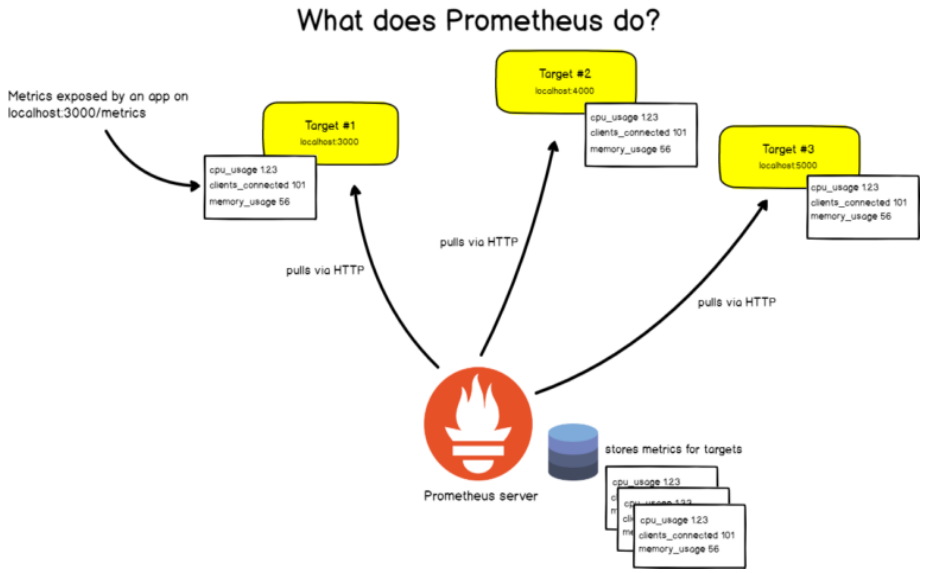
Το Prometheus είναι ένα από τα πιο διαδεδομένα εργαλεία εποπτείας και ειδοποίησης ανοιχτού κώδικα. Το λογισμικό δημιουργήθηκε λόγω της ανάγκης παρακολούθησης πολλαπλών μικροϋπηρεσιών (microservices) που ενδέχεται να εκτελούνται σε ένα σύστημα. Η αρχιτεκτονική του είναι δομοστοιχειωτή (modular) και συμπληρώνεται από ευρέως διαθέσιμες λειτουργικές μονάδες (modules) που ονομάζονται εξαγωγείς (exporters). Οι εξαγωγείς είναι υπεύθυνοι για την καταγραφή μετρήσεων και είναι συμβατοί με τα πιο δημοφιλή λογισμικά [12]. Η ανάπτυξη του Prometheus είναι βασισμένη στη γλώσσα προγραμματισμού Go και συνοδεύεται από εκτελέσιμα προγράμματα (distributed binaries) που χρησιμοποιούνται για τη γρήγορη εγκατάσταση και την άμεση λειτουργία του. Έχει τις ρίζες του στο SoundCloud και από την ίδρυση του το 2012 έχει υιοθετηθεί από πολλές εταιρείες και οργανισμούς αλλά και από μια πολύ ενεργή κοινότητα προγραμματιστών και χρηστών. Το Prometheus προσχώρησε στο Ίδρυμα Cloud Native Computing το 2016 ως το δεύτερο φιλοξενούμενο έργο, μετά το Kubernetes [13].

Η αρχιτεκτονική του Prometheus, όπως λαμβάνεται από την επίσημη ιστοσελίδα, παρουσιάζεται στην Εικόνα 2. Η αρχιτεκτονική του φαίνεται περίπλοκη, αλλά χωρίζεται σε ενότητες καθεμία από τις οποίες έχει συγκεκριμένο ρόλο στο συνολικό σύστημα. Το Prometheus είναι χωρισμένο σε δύο μεγάλα τμήματα. (α) το πρώτο και σημαντικότερο αφορά τη συλλογή μετρήσεων και (β) το δεύτερο αφορά την αποστολή ειδοποιήσεων. Οι βασικές ενότητες είναι οι ακόλουθες:




*Εικόνα 2. Αρχιτεκτονική του Prometheus*

**Prometheus server:** Αποτελεί την καρδιά του συστήματος αφού είναι υπεύθυνος για τη συλλογή μετρήσεων από έναν αριθμό κόμβων και την αποθήκευση τους τοπικά. Είναι σχεδιασμένο να παρακολουθεί μεγάλο εύρος μηχανημάτων και λογισμικού όπως διακομιστές, βάσεις δεδομένων, αυτόνομες εικονικές μηχανές κ.λπ. Το Prometheus λειτουργεί με βάση την αρχή του scraping, δηλαδή, αναμένει να ανακτήσει μετρήσεις μέσω HTTP calls που γίνονται σε συγκεκριμένα τελικά σημεία τα όποια ορίζονται κατά τη ρύθμιση της διαδικασίας ανάκτησης. Η συλλογή τους γίνεται ανά τακτά χρονικά διαστήματα από συγκεκριμένους εξαγωγείς, οι οποίοι συλλέγουν την πληροφορία και τη μεταφράζουν σε μορφή κατάλληλη ώστε να μπορεί στη συνέχεια να τη λάβει ο διακομιστής [16]. (Εικόνα 3)




*Εικόνα 3. Παράδειγμα λειτουργίας Prometheus*

**Push gateway:** Είναι μια ενδιάμεση υπηρεσία που επιτρέπει την αποστολή (push) μετρήσεων χρονοσειρών από συνήθως εφήμερες εργασίες και όπου δεν μπορεί να γίνει ανάκτηση δεδομένων.

**Alert manager:** Το τμήμα για την αποστολή ειδοποιήσεων. Δίνεται ιδιαίτερη βαρύτητα ώστε να παρέχεται στο χρήστη η δυνατότητα να ορίσει τις δικές του ειδοποιήσεις σχετικά με αυτές τις μετρήσεις και να μπορεί να ενημερώνεται για αποκλίσεις ή αλλαγές στα επίπεδα τιμών. Τέλος, παρέχεται η δυνατότητα αποστολής αυτών των ειδοποιήσεων μέσω πολλαπλών καναλιών όπως SMS, email, Slack κ.λπ.

**Data visualization:** Το Prometheus έρχεται με τη δική του διεπαφή χρήστη (UI) που επιτρέπει τον έλεγχο των ρυθμίσεων, των κόμβων και των γραφημάτων των δεδομένων (χρονοσειρών) με εφαρμογή κατάλληλων φίλτρων. Επιπλέον, είναι συμβατό με το Grafana, μια κορυφαία ανοικτού λογισμικού εφαρμογή οπτικοποίησης δεδομένων, έτσι ώστε τα δεδομένα του Prometheus να είναι διαθέσιμα για προβολή. Το Prometheus εκθέτει επίσης μία

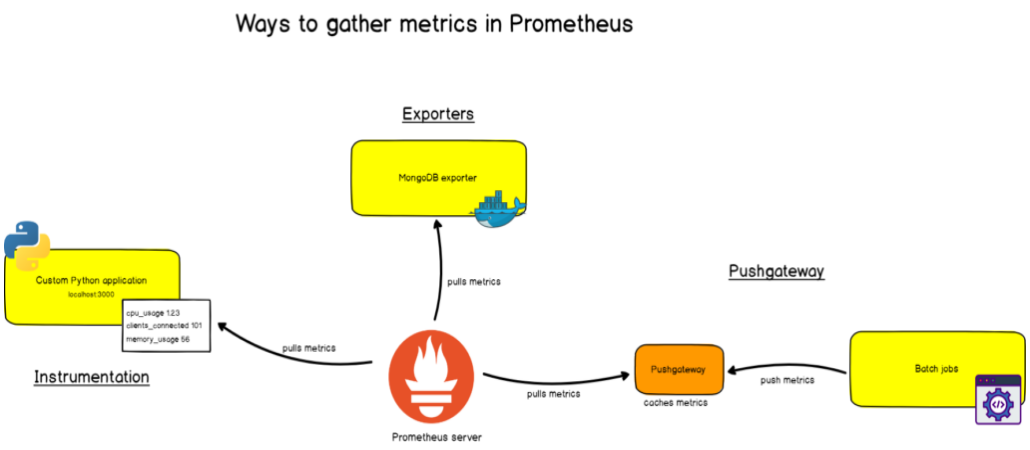
διεπαφή προγραμματισμού εφαρμογών (API), οπότε υπάρχει η δυνατότητα δημιουργίας εξειδικευμένων υλοποιήσεων.

**Service discovery:** Το Prometheus μπορεί να ανακαλύπτει δυναμικά τους στόχους και να βρίσκει νέους κατόπιν ζήτησης. Αυτό είναι ιδιαίτερα χρηστικό όταν χρησιμοποιούνται πολλαπλές απομονωμένες οντότητες περιοχής χρήστη, γνωστές ως containers, που μπορούν να αλλάξουν δυναμικά διευθύνσεις IP ανάλογα με τη ζήτηση. Το Prometheus λειτουργεί ικανοποιητικά για την καταγραφή αποκλειστικά αριθμητικών χρονοσειρών. Γίνεται καλή εφαρμογή του όταν πρόκειται τόσο για μηχανοκεντρική παρακολούθηση (machine-centric monitoring) όσο και για εποπτεία αρχιτεκτονικών υψηλής δυναμικότητας προσανατολισμένων στην υπηρεσία (highly dynamic service-oriented architectures monitoring).

### 3.1.2 Μοντέλο εξαγωγής δεδομένων

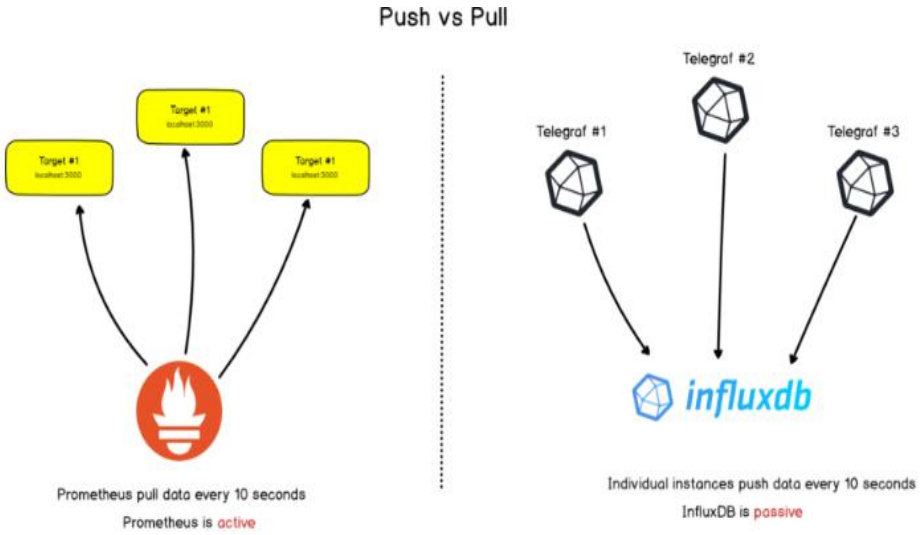
Η εξαγωγή δεδομένων χρονοσειρών από τα υπό εξέταση συστήματα γίνεται με διάφορους τρόπους (Εικόνα 4):

- Με την κατάλληλη παραμετροποίηση των εφαρμογών, όπου κάθε εφαρμογή εμφανίζει τις μετρήσεις της με συμβατή μορφή σε μια δεδομένη διεύθυνση (URL) και το Prometheus είναι υπεύθυνο να τις συλλέγει περιοδικά.
- Με τη χρήση ήδη διαθέσιμων λογισμικών εξαγωγής δεδομένων (prebuilt exporters). Διατίθεται μια ολοκληρωμένη συλλογή εξαγωγέων για υπάρχουσες τεχνολογίες, όπως παρακολούθηση μηχανών Linux (node exporter), βάσεις δεδομένων (SQL exporter / MongoDB exporter) ακόμα και HTTP load balancers (HAProxy exporter).
- Με τη βοήθεια του Push Gateway. Μερικές φορές, οι εφαρμογές ή εργασίες δεν εκθέτουν άμεσα μετρήσεις. Αυτές οι εφαρμογές είτε δεν έχουν σχεδιαστεί για αυτό (π.χ. batch jobs), είτε έχει επιλέξει ο διαχειριστής να μην εκθέτει αυτές τις μετρήσεις απευθείας μέσω της εφαρμογής.




*Εικόνα 4. Τρόποι συλλογής δεδομένων στο Prometheus*

Με εξαίρεση την τελευταία περίπτωση πρέπει να επισημανθεί ότι το Prometheus είναι ένα pull-based σύστημα παρακολούθησης. Αυτή είναι και η αξιοσημείωτη διαφορά μεταξύ του τρόπου λειτουργίας του Prometheus και άλλων βάσεων δεδομένων χρονοσειρών: το Prometheus ελέγχει ενεργά τους στόχους για να ανακτήσει μετρήσεις από αυτούς. Αυτό είναι πολύ διαφορετικό από το InfluxDB για παράδειγμα, όπου χρησιμοποιεί push data μοντέλο για την εισαγωγή δεδομένων (Εικόνα 5).




*Εικόνα 5. Push/Pull μοντέλα*

Από τη διαθέσιμη βιβλιογραφία, προκύπτουν οι λόγοι επιλογής του Prometheus στην αρχιτεκτονική που ακολουθήθηκε στα πλαίσια της διπλωματικής εργασίας:

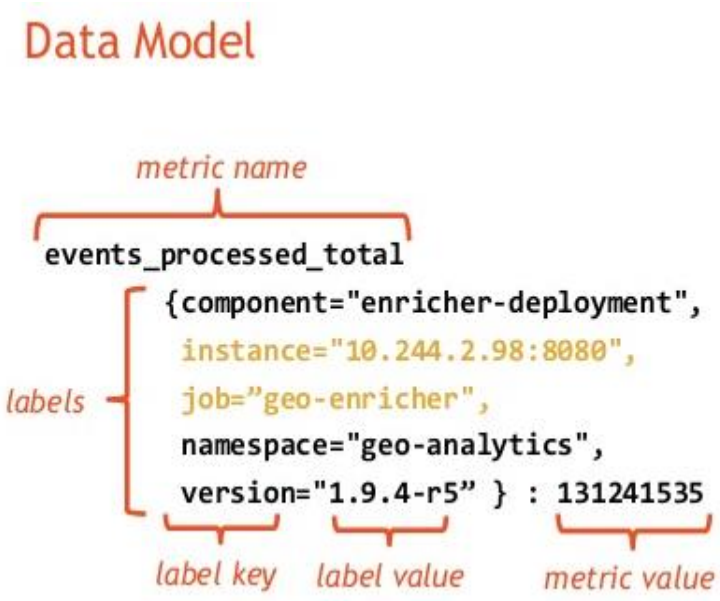
- Centralized control: Όταν το Prometheus ξεκινά τα ερωτήματα (queries) στους διάφορους στόχους (targets), ολόκληρη η διαμόρφωσή τους γίνεται στην πλευρά του διακομιστή και όχι στους επιμέρους στόχους. Επομένως αυτός καθορίζει σε ποιους στόχους και κάθε πότε θα γίνεται ανάκτηση δεδομένων. Ένα σύστημα που βασίζεται σε push-data μοντέλο ενέχει τον κίνδυνο να σταλούν πάρα πολλά δεδομένα προς τον διακομιστή και ουσιαστικά να καταστραφούν. Ένα σύστημα βασισμένο στο pull-data μοντέλο επιτρέπει τον έλεγχο του ρυθμού κάτι που του προσδίδει την ευελιξία να έχει πολλαπλές ρυθμίσεις ανάκτησης δεδομένων και έτσι πολλαπλές τιμές για διαφορετικούς στόχους.
- Το Prometheus προορίζεται για την αποθήκευση συγκεντρωτικών μετρήσεων και δεν αποτελεί ένα σύστημα βασισμένο σε συμβάντα (event-based system), γεγονός που το καθιστά πολύ διαφορετικό από άλλες βάσεις δεδομένων χρονοσειρών. Δεν έχει σχεδιαστεί για να συλλαμβάνει γεγονότα εγκαίρως (όπως για παράδειγμα, μια διακοπή υπηρεσίας) αλλά για να συγκεντρώνει προ-συγκεντρωμένες μετρήσεις για τις υπηρεσίες.

Αυτές είναι οι βασικές διαφορές μεταξύ μιας βάσης δεδομένων χρονοσειρών που στοχεύει για συγκεντρωτικές μετρήσεις και μιας που έχει σχεδιαστεί για να συγκεντρώνει "ακατέργαστες μετρήσεις".

### 3.1.3 Έννοιες εποπτείας

#### *3.1.3.1 Μοντέλο δεδομένων Κλειδιού-Τιμής*

Το Prometheus αποθηκεύει τα δεδομένα ως χρονοσειρές. Κάθε χρονοσειρά χαρακτηρίζεται από μία μέτρηση και ένα προαιρετικό σύνολο ετικετών ζεύγους κλειδιού-τιμής (key-value). Επομένως μια χρονοσειρά μπορεί τυπικά να οριστεί ως <metric> {<label_1> = <value1>, <label_2> = <value2> ...} <metric_value>. Ένα δείγμα δεδομένων σε μια χρονοσειρά περιέχει μια τιμή τύπου float64, μια χρονική σήμανση (timestamp) σε μορφή unix και τις τιμές ετικέτας για τη μέτρηση αυτή.(Εικόνα 6).




*Εικόνα 6. Παράδειγμα μοντέλου δεδομένων*

#### *3.1.3.2 Τύποι μετρητών*

Το Prometheus επιτρέπει τη χρήση τεσσάρων (4) τύπων για την περιγραφή των μετρήσεων:

**Counters:** Η απλούστερη μορφή μετρικού τύπου που μπορεί να χρησιμοποιηθεί είναι ο αθροιστής (counter). Ένας τέτοιος τύπος μετράει τα στοιχεία με την πάροδο του χρόνου.

**Gauges:** Οι μετρητές τύπου δείκτη (gauge) έχουν σχεδιαστεί για να παρακολουθούν τιμές που μπορεί να αυξομειώνονται με την πάροδο του χρόνου. Οπτικά, είναι σαν τα θερμόμετρα: σε κάθε δεδομένη στιγμή, εάν παρατηρηθεί το θερμόμετρο, φαίνεται η τρέχουσα τιμή θερμοκρασίας. Με μια πρώτη ματιά ο συγκεκριμένος τύπος φαίνεται να

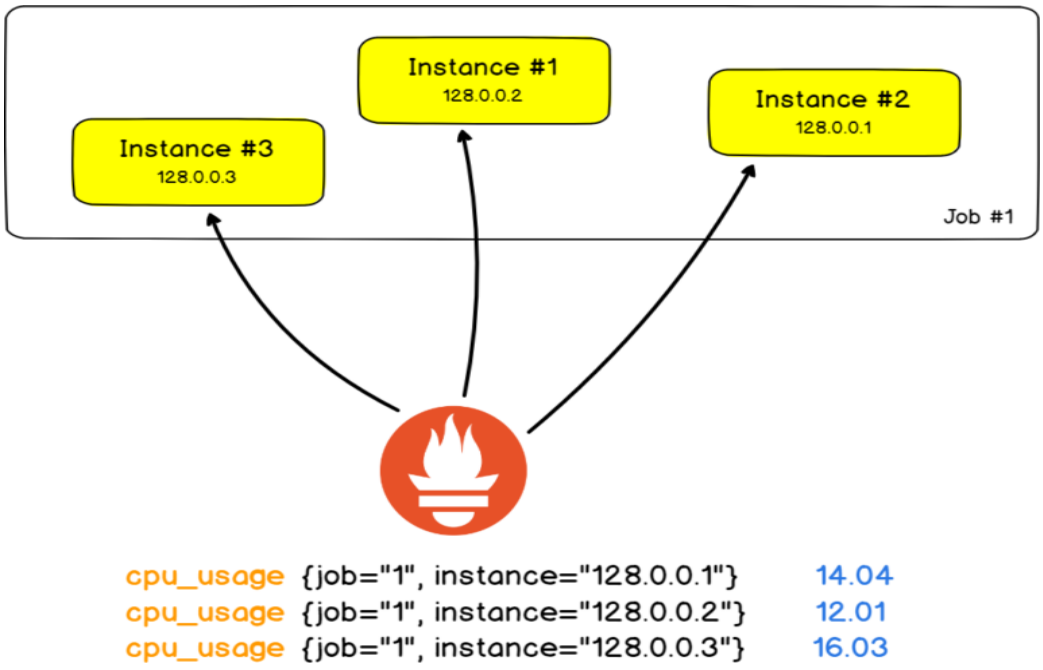
αναιρεί τη χρηστικότητα των counters. Ωστόσο υπάρχει μια ειδοποιός διαφορά: τα gauges δεν επιτρέπουν την παρακολούθηση της εξέλιξης μια τιμής με την πάροδο του χρόνου. Πιο συγκεκριμένα εάν το σύστημα στέλνει μετρήσεις ανά 5 δευτερόλεπτα, αλλά το Prometheus κάνει ανάκτηση ανά 15, χάνονται ορισμένες μετρήσεις και άρα σε περίπτωση που γίνεται επεξεργασία πάνω σε αυτά, τα αποτελέσματα που εξάγονται θα είναι λιγότερο ακριβή.

**Histograms:** Πρόκειται για ένα πιο σύνθετο μετρικό τύπο που παρέχει πρόσθετες πληροφορίες για τις μετρήσεις αθροίζοντας τες ανά περιοχή τιμών, Επιπλέον παρέχει και το συνολικό άθροισμα των παρατηρήσεων αυτών.

**Summaries:** Ο συγκεκριμένος μετρικός τύπος αποτελεί επέκταση των ιστογραμμάτων όπου εκτός από την παροχή του αθροίσματος και του αριθμού των παρατηρήσεων ανά περιοχή τιμών, παρέχει ποσοστιαίες μετρήσεις (quantiles metrics) σε ολισθαίνοντα παράθυρα (sliding windows).

### *3.1.3.3 Jobs και Instances*

Με τις πρόσφατες εξελίξεις που έγιναν σε κατανεμημένες αρχιτεκτονικές και την αύξηση της δημοτικότητας των λύσεων νέφους (cloud), δεν υπάρχει απλά ένας διακομιστής. Οι διακομιστές αντιγράφονται και διανέμονται σε όλο τον κόσμο. Σε όρους του Prometheus, ένα τελικό σημείο (endpoint) από όπου μπορεί να γίνει ανάκληση δεδομένων ονομάζεται instance, και συνήθως αντιστοιχεί σε μία διαδικασία. Μια συλλογή από instances με τον ίδιο σκοπό ονομάζεται εργασία (job). Το μεγάλο όφελος είναι ότι οι εργασίες και τα instances γίνονται πεδία στις ετικέτες (labels) και επιτρέπουν το καλύτερο φιλτράρισμα των αποτελεσμάτων (Εικόνα 7).




*Εικόνα 7. Filtering με τη χρήση job και instances πεδίων*

#### *3.1.3.4 PromQL*

Το Prometheus έχει τη δική του ενσωματωμένη γλώσσα που διευκολύνει την αναζήτηση και την ανάκτηση δεδομένων από τους διακομιστές γνωστή ως PromQL. Τα δεδομένα αντιπροσωπεύονται χρησιμοποιώντας τα ζεύγη κλειδιών-τιμών. Το PromQL δεν απέχει πολύ από αυτό, καθώς διατηρεί την ίδια σύνταξη και επιστρέφει τα αποτελέσματα ως διανύσματα. Με τη συγκεκριμένη γλώσσα ο χρήστης έχει στη διάθεση του δύο (2) είδη διανυσμάτων.

**Instant vectors:** Δίνουν μια αναπαράσταση όλων των μετρήσεων που παρακολουθούνται στην πιο πρόσφατη χρονική σήμανση.

**Time ranged vectors:** Αν ο χρήστης θέλει να παρακολουθήσει την εξέλιξη μιας μέτρησης με την πάροδο του χρόνου, μπορεί να αναζητήσει στον Prometheus με προσαρμοσμένα χρονικά διαστήματα. Το αποτέλεσμα θα είναι ένας διανυσματικός συνδυασμός όλων των τιμών που καταγράφηκαν για την επιλεγμένη περίοδο.

Το PromQL API εκθέτει ένα σύνολο λειτουργιών που διευκολύνουν την επεξεργασία δεδομένων για τα ερωτήματα. Παρέχεται η δυνατότητα ταξινόμησης των μετρήσεων, εφαρμογής μαθηματικών λειτουργιών πάνω σε αυτά (όπως παράγωγος ή εκθετικές λειτουργίες) και λειτουργίες πρόβλεψης (όπως τη λειτουργία Holt Winters)

## 3.2 Influxdb

### 3.2.1 Influxdb

Με την τωρινή τους μορφή οι σχεσιακές βάσεις δεδομένων δεν κρίνονται ιδανικές για την αποθήκευση χρονοσειρών και για αυτό σχεδιάστηκαν βάσεις δεδομένων ειδικά για χρονοσειρές όπως το InfluxDB, η TimescaleDB κλπ που επιτυγχάνουν ικανοποιητικά την κατάλληλη διαχείριση τους. Το InfluxDB αναπτύχθηκε από την InfluxData και πρόκειται για ένα εργαλείο ανοικτού κώδικα για μεγάλα δεδομένα, μία βάση δεδομένων NoSQL με κύρια χαρακτηριστικά την μαζική κλιμάκωση και την υψηλή διαθεσιμότητα, που επιτρέπει την γρήγορη εγγραφή και ανάγνωση δεδομένων χρονοσειρών [14]. Το InfluxDB είναι επίσης γραμμένο σε Go που είναι κύριας σημασίας καθώς καθιστά εύκολες τις διαδικασίες μεταγλώττισης (compiling) και ανάπτυξης (deployment) χωρίς εξωτερικές εξαρτήσεις, προσφέροντας μια τύπου SQL-like query γλώσσα. Ως εκ τούτου βελτιστοποιείται για γρήγορη και υψηλής απόδοσης αποθήκευση και ανάκτηση δεδομένων χρονοσειρών σε πεδία όπως η παρακολούθηση λειτουργιών, οι μετρήσεις εφαρμογών και τα δεδομένα αισθητήρων του Internet of Things. Ο σχεδιασμός της αρχιτεκτονικής plug-in καθιστά πολύ ευέλικτη την ενσωμάτωση άλλων προϊόντων διαφορετικών κατασκευαστών σε αυτό. Όπως και σε άλλες βάσεις δεδομένων NoSQL, υποστηρίζονται διαφορετικοί clients όπως Go, Java, Python και

Node.js για να αλληλοεπιδράσουν με τη βάση δεδομένων [15]. Το εγγενές HTTP API μπορεί εύκολα να ενσωματωθεί σε προϊόντα που βασίζονται στο διαδίκτυο, όπως DevOps για την παρακολούθηση δεδομένων σε πραγματικό χρόνο. Δεδομένου ότι είναι ειδικά σχεδιασμένο για δεδομένα χρονοσειρών, έγινε όλο και πιο δημοφιλές σε περιπτώσεις όπως η παρακολούθηση DevOps, η παρακολούθηση του διαδικτύου των πραγμάτων (IoT) και η εφαρμογή αναλυτικών δεδομένων βάσει χρονοσειρών. Η κλασική περίπτωση χρήσης δεδομένων χρονοσειρών περιλαμβάνει τα ακόλουθα:

- Μητρώα συστήματος και παρακολούθησης
- Χρηματοπιστωτικές / χρηματιστηριακές συναλλαγές με την πάροδο του χρόνου στις χρηματαγορές
- Παρακολούθηση απογραφής προϊόντων στο σύστημα λιανικής πώλησης
- Καταγραφή δεδομένων αισθητήρων σε διαδίκτυο και βιομηχανικό διαδίκτυο των πραγμάτων (IoT)
- Γεωτοποθέτηση και παρακολούθηση στον κλάδο των μεταφορών

Τα δεδομένα για κάθε μία από αυτές τις περιπτώσεις χρήσης είναι διαφορετικά, αλλά συχνά έχουν παρόμοιο μοτίβο. Στην περίπτωση του συστήματος και των αρχείων παρακολούθησης, ο χρήστης λαμβάνει τακτικές μετρήσεις για την παρακολούθηση διαφόρων υπηρεσιών παραγωγής όπως Apache, Tomcat, MySQL, Hadoop, Kafka, Spark, Hive, Web εφαρμογές κλπ. Οι χρονοσειρές συνήθως έχουν πληροφορίες μεταδεδομένων όπως το όνομα του διακομιστή, το όνομα της υπηρεσίας και τη μέτρηση που λαμβάνεται.

