ΕΘΝΙΚΟ ΔΙΚΤΥΟ ΥΠΟΔΟΜΩΝ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΡΕΥΝΑΣ – ΠΑΡΑΔΟΤΕΟ 3.1.3

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΕΞΥΠΝΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΦΟΡΤΙΩΝ ΚΕΝΤΡΩΝ ΔΕΔΟΜΕΝΩΝ

Ημερομηνία: 31-03-2020

Συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (ΕΤΠΑ) και από Εθνικούς Πόρους της Ελλάδας και της Κύπρου

Υπεύθυνος Συντάκτης Παραδοτέου:

Ομάδα Εργασίας: Αλέξανδρος Κιούσης

Δημήτρης Τσιρώνης

Θεοδώρα Μόρφη

Νίκος Αργυρόπουλος

Έκδοση: Τελική

Ημερομηνία: 31 Μαρτίου 2020

Περίληψη: Παραδοτέο 3.1.3 – Σκοπός του παρόντος παραδοτέου είναι η παρουσίαση του σχεδιασμού του συστήματος έξυπνης διαχείρισης φορτίων κέντρων δεδομένων, η παρούσιαση της υλοποίησης και η αναφορά για την πειραματική λειτουργία του συστήματος έξυπνης διαχείρισης των φορτίων, και τέλος η παρουσίας της λειτουργίας του συστήματος.

Η Πράξη “Εξοικονόμηση ενέργειας σε δημόσια Πανεπιστημιακά κτίρια με κέντρα δεδομένων - ΕΝΕΔΗ” του Προγράμματος Συνεργασίας INTERREG V-A Ελλάδα – Κύπρος 2014-2020 με κωδικό MIS 5028274 συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (ΕΤΠΑ) και από Εθνικούς πόρους της Ελλάδας και της Κύπρου.

Η Πράξη ΕΝΕΔΗ συμβάλλει στην επίτευξη των στόχων που καθορίζονται στο Πρόγραμμα Συνεργασίας. Η ακαδημαϊκή/ερευνητική κοινότητα παγκόσμια χρησιμοποιεί μεγάλα κέντρα δεδομένων που αυξάνουν το ενεργειακό αποτύπωμα. Στην Ελλάδα η ΕΔΕΤ λειτουργεί τα τρία μεγαλύτερα datacenters, αν και λαμβάνει όλα τα δυνατά μέτρα για μείωση της κατανάλωσης τους, αυτή παραμένει υψηλή. Δεδομένης της πρόβλεψης για αύξηση ζήτησης σε πόρους είναι απαραίτητο να παρθούν ειδικά μέτρα. Οι ενεργειακές ανάγκες των πανεπιστήμιων Κρήτης και Κύπρου αποτελούν σημαντικό τμήμα του λειτουργικού τους κόστους. Για τις ανάγκες τους τα πανεπιστήμια λειτουργούν κέντρα δεδομένων και επιπλέον η ΕΔΕΤ έχει εγκαταστήσει μεγάλο κέντρο δεδομένων που εξυπηρετεί τις ανάγκες δεκάδων νοσοκομείων της Ελλάδος σε κτήριο του Παν. Κρήτης στο Ηράκλειο.

Οι τρείς δημόσιοι φορείς από κοινού προτείνουν να προχωρήσουν σε παρεμβάσεις εξοικονόμησης ηλεκτρικής ενέργειας και παραγωγής ΑΠΕ, ενταγμένες σε μια ευρύτερη στρατηγική εξοικονόμησης ενέργειας και περιβαλλοντικής ευαισθητοποίησης της ακαδημαϊκής κοινότητας και του ευρύτερου δημόσιου τομέα. Η συλλογή/ανάλυση δεδομένων κατανάλωσης ενέργειας αποτελεί εξαιρετικά σημαντικό στάδιο στην λήψη ορθών αποφάσεων. Θα βοηθήσει τον στρατηγικό σχεδιασμό και την αποφυγή αποσπασματικών παρεμβάσεων για μεγιστοποίηση του καθαρού οφέλους και επίτευξη των απαραίτητων συνεργιών σε ένα ευρύτερο σύνολο των δημόσιων κτηρίων. Η γεωγραφική θέση των περιοχών ευνοεί τις υψηλές θερμοκρασίες το μεγαλύτερο μέρος του έτους κάνοντας αναγκαία την χρήση σχετικά μεγαλύτερων συστημάτων απαγωγής θερμότητας στα κέντρα δεδομένων αλλά ταυτόχρονα οι μεγάλες περίοδοι ηλιοφάνειας ευνοούν την παραγωγή ρεύματος μέσω φωτοβολταϊκών.