Σε μια σχεσιακή βάση δεδομένων, αν και οι τρόποι για να δομηθούν τα δεδομένα είναι συγκεκριμένοι, υπάρχουν ορισμένες προκλήσεις:

- Δημιουργία ενός ενιαίου διανεμημένου πίνακα για την αποθήκευση όλων των δεδομένων με το όνομα της σειράς, την τιμή και την ώρα που θα προκαλούσε γρήγορα πρόβλημα απόδοσης λόγω του μεγέθους του πίνακα.
- Δημιουργία ενός ξεχωριστού πίνακα ανά χρονική περίοδο (ημέρα, μήνα και ούτω καθεξής).

### 3.2.2 Μοντέλο δεδομένων

Τα δεδομένα που είναι αποθηκευμένα στο InfluxDB είναι γενικά μια ακολουθία δεδομένων με βάση το χρόνο. Τα αρχεία έχουν συνήθως εκατοντάδες εκατομμύρια σειρές, συμπεριλαμβανομένων των timestamps και άλλων πεδίων και ετικετών. Συνήθως, το σημείο δεδομένων είναι αμετάβλητο και μόνο για ανάγνωση. Το νέο σημείο θα γραφτεί αυτόματα και θα συνεχίσει να προσαρτάται στη μέτρηση. Για ένα μεγάλο όγκο δεδομένων, είναι απαραίτητος ο προσεκτικός σχεδιασμός για τη συλλογή δεδομένων. Απαιτείται σωστός

καθορισμός του χαρακτηριστικού που θέλει ο χρήστης να χρησιμοποιήσει για να γίνει κατηγοριοποίηση (indexing), έτσι ώστε τα υπόλοιπα πεδία να μην συμμετέχουν σε αυτή και προκαλούν επιβάρυνση στο σύστημα. Τα παραπάνω βήματα είναι κρίσιμα για την απόδοση των ερωτημάτων.

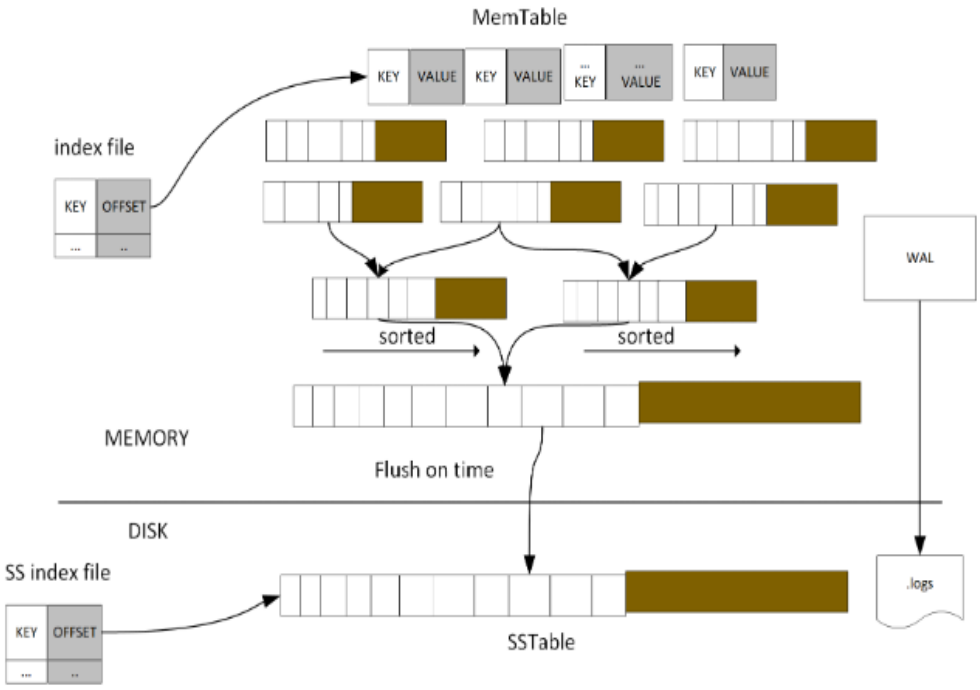
Στην επίσημη σελίδα τονίζεται η αποφυγή πολλών σειρών, καθώς οι ετικέτες περιέχουν πληροφορίες μεγάλων συμβολοσειρών, όπως hashes και καθολικά μοναδικά αναγνωριστικά (universally unique identifiers-UUID), τα οποία μπορούν να προκαλέσουν μεγάλη χρήση μνήμης και επιπλέον φόρτο εργασίας στις βάσεις δεδομένων. Ένας από τους βασικούς ρόλους που επηρεάζουν την απόδοση είναι ο υψηλός αριθμός στοιχείων, γνωστό ως cardinality της σειράς και συχνά προκαλεί υψηλή χρήση RAM. Με βάση τις πρόσφατες οδηγίες για το hardware, για λιγότερο από 100k μοναδικές σειρές συνιστάται περίπου 2-4 GB μνήμης RAM. Όταν μια μέτρηση έχει πολύ δυναμικές τιμές ετικετών, όπως περισσότερες από χιλιάδες, μπορεί εύκολα να καταναλώσει περισσότερα από 32 GB χρήση μνήμης. Από την άλλη πλευρά, όταν τα κλειδιά και οι τιμές ετικετών αποθηκεύονται μόνο μία φορά, χρειάζονται μόνο περισσότερο χώρο αποθήκευσης και δεν επηρεάζουν το αποτύπωμα μνήμης (memory footprint).

### 3.2.3 Μηχανισμός αποθήκευσης

Η παλαιότερη έκδοση InfluxDB υποστηρίζει διάφορες μηχανές αποθήκευσης, όπως LevelDB, RocksDB, HyperLevelDB και LMDB. Οι περισσότερες από αυτές βασίζονται σε δέντρο συγχώνευσης με δομή καταγραφής (log-structured merge-tree/LSM Tree). Από την έκδοση 0.9.5, το InfluxDB έχει τη δική του μηχανή αποθήκευσης που ονομάζεται Δέντρο δομημένης συγχώνευσης (Tree Structured Merge Tree/TSM Tree).

Το LSM Tree δημιουργεί αρχεία ευρετηρίου, τα οποία παρέχουν αποτελεσματική κατηγοριοποίηση για δεδομένα υψηλών συναλλαγών, όπως δεδομένα καταγραφής. Μία από τις υλοποιήσεις του LSM Tree ονομάζεται Sorted String Table (SSTable). Όταν τα δεδομένα αποθηκεύονται σε SSTable, αποθηκεύονται ως ζεύγος κλειδιού-τιμής. Τα αρχεία ευρετηρίου περιέχουν αλλαγές που αφορούν δεδομένη δέσμη (batch data) για ορισμένη διάρκεια. Το LSM Tree χρησιμοποιεί πληροφορίες δέσμης (batch information) από αρχεία ευρετηρίου για συγχώνευση-ταξινόμηση για κάθε θραύσμα αρχείου δεδομένων και προσωρινή μνήμη. Επίσης παρέχει υψηλής απόδοσης ανάκτηση δεδομένων για μεταγενέστερη αναζήτηση. Δεδομένου ότι το αρχείο προσωρινής μνήμης είναι αμετάβλητο, νέα στοιχεία θα εισαχθούν σε νέα αρχεία. Το LSM Tree χρησιμοποιεί πληροφορίες δέσμης (batch information) από αρχεία ευρετηρίου για συγχώνευση-ταξινόμηση κάθε θραύσματος αρχείου δεδομένων και προσωρινή μνήμη. Επίσης παρέχει ανάκτηση δεδομένων υψηλής απόδοσης για

μεταγενέστερη αναζήτηση. Δεδομένου ότι το αρχείο προσωρινής μνήμης είναι αμετάβλητο, οι νέες εγγραφές θα εισαχθούν σε νέα αρχεία. Περιοδικά, ο αλγόριθμος συγχωνεύει τα αρχεία μαζί για να διατηρήσει τον αριθμό τους σε μικρά νούμερα. Ωστόσο, η αναζήτηση με συγκεκριμένα φίλτρα απαιτεί άμεση ανταπόκριση και άρα θα χάσει σε απόδοση I / O σε ορισμένες περιπτώσεις. Οπότε το δέντρο LSM είναι πιο χρήσιμο για εισαγωγές με κατηγοριοποίηση παρά για ανάκτηση των καταχωρήσεων. Για να γίνει ταχύτερη ανάγνωση των δέντρων LSM, η κοινή προσέγγιση είναι η διατήρηση ενός δείκτη σελίδας στη μνήμη.




*Εικόνα 8. LSM Tree- Continuous merge sort και δημιουργία indexing*

Το TSM Tree είναι παρόμοιο με το LSM Tree. Το TSM Tree έχει ένα αρχείο καταγραφής (Write-Ahead-Log WAL) και μια συλλογή αρχείων δεδομένων μόνο για ανάγνωση ή αρχείων TSM, τα οποία είναι παρόμοια σε έννοια με SSTables σε δέντρο LSM.

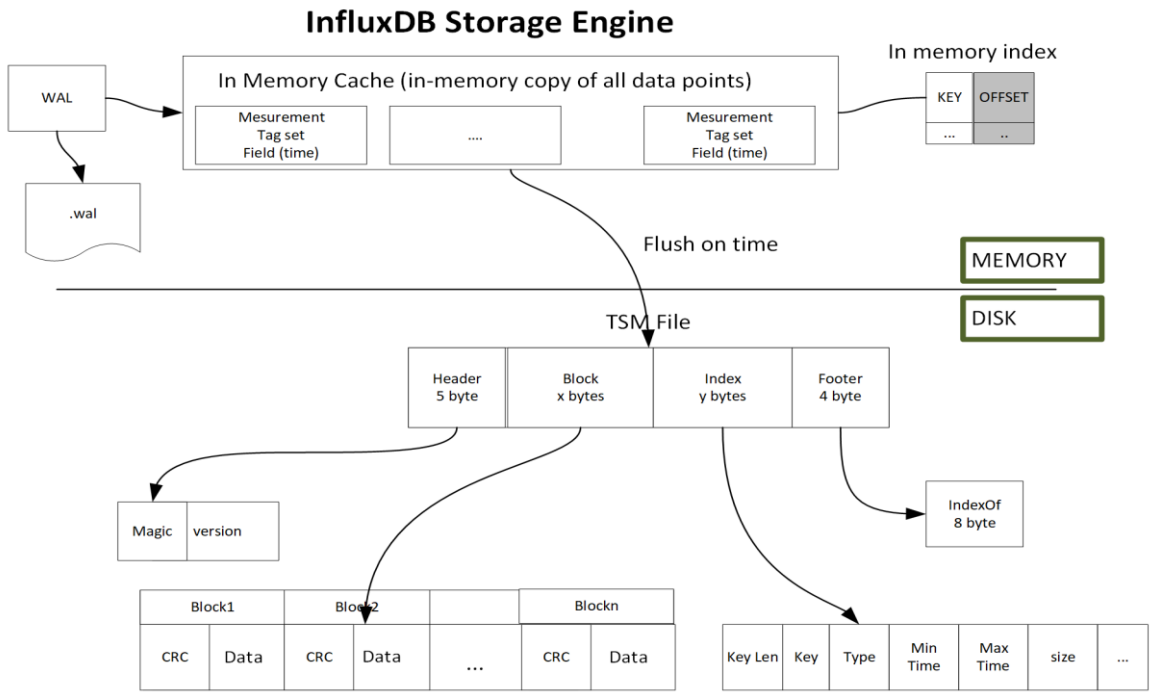
Τα WALs έχουν αρκετά πλεονεκτήματα, αλλά χρησιμοποιούνται κυρίως για τη διατήρηση της αντοχής στην εγγραφή και της ατομικότητας σε συστήματα βάσεων δεδομένων.

- Ανθεκτικότητα - Οι συνεχείς ενέργειες στο WAL εξασφαλίζουν πρώτα ότι αυτές οι ενέργειες θα εκτελεστούν ακόμα και αν η βάση δεδομένων καταρρεύσει. Καταγράφοντας τις ενέργειες σε WAL πριν πραγματοποιηθούν αλλαγές εντός της μνήμης, οι ενέργειες μπορούν να ανακτηθούν και να εφαρμοστούν ξανά, αν χρειαστεί, εξασφαλίζοντας έτσι την αντοχή εγγραφής.
- Ατομικότητα - Η ατομικότητα αναφέρεται σε μια ιδιότητα ενός συστήματος βάσης δεδομένων που εγγυάται ότι θα πραγματοποιηθεί πλήρως. Εάν η λειτουργία ενός

διακομιστή διακοπεί στο μέσο της εκτέλεσης διαφόρων ενεργειών, η βάση δεδομένων μπορεί να εξετάσει το WAL για να εντοπίσει το σημείο όπου έπαψε και να ολοκληρώσει την εργασία, εξασφαλίζοντας ότι η εργασία θα εκτελεστεί πλήρως.

Το αρχείο .wal δημιουργείται και οργανώνεται από το WAL, το μέγεθος του οποίου αυξάνεται σταδιακά. Όταν το μέγεθος γίνει 10 MB, το τρέχον αρχείο .wal θα κλείσει και θα δημιουργηθεί ένα νέο αρχείο .wal. Κάθε αρχείο τμήματος .wal περιέχει ένα μπλοκ συμπίεσης της εγγραφής και διαγραφής. Κάθε νέα άφιξη σημείου δεδομένων, θα συμπιεστεί χρησιμοποιώντας το Snappy και θα γραφτεί ως αρχείο .wal. Μια καταχώρηση θα προστεθεί σε ένα αρχείο ευρετηρίου μνήμης. Όταν φτάσει σε ένα ορισμένο αριθμό σημείων δεδομένων, το σημείο δεδομένων θα απομακρυνθεί (flush) από τον αποθηκευτικό δίσκο. Αυτό το χαρακτηριστικό γνώρισμα (batch feature) επιτυγχάνει υψηλή απόδοση. Τα σημεία δεδομένων αποθηκεύονται ως αρχεία TSM, τα οποία έχουν συμπιεσμένα δεδομένα σειράς σε στήλη. Κάθε αρχείο TSM έχει τέσσερα τμήματα: Header, Blocks, Index και Footer.

- Το τμήμα Header, αποτελείται από έναν "μαγικό αριθμό" για τον προσδιορισμό του τύπου αρχείου και έναν αριθμό έκδοσης.
- Το τμήμα Blocks, περιέχει CRC (Cyclic redundancy check) και δεδομένα.
- Το τμήμα Index, περιέχει το μήκος κλειδιού, το κλειδί (περιλαμβάνει το όνομα μέτρησης, το σύνολο ετικετών και ένα πεδίο που σχετίζεται με το χρόνο), το ελάχιστο μπλοκ και μέγιστο χρόνο κ.ο.κ.
- Το τμήμα Footer, αποθηκεύει την μετατόπιση του δείκτη όπως φαίνεται στην Εικόνα 9.

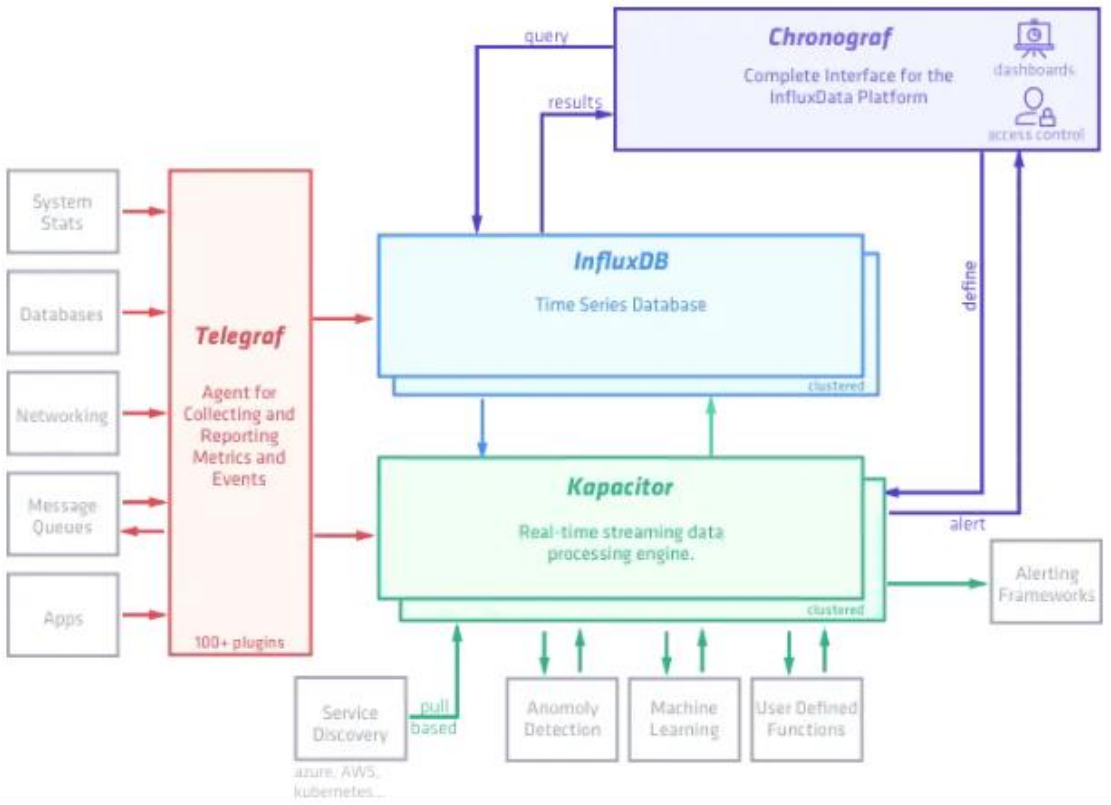



*Εικόνα 9. Μηχανή αποθήκευσης στο Influxdb*

### 3.2.4 Στοίβα TICK

Το InfluxData δημιούργησε μια πλατφόρμα βασισμένη σε τέσσερα εργαλεία ανοιχτού κώδικα που λειτουργούν καλά μεταξύ τους για δεδομένα χρονοσειρών. Όταν χρησιμοποιούνται μαζί, επιτρέπουν τη συλλογή, αποθήκευση, επεξεργασία και προβολή των δεδομένων με ευκολία. Τα τέσσερα κομμάτια της πλατφόρμας είναι γνωστά ως η στοίβα TICK:

- Telegraf: Διακομιστής με πρόσθετες μονάδες για τη συλλογή / αναφορά μετρήσεων
- InfluxDB: Κλιμακούμενος χώρος αποθήκευσης δεδομένων για μετρήσεις, συμβάντα και αναλύσεις σε πραγματικό χρόνο
- Chronograf: Περιβάλλον εργασίας χρήστη για παρακολούθηση και απεικόνιση του TICK stack
- Kapacitor: Πλαίσιο εργασίας (Framework) για την επεξεργασία, την παρακολούθηση και την ειδοποίηση σχετικά με δεδομένα χρονοσειρών




*Εικόνα 10. Αρχιτεκτονική στοίβας TICK*

Ωστόσο στην επίσημη ιστοσελίδα τονίζεται και η δυνατότητα χρήσης του influxdb ως αποθηκευτικού μέσου του συστήματος (backend storage) με τη χρήση άλλων εργαλείων εποπτείας με τα οποία έχει επίσημη υποστήριξη.

## 3.3 Kafka

### 3.3.1 Το Kafka σαν εργαλείο

Το Apache Kafka είναι μία πλατφόρμα ανοικτού κώδικα που ανήκει στο Apache Software Foundation, που επικεντρώνεται στην επεξεργασία και μετάδοση ροής δεδομένων [21]. Το έργο ξεκίνησε το 2011 από τη LinkedIn, την εταιρεία πίσω από το κοινωνικό δίκτυο για επαγγελματίες και στόχος της ήταν να αναπτυχθεί μια ουρά μηνυμάτων. Από την πρώτη δωρεάν (license-free) διανομή (Apache 2.0), οι δυνατότητες αυτού του λογισμικού έχουν επεκταθεί σε μεγάλο βαθμό, μετατρέποντας την από μια απλή εφαρμογή ουράς αναμονής σε μια ισχυρή και κατανεμημένη πλατφόρμα streaming με ένα ευρύ φάσμα λειτουργιών. Χρησιμοποιείται από γνωστές εταιρείες όπως η Netflix, η Microsoft και η Airbnb. Μια πλατφόρμα ροής έχει τρεις βασικές δυνατότητες:

- Δημοσίευση και εγγραφή (Publish and subscribe) σε ροές εγγραφών, παρόμοια με μια ουρά μηνυμάτων ή ένα σύστημα ανταλλαγής μηνυμάτων επιχείρησης.
- Αποθήκευση κάθε ροής εγγραφών με ανοχή σε σφάλματα (fault-tolerant).
- Επεξεργασία σε ροές εγγραφών καθώς συμβαίνουν.

Το Kafka χρησιμοποιείται γενικά για δύο μεγάλες κατηγορίες εφαρμογών:

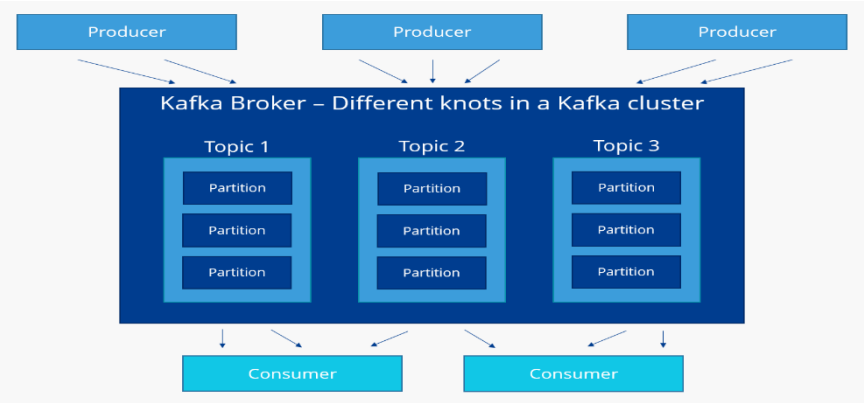
- Δημιουργία αγωγών ροής δεδομένων σε πραγματικό χρόνο, για την ανταλλαγή δεδομένων με αξιοπιστία μεταξύ συστημάτων ή εφαρμογών.
- Δημιουργία εφαρμογών ροής σε πραγματικό χρόνο που μετασχηματίζουν ή αντιδρούν στις ροές δεδομένων.

Το Apache Kafka έχει σχεδιαστεί κυρίως για τη βελτιστοποίηση της μετάδοσης και επεξεργασίας των ροών δεδομένων που μεταφέρονται μέσω απευθείας σύνδεσης μεταξύ του δέκτη δεδομένων και της πηγής δεδομένων. Το Kafka ενεργεί ως σημείο αναφοράς ανταλλαγής μηνυμάτων μεταξύ του αποστολέα και του δέκτη, παρέχοντας λύσεις στις κοινές προκλήσεις που αντιμετωπίζει ο συγκεκριμένος τύπος σύνδεσης. Για παράδειγμα, η πλατφόρμα Apache παρέχει λύση στην αδυναμία αποθήκευσης δεδομένων ή μηνυμάτων σε προσωρινή μνήμη όταν ο δέκτης δεν είναι διαθέσιμος (π.χ. λόγω προβλημάτων δικτύου). Επιπλέον, μια σωστά ρυθμισμένη ουρά Kafka αποτρέπει την υπερφόρτωση του δέκτη από τον αποστολέα. Αυτό συμβαίνει πάντα όταν η πληροφορία αποστέλλεται γρηγορότερα από ό,τι μπορεί να ληφθεί και να υποβληθεί σε επεξεργασία κατά τη διάρκεια μίας απευθείας σύνδεσης. Τέλος, το λογισμικό Kafka είναι ιδανικό για καταστάσεις στις οποίες το στοχευόμενο σύστημα λαμβάνει το μήνυμα, αλλά καταρρέει κατά τη διάρκεια της επεξεργασίας. Ενώ ο αποστολέας κανονικά υποθέτει ότι η επεξεργασία έχει συμβεί παρά το πρόβλημα που συνέβη, το Apache Kafka αναφέρει την αποτυχία στον αποστολέα.

Σε αντίθεση με τις καθαρές υπηρεσίες ερωτημάτων/μηνυμάτων όπως οι βάσεις δεδομένων, το Apache Kafka είναι ανθεκτικό σε σφάλματα. Αυτό σημαίνει ότι το λογισμικό ικανοποιεί τις απαιτήσεις για τη συνέχιση της επεξεργασίας των μηνυμάτων και των δεδομένων. Το Apache Kafka, σε συνδυασμό με την υψηλή επεκτασιμότητα και την ικανότητά του να διανέμει μεταδιδόμενες πληροφορίες σε οποιοδήποτε σύστημα (διανεμημένο αρχείο καταγραφής συναλλαγών), αποτελεί εξαιρετική λύση για όλες τις υπηρεσίες που πρέπει να διασφαλίζουν ότι τα δεδομένα αποθηκεύονται και επεξεργάζονται γρήγορα διατηρώντας παράλληλα υψηλή διαθεσιμότητα.

### 3.3.2 Επισκόπηση αρχιτεκτονικής Apache kafka

Το Apache Kafka λειτουργεί ως σύμπλεγμα (cluster) σε έναν ή περισσότερους διακομιστές που μπορούν να ευρίσκονται σε πολλαπλά κέντρα δεδομένων. Οι μεμονωμένοι διακομιστές στο σύμπλεγμα, γνωστοί ως brokers, αποθηκεύουν και κατηγοριοποιούν τις εισερχόμενες ροές δεδομένων σε θέματα (topics). Τα δεδομένα χωρίζονται σε διαμερίσματα (partitions), αναπαράγονται και διανέμονται μέσα στο cluster και τους αποδίδεται μία χρονική σφραγίδα (timestamp). Ως αποτέλεσμα, η πλατφόρμα ροής εξασφαλίζει υψηλή διαθεσιμότητα (availability) και γρήγορο χρόνο πρόσβασης ανάγνωσης [20].

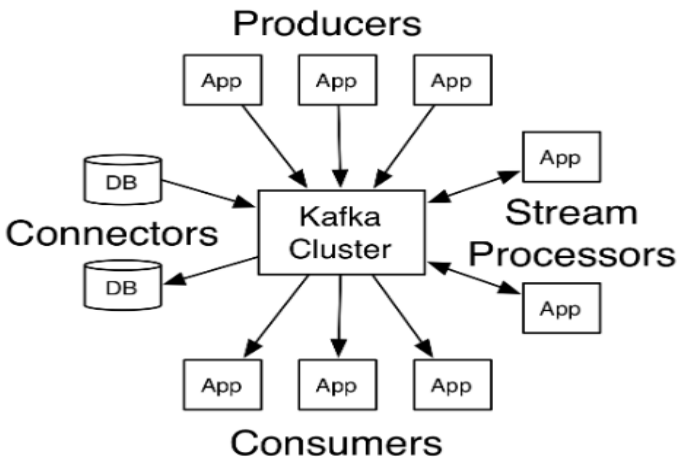



*Εικόνα 11. Κατανομή θεμάτων σε broker και διανομή κάθε partition σε καταναλωτές*

Οι εφαρμογές που γράφουν δεδομένα σε ένα Kafka cluster ονομάζονται παραγωγοί (producers), ενώ οι εφαρμογές που διαβάζουν δεδομένα από ένα Kafka cluster ονομάζονται καταναλωτές (consumers). Το κεντρικό στοιχείο που χρησιμοποιούν οι παραγωγοί και οι καταναλωτές κατά την επεξεργασία των ροών δεδομένων είναι μια βιβλιοθήκη Java που ονομάζεται Kafka Streams. Υποστηρίζοντας συναλλακτική γραφή (transactional writing), εξασφαλίζεται ότι τα μηνύματα παραδίδονται μόνο μία φορά (χωρίς διπλότυπα). Το Kafka έχει τέσσερα βασικά API (Εικόνα 12):

- Το Producer API επιτρέπει σε μια εφαρμογή να δημοσιεύει μια ροή αρχείων σε ένα ή περισσότερα topics του Kafka.

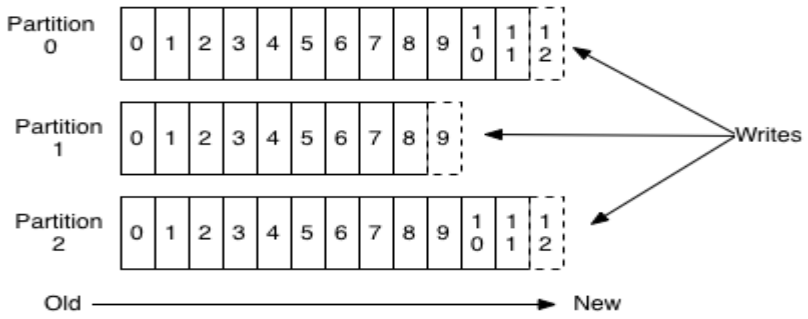
- Το Consumer API επιτρέπει σε μια εφαρμογή να εγγράφει σε ένα ή περισσότερα θέματα και να επεξεργάζεται τη ροή των αρχείων που παράγονται σε αυτά.
- Το Streams API επιτρέπει σε μια εφαρμογή να ενεργεί ως επεξεργαστής ροής, καταναλώνοντας μια ροή εισόδου από ένα ή περισσότερα topics και δημιουργώντας μια ροή εξόδου σε ένα ή περισσότερα topics, μετατρέποντας αποτελεσματικά τις ροές εισόδου σε ροές εξόδου.
- Το Connector API επιτρέπει την κατασκευή και λειτουργία επαναχρησιμοποιήσιμων παραγωγών ή καταναλωτών που συνδέουν τα Kafka topics με υπάρχουσες εφαρμογές ή συστήματα δεδομένων. Για παράδειγμα, ένας σύνδεσμος σε μια σχεσιακή βάση δεδομένων μπορεί να καταγράψει κάθε αλλαγή σε έναν πίνακα.




*Εικόνα 12. Kafka APIs*

### 3.3.3 Topics και Logs

Ένα θέμα (topic) είναι μια κατηγορία ή μια ονομασία ροής στην οποία δημοσιεύονται οι εγγραφές. Τα topics στο Kafka είναι πάντα πολυ-συνδρομητικά, δηλαδή, ένα topic μπορεί να έχει μηδέν, έναν ή πολλούς καταναλωτές που εγγράφονται στα δεδομένα αυτού. Για κάθε topic, το Kafka cluster διατηρεί ένα τεμαχισμένο σε διαμερίσματα αρχείο καταγραφής που μοιάζει με αυτό της Εικόνας 13. Τα διαμερίσματα στο αρχείο καταγραφής εξυπηρετούν διάφορους σκοπούς. Επιτρέπουν στο αρχείο καταγραφής να υπερβαίνει το μέγεθος που θα χωρέσει σε ένα μόνο διακομιστή. Κάθε μεμονωμένο διαμέρισμα πρέπει να ταιριάζει στους διακομιστές που το φιλοξενούν, αλλά ένα topic μπορεί να έχει πολλά διαμερίσματα ώστε να μπορεί να χειριστεί ένα αυθαίρετο μέγεθος δεδομένων.

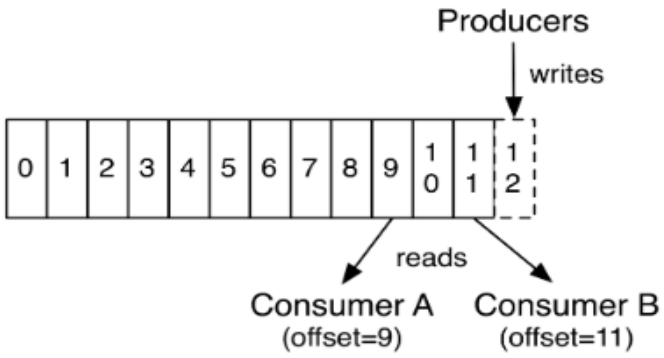



*Εικόνα 13. Εγγραφή/Ανάγνωση σε partitions των topics*

Κάθε διαμέρισμα είναι μια ταξινομημένη, αμετάβλητη ακολουθία αρχείων που προσαρτάται συνεχώς σε ένα δομημένο αρχείο καταγραφής. Οι εγγραφές στα διαμερίσματα έχουν κάθεμια ένα αναγνωριστικό αριθμό που ονομάζεται μετατόπιση (offset) και προσδιορίζει με μοναδικό τρόπο κάθε εγγραφή μέσα στο διαμέρισμα.

Το Kafka cluster διατηρεί όλες τις δημοσιευμένες εγγραφές - ανεξάρτητα από το αν έχουν καταναλωθεί ή όχι - σύμφωνα με μια προσδιοριζόμενη περίοδο διατήρησης. Για παράδειγμα, αν η πολιτική διατήρησης έχει οριστεί σε δύο ημέρες, τότε για δύο ημέρες μετά τη δημοσίευση η εγγραφή θα είναι διαθέσιμη για κατανάλωση και μετά θα απορριφθεί για να ελευθερωθεί χώρος. Η απόδοση του Kafka είναι ουσιαστικά σταθερή σε σχέση με το μέγεθος των δεδομένων, οπότε η αποθήκευση δεδομένων για μεγάλο χρονικό διάστημα δεν αποτελεί πρόβλημα.

Στην πραγματικότητα, τα μόνα μεταδεδομένα που διατηρούνται ανά καταναλωτή είναι το offset ή η θέση του εν λόγω καταναλωτή στο log. Το offset ελέγχεται από τον καταναλωτή: υπό κανονικές συνθήκες ο καταναλωτής θα αυξήσει το offset του γραμμικά, καθώς διαβάζει τα αρχεία, αλλά στην πραγματικότητα, εφόσον η θέση ελέγχεται από τον καταναλωτή, μπορεί να καταναλώνει αρχεία με οποιαδήποτε σειρά επιθυμεί. Για παράδειγμα, ένας καταναλωτής μπορεί να ανατρέξει σε παλαιότερο offset για να επανεπεξεργαστεί δεδομένα από το παρελθόν ή να προχωρήσει στην πιο πρόσφατη εγγραφή και να αρχίσει να καταναλώνει τα τωρινά δεδομένα.




*Εικόνα 14. Παράδειγμα χρήσης offset κατά την ανάγνωση από δύο διαφορετικούς καταναλωτές*

Αυτός ο συνδυασμός χαρακτηριστικών σημαίνει ότι οι καταναλωτές του Kafka δεν έχουν κάποια ουσιαστική επίδραση στο σύστημα - μπορούν να εισέλθουν και να αποχωρούν χωρίς μεγάλη επίδραση στο cluster ή σε άλλους καταναλωτές. Για παράδειγμα, μπορούν να χρησιμοποιηθούν τα εργαλεία της γραμμής εντολών για να διαβαστούν τα περιεχόμενα οποιουδήποτε topic χωρίς να αλλάξει τι καταναλώνεται από τους υπάρχοντες καταναλωτές.

### 3.3.4 Διανομή

Τα διαμερίσματα του αρχείου καταγραφής διανέμονται στους διακομιστές του Kafka cluster με κάθε διακομιστή να χειρίζεται δεδομένα και αιτήματα για ένα μέρος των κατατμήσεων. Κάθε διαμέρισμα αντιγράφεται σε έναν διαμορφωμένο αριθμό διακομιστών για ανοχή σφάλματος. Κάθε κατάτμηση έχει έναν εξυπηρετητή που ενεργεί ως "leader" και οι υπόλοιποι εξυπηρετητές που λειτουργούν ως "followers". Ο leader χειρίζεται όλα τα αιτήματα ανάγνωσης και εγγραφής για το διαμέρισμα, ενώ οι followers λειτουργούν παθητικά με τον εκάστοτε leader. Εάν ο επιλεγμένος leader αποτύχει, ένας από τους followers θα γίνει αυτομάτως ο νέος leader. Κάθε διακομιστής λειτουργεί ως leader για μερικά από τα διαμερίσματά του και follower για άλλους, ώστε το φορτίο να είναι καλά ισορροπημένο μέσα στο cluster.

### 3.3.5 Γεωγραφική Αναπαραγωγή

Το Kafka MirrorMaker είναι ένα εργαλείο που παρέχει υποστήριξη γεωγραφικής αναπαραγωγής για τα clusters. Με το MirrorMaker, τα μηνύματα αναπαράγονται σε πολλαπλά κέντρα δεδομένων ή Cloud Regions. Ο χρήστης μπορεί να το χρησιμοποιήσει σε ενεργά / παθητικά σενάρια για δημιουργία αντιγράφων ασφαλείας και ανάκτηση τους, ή σε ενεργά / παθητικά σενάρια για να τοποθετήσει δεδομένα πιο κοντά στους πελάτες ή για να υποστηρίξει τις απαιτήσεις της τοποθεσίας δεδομένων.

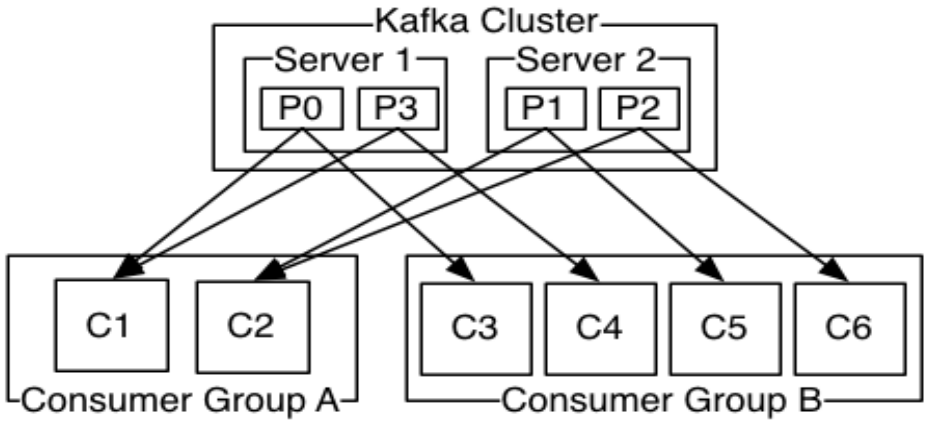
### 3.3.6 Παραγωγοί

Οι παραγωγοί (producers) δημοσιεύουν δεδομένα στα θέματα της επιλογής τους. Ο παραγωγός είναι υπεύθυνος για την επιλογή του αρχείου που θα αντιστοιχίσει σε κάποιο

διαμέρισμα και θέμα. Αυτό μπορεί να γίνει με μέθοδο round-robin για να εξισορροπηθεί το φορτίο ή να γίνει σύμφωνα με κάποια συνάρτηση σημασιολογικού διαμερισμού (για παράδειγμα με βάση κάποιο κλειδί στο αρχείο).

### 3.3.7 Καταναλωτές

Οι καταναλωτές (consumers) επισημαίνουν τον εαυτό τους με ένα όνομα ομάδας καταναλωτών και κάθε εγγραφή (record) που δημοσιεύεται σε ένα θέμα παραδίδεται σε υποσύνολο καταναλωτών εντός κάθε ομάδας καταναλωτών. Οι καταναλωτές μπορούν να βρίσκονται σε ξεχωριστές διαδικασίες ή ακόμα και σε ξεχωριστές μηχανές. Εάν όλα τα μέλη έχουν την ίδια ομάδα καταναλωτών, τότε τα αρχεία θα είναι αποτελεσματικά και ισόρροπα διαμοιρασμένα σε σχέση με τους καταναλωτές. Εάν όλα τα consumer instances έχουν διαφορετικές ομάδες καταναλωτών, τότε κάθε αρχείο θα μεταδοθεί σε όλες τις διεργασίες των καταναλωτών (consumer processes).




*Εικόνα 15. Παράδειγμα ομάδων καταναλωτών σε ένα kafka cluster*

Συνηθέστερα, ωστόσο, τα θέματα έχουν μικρό αριθμό ομάδων καταναλωτών, δηλαδή μία για κάθε λογικό συνδρομητή. Κάθε ομάδα αποτελείται από πολλούς καταναλωτές για επεκτασιμότητα και ανοχή σφάλματος. Ο τρόπος με τον οποίο εφαρμόζεται η κατανάλωση στο Kafka είναι η διαίρεση των κατατμήσεων στο αρχείο καταγραφής, έτσι ώστε κάθε ένας καταναλωτής να είναι ο αποκλειστικός καταναλωτής ενός δίκαιου μεριδίου διαμερισμάτων ανά πάσα στιγμή. Αυτή η διαδικασία διατήρησης της ιδιότητας μέλους της ομάδας αντιμετωπίζεται δυναμικά από το πρωτόκολλο Kafka. Εάν νέοι καταναλωτές συμμετέχουν στην ομάδα, θα αναλάβουν ορισμένα διαμερίσματα από άλλα μέλη της ομάδας και αν ένα instance τερματιστεί, τα κατατμήματά του θα διανεμηθούν στα υπόλοιπα.

Το Kafka παρέχει μόνο μία συνολική διάταξη για τα αρχεία μέσα σε ένα διαμερίσμα και όχι ανάμεσα σε διαφορετικά διαμερίσματα σε ένα topic. Η διάταξη ανά διαμέρισμα σε συνδυασμό με τη δυνατότητα κατανομής δεδομένων ανά κλειδί είναι αρκετή για τις

περισσότερες εφαρμογές. Ωστόσο, αν χρειάζεται συνολική διάταξη (ordering) για τα αρχεία, αυτό μπορεί να επιτευχθεί με ένα topic που έχει μόνο ένα διαμέρισμα, αν και αυτό θα σημαίνει μόνο μία διεργασία ανά ομάδα καταναλωτών.

### 3.3.8 Εγγυήσεις

Σε υψηλό επίπεδο (high-level), το Kafka δίνει τις ακόλουθες εγγυήσεις:

- Τα μηνύματα που αποστέλλονται από έναν παραγωγό σε ένα συγκεκριμένο διαμέρισμα ενός θέματος (topic partition) θα επισυνάπτονται με τη σειρά που αποστέλλονται. Δηλαδή, εάν από τον ίδιο παραγωγό η εγγραφή M1 αποστέλλεται πριν από την εγγραφή M2 , τότε η M1 θα έχει χαμηλότερη μετατόπιση (offset) από τη M2 και θα εμφανίζεται νωρίτερα στο log.
- Ένα καταναλωτής βλέπει τα αρχεία με τη σειρά που αποθηκεύονται στο log.
- Για ένα topic με συντελεστή αναπαραγωγής N (replication factor), θα παρέχεται ανοχή έως και N-1 αποτυχίες διακομιστή χωρίς να χαθεί κανένα αρχείο που δεσμεύεται στο αρχείο καταγραφής.

### 3.3.9 Χρήση ως σύστημα μετάδοσης μηνυμάτων

Η διαδικασία παράδοσης μηνυμάτων έχει παραδοσιακά δύο μοντέλα: queuing (ουρά) και publish-subscribe (δημοσίευση-εγγραφή). Σε μια ουρά, μια ομάδα καταναλωτών μπορεί να διαβάσει από ένα διακομιστή και κάθε εγγραφή πηγαίνει σε μία από αυτές. Στη δημοσίευση-εγγραφή, το αρχείο μεταδίδεται σε όλους τους καταναλωτές. Κάθε ένα από αυτά τα μοντέλα έχει πλεονεκτήματα και μειονεκτήματα. Η ουρά επιτρέπει τη διαίρεση της επεξεργασίας των δεδομένων σε πολλαπλούς καταναλωτές, γεγονός που διευκολύνει την κλιμάκωση της επεξεργασία. Δυστυχώς, οι ουρές δεν υποστηρίζουν πολλαπλούς συνδρομητές (multi-subscribers), άρα μόλις μια διαδικασία διαβάσει τα δεδομένα αυτά παύουν να υπάρχουν. Η δημοσίευση-εγγραφή επιτρέπει τη μετάδοση δεδομένων σε πολλαπλές διεργασίες, αλλά δεν υπάρχει τρόπος κλιμάκωσης, αφού κάθε μήνυμα μεταφέρεται σε κάθε συνδρομητή.

Η έννοια της ομάδας των καταναλωτών στο Kafka γενικεύει αυτές τις δύο έννοιες. Όπως και με μια ουρά, η ομάδα καταναλωτών επιτρέπει τη διαίρεση της επεξεργασία μέσω μιας συλλογής διαδικασιών (τα μέλη της ομάδας καταναλωτών). Όπως και με τη δημοσίευση-εγγραφή, το Kafka επιτρέπει τη μετάδοση μηνυμάτων σε πολλαπλές ομάδες καταναλωτών. Το πλεονέκτημα του μοντέλου του Kafka είναι ότι κάθε topic έχει και αυτές τις ιδιότητες - μπορεί να πραγματοποιήσει κλιμάκωση και είναι επίσης πολυ-συνδρομητής - και δεν χρειάζεται να γίνεται επιλογή στο ένα ή στο άλλο.

Το Kafka προσφέρει ισχυρότερες εγγυήσεις σε σχέση με ένα παραδοσιακό σύστημα ανταλλαγής μηνυμάτων. Μια παραδοσιακή ουρά διατηρεί τα κατά σειρά (in-order) αρχεία

στον εξυπηρετητή και εάν πολλαπλοί καταναλωτές καταναλώνουν από την ουρά τότε ο διακομιστής εκχωρεί αρχεία με τη σειρά που αυτά αποθηκεύονται. Ωστόσο, παρόλο που ο διακομιστής παραδίδει τα αρχεία με τη σειρά, τα αρχεία διανέμονται ασύγχρονα στους καταναλωτές, έτσι να μπορούν να φτάσουν εκτός σειράς σε διαφορετικούς καταναλωτές.

## 3.4 Grafana

### 3.4.1 Λειτουργία Grafana

Μία σύγχρονη διεπαφή χρήστη είναι απαραίτητο να παρουσιάζει την εκάστοτε ζητούμενη πληροφορία με εύληπτο και πρακτικό τρόπο στον τελικό χρήστη, ώστε να είναι σε θέση να κατανοεί πλήρως την τρέχουσα κατάσταση ενός συστήματος και να εξάγει συμπεράσματα. Η ενσωμάτωση γραφημάτων για οπτική απεικόνιση δεδομένων επιτρέπει τον άμεσο εντοπισμό προβλημάτων, διευκολύνει την ιστορική ανάλυση των μετρήσεων και την ανακάλυψη μοτίβων. Επομένως απλοποιείται σημαντικά η διαχείριση του συστήματος στόχου (στην περίπτωση μας ενός κέντρου δεδομένων), η μελέτη της συμπεριφοράς συγκεκριμένων μετρικών και η διαδικασία πρόβλεψης μελλοντικής συμπεριφοράς κάθε στοιχείου.

Το Grafana είναι μια πλατφόρμα ανοιχτού κώδικα για οπτικοποίηση δεδομένων χρονοσειρών γραμμένη σε Go και Javascript για την ανάλυση δεδομένων μέσω πολυάριθμων λειτουργιών όπως μαθηματικές συναρτήσεις, φίλτρα, συνάθροιση κλπ, την εξαγωγή συγκεκριμένων μετρήσεων από τεράστιες ποσότητες δεδομένων και την παρακολούθηση εφαρμογών με τη βοήθεια λεπτομερών και διαδραστικών γραφημάτων μέσα από εύκολα προσαρμοζόμενα ταμπλό (dashboards)[19]. Ένα από τα κεντρικά σημεία του Grafana είναι η ανακατασκευή των γραφημάτων στην πλευρά του πελάτη, γεγονός που μειώνει τον υπολογιστικό φόρτο του εξυπηρετητή. Αυτό επιτυγχάνεται ως εξής: Ο εξυπηρετητής της Grafana απαντά στην αρχική αίτηση του πελάτη με το βασικό HTML σκελετό-κομμάτι του γραφήματος και το υπόλοιπο περιεχόμενο φορτώνεται δυναμικά μέσω Javascript. Ακόμα υποστηρίζεται ο ορισμός χρηστών και ομάδων καθώς και η ανάθεση ρόλων και δικαιωμάτων σε κάθε μία από αυτές ώστε η πρόσβαση στα ταμπλό να είναι πλήρως ελεγχόμενη. Επιπρόσθετα έχει σε μεγάλο βαθμό συμβατότητα με σχεδόν όλα τα backend περιβάλλοντα αποθήκευσης χρονοσειρών, όπως Graphite, Prometheus, InfluxDB, ElasticSearch, MySQL, PostgreSQL κλπ[17].Το Grafana επιτρέπει επίσης τη δημιουργία plugins για ενσωμάτωση με πολλές διαφορετικές πηγές δεδομένων.

### 3.4.2 Χαρακτηριστικά του Grafana

Το εργαλείο Grafana διαθέτει πληθώρα χαρακτηριστικών μερικά εκ των οποίων είναι τα ακόλουθα:

- Απεικόνιση (Visualize): Διατίθεται μεγάλη ποικιλία σε γρήγορα και ευέλικτα γραφήματα προς τη πλευρά του πελάτη (client-side) με πολλές επιλογές. Παρέχει (panel plugins) για πολλούς διαφορετικούς τρόπους απεικόνισης των μετρήσεων και των αρχείων καταγραφής(logs).
- Ειδοποίηση (Alerting): Οπτικός καθορισμός κανόνων ειδοποιήσεων για τις σημαντικότερες μετρήσεις. Το Grafana είναι σε θέση να αξιολογεί συνεχώς τις μετρήσεις και μπορεί να αποστέλλει ειδοποιήσεις ανάλογα με τους κανόνες που θέτει ο χρήστης μέσω ηλεκτρονικού ταχυδρομείου ή μέσω των εφαρμογών Slack, PagerDuty, VictorOps, OpsGenie, webhook κλπ.
- Δυναμικοί πίνακες ελέγχου (Dashboards): Ο χρήστης έχει τη δυνατότητα να κατασκευάσει δυναμικούς και επαναχρησιμοποιήσιμους πίνακες ελέγχου με προτυποποιημένες μεταβλητές που εμφανίζονται σε ένα dropdown menu στο πάνω μέρος του πίνακα ελέγχου.
- Μικτές πηγές δεδομένων (Mixed Data Sources): Επιτρέπεται η εισαγωγή από διαφορετικές πηγές δεδομένων στο ίδιο γράφημα, με την προϋπόθεση ότι καθορίζεται μία πηγή δεδομένων σε μία βάση ανά query. Αυτό λειτουργεί ακόμα και για προσαρμοσμένες πηγές δεδομένων.
- Σχολιασμοί (Annotations): Ο χρήστης μπορεί να σχολιάσει γραφήματα με πληθώρα συμβάντων από διαφορετικές πηγές δεδομένων. Με τον δείκτη του ποντικιού πάνω στα συμβάντα εμφανίζονται τα πλήρη μετα-δεδομένα (metadata) και οι πλήρεις ετικέτες του συμβάντος.
- Ad-hoc φίλτρα (ad-hoc Filters): Τα ad-hoc φίλτρα επιτρέπουν τη δημιουργία νέων φίλτρων ζεύγους κλειδιού/τιμής (key/value), τα οποία εφαρμόζονται αυτόματα σε όλα τα ερωτήματα που χρησιμοποιούν αυτή την πηγή δεδομένων.
- Εξέταση μετρήσεων (Explore Metrics): Το Grafana επιτρέπει την εξέταση των μετρήσεων μέσω ερωτημάτων ad-hoc και δυναμική ανίχνευση (dynamic drilldown). Ο χρήστης μπορεί να διαιρέσει το παράθυρο σε επιμέρους και να συγκρίνει διαφορετικά χρονικά διαστήματα, ερωτήματα και πηγές δεδομένων σε παράθεση. Το ίδιο μπορεί να εφαρμοστεί και σε αρχεία καταγραφής ( Logs).

# 4 Ανάλυση Υλοποίησης

## 4.1 Είδη δεδομένων

### 4.1.1 Περιβαλλοντικοί παράγοντες

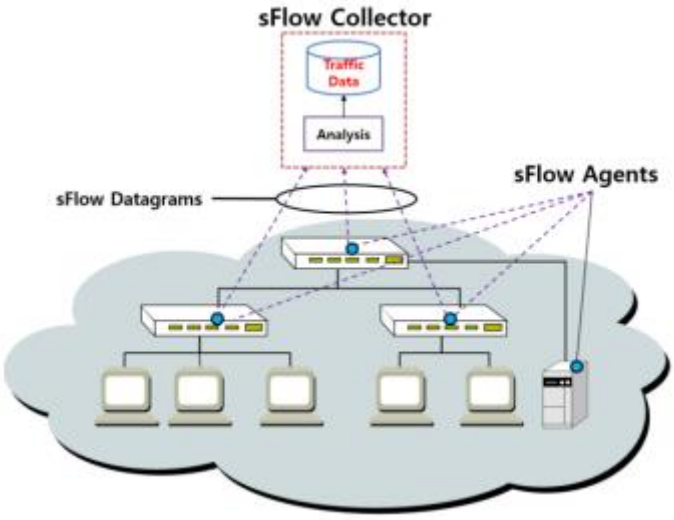
Ο περιβαλλοντικός έλεγχος του κέντρου δεδομένων για τη διατήρηση της θερμοκρασίας, της υγρασίας και άλλων φυσικών ιδιοτήτων του αέρα σε συγκεκριμένα επίπεδα, είναι ζωτικής σημασίας προκειμένου ο εξοπλισμός που στεγάζεται σε ένα κέντρο δεδομένων να είναι λειτουργικός κατά τη διάρκεια της χρήσης του. Ένα μεγάλο μέρος του προϋπολογισμού των κέντρων δεδομένων αφορά την ενέργεια που καταναλώνεται στη λειτουργία του εξοπλισμού και στο σύστημα ψύξης [22]. Προηγούμενες μελέτες το 2003 [23] και το 2005 [24] αποκάλυψαν ότι ορισμένα κέντρα δεδομένων ξοδεύουν περισσότερο από το 50% του συνολικού προϋπολογισμού ισχύος τους για ψύξη στις εγκαταστάσεις τους. Οι διαχειριστές έφτασαν στο συμπέρασμα ότι η ψύξη αποτελεί σημαντικό παράγοντα ενεργειακής κατανάλωσης και κατά συνέπεια η βιομηχανία έχει καταβάλει τεράστιες προσπάθειες για τον περιορισμό αυτού. Η Google ανακοίνωσε το 2012 ότι το νέο datacenter της στη Φινλανδία καταναλώνει μόνο περίπου το 11% του συνολικού προϋπολογισμού του για την ψύξη, το κέντρο δεδομένων της Yahoo ανακοίνωσε 7% το 2011 και το κέντρο δεδομένων Facebook το 2013 ανέφερε 8%. Με άλλα λόγια, για αυτά τα πρόσφατα κατασκευασμένα μεγάλης κλίμακας κέντρα, η ισχύς ψύξης δεν αποτελεί πλέον ένα σημαντικό πρόβλημα, σε αντίθεση με τα κέντρα μικρής και μεσαίας κλίμακας για τα οποία αποτελεί ένα μείζον ζήτημα, όπως είναι ο χώρος του εργαστηρίου Δικτύου Υπολογιστών στο οποίο πρόκειται να γίνει η μελέτη. Για το λόγο αυτό έγινε συλλογή δεδομένων θερμοκρασίας και υγρασίας από κατάλληλους αισθητήρες από το εγκατεστημένο APC Netbotz με τη χρήση SNMP.