Η διασύνδεση των κέντρων μεταξύ τους και ο συνδυασμός των μεθόδων και μηχανισμών βελτιστοποίησης της ενεργειακής απόδοσης και μείωσης του κόστους ηλεκτρικής ενέργειας αναμένεται να έχει πολλαπλασιαστικά οφέλη. Θα μελετηθούν και θα εφαρμοστούν νεωτερικά συστήματα ενεργής διαχείρισης της κατανομής υπολογιστικού φορτίου ανάμεσα στις εγκαταστάσεις σε Ηράκλειο και Λευκωσία που θα έχουν ως αποτέλεσμα την συνολική μείωση της κατανάλωσης, και θα συντονιστεί η παραγωγή ενέργειας των φωτοβολταϊκών και μέσω της έξυπνης κατανομής φορτίου.

Περιεχόμενα

1. Σχεδιασμός μεθοδολογίας διαχείρισης φορτίων κέντρων δεδομένων

1.1 Περίληψη κεφαλαίου

Σκοπός της παρούσας αναφοράς είναι η μελέτη των προδιαγραφών για την εφαρμογή εξελιγμένων τεχνικών για την έξυπνη τοποθέτηση και κατανομή φορτίων στα κέντρα δεδομένων. Η μελέτη επικεντρώνεται στο σχεδιασμό ενός βρόγχου ανάδρασης, ο οποίος θα είναι υπεύθυνος για τη έξυπνη διαχείριση των φορτίων στα κέντρα δεδομένων.

Η βασική αρμοδιότητα του βρόχου ανάδρασης είναι η συλλογή μετρικών από το εκάστοτε κέντρο δεδομένων με σκοπό την ανάλυση των φορτίων αρχικά και στη συνέχεια την κατανομή τους στους διαθέσιμους πόρους του κέντρου δεδομένων. Μέσω της παραπάνω ανάλυσης είναι δυνατή η λήψη αποφάσεων που οδηγούν σε ενέργειες για τη βελτιστοποίηση της λειτουργίας της υποδομής.

Το κεφάλαιο επομένως, περιλαμβάνει δύο ενότητες που αφορούν:

- Την αναγνώριση των κύριων μετρικών των εικονικών μηχανών προς επίβλεψη και συλλογή καθώς και το σχεδιασμό μίας ισχυρής και κλιμακώσιμης πλατφόρμας για τη συλλογή και αποθήκευση των δεδομένων για επιπλέον ανάλυση και επεξεργασία.

- Την εφαρμογή της τεχνικής υπερδέσμευσης των υπολογιστικών πόρων (κυρίως CPU) με σκοπό την αποτελεσματικότερη χρήση της υποδομής ενώ ταυτόχρονα παρέχεται εγγύηση ελάχιστου αριθμού διαθέσιμων υπολογιστικών πόρων.