### 4.1.2 Network Traffic

Η παρακολούθηση της κυκλοφορίας εντός ενός δικτύου είναι ένα σημαντικό μέρος της διαχείρισης δικτύων. Έχει χρησιμοποιηθεί ευρέως και σε άλλους τομείς όπως στην παρακολούθηση προσπαθειών εισβολής και ανάλυση πρωτοκόλλων. Επιπλέον, η παρακολούθηση της δικτυακής κίνησης μπορεί να αποκαλύψει τις ευπάθειες του λογισμικού του συστήματος, τα σφάλματα των χρηστών και άλλα ζητήματα ασφάλειας για την προστασία του δικτύου. Εκτός αυτού, η συλλογή των χαρακτηριστικών κυκλοφορίας επιτρέπει την αξιολόγηση της υπάρχουσας υποδομής ασφάλειας και την εφαρμογή άλλων μεθόδων και αλγόριθμων ασφαλείας ανώτατου επιπέδου [25]. Οι τεχνολογίες εποπτείας που χρησιμοποιήθηκαν στην υπάρχουσα υλοποίηση είναι το SNMP, το Sflow και το Netflow.

*4.1.2.1 Sflow*

Το sFlow, συντομογραφία του "sampled flow", είναι ένα βιομηχανικό πρότυπο για την εξαγωγή πακέτων στο στρώμα (Layer) 2 του μοντέλου OSI και είναι ενσωματωμένο σε δρομολογητές και μεταγωγείς. Παρέχει ένα μέσο για την εξαγωγή περικομμένων πακέτων, μαζί με μετρητές σε όλες τις διεπαφές ταυτόχρονα με σκοπό την παρακολούθηση του δικτύου [26]. Ο sflow πράκτορας (agent) είναι ένα λογισμικό που τρέχει ως τμήμα του λογισμικού διαχείρισης του δικτύου σε μία συσκευή όπως φαίνεται στην Εικόνα 16 [31]. Ο πράκτορας sFlow συνδυάζει μετρητές από τις διεπαφές και δείγματα ροών σε sFlow δεδομενογράμματα (datagrams), τα οποία αποστέλλονται μέσω του δικτύου σε έναν συλλέκτη sFlow. Η δειγματοληψία πακέτων (Packet sampling) εκτελείται συνήθως από τον μεταγωγέα / δρομολογητή με ASICs. Επίσης καταγράφεται η κατάσταση των καταχωρημένων συνδέσεων στον πίνακα προώθησης / δρομολόγησης που συσχετίζεται με κάθε δειγματοληπτημένο πακέτο.




*Εικόνα 16. Παράδειγμα αρχιτεκτονικής sFlow*

Χρησιμοποιώντας το sFlow, μπορούν να συλλεχθούν δείγματα κυκλοφορίας (traffic samples) από μια ευρεία γκάμα συσκευών δικτύου, όπως ανοιχτούς εικονικούς μεταγωγείς (OVS), φυσικούς μεταγωγείς, κεντρικούς υπολογιστές, κ.λπ.. Το sFlow μπορεί να ρυθμιστεί σε όλες τις διεπαφές μιας συσκευής με μικρή επιβάρυνση και ο ρυθμός δειγματοληψίας για κάθε σύνδεση μπορεί να αποφασιστεί σύμφωνα με την πολιτική παρακολούθησης.

Οι πράκτορες sFlow σε συσκευές δικτύου χρησιμοποιούν τυχαία δειγματοληψία σύμφωνα με τον καθορισμένο ρυθμό δειγματοληψίας και συνεπώς, μπορούν να χρησιμοποιηθούν για την παρακολούθηση δικτύων υψηλής ταχύτητας (ταχύτητες Gbps και υψηλότερες) με ποσοτικοποιήσιμη ακρίβεια. Τα δεδομένα δειγματοληψίας αποστέλλονται ως UDP πακέτα στον καθορισμένο κεντρικό υπολογιστή και στη θύρα όπου είναι ενεργοποιημένος ο συλλέκτης sFlow.

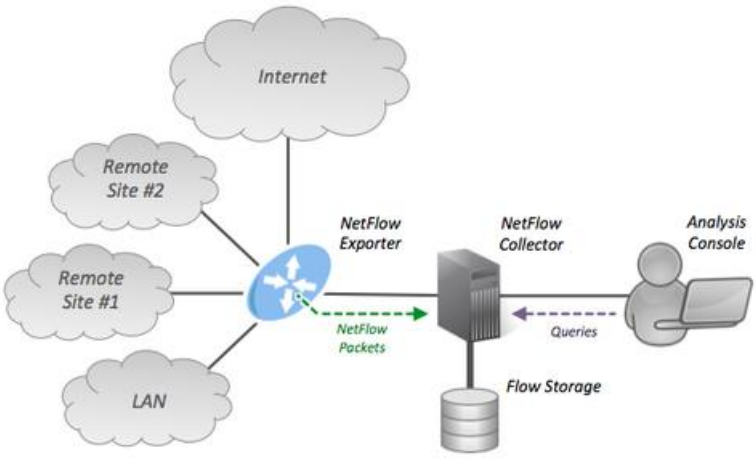
Συνοψίζοντας, οι πράκτορες sFlow που βρίσκονται σε όλο το δίκτυο στέλνουν συνεχώς ροές δεδομένων sFlow σε έναν κεντρικό συλλέκτη sFlow όπου αναλύονται από μια μηχανή ανάλυσης για την παραγωγή διαγραμμάτων κυκλοφοριακών ροών σε πραγματικό χρόνο σε όλο το δίκτυο.

Ένα ευρέως χρησιμοποιούμενο εργαλείο για την επεξεργασία πακέτων sFlow που ελήφθησαν από τις συσκευές δικτύου είναι το sFlow-RT. Δίνει τη δυνατότητα απεικόνισης στοιχείων σε δίκτυα καθορισμένα από λογισμικό (Software Defined Networks-SDN) σε πραγματικό χρόνο. Το sFlow-RT βρίσκεται στο επίπεδο ελέγχου της στοίβας SDN. Μετατρέπει τα ληφθέντα δεδομενογράμματα σε μετρήσεις ή σε συγκεντρωτικά στατιστικά στοιχεία βασισμένα στις ροές όπως ορίζονται από τον χρήστη. Μια ροή κίνησης είναι ένα σύνολο πακέτων με κοινή ιδιότητα, γνωστή ως κλειδί ροής, που παρατηρείται σε μια χρονική περίοδο. Το κλειδί ροής καθορίζεται από πεδία της επικεφαλίδας πακέτου, όπως τη διεύθυνση προέλευσης και προορισμού IP και τους αριθμούς θύρας TCP / UDP. Τα ονόματα ροών χρησιμοποιούνται συνήθως ως μετρήσεις που είναι προγραμματιζόμενες μέσω του RESTful Northbound APIs. Οποιαδήποτε γλώσσα υποστηρίζει μηνύματα αίτησης HTTP (Perl, Python, Java, Java script, bash κ.α.) μπορεί να χρησιμοποιηθεί για την ανάκτηση μετρήσεων από το sFlow-RT. Τα στατιστικά στοιχεία sFlow-RT μπορούν να ανακτηθούν σε μορφή JSON. Τα JSON κωδικοποιημένα αποτελέσματα βάσει κειμένου είναι ευανάγνωστα και υποστηρίζονται ευρέως από εργαλεία προγραμματισμού.

Για την υλοποίηση της εφαρμογής έγινε χρήση του sFlow-RT με κατάλληλη ενσωμάτωσή του στο Prometheus για τη συλλογή των δεδομένων από την θύρα 5600 που ήταν ενεργοποιημένη για τη συλλογή των datagrams.

#### *4.1.2.2 Netflow*

Το Netflow είναι ένα πρωτόκολλο που αναπτύχθηκε από τη Cisco και χρησιμοποιείται για τη συλλογή και καταγραφή της δικτυακής κίνησης, που εισέρχεται σε ή εξέρχεται από μία διεπαφή (interface) ενός δρομολογητή όπου το πρωτόκολλο είναι ενεργοποιημένο. Στην Εικόνα 17 φαίνεται η αρχιτεκτονική του Netflow. Η συλλογή της κίνησης γίνεται με τη μορφή ροών. Μια ροή ορίζεται ως η ακολουθία πακέτων μιας κατεύθυνσης, με κοινά χαρακτηριστικά, που διέρχονται μέσω μιας συσκευής ενός δικτύου. Το πλεονέκτημα που παρέχεται έτσι, δηλαδή, η καταγραφή της κίνησης σε ροές και όχι πακέτα, δίνει τη δυνατότητα αποθήκευσης μεγαλύτερου όγκου πληροφορίας στο ίδιο χρονικό διάστημα, με μεγαλύτερη ευκολία και μικρότερες απαιτήσεις σε αποθηκευτικό χώρο.




*Εικόνα 17. Παράδειγμα αρχιτεκτονικής Netflow*

Υπάρχουν διάφορες εκδόσεις του πρωτοκόλλου Netflow. Οι σημαντικότερες ωστόσο είναι οι 5 και 9, με την τελευταία να είναι η πιο διαδεδομένη στις σύγχρονες συσκευές. Μια τυπική ρύθμιση παρακολούθησης ροής (χρησιμοποιώντας το NetFlow) αποτελείται από τρία βασικά στοιχεία [30]:

- **Εξαγωγέας ροής (flow exporter):** Το εργαλείο που συσσωρεύει πακέτα σε ροές και στέλνει τις καταγραφές ροών προς έναν ή περισσότερους συλλέκτες ροής.
- **Συλλέκτης ροής (flow collector):** Το εργαλείο που είναι υπεύθυνο για τη λήψη, την αποθήκευση και την προεπεξεργασία των δεδομένων ροής που λαμβάνονται από έναν εξαγωγέα ροής.
- **Εφαρμογή ανάλυσης (analysis application):** Οι αναλύσεις λαμβάνουν δεδομένα ροής στο πλαίσιο της ανίχνευσης εισβολών ή της μορφοποίησης της κυκλοφορίας.

Η δομή ενός πακέτου Netflow περιέχει δύο στοιχεία: την επικεφαλίδα και τις εγγραφές ροών. Συγκεκριμένα, η επικεφαλίδα περιέχει πληροφορίες σχετικά με την έκδοση του πρωτοκόλλου, τον αριθμό σειράς του πακέτου, τον χρόνο εξαγωγής του πακέτου όπως αυτός προκύπτει από τον δρομολογητή και τον αριθμό των ροών που ακολουθούν. Από την άλλη πλευρά η εγγραφή για ροές περιέχει τη διεπαφή εισόδου όπως αυτή αναγράφεται στο SNMP, τη διεπαφή εξόδου ή τον αριθμό μηδέν, αν το πακέτο απορρίφθηκε, τις χρονικές στιγμές εκκίνησης και λήξης της ροής, τον αριθμό των πακέτων και των bytes που παρατηρήθηκαν στην αντίστοιχη ροή, τη διεύθυνση πηγής, προορισμού και την τιμή TOS (Type of service), πακέτων IP, τον τύπο για πακέτα ICMP, τις θύρες πηγής και προορισμού των πρωτοκόλλων TCP, UDP, SCTP. Σε περίπτωση ροών TCP περιέχονται πληροφορίες για τις σημαίες (flags) των πακέτων TCP. Δίνεται επίσης η δυνατότητα παροχής πληροφορίας σχετικά με το

αυτόνομο σύστημα από το οποίο προήλθε το πακέτο είτε το άμεσα γειτονικό, είτε το αυτόνομο σύστημα από το οποίο ξεκίνησε το πακέτο.[27]

Στην εφαρμογή της παρούσας διπλωματικής χρησιμοποιήθηκαν κατάλληλοι exporters του Prometheus, η λειτουργία των οποίων θα αναλυθεί διεξοδικά στις επόμενες ενότητες, με τους οποίους έγινε συλλογή της δικτυακής κίνησης στο χώρο του Εργαστηρίου.

### 4.1.3 Μετρήσεις κατανάλωσης ενέργειας

Ένα ολοκληρωμένο σύστημα εποπτείας, πέρα από όσα αναφέρθηκαν στις προηγούμενες ενότητες, πρέπει να παρέχει στο χρήστη πλήρη εικόνα της τάσης, του ρεύματος και της καταναλισκόμενης ισχύος σε κάθε συσκευή που είναι εγκατεστημένη. Πιο συγκεκριμένα ο χρήστης πρέπει να γνωρίζει ανά πάσα στιγμή την εκάστοτε πραγματικού χρόνου τιμή των παραγόντων αυτών προκειμένου να προβεί σε απαραίτητες ενέργειες καθώς και να έχει πρόσβαση σε προηγούμενες μετρήσεις ώστε να είναι σε θέση να βρει την πραγματική αιτία μη ορθής λειτουργίας και να σχεδιάσει κατάλληλους τρόπους προς αποφυγή επανάληψης της. Επιπροσθέτως αυτές οι μετρήσεις είναι ιδιαίτερα σημαντικές για τη δημιουργία μοντέλων πρόβλεψης της ενεργειακής κατανάλωσης, σχεδίαση μοντέλων ελαχιστοποίησής της και συσχέτισή της με άλλους παράγοντες οι οποίοι μπορεί να επηρεάζουν. Για το λόγο αυτό έγινε εγκατάσταση smart-plugs (TP-LINK HS110) σε συγκεκριμένα μηχανήματα στο χώρο του Εργαστηρίου για καταγραφή αυτών των δεδομένων. Παρότι το HS100 χρησιμοποιείται σε συνδυασμό με εφαρμογή cloud της TP-Link, είναι εφικτή η εξαγωγή των μετρήσεων τάσης, ρεύματος και ισχύος με εντολές JSON σύμφωνα με την ανάλυση reverse engineering_της SoftCheck GmbH [32]. Με τη βοήθεια ενός exporter [33] που βασίζεται σε αυτήν, οι μετρήσεις αυτές ανακτώνται ανά τακτά χρονικά διαστήματα, που ορίζονται από το χρήστη, και γίνονται διαθέσιμες στο Prometheus.

## 4.2 Εγκατάσταση και ρύθμιση παραμέτρων

Αρχικά έγινε εγκατάσταση του Prometheus ως υπηρεσία [34] στο εικονικό μηχάνημα (VM) με διεύθυνση IP 147.102.7.56. Έγινε χρήση της έκδοσης 2.11.2 με τη βοήθεια των εντολών wget για λήψη και tar για αποσυμπίεση:

```
$ wget https://github.com/prometheus/prometheus/releases/download/v2.11.2/prometheus-2.11.2.linux-amd64.tar.gz
$ tar xvzf prometheus-2.11.2.linux-amd64.tar.gz
```

Τα κυριότερα αρχεία για τη λειτουργία του είναι το prometheus.yml, που είναι το αρχείο ρυθμίσεων του διακομιστή Prometheus, το promtool, που είναι το εργαλείο που επαληθεύει την ορθή ρύθμιση του διακομιστή, και το Prometheus, που είναι το μεταγλωττισμένο αρχείο με το οποίο θα τρέξει η υπηρεσία Prometheus στο σύστημα. Αρχικά για λόγους ασφάλειας έγινε δημιουργία ενός χρήστη Prometheus

```
$ sudo useradd -rs /bin/false prometheus
```

Στη συνέχεια δόθηκαν οι ακόλουθες εντολές για τη δημιουργία κατάλληλων φακέλων και απόδοση δικαιωμάτων στο χρήστη Prometheus

```
$ cd Prometheus/prometheus-2.11.2.linux-amd64/
$ sudo cp prometheus promtool /usr/local/bin
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
$ sudo mkdir /etc/prometheus
$ sudo cp -R consoles/ console_libraries/ prometheus.yml /etc/prometheus
$ sudo mkdir -p data/prometheus
$ sudo chown -R prometheus:prometheus data/prometheus /etc/prometheus/*
```

Στο φάκελο /lib/systemd/system δημιουργήσαμε ένα νέο αρχείο prometheus.service

```
$ cd /lib/systemd/system
$ sudo touch prometheus.service
```

και έπειτα γράφτηκε ο ακόλουθος κώδικας στο prometheus.service

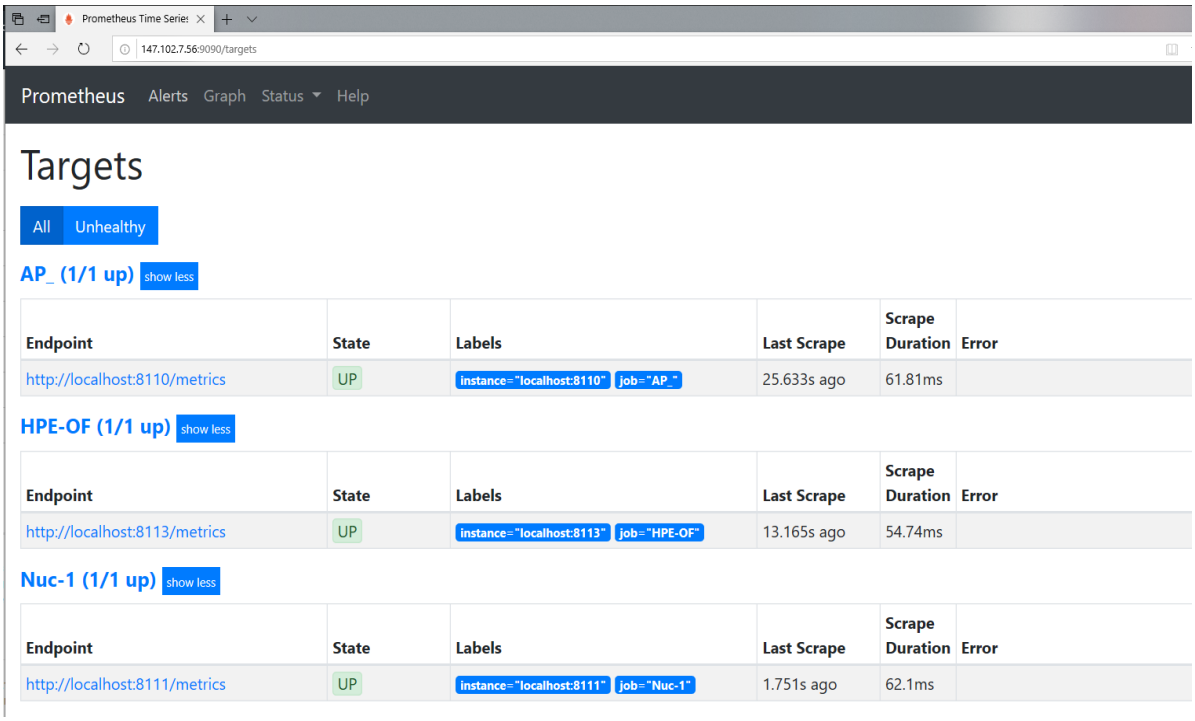
```
$ sudo nano prometheus.service
```

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=prometheus
Group=prometheus
ExecStart=/usr/local/bin/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path="/data/prometheus" \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries \
--web.listen-address=0.0.0.0:9090 \
--web.enable-admin-api

Restart=always

[Install]
WantedBy=multi-user.target
```

Τέλος με τις δύο ακόλουθες εντολές έγινε ενεργοποίηση κατά την έναρξη του VM και ξεκίνημα του Prometheus

```
$ sudo systemctl enable prometheus
$ sudo systemctl start prometheus
```

Ο διακομιστής Prometheus τρέχει και είναι διαθέσιμος όπως φαίνεται στην Εικόνα 18 στο URL http://147.102.7.56:9090.




*Εικόνα 18. Prometheus GUI*

Το αρχείο διάρθρωσης Prometheus.yml του Prometheus έχει ένα στατικό κομμάτι και ένα κομμάτι που αφορά τα smartplugs το οποίο παράγεται δυναμικά. Πιο συγκεκριμένα εκτελώντας το discovery.py ανακαλύπτουμε τη διεύθυνση IP και το alias κάθε smartplug. Τα στοιχεία αυτά εγγράφονται με συγκεκριμένο τρόπο στο Prometheus.yml για να γίνει η ανάγνωση τους, όπως φαίνεται παρακάτω:

```
#Smartplug_IP: <device_ip>
 - job_name: '<device_name>'
 static_configs:
 - targets: ['localhost:<port>']
```

Το στατικό μέρος που αφορά τους εξαγωγείς, την επικοινωνία με τη βάση δεδομένων και τη μεταφορά δεδομένων σε άλλες υποδομές μέσω Kafka αναλύεται πιο κάτω στις αντίστοιχες ενότητες. Το τελικό αρχείο διάρθρωσης του Prometheus (prometheus.yml) είναι διαθέσιμο στο Παράρτημα με τον κώδικα.

Όσον αφορά τους εξαγωγείς χρησιμοποιήθηκαν τέσσερα διαφορετικά είδη που σχετίζονται με snmp, netflow, sflow και smatplug, όπου ο τελευταίος είναι μια υλοποίηση για εξαγωγή δεδομένων από το TP-Link HS110.

Το snmp exporter αποτελεί επίσημη υλοποίηση του Prometheus [37] και έγινε κατάλληλη διαμόρφωση έτσι ώστε να τρέχει ως υπηρεσία στο VM. Για τη λειτουργία του απαιτείται το snmp.yml, που είναι το αρχείο διάρθρωσης και παράγεται με τη βοήθεια ενός generator [38]. Αρχικά έγινε εγκατάσταση των MIBs που ήταν απαραίτητες για τη συλλογή των δεδομένων θερμοκρασίας, υγρασίας, κατανάλωσης ισχύος, διαθέσιμης μνήμης, υπολογιστικής ισχύος και δημιουργία βάσει αυτών ενός αρχείου generator.yml, το οποίο

διαβάζει ο generator για να παράγει το κατάλληλο snmp.yml. Ακολουθούν οι σχετικές εντολές:

```
$ export MIBDIRS=mibs
$ ./generator generate
```

Στη συγκεκριμένη υλοποίηση πήραμε δεδομένα ισχύος, μνήμης, υπολογιστικής ισχύος από τους DELL εξυπηρετητές dragon (ports 9992 9990), aether (port 9995), orestes (ports 9996 και 9991), trillium (port 9993), pegasus (port 9994) και θερμοκρασίας και υγρασίας από αισθητήρες APC Netbotz (port 9116). Το αρχείο snmp.yml (διαθέσιμο στο Παράρτημα) διαβάζεται από το εκτελέσιμο snmp_exporter προκειμένου να γνωρίζει ποιες μετρήσεις πρέπει να συλλέξει και σε ποια πόρτα για το Prometheus.

```
$ snmp_exporter --config.file=snmp.yml --web.listen-address=":<port>"
```

Προκειμένου η συλλογή στοιχείων ανά συσκευή να λειτουργεί ως υπηρεσία (Service) δόθηκαν τα απαραίτητα δικαιώματα στην εφαρμογή μέσω της εντολής:

```
$ chmod +x snmp_exporter
```

και εν συνεχεία για κάθε επιτηρούμενη συσκευή δόθηκαν οι παρακάτω εντολές

```
$ cd /etc/systemd/system
$ sudo vi snmp_<name>.service
```

όπου <name> το όνομα της εκάστοτε συσκευής, π.χ dragon, aether, κλπ και snmp_<name>.service το αντίστοιχο αρχείο διάθρωσης ως εξής:

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-
address=":<port>"
[Install]
WantedBy=multi-user.target
```

Κατόπιν εκτελέστηκαν οι επόμενες εντολές για να ενεργοποιηθούν οι υπηρεσίες και να ξεκινάνε κατά την εκκίνηση (boot).

```
$ sudo systemctl enable snmp_<name>
$ sudo systemctl start snmp_<name>
```

Για να δούμε αν όντως λειτουργεί ορθά χρησιμοποιούμε την επόμενη εντολή η οποία μας δίνει την κατάσταση της εκάστοτε υπηρεσίας όπως φαίνεται στην Εικόνα 19.

```
$ sudo systemctl status snmp_<name>
```

```
● snmp_dragon1.service
   Loaded: loaded (/etc/systemd/system/snmp_dragon1.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-10-15 13:46:19 EEST; 2h 48min ago
 Main PID: 6651 (snmp_exporter)
    Tasks: 20 (limit: 4915)
   CGroup: /system.slice/snmp_dragon1.service
           └─6651 /home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter --config.file=/home/dimitris/snmp_expo

Oct 15 13:46:19 powermonitor systemd[1]: Started snmp_dragon1.service.
Oct 15 13:46:19 powermonitor snmp_exporter[6651]: time="2019-10-15T13:46:19+03:00" level=info msg="Starting snmp expor
Oct 15 13:46:19 powermonitor snmp_exporter[6651]: time="2019-10-15T13:46:19+03:00" level=info msg="Build context (go=g
Oct 15 13:46:19 powermonitor snmp_exporter[6651]: time="2019-10-15T13:46:19+03:00" level=info msg="Listening on :9990"
~
```

*Εικόνα 19. Κατάσταση λειτουργίας υπηρεσίας*

Προκειμένου να μπορεί το Prometheus να λαμβάνει δεδομένα SNMP συμπληρώθηκαν οι ακόλουθες γραμμές στο Prometheus.yml:

```
metrics_path: /snmp
 params:
 module: [<module_name>]
 relabel_configs:
 - source_labels: [__address__]
 target_label: __param_target
 - source_labels: [__param_terget]
 target_label: instane
 - target_label: __address__
 replacement: localhost:<port>
 - job_name: '<job_name'
 static_configs:
 - targets:
 - <ip>
```

Αντίστοιχη διαδικασία ακολουθήθηκε για τους exporter που χρειάζονται για τη συλλογή στοιχείων από τις υπόλοιπες συσκευές, όπως περιγράφεται στη συνέχεια. Τα σχετικά αρχεία διάρθρωσης *.service είναι διαθέσιμα στο Παράρτημα. Το neflow-exporter [40] επιτρέπει τη συλλογή δεδομένων netflow στη θύρα 2300, και πιο συγκεκριμένα τα octetDeltaCount και packetDeltaCount που είναι το μέγεθος και το πλήθος πακέτων αντίστοιχα. Τα δεδομένα αφορούν ένα δρομολογητή Cisco με IP διεύθυνση 147.102.40.203 και ένα OVS smartnic με IP διεύθυνση 147.102.40.78. Αυτά μπορεί ο χρήστης να τα δει στο URL http://147.102.7.56:9191/metrics όπως φαίνεται παρακάτω στην Εικόνα 20, αφού πρώτα έχει ξεκινήσει το εκτελέσιμο του exporter.

```
netflow_14710240203_TemplateID256_octetDeltaCount{From="147.102.40.203",NetflowVersion="9",TemplateID="256",bgpDestinationAsNumber="0"
,bgpSourceAsNumber="0",classId="0",destinationIPv4Address="106.39.15.168",destinationIPv4PrefixLength="0",destinationTransportPort="50
111",egressInterface="13",flowDirection="0",ingressInterface="14",ipClassOfService="0",ipNextHopIPv4Address="147.102.40.200",protocolI
dentifier="6",samplerId="0",sourceIPv4Address="147.102.39.6",sourceIPv4PrefixLength="26",sourceTransportPort="22",tcpControlBits="27"}
2880 1577183991230
netflow_14710240203_TemplateID256_octetDeltaCount{From="147.102.40.203",NetflowVersion="9",TemplateID="256",bgpDestinationAsNumber="0"
,bgpSourceAsNumber="0",classId="0",destinationIPv4Address="109.74.75.55",destinationIPv4PrefixLength="0",destinationTransportPort="386
80",egressInterface="13",flowDirection="0",ingressInterface="14",ipClassOfService="0",ipNextHopIPv4Address="147.102.40.200",protocolId
entifier="6",samplerId="0",sourceIPv4Address="147.102.39.6",sourceIPv4PrefixLength="26",sourceTransportPort="22",tcpControlBits="27"}
2880 1577183996230
netflow_14710240203_TemplateID256_octetDeltaCount{From="147.102.40.203",NetflowVersion="9",TemplateID="256",bgpDestinationAsNumber="0"
,bgpSourceAsNumber="0",classId="0",destinationIPv4Address="112.3.27.68",destinationIPv4PrefixLength="0",destinationTransportPort="80",
egressInterface="13",flowDirection="0",ingressInterface="14",ipClassOfService="128",ipNextHopIPv4Address="147.102.40.200",protocolIden
tifier="6",samplerId="0",sourceIPv4Address="147.102.39.62",sourceIPv4PrefixLength="26",sourceTransportPort="6203",tcpControlBits="4"}
40 1577183991230
netflow_14710240203_TemplateID256_octetDeltaCount{From="147.102.40.203",NetflowVersion="9",TemplateID="256",bgpDestinationAsNumber="0"
,bgpSourceAsNumber="0",classId="0",destinationIPv4Address="112.80.54.62",destinationIPv4PrefixLength="0",destinationTransportPort="420
70",egressInterface="13",flowDirection="0",ingressInterface="14",ipClassOfService="0",ipNextHopIPv4Address="147.102.40.200",protocolId
entifier="6",samplerId="0",sourceIPv4Address="147.102.39.6",sourceIPv4PrefixLength="26",sourceTransportPort="22",tcpControlBits="27"}
2976 1577184152237
netflow_14710240203_TemplateID256_octetDeltaCount{From="147.102.40.203",NetflowVersion="9",TemplateID="256",bgpDestinationAsNumber="0"
,bgpSourceAsNumber="0",classId="0",destinationIPv4Address="112.80.54.62",destinationIPv4PrefixLength="0",destinationTransportPort="454
10",egressInterface="13",flowDirection="0",ingressInterface="14",ipClassOfService="0",ipNextHopIPv4Address="147.102.40.200",protocolId
entifier="6",samplerId="0",sourceIPv4Address="147.102.39.6",sourceIPv4PrefixLength="26",sourceTransportPort="22",tcpControlBits="27"}
3248 1577183905226
```

*Εικόνα 20. Netflow δεδομένα*

Για το netflow, στο αρχείο Prometheus.yml προστέθηκαν οι ακόλουθες γραμμές:

```
- job_name: 'netflow'
static_configs:
- targets: ['localhost:9191']
metric_relabel_configs:
- regex: '<excluded_label>'
action: labeldrop
```

όπου οι 2 τελευταίες γραμμές αφορούν ετικέτες που δεν μας είναι χρήσιμες αλλά επιτρέπουν την πιο ομαλή λειτουργία του εξαγωγέα.

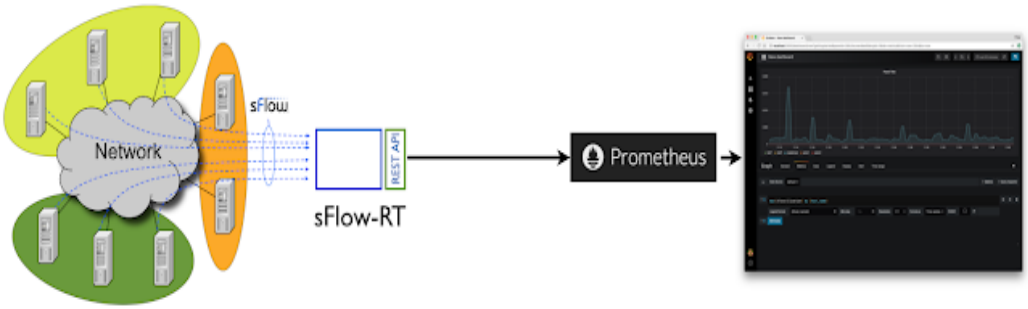
Αντίστοιχα το sflow-exporter [41] υλοποιήθηκε με τη βοήθεια του sflow-rt το οποίο μετατρέπει τις μετρήσεις σε κατάλληλη μορφή και τις εκθέτει σε ένα REST API για το Prometheus (Εικόνα 21). Για αυτό στο Prometheus.yml προστεθήκαν τα εξής:

```
- job_name: 'sflow-rt'
metrics_path: /app/prometheus/scripts/export.js/dump/ALL/ALL/txt
static_configs:
- targets: ['localhost:8008']
```

Εκτελώντας την εντολή

```
$ sflow-rt/start.sh -Dsflow.port=5600
```

Συλλέγονται τα δεδομένα που σχετίζονται με το sFlow στη θύρα 5600 και είναι διαθέσιμα για ανάγνωση στο http://147.102.7.56:8008/app/prometheus/scripts/export.js/dump/ALL/ALL/txt.




*Εικόνα 21. Sflow-RT και Prometheus REST API*

Μερικά από αυτά φαίνονται στην Εικόνα 22. Το sflow-exporter χρησιμοποιήθηκε για δεδομένα που αφορούσαν ένα μεταγωγέα Juniper με IP διεύθυνση 147.102.7.207 και ένα OVS smartnic με IP διεύθυνση 147.102.40.78.

```
ifinucastpkts{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0.5
ifoutdiscards{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifoututilization{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifinerrors{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
eth_fcserrors{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifspeed{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 10000000000
ifoutpkts{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifindiscards{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifouterrors{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifoutucastpkts{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifinutilization{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0.00000256
ifinoctets{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 32
ifinmulticastpkts{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifoutoctets{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifinpkts{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0.5
eth_alignmenterrors{agent="147.102.40.78",datasource="4",ifname="enp1s0np0"} 0
ifinucastpkts{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifoutdiscards{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifoututilization{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifinerrors{agent="147.102.40.78",datasource="8",ifname="br0"} 0
eth_fcserrors{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifspeed{agent="147.102.40.78",datasource="8",ifname="br0"} 100000000
ifoutpkts{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifindiscards{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifouterrors{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifoutucastpkts{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifinutilization{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifinoctets{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifinmulticastpkts{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ifoutoctets{agent="147.102.40.78",datasource="8",ifname="br0"} 0
eth_alignmenterrors{agent="147.102.40.78",datasource="8",ifname="br0"} 0
ovs_dp_missrate{agent="147.102.40.78",datasource="2.1000"} 16.666666666666664
```

*Εικόνα 22. Δεδομένα Sflow*

Τέλος έχουμε τα smartplug-exporters [33] από τα οποία έγινε συλλογή δεδομένων ρεύματος, τάσης και καταναλισκόμενης ισχύος. Τοποθετήθηκαν σε διάφορες συσκευές του εργαστηρίου και πιο συγκεκριμένα στους εξυπηρετητές Nuc-1, Alderaan, HPE-OF, Pegasus, Aether, Trillium και τον ανεμιστήρα του Rack. Στο Πίνακα 1 φαίνεται η περιγραφή των συσκευών με τα ονόματα τους. Στον Πίνακα 2 φαίνεται η αντιστοιχία των συσκευών με τις διευθύνσεις IP των smartplugs με τα οποία είναι συνδεδεμένες στο χώρο του εργαστηρίου καθώς και οι θύρες επικοινωνίας με τους εξαγωγείς από τους οποίους λαμβάνει τις μετρήσεις το Prometheus. Η εντολή για τη εξαγωγή δεδομένων είναι η εξής:

```
$ hs110-exporter.py -t <ip> -p <port. -f <time_sec>
```

| DEVICE_NAME | Περιγραφή |
|---|---|
| AP | Cisco Access Point |
| Nuc-1 | Intel NUC και SDR Ettus B-210 |
| Alderaan | Cisco router 2821 |
| HPE-OF | HP switch |

| | |
|---|---|
| Aether | Dell Server |
| Trillium | Dell Server |
| Pegasus | Dell Server |
| Dragon | Dell Server |
| Orestes | Dell Server |
| Rack-Fan | Cooling System |

*Πίνακας 1. Περιγραφή συσκευών*

| DEVICE_NAME | SMARTPLUG IP | EXPORTER PORT |
|---|---|---|
| AP | 147.102.40.102 | 8110 |
| Nuc-1 | 147.102.40.188 | 8111 |
| Alderaan | 147.102.39.231 | 8112 |
| HPE-OF | 147.102.39.230 | 8113 |
| Pegasus | 147.102.39.224 | 8114 |
| Aether | 147.102.39.227 | 8115 |
| Rack-Fan | 147.102.39.228 | 8116 |
| Trillium | 147.102.39.229 | 8117 |

*Πίνακας 2. Αντιστοιχία συσκευών με διευθύνσεις IP και θύρες*

Ακολούθησε η εγκατάσταση του InfluxDB 1.7.8 με βάση την επίσημη τεκμηρίωση ως εξής:

```
$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,}
$DISTRIB_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.lis
$ sudo apt-get update && sudo apt-get install influxdb
$sudo service influxdb start
```

Για τη χρήση του InfluxDB ως αποθηκευτικού χώρου του Prometheus έγινε η κατάλληλη ρύθμιση του prometheus.yml [34], [35] εισάγοντας τις παρακάτω γραμμές που ουσιαστικά δημιουργούν ένα Database στο influxDB με όνομα prometheus όπου το Prometheus μπορεί να γράψει και διαβάσει.

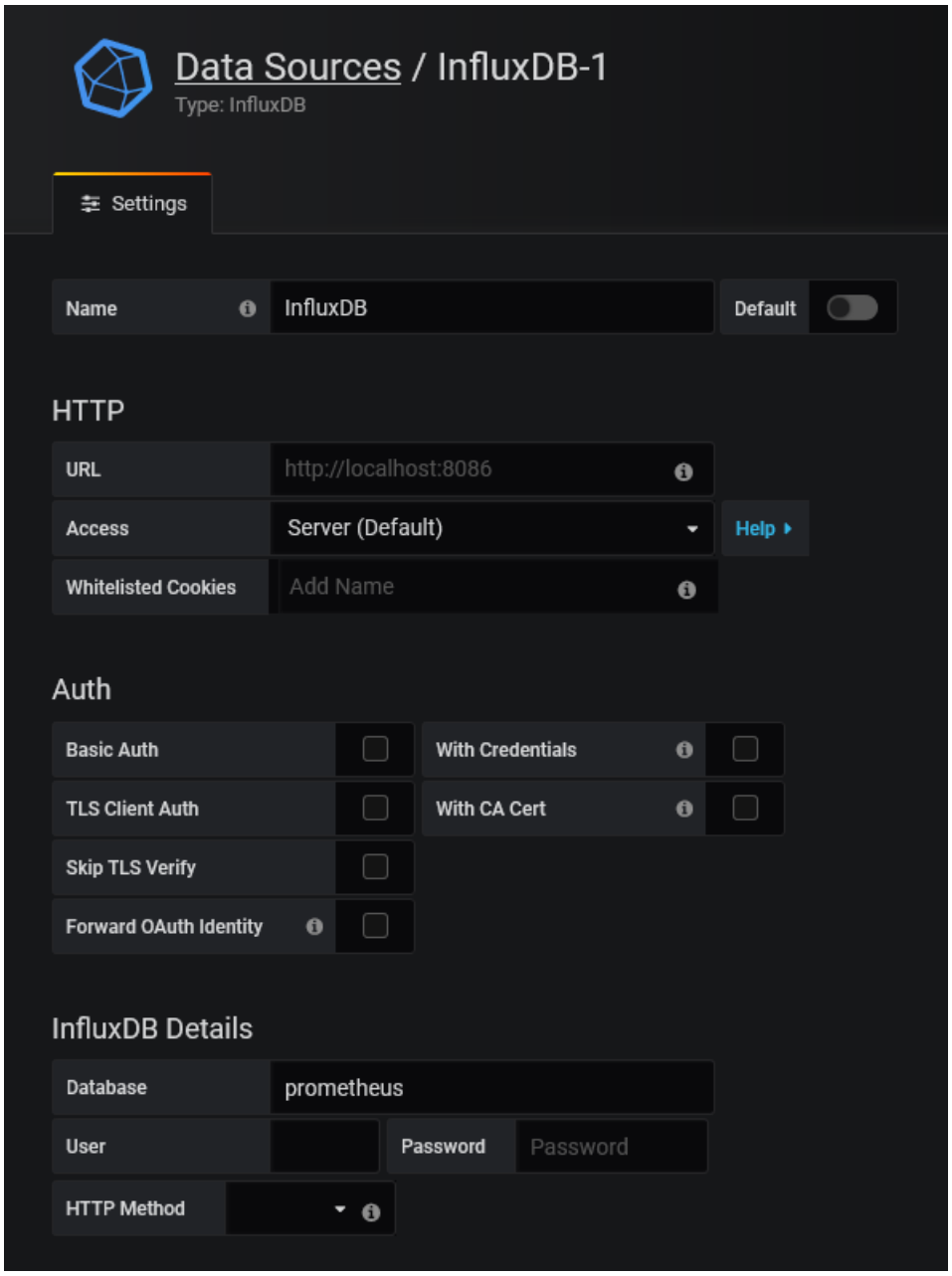
```
remote_write:
- url: "http://localhost:8086/api/v1/prom/write?db=prometheus"
remote_read:
- url: "http://localhost:8086/api/v1/prom/read?db=prometheus"
```

Η εγκατάσταση του Grafana όπως αναφέρεται στην επίσημη τεκμηρίωση έγινε όπως παρουσιάζεται παρακάτω.

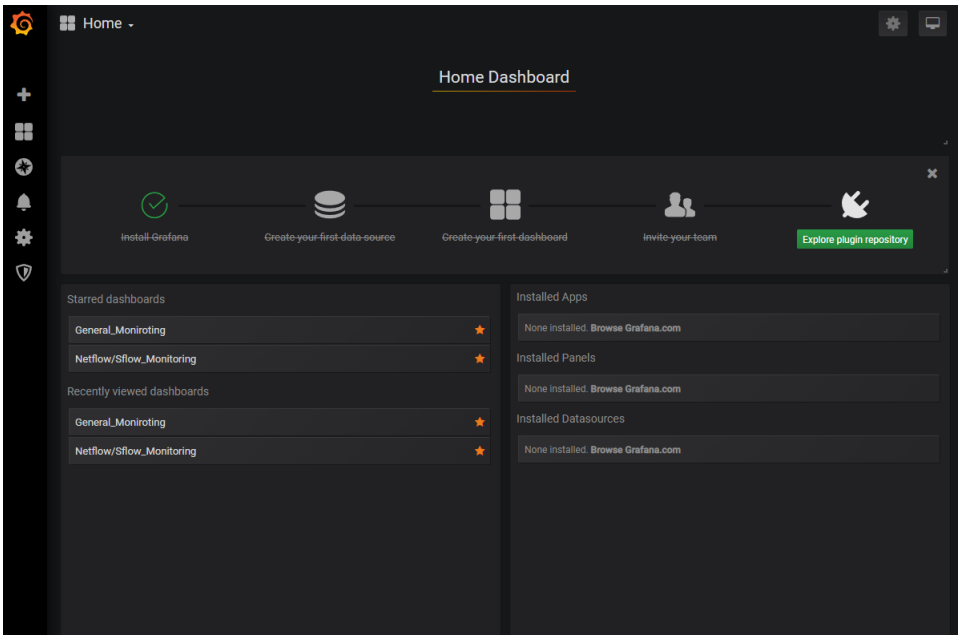
```
$ apt-get install -y software-properties-common
$ sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install grafana
```

$ sudo service grafana-server start

Για την οπτικοποίηση του της βάση δεδομένων “prometheus” του InfluxDB από το Grafana απαιτείται η επιλογή του InfluxDB ως Data Source και η δήλωση της επιθυμητής βάσης (Εικόνα 23). Έπειτα έγινε κατασκευή διαφορετικών καρτελών. Ειδικότερα έγιναν δύο όπου η μία αφορούσε τις περιβαλλοντικές συνθήκες και την κατανάλωση (general monitoring) και η άλλη αφορούσε τη δικτυακή κίνηση netflow και sflow (Netflow/sflow Monitoring) (Εικόνα 24).




*Εικόνα 23. Ρύθμιση Grafana για εισαγωγή δεδομένων από influxdb*




*Εικόνα 24. Grafana Dashboard*

Πρέπει να τονιστεί ότι ο δαίμονας της Grafana επικοινωνεί απευθείας μέσω ερωτημάτων HTTP με τον εξυπηρετητή της InfluxDB, κάνοντας ανάκτηση των πιο πρόσφατων δεδομένων μέσω του HTTP API της σε τακτά χρονικά διαστήματα. Έτσι δεν εμπλέκεται κάποιο ενδιάμεσο εργαλείο και δεν μεταφέρεται μεγάλος όγκος δεδομένων και, κατα συνέπεια, δεν προκαλείται υπερφόρτωση του δικτύου. Με αυτά τα δεδομένα δημιουργεί γραφήματα πραγματικού χρόνου, τα οποία ανανεώνονται αυτόματα καθώς εισέρχονται νέα δεδομένα, κάνοντας σκιαγράφηση της συμπεριφοράς διαφόρων δομών σε δεδομένα χρονικά παράθυρα. Ο δαίμονας έπειτα κατασκευάζει δυναμικά dashboards και panels για κάθε μέτρηση που επιθυμεί ο χρήστη. Ακόμα κάθε panel αντιστοιχίζεται με ένα μοναδικό URL, μέσω του οποίου επιτρέπεται η μελλοντική ενσωμάτωσή του και σε άλλες εφαρμογές.

Το Kafka σύμφωνα με τις οδηγίες της επίσημης ιστοσελίδας απαιτεί τη χρήση του Zookeeper Server πριν την έναρξη του Kafka Server όπως φαίνεται παρακάτω.

```
$ nohup bin/zookeeper-server-start.sh config/zookeeper.properties >/dev/null 2>&1 &
$ nohup bin/kafka-server-start.sh config/server.properties >/dev/null 2>&1 &
```