1.1.1 Δομή κεφαλαίου

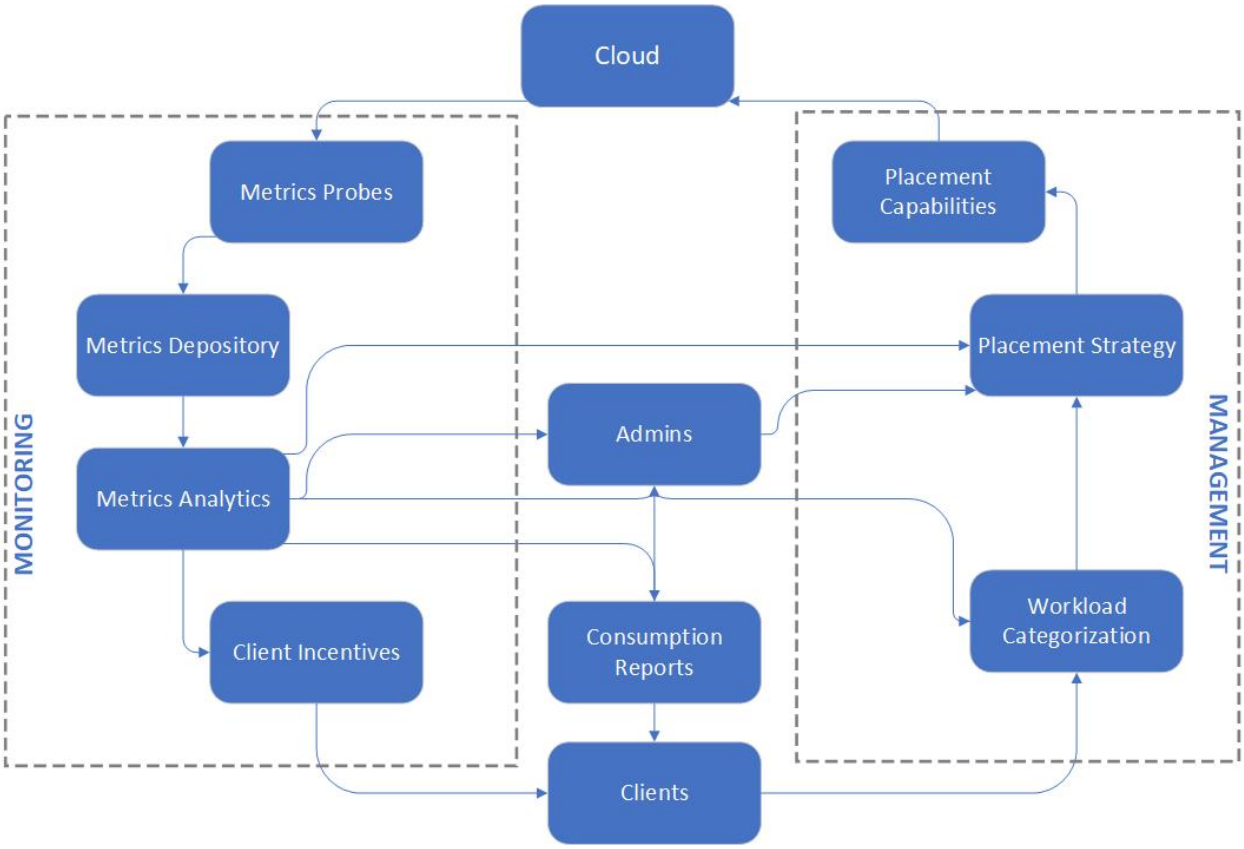
Το κεφάλαιο περιλαμβάνει τις παρακάτω τρεις ενότητες:

- Περιγραφή του βρόχου ανάδρασης και των στοιχείων του, όπως αυτά παρουσιάζονται στις ενότητες της Εποπτείας (Monitoring) και της Διαχείρισης (Management) αντίστοιχα.
- Καθορισμός απαιτήσεων και αρχιτεκτονική της πλατφόρμας ΕΝΕΔΗ (Εποπτείας (Monitoring) και Διαχείρισης (Management), όπως αποτυπώνεται στο βρόχο ανάδρασης, στο αριστερό τμήμα στην Εικόνα 1.
- Ανάλυση της μεθόδου διαχείρισης των υπολογιστικών πόρων βάσει της μελέτης για την εφαρμογή της υπερδέσμευσης των πόρων (CPU) όπως παρουσιάζεται στο δεξί τμήμα στην Εικόνα 1.

1.2.Αντικείμενο εργασίας

1.2.1.Μοντελοποίηση συστήματος έξυπνης διαχείρισης φορτίων

Η μοντελοποίηση του συστήματος έξυπνης διαχείρισης ΕΝΕΔΗ παρουσιάζεται στην Εικόνα 1.