Όλα τα διαθέσιμα topics μπορούν να εμφανιστούν στη γραμμή εντολών με την εντολή

```
$ bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

και στην περίπτωση που κάποιος χρήστης θέλει να διαβάσει τι είναι γραμμένο σε ένα topic, π.χ. με το όνομα metrics, μπορεί να εκτελέσει την ακόλουθη εντολή:

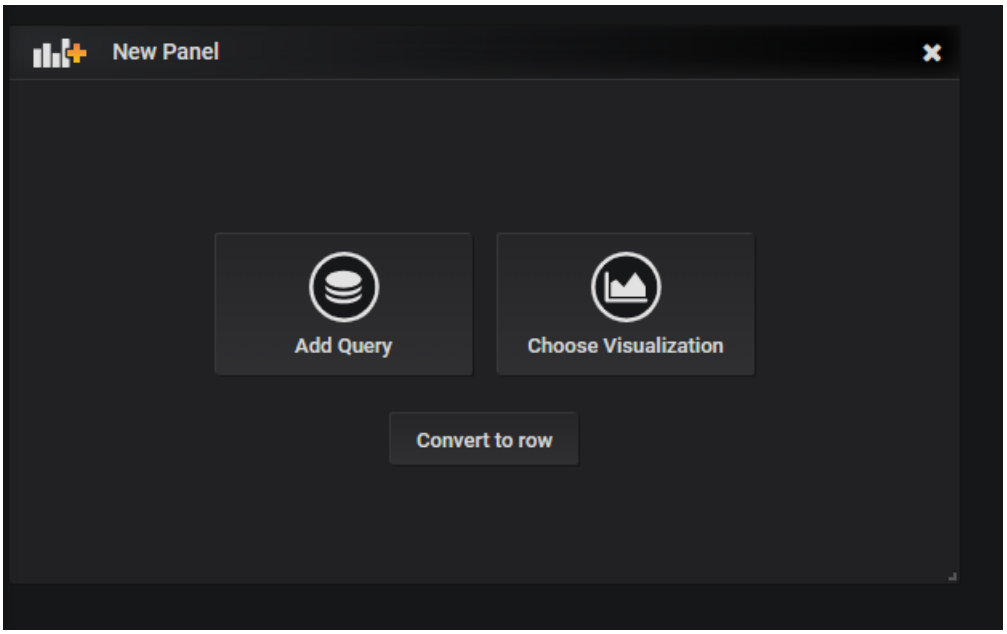
```
$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic metrics --from-beginning
```

Στην παρούσα εργασία χρησιμοποιήθηκε ένας prometheus-kafka-adapter [36] που γράφει τις μετρήσεις από το prometheus στο Kafka στο topic metrics. Πρέπει να τονιστεί στο σημείο αυτό ότι πρόκειται αποκλειστικά για εγγραφή από το Prometheus στο Kafka και όχι

για ανάγνωση. Το Kafka όπως περιγράφηκε είναι ένα κατανεμημένο σύστημα μεταφοράς μηνυμάτων που αξιοποιείται για τη συλλογή και την παράδοση μεγάλου όγκου δεδομένων με μικρές καθυστερήσεις (low latency), ανθεκτικότητα στα σφάλματα (Fault-Tolerant) και μεγάλο throughput χωρίς την ύπαρξη ιδιαίτερου εξοπλισμού. Το Kafka επιτρέπει την μεταφορά μετρήσεων σε πολλούς καταναλωτές (scalibality) σε μορφή JSON, ασύγχρονα, με ασφάλεια και με κατάλληλες εγγυήσεις. Έτσι πολλοί καταναλωτές μπορούν να διαβάζουν από την ίδια ροή δεδομένων σε διαφορετικές θέσεις και με διαφορετικές ταχύτητες ώστε να μην επιβαρύνεται η λειτουργία του InfluxDB ως μόνιμου αποθηκευτικού χώρου, καθώς η λειτουργία του καταναλώνει αρκετούς υπολογιστικούς πόρους και επιπλέον ερωτήματα ενδέχεται να οδηγήσουν ακόμα και σε διακοπή της λειτουργίας του. Επιπρόσθετα, οι διεπαφές προγραμματισμού εφαρμογών καταναλωτή και παραγωγού είναι με τέτοιο τρόπο διαμορφωμένες ώστε να επιτρέπουν την εύκολη ενσωμάτωση τους σε ήδη υπάρχουσες εφαρμογές, όπως είναι για παράδειγμα η στοίβα ELK, που είναι ο συνδυασμός των εφαρμογών Elasticsearch, Logstash και Kibana. Για την επίτευξη αυτού στο prometheus.yml προστέθηκε η ακόλουθη γραμμή:

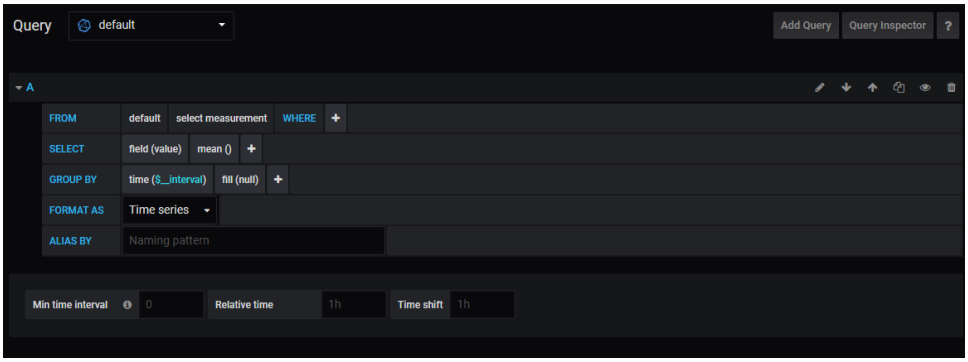
```
remote_write:
- url: "localhost:8080/receive"
```

Όπως αναφέρθηκε στις προηγούμενες παραγράφους χρησιμοποιήθηκαν εξαγωγείς για τη συλλογή δεδομένων. Όλα τα δεδομένα που συλλέχθηκαν είναι διαθέσιμα μέσω του Grafana κάνοντας login με τα κατάλληλα στοιχεία χρήστη-κωδικού (Εικόνα 27). Τα δεδομένα εμφανίζονται ως στοιχεία ταμπλό (dashboards) όπως φαίνεται στις Εικόνες 28, 29, 30, 31). Για τη διαδικασία κατασκευής των ταμπλό επιλέγουμε τι δεδομένα θέλουμε να προβάλλουμε και σε τι μορφή όπως φαίνεται στην Εικόνα 25.




*Εικόνα 25. Δημιουργία γραφημάτων*

Έπειτα στην επιλογή Add Query επιλέγουμε τα κατάλληλα φίλτρα ώστε να γίνει λήψη των δεδομένων προς απεικόνιση από την προεπιλεγμένη βάση δεδομένων που στην περίπτωση μας είναι η InfluxDB. Το Grafana επιτρέπει ως ένα βαθμό και επεξεργασία των μετρήσεων αυτών π.χ. υπολογισμός μέσου όρου κλπ (Εικόνα 26). Η επιλογή add visualization παρουσιάζει τους διάφορους τρόπους απεικόνισης των δεδομένων και παραμετροποίηση αυτών για να είναι πιο ευανάγνωστα από τον χρήστη.




*Εικόνα 26. Επιλογή μετρήσεων προς απεικόνιση*


*Εικόνα 27. Εισαγωγή στοιχείων χρήστη-κωδικού*

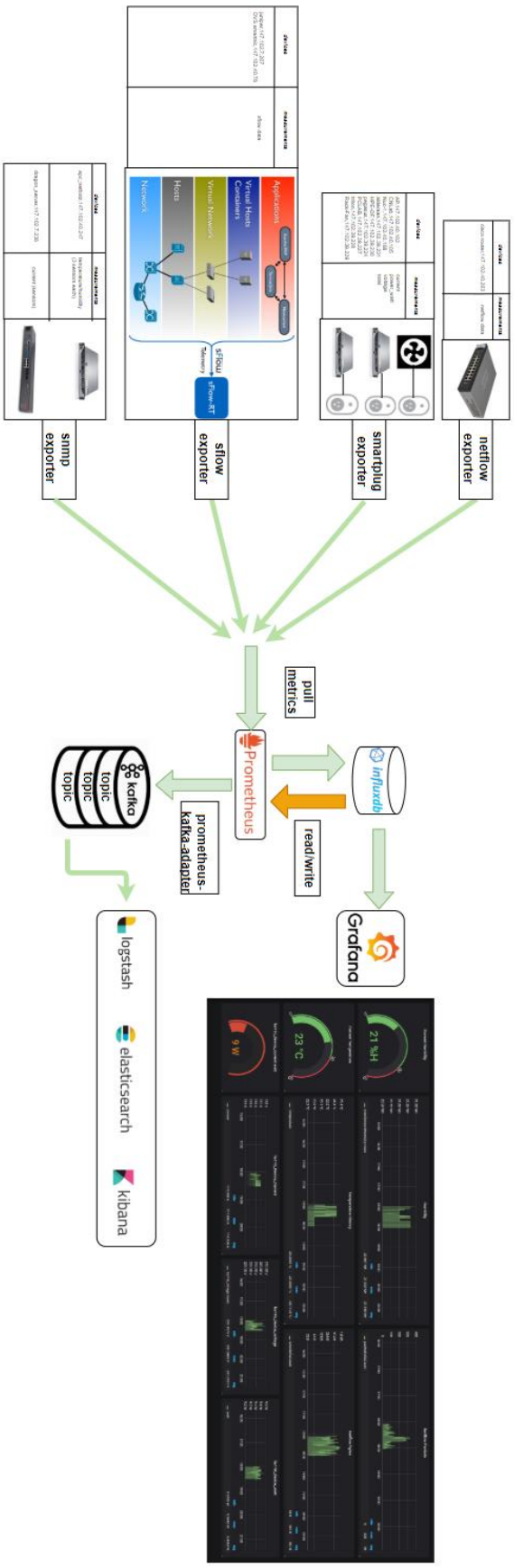

*Εικόνα 28. Environmental Variables*


*Εικόνα 29. Connected Smartplugs*


*Εικόνα 30. Sflow Monitoring*


*Εικόνα 31. Netflow Monitoring*

Η τελική εικόνα της αρχιτεκτονικής της υλοποίησης για την εποπτεία του εργαστηρίου Δικτύου υπολογιστών σύμφωνα με όσα αναφέρθηκαν φαίνεται στην Εικόνα 32.




*Εικόνα 32. Τελική αρχιτεκτονική δομή εφαρμογής*

# 5 Αξιολόγηση Υλοποίησης

## 5.1 Μοντέλο Ενεργειακής Βελτιστοποίησης

Τα δεδομένα που συλλέγονται από μια εφαρμογή παρακολούθησης μιας υποδομής αποτελούν σημαντική πηγή άντλησης πληροφοριών για το πως λειτουργεί η κάθε συσκευή σε σχέση με τις υπόλοιπες, αλλά και κάτω από διαφορετικές συνθήκες τόσο περιβαλλοντικές όσο και μεγάλου δικτυακού φορτίου. Ένας καλός διαχειριστής υποδομής αξιοποιεί κατάλληλα όλα όσα συλλέγονται και εφαρμόζει κατάλληλες τεχνικές για να βρει τη συσχέτιση μεταξύ αυτών και του πόσο πολύ επηρεάζουν το ένα το άλλο, προκειμένου να βελτιστοποιήσει τη λειτουργία της υποδομής και να μειώσει τα λειτουργικά έξοδα.

Σύμφωνα με προηγούμενες μελέτες [42], [3] οι κυριότεροι παράγοντες που επηρεάζουν την ενεργειακή κατανάλωση ενός εξυπηρετητή (server), είναι πρωτίστως η κεντρική επεξεργαστική μονάδα CPU και σε μικρότερο βαθμό η μνήμη RAM. Τα περισσότερα μοντέλα βελτιστοποίησης της ενεργειακής κατανάλωσης έχουν επικεντρωθεί γύρω από αυτούς τους δύο όρους, το καθένα με διαφορετική προσέγγιση στην προσπάθεια αυτή.

Στην παρούσα διπλωματική, λοιπόν, χρησιμοποιήθηκε η τεχνική της πολλαπλής παλινδρόμησης με δύο ανεξάρτητες μεταβλητές (multiple regression with two independent variables), τον βαθμό χρησιμοποίησης της CPU και της μνήμης RAM για την πρόβλεψη της καταναλισκόμενης ισχύς σε κάθε εξυπηρετητή.

## 5.2 Multiple Linear Regression Model

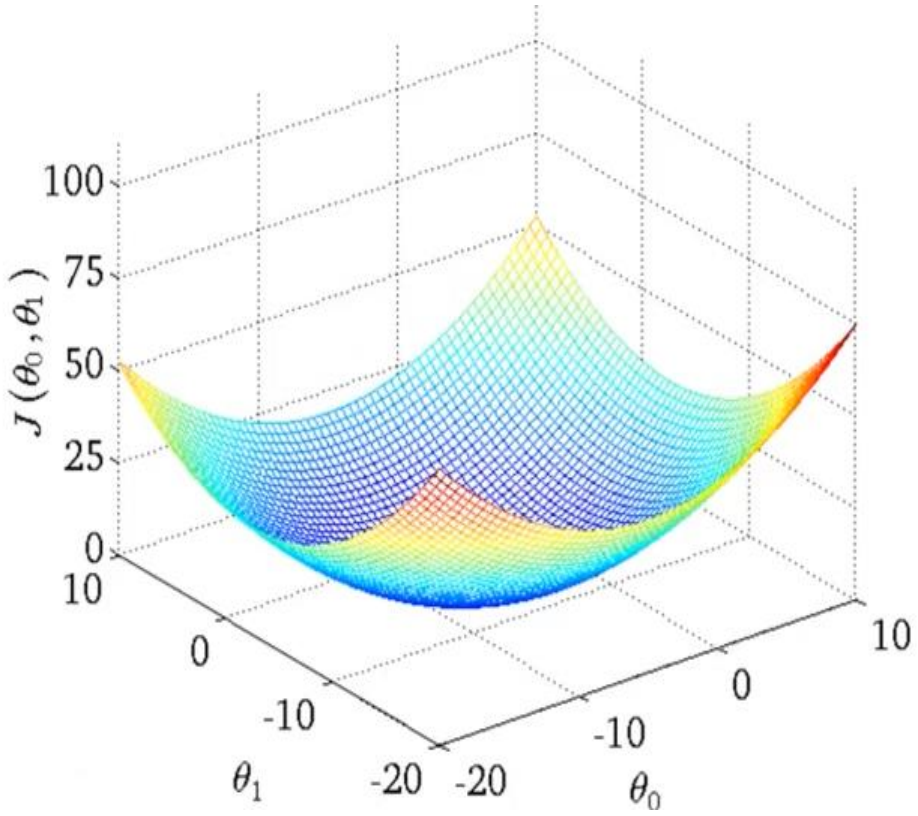
Η γραμμική παλινδρόμηση είναι μια γραμμική προσέγγιση για τη μοντελοποίηση της σχέσης μεταξύ μιας βαθμωτής απόκρισης (ή εξαρτημένης μεταβλητής) και μιας ή περισσοτέρων επεξηγηματικών μεταβλητών (ή ανεξάρτητων μεταβλητών). Η περίπτωση μιας επεξηγηματικής μεταβλητής ονομάζεται απλή γραμμική παλινδρόμηση. Για περισσότερες από μία επεξηγηματικές μεταβλητές, η διαδικασία ονομάζεται πολλαπλή γραμμική παλινδρόμηση. Στη γραμμική παλινδρόμηση, οι σχέσεις διαμορφώνονται χρησιμοποιώντας λειτουργίες γραμμικής πρόβλεψης των οποίων οι άγνωστες παράμετροι μοντέλου υπολογίζονται από τα δεδομένα. Αυτά τα μοντέλα ονομάζονται γραμμικά μοντέλα [43]. Στην απλή γραμμική παλινδρόμηση με μία ανεξάρτητη μεταβλητή έχουμε την ακόλουθη μαθηματική μορφή όπου οι παράμετροι $\theta_0$ και $\theta_1$ καθορίζουν το σημείο τομής με τον άξονα y και την κλίση αντίστοιχα. Για κάθε σύνολο δεδομένων ο αλγόριθμος μάθησης μπορεί να καθορίσει ποιες τιμές των $\theta_i$ παράγουν το κατάλληλο $h_\theta(x)$ για καλύτερη μοντελοποίηση των δεδομένων.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Ο τρόπος με τον οποίον διαπιστώνουμε πόσο αξιόπιστο είναι ένα μοντέλο $h_\theta(x)$ για ένα σύνολο δεδομένων ονομάζεται συνάρτηση κόστους J και ορίζεται όπως παρακάτω.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

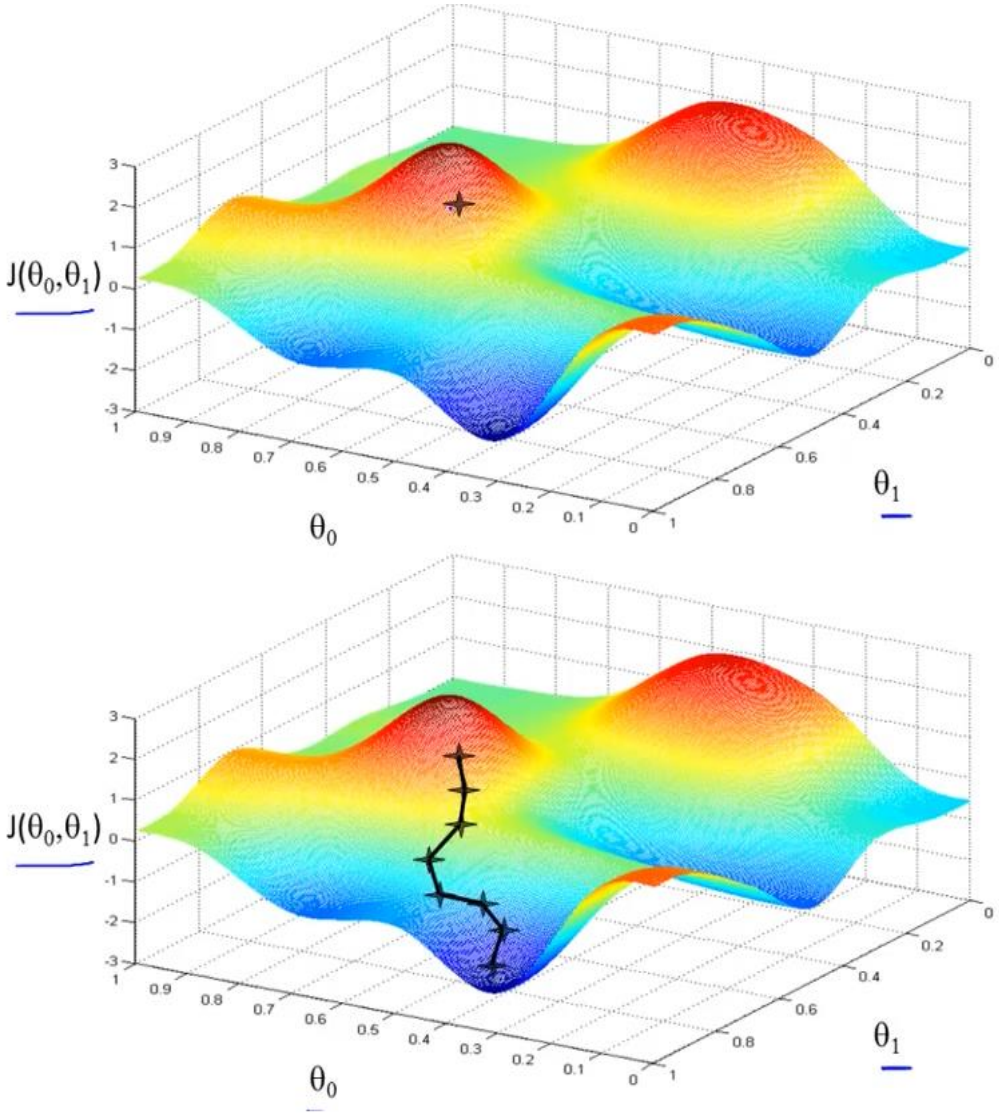
Δηλαδή, η συνάρτηση κόστους J στο μοντέλο γραμμικής παλινδρόμησης υπολογίζεται ως το άθροισμα τετραγώνων των διαφορών της εξαρτημένης μεταβλητής για κάθε είσοδο $x_i$ του συνόλου δεδομένων από την πραγματική τιμή της εξαρτώμενης μεταβλητής που σχετίζεται με το $x_i$ και διαιρώντας το συνολικό άθροισμα με 2m, όπου m το συνολικό πλήθος των δεδομένων. Σαφώς, εάν η υπόθεση γραμμικής παλινδρόμησης ήταν τέλεια, θα προέβλεπε πάντα τη σωστή τιμή της εξαρτώμενης μεταβλητής και θα οδηγούσε σε J = 0. Ανάλογα τις τιμές των $\theta_i$ που δίνουμε στο μοντέλο, το J αποκτά κάποια τιμή όπως φαίνεται στο επόμενο σχήμα για μονομεταβλητή γραμμική παλινδρόμηση (Εικόνα 33).




*Εικόνα 33. Σχηματική αναπαράσταση συνάρτησης κόστους*

Προκειμένου να επιτύχουμε ελαχιστοποίηση του J χωρίς να δοκιμάσουμε όλους τους δυνατούς συνδυασμούς των παραμέτρων $\theta_i$, χρειαζόμαστε έναν αλγόριθμο που να οδηγεί στις κατάλληλες τιμές $\theta_i$ ανεξάρτητα από τις αρχικές τους τιμές που έχουν. Ο αλγόριθμος που επιτυγχάνει αυτό ονομάζεται απότομης καθόδου (gradient descent). Ο αλγόριθμος Gradient Descent ενημερώνει τις τιμές $\theta_i$, επιλέγοντας εκείνες που μειώνουν το J με κάθε επανάληψη έως ότου επιτευχθεί ένα τοπικό ελάχιστο. Ανάλογα με τις αρχικές τιμές $\theta_i$, ο αλγόριθμος

μπορεί να συγκλίνει σε διαφορετικά τοπικά ελάχιστα, όπως φαίνεται στην Εικόνα 34 για μια αυθαίρετη συνάρτηση κόστους J, αλλά αυτό δεν αποτελεί πρόβλημα για μονομεταβλητή γραμμική παλινδρόμηση.




*Εικόνα 34. Εύρεση τοπικού ελαχίστου*

Ο κανόνας που ορίζει τον αλγόριθμο gradient descent έχει ως εξής:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Ένα βασικό χαρακτηριστικό του αλγορίθμου είναι ότι κάθε $\theta_i$ πρέπει να ενημερωθεί ταυτόχρονα έτσι ώστε η μερική παράγωγος να υπολογιστεί για το J με το ίδιο ζεύγος ($\theta_0$, $\theta_1$) πριν αλλάξουν οι τιμές τους. Ένα άλλο σημαντικό χαρακτηριστικό είναι η παράμετρος α, που είναι ισοδύναμη με το μέγεθος βήματος. Το α είναι γνωστό και ως ρυθμός εκμάθησης διότι καθορίζει το πόσο γρήγορα συγκλίνει ο αλγόριθμος. Αν το βήμα α είναι πολύ μικρό, ο αλγόριθμος θα πάρει πολύ χρόνο για να συγκλίνει και αν είναι πολύ μεγάλο, ο αλγόριθμος

μπορεί να υπερβεί το τοπικό ελάχιστο ή ακόμα και να αποτύχει και να αποκλίνει. Η μερική παράγωγος δίνει την κατεύθυνση προς την οποία αλλάζει η συνάρτηση J και ο αλγόριθμος κάνει ένα βήμα προς αυτήν, ενημερώνοντας το ζεύγος $\theta_i$ και ανανεώνονται τις τιμές του μέχρι να επιτευχθεί ένα τοπικό ελάχιστο [45].

Όταν έχουμε παραπάνω από μία ανεξάρτητες μεταβλητές, τότε έχουμε μοντέλο πολλαπλής γραμμικής παλινδρόμησης που είναι παρόμοιο με τη μονομεταβλητή γραμμική παλινδρόμηση. Κάθε μία από αυτές τις μεταβλητές έχει τη δική της παράμετρο και έτσι μπορούμε να ορίσουμε το $h_\theta(x)$ για ως εξής.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Το $\theta_0$ αντιπροσωπεύει την τιμή της υπόθεσης όταν όλες οι ανεξάρτητες μεταβλητές έχουν μηδενική τιμή. Κάθε άλλη $\theta_i$, μπορεί να θεωρηθεί ως το πόσο αυξάνεται η h(x) όταν το $x_i$ αυξάνεται κατά 1 μονάδα. Αυτό συμβαίνει επειδή, αν κρατάμε κάθε x σταθερή, εκτός από ένα $x_i$ που αυξάνουμε κατά 1, τότε μπορούμε να δούμε ότι $h_\theta(x_1,x_2,\ldots,x_i+1, x_{(i+1)}\ldots x_n) - h_\theta(x_1,x_2,\ldots x_i,x_{i+1},..x_n) = \theta_i(x_i+1) - \theta_i x_i = \theta_i$. Για μεγαλύτερη ευκολία μπορούμε να ομαδοποιήσουμε όλα τα $\theta_i$ σε ένα διάνυσμα $\theta$, όλες τις μεταβλητές σε ένα άλλο διάνυσμα $x$, και ορίζοντας $x_0 = 1$ μπορούμε να γράψουμε $h(x)$ συνοπτικά

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Όπως και πριν, θέλουμε να βρούμε τις παραμέτρους $\theta_i$ που ελαχιστοποιούν τη συνάρτηση κόστους J και το επιτυγχάνουμε μέσω του αλγόριθμου Gradient descent.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$\ldots$

}

Χρησιμοποιώντας τον αλγόριθμο gradient descent μπορούμε να ξεκινήσουμε με τυχαίες τιμές των $\theta_i$ και να οδηγούμαστε σε παραμέτρους που βελτιστοποιούν την

προσαρμογή στα δικά μας δεδομένα, όπως και στην περίπτωση μίας μεταβλητής θέτοντας κατάλληλα τον παράγοντα α [46].

Το ακόλουθο μοντέλο είναι ένα μοντέλο πολλαπλής γραμμικής παλινδρόμησης με μία εξαρτημένη μεταβλητή y και δύο μεταβλητές πρόβλεψης $x_1$ και $x_2$

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

όπου

y: η εξαρτημένη μεταβλητή

$\theta_0$ : η σταθερά

$\theta_1$ , $\theta_2$ : οι συντελεστές των $x_1$ και $x_2$ αντίστοιχα

## 5.3 Σενάριο χρήσης

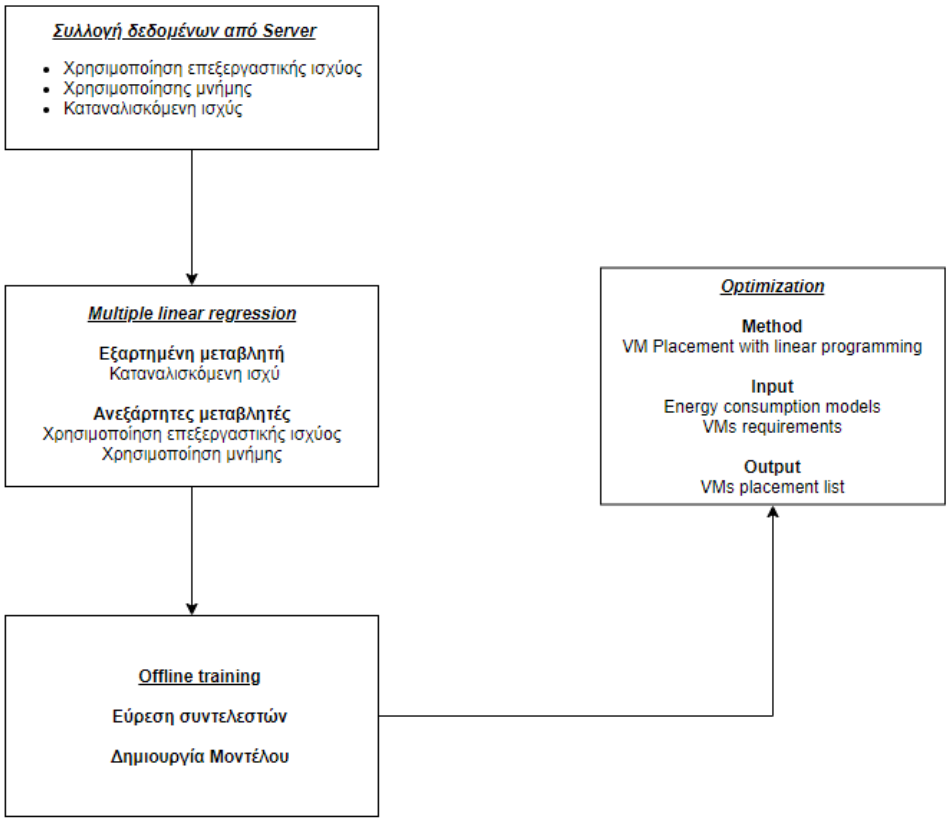
Με τη χρήση της γραμμικής παλινδρόμησης δημιουργήθηκε ένα σύστημα εξισώσεων πρόβλεψης ενεργειακής κατανάλωσης για κάθε ένα εξυπηρετητή, το οποίο με τη χρήση mixed integer programming μπορεί να οδηγήσει σε ελαχιστοποίηση της συνολικής κατανάλωσης. Το μοντέλο της γραμμικής παλινδρόμησης εφαρμόσθηκε στις τιμές που λάβαμε για τους εξυπηρετητές dragon, orestes, trillium, pegasus και aether του εργαστηρίου. Έτσι, λαμβάνοντας από τη βάση δεδομένων prometheus του InfluxDB, με κατάλληλα queries τις παραπάνω μετρήσεις για κάθε εξυπηρετητή κάτω από διαφορετικές συνθήκες, ανά τακτά χρονικά διαστήματα μέσα στην ημέρα, είναι εφικτή η βέλτιστη τοποθέτηση VM σε κάθε έναν από αυτούς ώστε να ελαχιστοποιηθεί το συνολικό κόστος λειτουργίας [42].

Αρχικά λαμβάνουμε μέσω της βάσης δεδομένων prometheus, που είναι κομμάτι του influxDB, τα δεδομένα που είναι απαραίτητα για τη βελτιστοποίηση για ένα χρονικό διάστημα που ορίζεται από τον χρήστη.

```
$influx -database 'prometheus' -execute ' select "value"::field from hrStorageUsed where hrStorageIndex='6' and job=<device> and time>= <time>

$influx -database 'prometheus' -execute ' SELECT * "value"::field from hrProcessorLoad where hrDeviceIndex='6' and job=<device> and time>= <time>

$influx -database 'prometheus' -execute ' SELECT * from amperageProbeReading where job=<device> and time>= <time>