Εικόνα 1 Βρόχος ανάδρασης ΕΝΕΔΗ. Στο πλαίσιο στα αριστερά παρουσιάζονται τα δομικά στοιχεία του συστήματος Εποπτείας (Monitoring) ενώ στο πλαίσιο στα δεξιά τα δομικά στοιχεία του συστήματος Διαχείρισης (Management) της υποδομής.

Το μοντέλο αποτελείται από δύο βασικά υποσυστήματα:

1. Το σύστημα Εποπτείας (Monitoring), αποτελείται από τα στοιχεία στο αριστερό πλαίσιο στην Εικόνα 1 και έχει ως κύριο σκοπό την εποπτεία και συλλογή πληροφορίας που σχετίζεται με την υποδομή.
2. Το σύστημα Διαχείρισης (Management) αποτελείται από τα στοιχεία στο δεξί πλαίσιο στην Εικόνα 1 και είναι υπεύθυνο για τη δρομολόγηση ενεργειών, για τη βελτιστοποίηση της κατανομής των φορτίων στο νεφοϋπολογιστικό περιβάλλον.

Στις επόμενες ενότητες περιγράφονται τα δομικά στοιχεία) που περιλαμβάνονται στα δύο βασικά υποσυστήματα του ΕΝΕΔΗ.

1.2.1.1. Metrics Probes

Τα σημεία μετρικών (Metric Probes) συλλέγουν τις απαραίτητες μετρικές που αφορούν στη χρήση των πόρων των εικονικών μηχανών και των εξυπηρετητών. Τα δεδομένα συλλέγονται βάσει των προκαθορισμένων ρυθμίσεων και στη συνέχεια αποστέλλονται στο αποθετήριο μετρικών (Metrics Depository).

1.2.1.2. Metrics Depository

Το αποθετήριο μετρικών (Metrics Depository) αποτελεί το σημείο αποθήκευσης των συλλεχθέντων δεδομένων (μετρικών) με σκοπό την περαιτέρω ανάλυσή τους.

1.2.1.3. Metrics Analytics

Το συγκεκριμένο δομικό στοιχείο αποτελεί τον πυρήνα του βρόχου ανάδρασης ΕΝΕΔΗ. Το στοιχείο για την ανάλυση των μετρικών (Metrics Analytics) εφαρμόζει ερωτήματα στο αποθετήριο μετρικών με σκοπό την ανάλυση των δεδομένων της υποδομής.

Στη συγκεκριμένη μελέτη, οι περιπτώσεις χρήσης δεν περιορίζονται στη διαχείριση φορτίων αλλά επεκτείνονται και στην εποπτεία και την καταγραφή των αποτελεσμάτων.

1.2.1.4. Consumption Reports

Η ανάλυση των δεδομένων συνοψίζεται σε αναφορές κατανάλωσης (consumption reports), οι οποίες μπορούν να αξιολογηθούν από το διαχειριστή της υπηρεσίας παρέχοντας συγκεντρωτικά αποτελέσματα που προκύπτουν από την πραγματική χρήση των υπολογιστικών πόρων.

1.2.1.5. Clients Incentives

Εφόσον οι χρήστες έχουν τον αποκλειστικό έλεγχο των εικονικών μηχανών, η νεφοϋπολογιστική υποδομή δεν έχει πρόσβαση σε επίπεδο εφαρμογών, ώστε να υπολογίσει την κατανάλωση τους. Για αυτό το λόγο πρέπει οι ίδιοι οι χρήστες να ωθούνται προς την κατεύθυνση να χρησιμοποιούν με βέλτιστο τρόπο τους διαθέσιμους υπολογιστικούς πόρους σε επίπεδο νέφους (Clients Incentives). Ένα τρόπος είναι η εφαρμογή ενός είδους χρέωσης με βάση την πραγματική χρήση των πόρων, όπως αυτά συλλέγονται από τα σημεία αναφοράς μετρικών (probes), επιβάλλοντας ποινές στις περιπτώσεις υποχρησιμοποίησης των πόρων.

1.2.1.6. Workload Categorization

Στο πλαίσιο της διαχείρισης των πόρων διακρίνουμε δύο κατηγορίες φόρτου εργασίας. Στην πρώτη κατηγορία εντάσσονται οι εργασίες οι οποίες παράγουν και αποστέλλουν άμεσα μία απάντηση σε κάποιο ερώτημα, όπως είναι μία εφαρμογή ή ένας εξυπηρετητής βάσης δεδομένων. Η δεύτερη κατηγορία περιλαμβάνει μεγάλους όγκους δεδομένων πάνω στα οποία θα πραγματοποιηθεί μεγάλης κλίμακας επεξεργασία και δεν υπάρχει η δυνατότητα άμεσης απάντησης. Παραδείγματα τέτοιων φορτίων είναι η επεξεργασία επιστημονικών συνόλων δεδομένων και η ανάλυση μεγάλου όγκου δεδομένων (big data analytics).

Η λειτουργική διαφοροποίηση στην υποδομή είναι ότι για φορτία της πρώτης κατηγορίας πρέπει να δεσμευτούν οι πόροι που απαιτούνται αμέσως ενώ στη δεύτερη κατηγορία τα φορτία μπορούν να τρέξουν με μικρότερη προτεραιότητα, δεσμεύοντας τους απαραίτητους πόρους όταν θα είναι διαθέσιμοι. Ονοματίζουμε την πρώτη κατηγορία φορτίων ως reserved και τη δεύτερη ως elastic. Με αυτή την ταξινόμηση, η υποδομή μπορεί να βελτιστοποιήσει τη λειτουργία της διασφαλίζοντας ότι όλα τα reserved φορτία μπορούν ταυτόχρονα να εξυπηρετηθούν στη συνολική τους απαίτηση, ενώ τα elastic φορτία μπορούν να δεσμεύσουν τους πόρους μόλις τα πρώτα γίνουν αδρανή.

Ένα βασικό ζήτημα στο βρόχο ανάδρασης είναι η δυνατότητα κατηγοριοποίησης των φορτίων με βασική πηγή πληροφόρησης τον ίδιο το χρήστη της υπηρεσίας. Η πληροφορία του είδους του φορτίου μπορεί μέσω της εποπτείας της χρήσης να συνεισφέρει στην εξισορρόπηση και τη δρομολόγηση μεταξύ reserved και elastic φορτίων.

1.2.1.7. Placement Strategy

Η μελέτη μας επικεντρώνεται στη τοποθέτηση των φορτίων στην υποδομή. Τα δεδομένα που συλλέγονται αναφορικά με την πραγματική χρήση των εξυπηρετητών αποτελούν το κριτήριο του αλγόριθμου κατανομής. Μία πρώτη προσέγγιση είναι η προσπάθεια ελαχιστοποίησης της τυπικής απόκλισης της μέσης κατανάλωσης πόρων στους κόμβους του νέφους. Οι συγκεκριμένες μετρικές μπορούν να χρησιμοποιηθούν για την εξοικονόμηση της ενεργειακής κατανάλωσης του κέντρου

δεδομένων, μετακινώντας τις αδρανείς εικονικές μηχανές από τους εξυπηρετητές και στην συνέχεια απενεργοποιώντας τους.

1.2.1.8 Placement Capabilities

Η ανάλυση της χρήσης των υπολογιστικών πόρων αποτελεί μόνο το ένα τμήμα του βρόχου ανάδρασης ΕΝΕΔΗ. Το δεύτερο τμήμα αποτελείται από το σύστημα ορισμού της πολιτικής για τη διαχείριση των πόρων στη λειτουργική κατάσταση της υποδομής. Βασικά κριτήρια αποτελούν οι διαφορετικές δυνατότητες τοποθέτησης καθώς και η μελέτη των περιορισμών, που σε κάθε περίπτωση τίθενται.