$influx -database 'prometheus' -execute ' SELECT * from hs110_power_watt where job=<device> and time>= <time>
```

Για το μοντέλο κατασκευάστηκε και χρησιμοποιήθηκε το MultRegression.py το οποίο παίρνει ως είσοδο τις παραπάνω τιμές καταναλισκόμενης ισχύος, τον βαθμό χρησιμοποίησης CPU και μνήμης RAM και δίνει τη σταθερά και τους συντελεστές του μοντέλου κατανάλωσης εντός Data Center. Ως έξοδο λαμβάνουμε για κάθε ένα εξυπηρετητή ένα

σύστημα εξισώσεων πρόβλεψης ενεργειακής κατανάλωσης κατά τη διάρκεια της ημέρας ανάλογα με την αξιοποίηση των διαθέσιμων πόρων, της μορφής $y_i = \theta_{i0} + \theta_{i1} x_{i1} + \theta_{i2} x_{i2}$, όπου i είναι ο κάθε εξυπηρετητής. Στη συγκεκριμένη υλοποίηση το $y_i$ είναι η προβλεπόμενη τιμή της ενεργειακής κατανάλωσης και $x_{i1}$ και $x_{i2}$ είναι ο βαθμός χρησιμοποίησης CPU και μνήμης RAM, αντίστοιχα. Η εύρεση της ελάχιστης τιμής της συνολικής ενεργειακής κατανάλωσης των εξυπηρετητών υπολογίζεται συνδυάζοντας τις απαιτήσεις των διαθέσιμων VMs, σύμφωνα με το παραπάνω σύστημα, και εφαρμόζοντας τη μέθοδο mixed integer programming. Ως έξοδο από τη διαδικασία αυτή λαμβάνουμε μια λίστα που περιέχει τη βέλτιστη τοποθέτηση των VMs στους εξυπηρετητές. Στην Εικόνα 35 παρουσιάζεται το διάγραμμα ροής της διαδικασίας που ακολουθήθηκε.




*Εικόνα 35. Διάγραμμα ροής Βελτιστοποίησης*

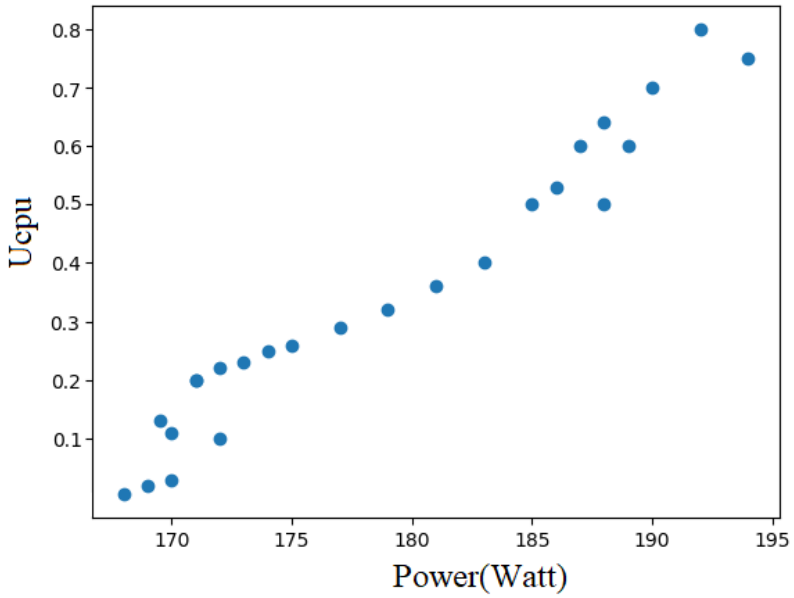
## 5.4 Παρουσίαση αποτελεσμάτων και συμπεράσματα

Τα αποτελέσματα από την εκτέλεση των ανωτέρων διαδικασιών φαίνονται στον Πίνακα 3 που περιέχει τις σταθερές και τους συντελεστές που αντιστοιχούν σε χρησιμοποίηση CPU και RAM για μια συγκεκριμένη ώρα.

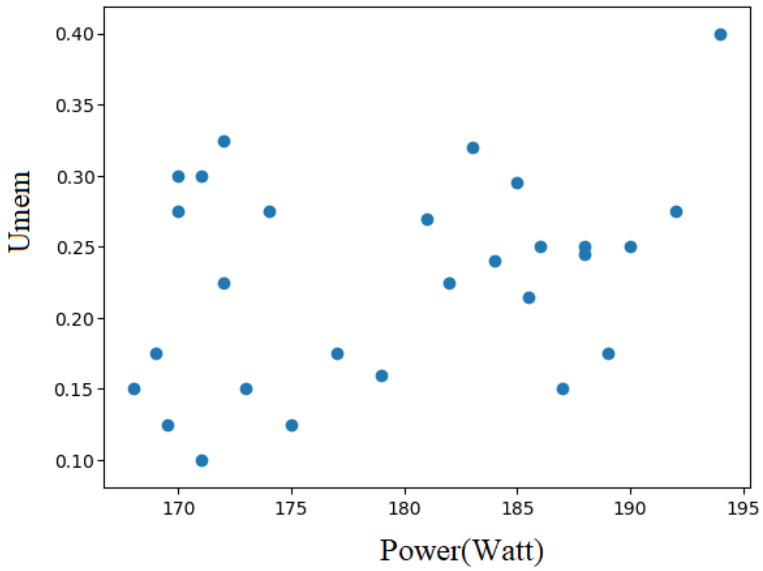
| Server_Name | Constant | Coefficient 1 | Coefficient 2 |
|---|---|---|---|
| Trillium | 120.1062 | 3.1411 | 0.045 |
| Dragon | 120.4146 | 6.2810 | 0.0842 |
| Aether | 69.4738 | 5.1521 | 0.1930 |
| Orestes | 123.06178 | 23.84775 | 0.7299 |
| Pegasus | 141.2994 | 3.6204 | 0.6637 |

*Πίνακας 3. Αποτελέσματα σταθερών και συντελεστών από γραμμική παλινδρόμηση*

Όπως φαίνεται στην Εικόνα 36 για τον εξυπηρετητή Pegasus παρατηρούμε ότι υπάρχει μια γραμμική σχέση της ενεργειακής κατανάλωσης με το βαθμό χρησιμοποίησης του επεξεργαστή. Στην Εικόνα 37 δεν είναι τόσο προφανής η γραμμική σχέση ενεργειακής κατανάλωσης με το βαθμό χρησιμοποίησης της μνήμης RAM, ωστόσο στο σημείο αυτό πρέπει να επισημανθεί ότι οι μνήμες RAM καταναλώνουν μικρά ποσά ενέργειας λόγω της χαμηλότερης τάσης λειτουργίας τους (1.5V-2.5V), οπότε λόγω της διακύμανσης της ενεργειακής κατανάλωσης από την επεξεργαστική ισχύ αρχικά φαίνεται ότι δεν επηρεάζει ιδιαίτερα και για αυτό απαιτείται ιδιαίτερη προσοχή στη μελέτη της. Στις μετρήσεις που πήραμε διαπιστώσαμε ότι η εισαγωγή του βαθμού χρησιμοποίησης της μνήμης RAM στο γραμμικό μας μοντέλο είχε ως αποτέλεσμα καλύτερη ακρίβεια στην πρόβλεψη.




*Εικόνα 36. Διάγραμμα ισχύς συναρτήσει της επεξεργαστικής ισχύος*




*Εικόνα 37. Διάγραμμα ισχύος συναρτήσει της μνήμης*

Για να γίνει αξιολόγηση ενός μοντέλου απαιτούνται κάποιοι παράγοντες αξιοπιστίας που δείχνουν πόσο καλά προσεγγίζει το μοντέλο τις πραγματικές τιμές του συστήματος. Ο πρώτος παράγοντας είναι το adjusted R-squared, δηλαδή, το ποσοστό διακύμανσης της εξαρτημένης μεταβλητής που είναι προβλέψιμο από τις ανεξάρτητες μεταβλητές. Όσο μεγαλύτερη τιμή έχει τόσο πιο καλά το μοντέλο προσεγγίζει την πραγματικότητα. Το ίδιο σημαντικός είναι και ο παράγοντας p-value, που είναι η πιθανότητα απόκτησης των παρατηρούμενων αποτελεσμάτων μιας δοκιμής, υποθέτοντας ότι η μηδενική υπόθεση είναι σωστή. Όσο μικρότερη είναι η τιμή του τόσο πιο σημαντικός είναι ο παράγοντας που μελετάμε για το μοντέλο μας. Από την εκτέλεση του αλγορίθμου προέκυψε ότι οι τιμές p-values είναι αρκετά μικρές (0.00 και 0.1-0.3 για την επεξεργαστική ισχύ και τη μνήμη αντίστοιχα) και οι τιμές adjusted R-squared κυμαίνονται στο εύρος 79% με 90%, το οποίο σημαίνει ότι το μοντέλο ικανοποιεί σε μεγάλο βαθμό τις περιπτώσεις που εξετάζουμε και είναι σε θέση να προβλέψει, με δεδομένες τιμές βαθμού χρησιμοποίησης επεξεργαστή και μνήμης, την απαιτούμενη ενέργεια [47]. Στον Πίνακα 4 φαίνονται οι τιμές αυτές για διάφορες στιγμές της μέρας. Επομένως από τα παραπάνω προκύπτει ότι η επιλογή του γραμμικού μοντέλου για τους εξυπηρετητές ήταν σωστή.

| ***time*** | ***const_p-value*** | ***coef1_p-value*** | ***coef2_p-value*** | ***adjusted R-squared*** |
|---|---|---|---|---|
| 1:00 | 0.00 | 0.00 | 0.112 | 0.901 |
| 2:00 | 0.00 | 0.00 | 0.153 | 0.869 |
| 3:00 | 0.00 | 0.00 | 0.221 | 0.843 |
| 4:00 | 0.00 | 0.00 | 0.251 | 0.826 |
| 5:00 | 0.00 | 0.00 | 0.233 | 0.813 |
| 6:00 | 0.00 | 0.00 | 0.272 | 0.798 |
| 7:00 | 0.00 | 0.00 | 0.142 | 0.876 |
| 8:00 | 0.00 | 0.00 | 0.236 | 0.837 |
| 9:00 | 0.00 | 0.00 | 0.152 | 0.871 |
| 10:00 | 0.00 | 0.00 | 0.164 | 0.862 |
| 11:00 | 0.00 | 0.00 | 0.219 | 0.839 |
| 12:00 | 0.00 | 0.00 | 0.191 | 0.851 |

| | | | | |
|---|---|---|---|---|
| 13:00 | 0.00 | 0.00 | 0.157 | 0.865 |
| 14:00 | 0.00 | 0.00 | 0.222 | 0.835 |
| 15:00 | 0.00 | 0.00 | 0.128 | 0.894 |
| 16:00 | 0.00 | 0.00 | 0.166 | 0.861 |
| 17:00 | 0.00 | 0.00 | 0.242 | 0.819 |
| 18:00 | 0.00 | 0.00 | 0.192 | 0.852 |
| 19:00 | 0.00 | 0.00 | 0.117 | 0.897 |
| 20:00 | 0.00 | 0.00 | 0.182 | 0.859 |
| 21:00 | 0.00 | 0.00 | 0.151 | 0.871 |
| 22:00 | 0.00 | 0.00 | 0.166 | 0.862 |
| 23:00 | 0.00 | 0.00 | 0.137 | 0.884 |
| 0:00 | 0.00 | 0.00 | 0.149 | 0.874 |

*Πίνακας 4. . p-values και adjusted R-squared για εγκυρότητα μοντέλου*

Η εφαρμογή για τη βέλτιστη τοποθέτηση των VM σε αντίστοιχους φυσικούς εξυπηρετητές βασίζεται στη λύση του προβλήματος βελτιστοποίησης που ορίζεται παρακάτω. Έστω

S: το σύνολο των εξυπηρετητών που είναι διαθέσιμοι

$\theta_i^{(s)}$: Οι τιμές που χαρακτηρίζουν την ενεργειακή κατανάλωση του εξυπηρετητή $s \in$ S , i $\in \{0,1,2\}$

$C_e^{(s)}$: Το δεσμευμένο πλήθος πυρήνων του VM $e \in$ E του εξυπηρετητή $s \in$ S

$C_s$ : Το σύνολο των διαθέσιμων πυρήνων του εξυπηρετητή $s \in$ S

$R_e^{(s)}$: Το δεσμευμένο μέγεθος μνήμης RAM του VM $e \in$ E του εξυπηρετητή $s \in$ S

$R_s$ : Η εγκατεστημένη μνήμη RAM του εξυπηρετητή $s \in$ S

E: Το σύνολο των VM που πρέπει να τοποθετηθούν στους εξυπηρετητές

$Z_s^{(e)}$: 1 αν το VM e όπου $e \in$ E βρίσκεται στον εξυπηρετητή $s \in$ S, 0 αλλιώς

N: Ο συντελεστής που περιγράφει την υπερχρησιμοποίηση (overcommition ratio) στους διαθέσιμους επεξεργαστές

Η συνάρτηση κέρδους που πρέπει να ελαχιστοποιηθεί είναι:

$$\sum_{s \in S}( \theta_0^{(s)} + \theta_1^{(s)} \cdot C_s + \theta_2^{(s)} \cdot R_s ) \qquad (1)$$

Με τους περιορισμούς

$$\sum_{e \in E, s \in S} R_e^{(s)} \leq R_s \ , \forall\, e \in \text{E}, s \in \text{S} \qquad (2)$$
$$\sum_{e \in E, s \in S} C_e^{(s)} \leq N \cdot C_s \ , \forall\, e \in \text{E}, s \in \text{S} \qquad (3)$$
$$\sum_{e \in E, s \in S} Z_s^{(e)} = 1 \qquad (4)$$

Η εξίσωση (2) εξασφαλίζει ότι δεν θα τοποθετηθούν VM με περισσότερη μνήμη από τη διαθέσιμη του εξυπηρετητή. Αντίστοιχα η εξίσωση (3) εξασφαλίζει ότι δεν θα ανατεθούν

περισσότερες CPU από τις διαθέσιμες πολλαπλασιασμένες με τον συντελεστή υπερχρησιμοποίσης των διαθέσιμων πόρων. Δηλαδή, επιτρέπεται η υπερχρησιμοποίηση των διαθέσιμων πυρήνων CPU, για παράδειγμα σε cloud περιβάλλοντα, και σε περίπτωση overcommision ratio 2:1 με 10 πυρήνες CPU διαθέσιμους να διατίθενται 20 vCPU στα VM. Τέλος, η εξίσωση (4) εξασφαλίζει ότι όλα τα VM θα τοποθετηθούν σε κάποιο εξυπηρετητή. Η ελάχιστη τιμή της παράστασης στην εξίσωσης (1) μπορεί να βρεθεί με τη χρήση λογισμικού επίλυσης προβλημάτων mixed integer programming, όπως η σουίτα λογισμικού CPLEX [44].

Η λίστα με τα VM που φιλοξενούνται στους εξυπηρετητές που αναφέρθηκαν προηγουμένως και οι αντίστοιχες απαιτήσεις τους σε vCPUs και RAM φαίνεται στο παρακάτω Πίνακα 5.

| ID | vCPU | RAM (GB) |
|---|---|---|
| 1 | 4 | 4 |
| 2 | 2 | 4 |
| 3 | 1 | 1 |
| 4 | 4 | 8 |
| 5 | 4 | 8 |
| 6 | 4 | 4 |
| 7 | 4 | 8 |
| 8 | 4 | 8 |
| 9 | 8 | 8 |
| 10 | 2 | 2 |
| 11 | 2 | 2 |
| 12 | 1 | 1 |
| 13 | 1 | 1 |
| 14 | 4 | 4 |
| 15 | 1 | 4 |
| 16 | 2 | 2 |
| 17 | 4 | 8 |
| 18 | 8 | 32 |
| 19 | 2 | 3 |
| 20 | 3 | 4 |
| 21 | 1 | 4 |
| 22 | 2 | 3 |
| 23 | 8 | 24 |
| 24 | 2 | 1 |
| 25 | 4 | 4 |
| 26 | 2 | 8 |
| 27 | 2 | 4 |
| 28 | 2 | 8 |
| 29 | 2 | 8 |
| 30 | 2 | 4 |
| 31 | 4 | 16 |
| 32 | 1 | 2 |
| 33 | 2 | 2 |
| 34 | 4 | 5 |
| 35 | 2 | 5 |
| 36 | 4 | 6 |
| 37 | 2 | 2 |
| 38 | 2 | 4 |

*Πίνακας 5. Λίστα διαθέσιμων VMs*

Η αρχική συνολική κατανάλωση όλων των εξυπηρετητών είναι 708 Watt σύμφωνα με την κατανομή των VMs στους εξυπηρετητές στον Πίνακα 6. Η εύρεση της βέλτιστης τιμής μέσω CPLEX προκύπτει σε 0.01 seconds για 5 εξυπηρετητές και 38 VM. Η συνολική κατανάλωση που μπορεί να προκύψει από τη νέα τοποθέτηση των VM ύστερα από την εφαρμογή της βελτιστοποίησης είναι 585.9 Watt με την κατανομή των VMs στους εξυπηρετητές όπως φαίνεται στον Πίνακα 7. Συγκεκριμένα στους πίνακες βλέπουμε ποια VMs εκτελούνται στον κάθε διαθέσιμο εξυπηρετητή, για παράδειγμα στη βέλτιστη λύση στον εξυπηρετητή Z1 εκτελούνται τα VMs 2, 3, 5, 7, 8, 15, 17, 21, 23, 26, 27, 28, 29, 30, 31, 32, 35 και 38 και ούτω καθεξής, όπου Z1=Trillium, κλπ.

| | VM1 | VM2 | VM3 | VM4 | VM5 | VM6 | VM7 | VM8 | VM9 | VM10 | VM11 | VM12 | VM13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Z2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | VM14 | VM15 | VM16 | VM17 | VM18 | VM19 | VM20 | VM21 | VM22 | VM23 | VM24 | VM25 | VM26 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Z4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Z5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | VM27 | VM28 | VM29 | VM30 | VM31 | VM32 | VM33 | VM34 | VM35 | VM36 | VM37 | VM38 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Z5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

*Πίνακας 6. Τοποθέτηση VM στους εξυπηρετητές πριν την εφαρμογή βελτιστοποίησης*

| | VM1 | VM2 | VM3 | VM4 | VM5 | VM6 | VM7 | VM8 | VM9 | VM10 | VM11 | VM12 | VM13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Z2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z5 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

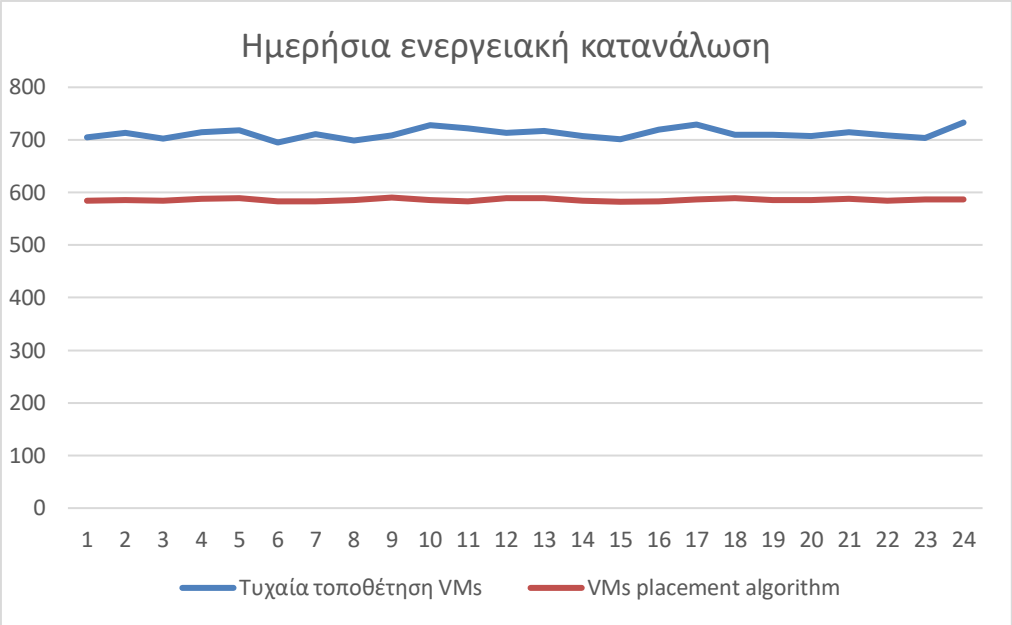
| | VM14 | VM15 | VM16 | VM17 | VM18 | VM19 | VM20 | VM21 | VM22 | VM23 | VM24 | VM25 | VM26 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| Z2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z5 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| | VM27 | VM28 | VM29 | VM30 | VM31 | VM32 | VM33 | VM34 | VM35 | VM36 | VM37 | VM38 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Z2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| Z3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

*Πίνακας 7. Τοποθέτηση VM στους εξυπηρετητές μετά την εφαρμογή βελτιστοποίησης*

Για μεγαλύτερη πληρότητα εφαρμόστηκε η μέθοδος της γραμμικής παλινδρόμησης για κάθε εξυπηρετητή ανά μία ώρα προκειμένου με την κατάλληλη τοποθέτηση των VM να επιτυγχάνεται η μέγιστη δυνατή ενεργειακή βελτιστοποίηση κατά τη διάρκεια μίας ημέρας.

Στον Πίνακα 8 φαίνεται η βέλτιστη κατανομή των VMs καθ' όλη τη διάρκεια της ημέρας ανά μία ώρα στους εξυπηρετητές. Στην Εικόνα 38 φαίνεται ξεκάθαρα η βελτιστοποίηση της κατανάλωσης, ενώ στον Πίνακα 9 καταγράφεται το ποσοστό εξοικονόμησης της συνολικής καταναλισκόμενης ισχύος που φτάνει μέχρι και 20%.




*Εικόνα 38. Σύγκριση ενεργειακής κατανάλωσης*

| Time | Trillium | Dragon | Aether | Orestes | Pegasus |
|---|---|---|---|---|---|
| 1:00 | 18 | 16 | 0 | 1 | 3 |
| 2:00 | 18 | 16 | 0 | 1 | 3 |
| 3:00 | 18 | 16 | 0 | 1 | 3 |
| 4:00 | 18 | 16 | 0 | 1 | 3 |
| 5:00 | 18 | 16 | 0 | 4 | 0 |
| 6:00 | 18 | 16 | 0 | 1 | 3 |
| 7:00 | 18 | 16 | 0 | 1 | 3 |
| 8:00 | 18 | 16 | 0 | 0 | 4 |
| 9:00 | 18 | 16 | 0 | 4 | 0 |
| 10:00 | 16 | 17 | 0 | 0 | 5 |
| 11:00 | 18 | 16 | 0 | 0 | 4 |
| 12:00 | 18 | 16 | 0 | 1 | 3 |
| 13:00 | 18 | 16 | 0 | 4 | 0 |
| 14:00 | 18 | 16 | 0 | 4 | 0 |
| 15:00 | 18 | 15 | 0 | 5 | 0 |
| 16:00 | 18 | 16 | 0 | 4 | 0 |
| 17:00 | 18 | 16 | 0 | 1 | 3 |
| 18:00 | 18 | 16 | 0 | 4 | 0 |
| 19:00 | 16 | 17 | 0 | 0 | 5 |
| 20:00 | 18 | 16 | 0 | 4 | 0 |
| 21:00 | 18 | 16 | 0 | 1 | 3 |
| 22:00 | 18 | 16 | 0 | 4 | 0 |
| 23:00 | 18 | 16 | 0 | 4 | 0 |
| 0:00 | 18 | 16 | 0 | 4 | 0 |

*Πίνακας 8. Κατανομή VMs σε εξυπηρετητές κατά τη διάρκεια της ημέρας*

| Time | Αρχική κατανάλωση | Τελική Κατανάλωση | Ποσοστό Βελτίωσης |
|---|---|---|---|
| 1:00 | 705 | 584.2 | 17.13 |
| 2:00 | 713 | 585.4 | 17.88 |
| 3:00 | 703 | 583.8 | 16.96 |
| 4:00 | 715 | 587.9 | 17.77 |
| 5:00 | 718 | 589.7 | 17.86 |
| 6:00 | 695 | 582.9 | 16.12 |
| 7:00 | 711 | 583.2 | 17.97 |
| 8:00 | 699 | 585.3 | 16.26 |
| 9:00 | 709 | 590.3 | 16.73 |
| 10:00 | 728 | 585.5 | 19.56 |
| 11:00 | 722 | 583.7 | 19.15 |
| 12:00 | 713 | 589.7 | 17.29 |
| 13:00 | 717 | 589.6 | 17.77 |
| 14:00 | 707 | 584.6 | 17.31 |
| 15:00 | 701 | 582.5 | 16.92 |
| 16:00 | 720 | 583.1 | 19.01 |
| 17:00 | 729 | 586.3 | 19.57 |
| 18:00 | 710 | 588.9 | 17.05 |
| 19:00 | 710 | 585.9 | 17.48 |
| 20:00 | 707 | 585.3 | 17.21 |
| 21:00 | 715 | 587.8 | 17.79 |
| 22:00 | 709 | 583.7 | 17.66 |
| 23:00 | 704 | 586.5 | 16.69 |
| 0:00 | 733 | 586.9 | 19.94 |

*Πίνακας 9. Ποσοστά βελτίωσης κατά τη διάρκεια της ημέρας*

Από τους παραπάνω πίνακες και διαγράμματα είναι εμφανές ότι ακόμα και με τη χρήση μόνο δύο παραγόντων όπως είναι η χρησιμοποίηση CPU και μνήμης RAM επιτυγχάνονται ικανοποιητικά αποτελέσματα, γεγονός που δείχνει ότι υπάρχουν σημαντικά περιθώρια βελτίωσης. Επιπλέον επειδή τα υπολογιστικά φορτία (workloads) δεν έχουν μεγάλες διακυμάνσεις κατά τη διάρκεια της ημέρας τα ποσοστά εξοικονόμησης είναι σχεδόν σταθερά. Μάλιστα, παρατηρείται ότι πάντα υπάρχει προτίμηση κατανομής των VMs πρώτα στους καλύτερα ενεργειακά εξυπηρετητές, με κάποιους να μην έχουν φορτίο. Όμως, η αρχική προσέγγιση εφαρμογής της μεθόδου μια φόρα την ημέρα ίσως να μην καλύπτει Data Center όπου παρατηρούνται διακυμάνσεις στα υπολογιστικά φορτία στη διάρκεια της ημέρας. Η περίοδος υπολογισμού του προβλήματος ελαχιστοποίησης της καταναλισκόμενη ενέργειας πρέπει επιλέγεται ανάλογα με την κάθε περίπτωση.

# 6 Επίλογος

## 6.1 Σύνοψη

Σκοπός της παρούσας διπλωματικής ήταν ο σχεδιασμός και η ανάπτυξη μιας εφαρμογής για τη συλλογή μετρήσεων κατανάλωσης, θερμοκρασίας και υγρασίας από ετερογενείς πηγές, και η αποθήκευση τους σε βάση δεδομένων κατάλληλη για τη διαχείριση χρονοσειρών. Η υλοποίηση βασίστηκε εξ ολοκλήρου σε λογισμικό ανοιχτού κώδικα και δόθηκε ιδιαίτερη έμφαση στα εργαλεία: Prometheus, που αποτέλεσε τη βάση του συστήματος, για την συλλογή δεδομένων, InfluxDB για την αποθήκευση τους, Kafka για την επιτυχή μεταφορά τους και Grafana για την οπτικοποίηση τους. Με τη χρήση αυτών επιτυγχάνεται ο αποτελεσματικός έλεγχος της υποδομής και ο διαχειριστής είναι σε θέση να γνωρίζει ανά πάσα στιγμή και σε πραγματικό χρόνο την ενεργειακή κατανάλωση, τη δικτυακή κίνηση και τις περιβαλλοντικές συνθήκες. Η εποπτεία και μελέτη των αναφερθέντων στοιχείων, όπως διαπιστώθηκε επιτρέπουν τη μείωση των λειτουργικών εξόδων των DC είτε αυτό αφορά έγκαιρο εντοπισμό των προβλημάτων που δυσχεραίνουν τη λειτουργία τους και άμεση επισκευή τους, είτε εξοικονόμηση καταναλισκόμενης ενέργειας με εφαρμογή αλγορίθμων και τεχνικών βελτιστοποίησης της κατανάλωσης, με ταυτόχρονη διατήρηση της ποιότητας λειτουργίας. Η αξία του εγχειρήματος είναι ιδιαίτερα σημαντική αν αναλογιστεί κάποιος τη ραγδαία αύξηση σε δεδομένα, που αναμένεται να φτάσει τα 175 zettabytes έως το 2025 [39], λόγω της ανάπτυξης τεχνολογιών NFV, της εισαγωγής του 5G και της ανάγκης για υπολογιστική ισχύ απομακρυσμένα μέσω του νέφους. Όλα αυτά καθιστούν την εποπτεία των DC απαραίτητη αφού είναι υπεύθυνα για τη λήψη, την επεξεργασία, την αποθήκευση και τη διαχείριση αυτού του τεράστιου όγκου πληροφορίας, και τα οποία με το πέρασμα του χρόνου γίνονται ολοένα και πιο περίπλοκα και εξελιγμένα και οι υπάρχουσες λύσεις θεωρούνται πλέον απαρχαιωμένες και ανεπαρκείς.

Η εφαρμογή χρησιμοποιήθηκε με μεγάλη επιτυχία στο χώρο του Εργαστηρίου Δικτύων Υπολογιστών και περιλαμβάνει ένα ευρύ φάσμα δεδομένων για την ολοκληρωμένη παρακολούθησή του υπό μελέτη χώρου από servers, εικονικούς μεταγωγείς (OVS), δρομολογητές, αισθητήρες καθώς και συσκευές NIC. Τα δεδομένα αυτά διαθέσιμα σε γραφήματα μέσω του Grafana. Επιπρόσθετα εκτελέστηκε και ένα σενάριο όπου οι παραπάνω μετρήσεις χρησιμοποιήθηκαν για την εφαρμογή ενός μοντέλου ενεργειακής βελτιστοποίησης που αποσκοπούσε στην κατάλληλη τοποθέτηση των VMs στους servers, ανάλογα με τις απαιτήσεις του κάθε VM και το υλικό (hardware) που διέθετε ο κάθε server.

## 6.2 Μελλοντικές επεκτάσεις

Στην παρούσα μελέτη, ο σχεδιασμός και η υλοποίηση του μοντέλου ενεργειακής βελτιστοποίησης στηρίχθηκε στην ενεργειακή κατανάλωση από τη χρησιμοποίηση (α) της κεντρικής υπολογιστικής μονάδας και (β) της μνήμης που αποτελούν τις κυρίαρχες συνιστώσες, στο μοντέλο γραμμικής παλινδρόμησης με δύο ανεξάρτητες μεταβλητές. Αντίστοιχα, στις ίδιες παραμέτρους βασίστηκε ο αλγόριθμος βελτιστοποίησης με έμφαση στο VM placement, το οποίο έδωσε θετικά αποτελέσματα στο σκοπό μας. Η μελέτη και η επεξεργασία περισσότερων παραγόντων, όπως είναι η δικτυακή κίνηση, οι παράγοντες θερμοκρασίας και υγρασίας, οι ταχύτητες εγγραφής και ανάγνωσης των δίσκων και εισαγωγής τους σε πιο σύνθετα μοντέλα πρόβλεψης με διάφορες μεθόδους όπως είναι η μηχανική μάθηση και τα νευρωνικά δίκτυα (χρήση Elman Neural Network[29]) πιθανώς να προσφέρει μεγαλύτερη ακρίβεια και κατ' επέκταση καλύτερη αξιοποίηση των πόρων. Λόγω του μεγάλου όγκου πληροφορίας που μεταδίδεται καθημερινά η γνώση των διαδρομών δικτυακής κίνησης επιτρέπει τη σχεδίαση αρχιτεκτονικών υποδομών που θα είναι σε θέση να διαμοιράζουν αποτελεσματικά το μεγάλο όγκο στα διάφορα κέντρα, με επιλογή εκείνου που στο σύνολο θα έχει μικρότερη ενεργειακή κατανάλωση. Επιπρόσθετα γνωρίζοντας τα επίπεδα της τάσης και του ρεύματος υπάρχει η δυνατότητα εύρεσης λειτουργίας των ιδανικών συνθηκών για τα διάφορα εξαρτήματα σε ένα κέντρο [48].

Το σύστημα που υλοποιήθηκε, πέρα από όσα αναφέρθηκαν, παρέχει τη δυνατότητα στον εκάστοτε χρήστη, εφόσον το επιθυμεί, να δημιουργήσει και να προσθέσει κατάλληλες ειδοποιήσεις ανάλογα με το αν οι μετρήσεις είναι μεγαλύτερες ή μικρότερες από ένα κατώφλι που έχει ορίσει αυτός ή είναι ανάμεσα στα επιθυμητά όρια. Έτσι μπορούν να εκτελούνται κατάλληλες ενέργειες χωρίς να απαιτείται κάθε φορά η συμβολή του χρήστη, παρέχοντας ένας είδος αυτοματοποίησης στην υποδομή. Ακόμα αν οι μετρήσεις αφορούν δικτυακή κίνηση επιτρέπουν την καλύτερη προστασία από επιθέσεις και εισβολές (πχ DDOS, διαρροή ευαίσθητων πληροφοριών κ.λπ) και την απρόσκοπτη λειτουργία του DC.

Τέλος, μπορεί να γίνει επέκταση της μεθόδου ώστε να μελετάται κάθε πότε πρέπει να γίνεται VM migration, σε περιπτώσεις που η μεταφορά ενός VM είναι χρονοβόρα και κοστοβόρα και αναιρεί την επιθυμητή βελτιστοποίηση. Επιπρόσθετα, επειδή με τη μέθοδο διαθέτουμε και τη χρησιμοποίηση του CPU και της μνήμης μετά την εφαρμογή του μοντέλου, μπορεί να γίνει πρόβλεψη αναδιανομής των VM, είτε σε περίπτωση εισαγωγής νέων VM είτε νέων Servers, επιτρέποντας τη δυνατότητα μελλοντικού σχεδιασμού επέκτασης. Αν συμπεριληφθούν και οι περιβαλλοντικές συνθήκες στον παραπάνω τρόπο, ίσως είναι εφικτό να επιτευχθούν ακόμα χαμηλότερα επίπεδα ενεργειακής κατανάλωσης.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Moises Levy, Jason O. Hallstrom, "A New Approach to Data Center Infrastructure Monitoring and Management (DCIMM)", 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp 1-2

[2] Wazir Zada Khan, M.Y. Aalsalem, H.M. Zangoti, M. Zahid, M. Khalil Afzal, "Internet of Things based Physical and Environmental Monitoring System for Data Centers", 2018 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), pp 41-42

[3] Data Center Energy Consumption Modeling: A Survey

[4] https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/

[5] https://www.techrepublic.com/article/why-time-series-databases-are-exploding-in-popularity/

[6] https://www.researchoptimus.com/article/what-is-time-series-analysis.php

[7] https://blog.nexcess.net/what-are-data-center-tiers-explained/

[8] https://www.hpe.com/us/en/what-is/data-center-tiers.html

[9] Salam Ismaee, Raed Karim, Ali Miri, "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres", Ismaeel et al. Journal of Cloud Computing: Advances, Systems and Applications (2018) 7:10 pp 1-2

[10] https://machinelearningmastery.com/findings-comparing-classical-and-machine-learning-methods-for-time-series-forecasting/

[11] https://spectrum.ieee.org/energywise/energy/environment/green-data-the-vext-step-to-zeroemissions-data-centers (Πρόσβαση Σεπτέμβριος 2019)

[12] https://opensourceforu.com/2017/04/prometheus/ (Πρόσβαση Σεπτέμβριος 2019)

[13] https://prometheus.io/docs/introduction/overview/ (Πρόσβαση Σεπτέμβριος 2019)

[14] Xun Wu, Sudarshan Kadambi, Devram Kandhare, Aaron Ploetz, "Seven NoSQL Databases in a Week", Publisher: Packt Publishing Release Date: March 2018 (Πρόσβαση Οκτώβριος 2019)

[15] https://devconnected.com/the-definitive-guide-to-influxdb-in-2019/ (Πρόσβαση Σεπτέμβριος 2019)

[16] https://devconnected.com/the-definitive-guide-to-prometheus-in-2019/ (Πρόσβαση Σεπτέμβριος 2019)

[17] https://grafana.com/docs/ (Πρόσβαση Σεπτέμβριος 2019)

[18] Mohammad M. Tajiki, Stefano Salsano, “ Joint Energy Efficent and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining

[19] https://www.8bitmen.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/ (Πρόσβαση Σεπτέμβριος 2019)

[20] https://www.ionos.com/digitalguide/server/know-how/apache-kafka/ (Πρόσβαση Σεπτέμβριος 2019)

[21] https://kafka.apache.org/documentation/ (Πρόσβαση Σεπτέμβριος 2019)

[22] Sungkap Yeo Mohammad M. Hossain Jen-Cheng Huang Hsien-Hsin S. Lee, “ATAC: Ambient Temperature-Aware Capping for Power Efficient Datacenters”

[23] Lawrence Berkeley National Laboratory. Data Center Energy Benchmarking Case Study — Facility 8, 2003.

[24] G. Shamshoian, M. Blazek, P. Naughton, R. S. Seese, E. Mills, and W. Tschudi. High-tech means high-efficiency: The business case for energy management in high-tech industries. 2005.

[25] https://www.dnsstuff.com/how-to-monitor-network-traffic

[26] sFlow, "Traffic Monitoring using sFlow," Available at: http://www.sflow.org/sFlowOverview.pdf, 2003

[27] ”Cisco Systems NetFlow Services Export Version 9” Available online: https://www.ietf.org/rfc/rfc3954.txt

[28] https://www.sciencedirect.com/science/article/pii/S1876610217355674

[29] Wentai Wu, Weiwei Lin, Ligang He, “A Power Consumption Model for Cloud Servers Based on Elman Neural Network

[30] https://en.wikipedia.org/wiki/NetFlow

[31] Shafqat Ur Rehman, Wang-Cheol Song, Mingoo Kang “Network-Wide Traffic Visibility in OF@TEIN SDN Testbed using sFlow “

[32] https://github.com/softScheck/tplink-smartplug

[33] https://github.com/sdelrio/hs110-prometheus-exporter/blob/master/hs110exporter.py

[34] https://devconnected.com/how-to-setup-grafana-and-prometheus-on-linux/

[35] https://docs.influxdata.com/influxdb/v1.7/supported_protocols/prometheus

[36] https://github.com/Telefonica/prometheus-kafka-adapter

[37] https://github.com/prometheus/snmp_exporter

[38] https://github.com/prometheus/snmp_exporter/tree/master/generator

[39] https://www.cbinsights.com/research/future-of-data-centers/

[40] https://github.com/paihu/netflow_exporter

[41] https://blog.sflow.com/2019/04/prometheus-exporter.html

[42] Fikru Feleke Moges, Surafel Lemma Abebe, “Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework”, Mogs and Abebe Journal of Cloud Computing: Advances, Systems and Applications (2019) 8:2

[43] https://en.wikipedia.org/wiki/Linear_regression

[44] https://en.wikipedia.org/wiki/CPLEX

[45] https://medium.com/@rgotesman1/learning-machine-learning-part-1-linear-regression-cost-functions-and-gradient-descent-7b67d6543aff

[46] https://medium.com/@rgotesman1/learning-machine-learning-part-2-multiple-linear-regression-41d227c28dac

[47] https://blog.minitab.com/blog/adventures-in-statistics-2/how-to-interpret-regression-analysis-results-p-values-and-coefficients

[48] Josip Lorincz, Antonio Capone, Jinsong Wu, “Greener, Energy-Efficient and Sustainable Networks: State-Of-The-Art and New Trends”, Published 8 November 2019

# ΠΑΡΑΡΤΗΜΑ

## *Snmp.yml*