Η μετακίνηση, η αδρανοποίηση, η συνέχιση, ή ακόμα και η υποχρεωτική καταστροφή του φορτίου, αποτελούν διαφορετικές δυνατότητες. Η υποδομή ακόμα μπορεί να έχει διαφορετικούς κόμβους που μπορούν να φιλοξενήσουν διαφορετικούς τύπους φορτίων έχοντας διαφορετικό υλικό, όπως δίσκους SSD, μονάδες επεξεργασίας γραφικών GPU, πιο αξιόπιστο υλικό αποθήκευσης ή δικτύου. Επομένως, τα ίδια τα χαρακτηριστικά των τύπων των φορτίων περιορίζουν τις δυνατότητες τοποθέτησης τονίζοντας έτσι την ανάγκη αυτόματης κατανομής των φορτίων στην υποδομή είτε κατά την αρχική τοποθέτηση είτε κατά την μετακίνηση τους. Η αποτύπωση των δυνατοτήτων τοποθέτησης της υποδομής αποτελεί κρίσιμο παράγοντα τόσο για την επιλογή της κατάλληλης πολιτικής όσο και για την ίδια την υλοποίηση της αυτοματοποιημένης εφαρμογής.

1.2.2. Μετρικές συστήματος

Οι τέσσερις βασικές κατηγορίες πόρων σε μία εικονική μηχανή είναι οι παρακάτω:

1. Κεντρική Μονάδα Επεξεργασίας (CPU)

Η χρήση της CPU αντιπροσωπεύει το υπολογιστικό φορτίο σε μία εικονική μηχανή. Κάθε εικονική μηχανή (VM) έχει μία ή περισσότερες εικονικές κεντρικές μονάδες επεξεργασίας (vCPU), οι οποίες είναι διαθέσιμες προς χρήση. Από την πλευρά του εξυπηρετητή, στον οποίο λαμβάνει χώρα και η εποπτεία, κάθε vCPU ή κάθε ομάδα vCPU πολυπλεγμένα, τρέχει σε μία φυσική CPU. Εφόσον, οι φυσικές CPU είναι περιορισμένες σε αριθμό, καθίσταται αναγκαία η εποπτεία της χρήσης της CPU σε κάθε εικονική μηχανή σε συνδυασμό με το κατώφλι εικονοποίησης του συστήματος.

2. Μνήμη RAM

Κάθε εικονική μηχανή δεσμεύει ένα μέγιστο μέγεθος μνήμης RAM που μπορεί να χρησιμοποιήσει από τον εξυπηρετητή. Ο εξυπηρετητής ωστόσο, δεν μπορεί να έχει πρόσβαση στη χρήση της δεσμευμένης μνήμης στο επίπεδο των εικονικών μηχανών ούτε μπορεί να αξιολογήσει αν το σύνολο αυτής είναι απαραίτητο για τη χρήση που ζητείται. Επομένως, η επίβλεψη της μνήμης εξωτερικά αποτελεί μία πολύ δύσκολη εργασία και μπορεί εν μέρει να αντιμετωπιστεί με την εποπτεία των στατιστικών της μνήμης από τα στατιστικά του λειτουργικού συστήματος της ίδιας της εικονικής μηχανής.

3. Δίκτυο

Κάθε εικονική μηχανή έχει διαθέσιμες μία ή περισσότερες κάρτες δικτύου που επιτρέπουν αφενός τη σύνδεση εντός ενός ιδιωτικού δικτύου και αφετέρου τη σύνδεση στον παγκόσμιο ιστό. Η δικτυακή κίνηση των δεδομένων μία εικονικής μηχανής δρομολογείται κυρίως μέσω του εξυπηρετητή, ο οποίος με τη σειρά του έχει πεπερασμένο αριθμό διαθέσιμων πόρων. Η μέτρηση της δικτυακής χρήσης αποτελεί βασικό παράγοντα για την κατανόηση των απαιτήσεων του δικτύου, τόσο σε επίπεδο διεκπεραιωτικής ικανότητας (throughput) όσο και στον αριθμό των πακέτων που έστειλε ή έλαβε η εικονική μηχανή.

4. Δίσκος

Οι βασικές περιπτώσεις χρήσης ενός δίσκου είναι η αποθήκευση/ανάκτηση ροών δεδομένων καθώς και η εκκίνηση του λειτουργικού συστήματος κατά την έναρξη (boot). Ο τρόπος με τον οποίο οι εικονικοί δίσκοι χρησιμοποιούνται σε επίπεδο λειτουργιών εισόδου/εξόδου ανά δευτερόλεπτο αποτελεί μία σημαντική μετρική για την αξιολόγηση των απαιτήσεων χωρητικότητας δίσκου των εικονικών μηχανών.

Η συλλογή, ανάλυση και αξιολόγηση των παραπάνω μετρικών αποτελεί ένα πολύτιμο εργαλείο για την κατανόηση των απαιτήσεων των εικονικών μηχανών. Ωστόσο, σπουδαιότερη είναι η συνεισφορά των μετρικών όταν αναλύονται συνδυαστικά με βάση χρονικούς και χωρικούς συσχετισμούς.

Για την αποτελεσματικότερη εξαγωγή συμπερασμάτων που περιλαμβάνουν και τη συσχέτιση της χρήσης των μετρικών των εικονικών μηχανών σε σχέση με τον εξυπηρετητή σε επίπεδο συστήματος και κατωφλίου εικονοποίησης, η συλλογή των δεδομένων των μετρικών εκτελείται σε επίπεδο εξυπηρετητών. Στη συνέχεια, με σκοπό την αποθήκευση των δεδομένων σε επίπεδο χρονοσειρών ανά εικονική μηχανή, σημειώνουμε με την προέλευση (το αναγνωριστικό του εξυπηρετητή) τα δεδομένα. Με αυτό τον τρόπο, είναι δυνατή η σύνταξη επιτελικών αναφορών ανά εξυπηρετητή ή εικονική μηχανή αλλά και η λήψη απαντήσεων σε σύνθετα ερωτήματα (complex queries) πάνω στα δεδομένα.

Τέλος, σημαντικός παράγοντας για την ανάλυση των δεδομένων αποτελεί ο βαθμός εμβάθυνσης κατά τη συλλογή των δεδομένων, δηλαδή το πόσο λεπτομερώς συλλέγονται, αποθηκεύονται και περιγράφονται τα δεδομένα. Βέβαια, ακόμα και σε αυτό το σημείο είναι απαραίτητη η εξισορρόπηση μεταξύ εμβάθυνσης και υπολογιστικής - αποθηκευτικής ισχύος.

1.3. Πλατφόρμα ΕΝΕΔΗ

Η συγκεκριμένη ενότητα περιγράφει τις ιδιότητες της πλατφόρμας που απαιτούνται για την πλήρωση των προδιαγραφών του συστήματος Εποπτείας για την έξυπνη τοποθέτηση βάσει του βρόχου ανάδρασης ΕΝΕΔΗ.

1.3.1. Απαιτήσεις και περιπτώσεις χρήσης

Μία πλατφόρμα Εποπτείας και Διαχείρισης μίας υπηρεσίας νέφους οφείλει να επεξεργάζεται μεγάλες ποσότητες δεδομένων, ανάλογες με το μέγεθος της νεφοϋπολογιστικής υποδομής. Μία τέτοια πλατφόρμα πρέπει να συλλέγει μερικές χιλιάδες bytes δεδομένων από κάθε υποσύστημα, εικονική μηχανή και εξυπηρετητή, να τα αποθηκεύει και να τα καθιστά διαθέσιμα προς περαιτέρω ανάλυση.

Η πλατφόρμα εποπτείας ΕΝΕΔΗ πρέπει να ικανοποιεί όλες τις ακόλουθες απαιτήσεις:

- **Κλιμακωσιμότητα (Scalability):** Η πλατφόρμα πρέπει να είναι ικανή να συλλέγει δεδομένα χρήσης από χιλιάδες εικονικές μηχανές κατανεμημένες σε χιλιάδες κόμβους σε τακτά χρονικά διαστήματα με διαφορά λίγων δευτερολέπτων, επιτρέποντας έτσι τη δυνατότητα επεξεργασίας των μετρικών αυτών σε επόμενο χρόνο.

- **Ανθεκτικότητα (Robustness):** Η πλατφόρμα πρέπει να είναι ικανή να αντιμετωπίζει τόσο δικτυακές δυσλειτουργίες όσο και διακοπές λειτουργίας διάφορων υποσυστημάτων της υποδομής. Τα παραπάνω σενάρια πρέπει να μην επηρεάζουν τη συνολική λειτουργία της πλατφόρμας και να μην οδηγούν σε απώλεια των μετρικών. Επίσης, τα προσωρινά αποθηκευμένα δεδομένα θα πρέπει να μπορούν να ανακτηθούν με την επαναφορά του συστήματος σε κατάσταση ομαλής λειτουργίας.