```
# WARNING: This file was auto-generated using snmp_exporter generator, manual changes will be lost.
apc_mon:
 walk:
 - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.5
 - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.6
 metrics:
 - name: memSensorsTemperature
 oid: 1.3.6.1.4.1.318.1.1.10.4.2.3.1.5
 type: gauge
 help: The sensor's current temperature reading - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.5
 indexes:
 - labelname: memSensorsStatusModuleNumber
 type: gauge
 - labelname: memSensorsStatusSensorNumber
 type: gauge
 - name: memSensorsHumidity
 oid: 1.3.6.1.4.1.318.1.1.10.4.2.3.1.6
 type: gauge
 help: The current humidity reading from the sensor. - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.6
 indexes:
 - labelname: memSensorsStatusModuleNumber
 type: gauge
 - labelname: memSensorsStatusSensorNumber
 type: gauge
 auth:
 community: n7uA-cN
server_aether:
 walk:
 - 1.3.6.1.2.1.25.2
 - 1.3.6.1.2.1.25.3.3
 metrics:
 - name: hrMemorySize
 oid: 1.3.6.1.2.1.25.2.2
 type: gauge
 help: The amount of physical read-write main memory, typically RAM, contained
 by the host. - 1.3.6.1.2.1.25.2.2
 - name: hrStorageIndex
 oid: 1.3.6.1.2.1.25.2.3.1.1
 type: gauge
 help: A unique value for each logical storage area contained by the host. - 1.3.6.1.2.1.25.2.3.1.1
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrStorageDescr
 oid: 1.3.6.1.2.1.25.2.3.1.3
 type: DisplayString
 help: A description of the type and instance of the storage described by this
 entry. - 1.3.6.1.2.1.25.2.3.1.3
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrStorageAllocationUnits
 oid: 1.3.6.1.2.1.25.2.3.1.4
 type: gauge
 help: The size, in bytes, of the data objects allocated from this pool - 1.3.6.1.2.1.25.2.3.1.4
```

```
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrStorageSize
 oid: 1.3.6.1.2.1.25.2.3.1.5
 type: gauge
 help: The size of the storage represented by this entry, in units of hrStorageAllocationUnits
 - 1.3.6.1.2.1.25.2.3.1.5
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrStorageUsed
 oid: 1.3.6.1.2.1.25.2.3.1.6
 type: gauge
 help: The amount of the storage represented by this entry that is allocated, in
 units of hrStorageAllocationUnits. - 1.3.6.1.2.1.25.2.3.1.6
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrStorageAllocationFailures
 oid: 1.3.6.1.2.1.25.2.3.1.7
 type: counter
 help: The number of requests for storage represented by this entry that could
 not be honored due to not enough storage - 1.3.6.1.2.1.25.2.3.1.7
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrProcessorLoad
 oid: 1.3.6.1.2.1.25.3.3.1.2
 type: gauge
 help: The average, over the last minute, of the percentage of time that this processor
 was not idle - 1.3.6.1.2.1.25.3.3.1.2
 indexes:
 - labelname: hrDeviceIndex
 type: gauge
 auth:
 community: Cn-LaB
server_dragon_1:
 get:
 - 1.3.6.1.4.1.674.10892.5.4.600.30.1.6.1.3
 metrics:
 - name: amperageProbeReading
 oid: 1.3.6.1.4.1.674.10892.5.4.600.30.1.6
 type: gauge
 help: 0600.0030.0001.0006 This attribute defines the reading for an amperage probe
 of type other than amperageProbeTypeIsDiscrete - 1.3.6.1.4.1.674.10892.5.4.600.30.1.6
 indexes:
 - labelname: amperageProbechassisIndex
 type: gauge
 - labelname: amperageProbeIndex
 type: gauge
 auth:
 community: public
server_dragon_2:
 walk:
 - 1.3.6.1.2.1.25.2
 - 1.3.6.1.2.1.25.3.3
 metrics:
 - name: hrMemorySize
 oid: 1.3.6.1.2.1.25.2.2
 type: gauge
 help: The amount of physical read-write main memory, typically RAM, contained
 by the host. - 1.3.6.1.2.1.25.2.2
```

```
- name: hrStorageIndex
oid: 1.3.6.1.2.1.25.2.3.1.1
type: gauge
help: A unique value for each logical storage area contained by the host. - 1.3.6.1.2.1.25.2.3.1.1
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageDescr
oid: 1.3.6.1.2.1.25.2.3.1.3
type: DisplayString
help: A description of the type and instance of the storage described by this
entry. - 1.3.6.1.2.1.25.2.3.1.3
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationUnits
oid: 1.3.6.1.2.1.25.2.3.1.4
type: gauge
help: The size, in bytes, of the data objects allocated from this pool - 1.3.6.1.2.1.25.2.3.1.4
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageSize
oid: 1.3.6.1.2.1.25.2.3.1.5
type: gauge
help: The size of the storage represented by this entry, in units of hrStorageAllocationUnits
- 1.3.6.1.2.1.25.2.3.1.5
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageUsed
oid: 1.3.6.1.2.1.25.2.3.1.6
type: gauge
help: The amount of the storage represented by this entry that is allocated, in
units of hrStorageAllocationUnits. - 1.3.6.1.2.1.25.2.3.1.6
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationFailures
oid: 1.3.6.1.2.1.25.2.3.1.7
type: counter
help: The number of requests for storage represented by this entry that could
not be honored due to not enough storage - 1.3.6.1.2.1.25.2.3.1.7
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrProcessorLoad
oid: 1.3.6.1.2.1.25.3.3.1.2
type: gauge
help: The average, over the last minute, of the percentage of time that this processor
was not idle - 1.3.6.1.2.1.25.3.3.1.2
indexes:
- labelname: hrDeviceIndex
type: gauge
auth:
community: CN-LaB
server_orestis:
get:
- 1.3.6.1.4.1.674.10892.5.4.600.30.1.6.1.3
metrics:
- name: amperageProbeReading
oid: 1.3.6.1.4.1.674.10892.5.4.600.30.1.6
type: gauge
```

```
help: 0600.0030.0001.0006 This attribute defines the reading for an amperage probe
of type other than amperageProbeTypeIsDiscrete - 1.3.6.1.4.1.674.10892.5.4.600.30.1.6
indexes:
- labelname: amperageProbechassisIndex
type: gauge
- labelname: amperageProbeIndex
type: gauge
auth:
community: CN-Lab
server_orestis_util:
walk:
- 1.3.6.1.2.1.25.2
- 1.3.6.1.2.1.25.3.3
metrics:
- name: hrMemorySize
oid: 1.3.6.1.2.1.25.2.2
type: gauge
help: The amount of physical read-write main memory, typically RAM, contained
by the host. - 1.3.6.1.2.1.25.2.2
- name: hrStorageIndex
oid: 1.3.6.1.2.1.25.2.3.1.1
type: gauge
help: A unique value for each logical storage area contained by the host. - 1.3.6.1.2.1.25.2.3.1.1
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageDescr
oid: 1.3.6.1.2.1.25.2.3.1.3
type: DisplayString
help: A description of the type and instance of the storage described by this
entry. - 1.3.6.1.2.1.25.2.3.1.3
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationUnits
oid: 1.3.6.1.2.1.25.2.3.1.4
type: gauge
help: The size, in bytes, of the data objects allocated from this pool - 1.3.6.1.2.1.25.2.3.1.4
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageSize
oid: 1.3.6.1.2.1.25.2.3.1.5
type: gauge
help: The size of the storage represented by this entry, in units of hrStorageAllocationUnits
- 1.3.6.1.2.1.25.2.3.1.5
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageUsed
oid: 1.3.6.1.2.1.25.2.3.1.6
type: gauge
help: The amount of the storage represented by this entry that is allocated, in
units of hrStorageAllocationUnits. - 1.3.6.1.2.1.25.2.3.1.6
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationFailures
oid: 1.3.6.1.2.1.25.2.3.1.7
type: counter
help: The number of requests for storage represented by this entry that could
not be honored due to not enough storage - 1.3.6.1.2.1.25.2.3.1.7
indexes:
```

```
- labelname: hrStorageIndex
type: gauge
- name: hrProcessorLoad
oid: 1.3.6.1.2.1.25.3.3.1.2
type: gauge
help: The average, over the last minute, of the percentage of time that this processor
was not idle - 1.3.6.1.2.1.25.3.3.1.2
indexes:
- labelname: hrDeviceIndex
type: gauge
auth:
community: CN-Lab
server_pegasus:
walk:
- 1.3.6.1.2.1.25.2
- 1.3.6.1.2.1.25.3.3
metrics:
- name: hrMemorySize
oid: 1.3.6.1.2.1.25.2.2
type: gauge
help: The amount of physical read-write main memory, typically RAM, contained
by the host. - 1.3.6.1.2.1.25.2.2
- name: hrStorageIndex
oid: 1.3.6.1.2.1.25.2.3.1.1
type: gauge
help: A unique value for each logical storage area contained by the host. - 1.3.6.1.2.1.25.2.3.1.1
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageDescr
oid: 1.3.6.1.2.1.25.2.3.1.3
type: DisplayString
help: A description of the type and instance of the storage described by this
entry. - 1.3.6.1.2.1.25.2.3.1.3
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationUnits
oid: 1.3.6.1.2.1.25.2.3.1.4
type: gauge
help: The size, in bytes, of the data objects allocated from this pool - 1.3.6.1.2.1.25.2.3.1.4
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageSize
oid: 1.3.6.1.2.1.25.2.3.1.5
type: gauge
help: The size of the storage represented by this entry, in units of hrStorageAllocationUnits
- 1.3.6.1.2.1.25.2.3.1.5
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageUsed
oid: 1.3.6.1.2.1.25.2.3.1.6
type: gauge
help: The amount of the storage represented by this entry that is allocated, in
units of hrStorageAllocationUnits. - 1.3.6.1.2.1.25.2.3.1.6
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationFailures
oid: 1.3.6.1.2.1.25.2.3.1.7
type: counter
```

```
help: The number of requests for storage represented by this entry that could
not be honored due to not enough storage - 1.3.6.1.2.1.25.2.3.1.7
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrProcessorLoad
oid: 1.3.6.1.2.1.25.3.3.1.2
type: gauge
help: The average, over the last minute, of the percentage of time that this processor
was not idle - 1.3.6.1.2.1.25.3.3.1.2
indexes:
- labelname: hrDeviceIndex
type: gauge
auth:
community: Cn-LaB
server_trillium:
walk:
- 1.3.6.1.2.1.25.2
- 1.3.6.1.2.1.25.3.3
metrics:
- name: hrMemorySize
oid: 1.3.6.1.2.1.25.2.2
type: gauge
help: The amount of physical read-write main memory, typically RAM, contained
by the host. - 1.3.6.1.2.1.25.2.2
- name: hrStorageIndex
oid: 1.3.6.1.2.1.25.2.3.1.1
type: gauge
help: A unique value for each logical storage area contained by the host. - 1.3.6.1.2.1.25.2.3.1.1
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageDescr
oid: 1.3.6.1.2.1.25.2.3.1.3
type: DisplayString
help: A description of the type and instance of the storage described by this
entry. - 1.3.6.1.2.1.25.2.3.1.3
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageAllocationUnits
oid: 1.3.6.1.2.1.25.2.3.1.4
type: gauge
help: The size, in bytes, of the data objects allocated from this pool - 1.3.6.1.2.1.25.2.3.1.4
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageSize
oid: 1.3.6.1.2.1.25.2.3.1.5
type: gauge
help: The size of the storage represented by this entry, in units of hrStorageAllocationUnits
- 1.3.6.1.2.1.25.2.3.1.5
indexes:
- labelname: hrStorageIndex
type: gauge
- name: hrStorageUsed
oid: 1.3.6.1.2.1.25.2.3.1.6
type: gauge
help: The amount of the storage represented by this entry that is allocated, in
units of hrStorageAllocationUnits. - 1.3.6.1.2.1.25.2.3.1.6
indexes:
- labelname: hrStorageIndex
type: gauge
```

```
 - name: hrStorageAllocationFailures
 oid: 1.3.6.1.2.1.25.2.3.1.7
 type: counter
 help: The number of requests for storage represented by this entry that could
 not be honored due to not enough storage - 1.3.6.1.2.1.25.2.3.1.7
 indexes:
 - labelname: hrStorageIndex
 type: gauge
 - name: hrProcessorLoad
 oid: 1.3.6.1.2.1.25.3.3.1.2
 type: gauge
 help: The average, over the last minute, of the percentage of time that this processor
 was not idle - 1.3.6.1.2.1.25.3.3.1.2
 indexes:
 - labelname: hrDeviceIndex
 type: gauge
 auth:
 community: Cn-LaB
```

## *Generator.yml*

```
modules:
 apc_mon:
 walk:
 - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.5 #memsensor_temp
 - 1.3.6.1.4.1.318.1.1.10.4.2.3.1.6 #memsensor_humid
 auth:
 community: n7uA-cN
 server_dragon:
 walk:
 - iso.3.6.1.4.1.674.10892.5.4.600.30.1.6.1.3 # Power_Watt
 auth:
 community: public
 server_orestis:
 walk:
 - iso.3.6.1.4.1.674.10892.5.4.600.30.1.6 # Power_Watt
 auth:
 community: public
```

## *Prometheus.yml*

```
# my global config
global:
 scrape_interval: 30s #15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
 evaluation_interval: 30s #15s # Evaluate rules every 15 seconds. The default is every 1 minute.
 # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
 alertmanagers:
 - static_configs:
 - targets:
 # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
 # - "first_rules.yml"
 # - "second_rules.yml"

 ############# WRITE/READ INFLUXDB/WRITE KAFKA#################################
remote_write:
 - url: "http://localhost:8086/api/v1/prom/write?db=prometheus"
 - url: "http://localhost:8080/receive"
```

```
 #remote_read:
 #- url: "http://localhost:8086/api/v1/prom/read?db=prometheus"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
 # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
 - job_name: 'prometheus'

 # metrics_path defaults to '/metrics'
 # scheme defaults to 'http'.

 static_configs:
 - targets: ['localhost:9090']

#Smartplug_IP: 147.102.40.102
 - job_name: 'AP_'
 static_configs:
 - targets: ['localhost:8110']
#Smartplug_IP: 147.102.40.188
 - job_name: 'Nuc-1'
 static_configs:
 - targets: ['localhost:8111']
#Smartplug_IP: 147.102.39.224
 - job_name: 'pegasus'
 static_configs:
 - targets: ['localhost:8114']
#Smartplug_IP: 147.102.39.227
 - job_name: 'aether'
 static_configs:
 - targets: ['localhost:8115']
#Smartplug_IP: 147.102.39.228
 - job_name: 'Rack-Fan'
 static_configs:
 - targets: ['localhost:8116']
#Smartplug_IP: 147.102.39.229
 - job_name: 'trillium'
 static_configs:
 - targets: ['localhost:8117']
#Smartplug_IP: 147.102.39.230
 - job_name: 'HPE-OF'
 static_configs:
 - targets: ['localhost:8113']
#Smartplug_IP: 147.102.39.231
 - job_name: 'alderaan'
 static_configs:
 - targets: ['localhost:8112']
 - job_name: 'netflow'
 static_configs:
 - targets: ['localhost:9191']
 metric_relabel_configs:
 - regex: 'tcpControlBits'
 action: labeldrop
 - regex: 'NetflowVersion'
 action: labeldrop
 - regex: 'TemplateID'
 action: labeldrop
 - regex: 'bgpDestinationAsNumber'
 action: labeldrop
 - regex: 'bgpSourceAsNumber'
 action: labeldrop
 - regex: 'destinationIPv4PrefixLength'
 action: labeldrop
```

```
- regex: 'classId'
action: labeldrop
- regex: 'flowDirection'
action: labeldrop
- regex: 'ingressInterface'
action: labeldrop
- regex: 'egressInterface'
action: labeldrop
- regex: 'instance'
action: labeldrop
- regex: 'ipClassOfService'
action: labeldrop
- regex: 'ipNextHopIPv4Address'
action: labeldrop
- regex: 'job'
action: labeldrop
- regex: 'protocolIdentifier'
action: labeldrop
- regex: 'samplerId'
action: labeldrop
- regex: 'sourceIPv4PrefixLength'
action: labeldrop
- job_name: 'sflow-rt'
metrics_path: /app/prometheus/scripts/export.js/dump/ALL/ALL/txt
static_configs:
- targets: ['localhost:8008']



- job_name: 'snmp_apc_mon'
static_configs:
- targets:
- 147.102.40.247

metrics_path: /snmp
params:
module: [apc_mon]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_target]
target_label: instance
- target_label: __address__
replacement: localhost:9116
##############################
- job_name: 'snmp_dragon_server_1'
static_configs:
- targets:
- 147.102.7.238

metrics_path: /snmp
params:
module: [server_dragon_1]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9990
- job_name: 'snmp_dragon_server_2'
static_configs:
- targets:
```

```
- 147.102.40.33

metrics_path: /snmp
params:
module: [server_dragon_2]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9992
- job_name: 'snmp_orestis_server'
static_configs:
- targets:
- 147.102.7.234

metrics_path: /snmp
params:
module: [server_orestis]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9991
- job_name: 'trillium_snmp'
static_configs:
- targets:
- 147.102.40.16

metrics_path: /snmp
params:
module: [server_trillium]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9993


- job_name: 'pegasus_snmp'
static_configs:
- targets:
- 147.102.40.17

metrics_path: /snmp
params:
module: [server_pegasus]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9994
- job_name: 'aether_snmp'
static_configs:
- targets:
- 147.102.40.34
```

```
metrics_path: /snmp
params:
module: [server_aether]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9995
- job_name: 'orestes_snmp_util'
static_configs:
- targets:
- 147.102.7.16

metrics_path: /snmp
params:
module: [server_orestis_util]
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_terget]
target_label: instane
- target_label: __address__
replacement: localhost:9996
```

## *Discovery.py*

```
import xml.etree.ElementTree as ET
from pyHS100 import *
import sys

file_xml = sys.argv[1]
port = int(sys.argv[2])
root = ET.parse(file_xml).getroot()

f1 = open("prometheus.yml", "a") #write to the end of prometheus.yml
for type_tag in root.findall('host/address'):
 try:
 value = type_tag.get('addr')
 plug = SmartPlug(value)
 alias=plug.alias
 f1.write("#Smartplug_IP: %s" '\n' % value)
 f1.write(" - job_name: '%s'" '\n' % alias)
 f1.write(" static_configs:" '\n')
 f1.write(" - targets: ['localhost:%d']" '\n' % port)
 port = port + 1
 except Exception:
 pass
```

## *MultRegression.py*

```
from pandas import DataFrame
from sklearn import linear_model
import statsmodels.api as sm

Power_prediction = {'Power':[],'Ucpu':[],'Umem':[]
 }
df = DataFrame(Power_prediction,columns=['Power','Ucpu','Umem'])

X = df[['Ucpu','Umem']]
Y = df['Power']

#sklearn
```

```
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

#new_ucpu=
#new_umem=
#print('Predicted_Power: \n',regr.predict([[new_ucpu,new_umem]]))

# with statsmodels
X = sm.add_constant(X.to_numpy()) # adding a constant

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

*ex_AP.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.40.102 -p 8110 -f 40
[Install]
WantedBy=multi-user.target
```

*prometheus.service*

```
[Service]
Type=simple
User=dimitris
ExecStart= /home/dimitris/prometheus-2.11.2.linux-amd64/prometheus\
 --config.file=/home/dimitris/prometheus-2.11.2.linux-amd64/prometheus.yml \
 --storage.tsdb.path="/data/prometheus" \
 --web.console.templates=/home/dimitris/prometheus-2.11.2.linux-amd64//consoles \
 --web.console.libraries=/home/dimitris/prometheus-2.11.2.linux-amd64/console_libraries \
 --web.listen-address=0.0.0.0:9090 \
 --web.enable-admin-api

Restart=always

[Install]
WantedBy=multi-user.target
```

*ex_trillium.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.229 -p 8117 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_pegasus.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.224 -p 8114 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_alderaan.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.231 -p 8112 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_aether.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.227 -p 8115 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_Rack_Fan.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.228 -p 8116 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_Nuc-1.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.40.188 -p 8111 -f 40
[Install]
WantedBy=multi-user.target
```

*ex_HPE-OF.service*

```
[Service]
User=dimitris
ExecStart=/usr/bin/python3 /home/dimitris/hs110-prometheus-exporter/hs110exporter.py -t
147.102.39.230 -p 8113 -f 40
[Install]
WantedBy=multi-user.target
```

*netflow.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/go/src/github.com/netflow_exporter/netflow_exporter
[Install]
WantedBy=multi-user.target
```

*sflow_rt.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/sflow-rt/start.sh -Dsflow.port=5600
[Install]
WantedBy=multi-user.target
```

*snmp_aether.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9995"
[Install]
WantedBy=multi-user.target
```

*snmp_apc_mon.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9116"
[Install]
WantedBy=multi-user.target
```

*snmp_dragon1.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9990"
[Install]
WantedBy=multi-user.target
```

*snmp_dragon2.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9992"
[Install]
WantedBy=multi-user.target
```

*snmp_orestes_server.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9991"
[Install]
WantedBy=multi-user.target
```

*snmp_orestes_util.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9996"
[Install]
WantedBy=multi-user.target
```

*snmp_pegasus.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9994"
[Install]
WantedBy=multi-user.target
```

*snmp_trillium.service*

```
[Service]
User=dimitris
ExecStart=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp_exporter
--config.file=/home/dimitris/snmp_exporter-0.15.0.linux-amd64/snmp.yml --web.listen-address=":9993"
[Install]
WantedBy=multi-user.target
```