- **Διατήρηση δεδομένων (Retention)**: Οι μετρικές θα πρέπει να είναι καταχωρισμένες στη πλατφόρμα για όσο χρόνο απαιτείται. Η συλλογή κακής ποιότητας μετρικών από μία νεφοϋπολογιστική υποδομή μπορεί να οδηγήσει σε μεγάλο, μη διαχειρίσιμο όγκο δεδομένων για τα οποία τίθενται περιορισμοί που αφορούν το λειτουργικό κόστος της υποδομής και δημιουργούν σχεδιαστικές προκλήσεις. Μία αποτελεσματική πολιτική διαχείρισης είναι η σταδιακή απλοποίηση των δεδομένων με το χρόνο. Τα πιο πρόσφατα δεδομένα θα αποθηκεύονται με μεγάλη λεπτομέρεια, ώστε να μπορούν να αναλυθούν και να επεξεργαστούν πλήρως, ενώ παλαιότερες μετρήσεις θα υπόκεινται σε δειγματοληψία ώστε μόνο η κρίσιμη πληροφορία να συγκρατείται για μελλοντική αξιολόγηση.

- **Υποβολή Ερωτημάτων (Querying)**: Μία διεπαφή προγραμματισμού εφαρμογών (API) πρέπει να παρέχεται στους διαχειριστές και τους χρήστες της υποδομής ώστε να είναι δυνατή η ανάκτηση των μετρικών για επιπλέον ανάλυση και επεξεργασία, διασφαλίζοντας ταυτόχρονα την ορθή λειτουργία χωρίς διακοπές της υπηρεσίας.

- **Συνδυαστικά Ερωτήματα (Aggregations)**: Η πλατφόρμα ερωτημάτων επιτρέπει στους χρήστες τη δημιουργία συνδυαστικών ερωτημάτων που οδηγούν σε υψηλής ποιότητας επιτελικές αναφορές. Τα ερωτήματα τα οποία παρουσιάζονται συχνά ή αυτά τα οποία ο διαχειριστής της υποδομής θεωρεί ότι παρέχουν ουσιώδη πληροφορία, μπορούν να προϋπολογιστούν ώστε να μειώσουν τον υπολογιστικό φόρτο και το χρόνο απόκρισης κατά τη διεπαφή με το χρήστη.

- **Χρήστες (Users)**: Δεδομένης της συλλογής προσωπικών δεδομένων των χρηστών, είναι απαραίτητη η ενσωμάτωση δικλείδων ασφαλείας με τη διαβαθμισμένη πρόσβαση στους χρήστες τόσο σε επίπεδο λογαριασμών όσο και στο επίπεδο των δεδομένων με τα οποία μπορούν να αλληλεπιδρούν.

Το επίπεδο συνεισφοράς των παραπάνω προδιαγραφών διαφέρει ανάλογα με το σενάριο χρήσης που εξετάζουμε. Ένα μεγάλο εύρος τέτοιων σεναρίων που έχουν εποπτευθεί κατά τη χρήση του συστήματος Ωκεανός (~okeanos), παρέχεται στους χρήστες και τους διαχειριστές ως μία αρχική βάση για τη λήψη επιχειρησιακών αποφάσεων.

1.3.1.1. Αρχιτεκτονική της υπηρεσίας Ωκεανός (~okeanos)

Η νεφοϋπολογιστική υπηρεσία ~okeanos είναι μία εγκατάσταση της στοίβας λογισμικού Synnefo της ΕΔΕΤ, ώστε να διαχειρίζεται και να προσφέρει υπολογιστικούς πόρους στην ακαδημαϊκή και ερευνητική κοινότητα στην Ελλάδα και πανευρωπαϊκά. Η τεχνολογία στην οποία βασίζεται η υπηρεσία παρουσιάζεται συνοπτικά στις παρακάτω ενότητες.

Λογισμικό Google Ganeti και Ceph RADOS

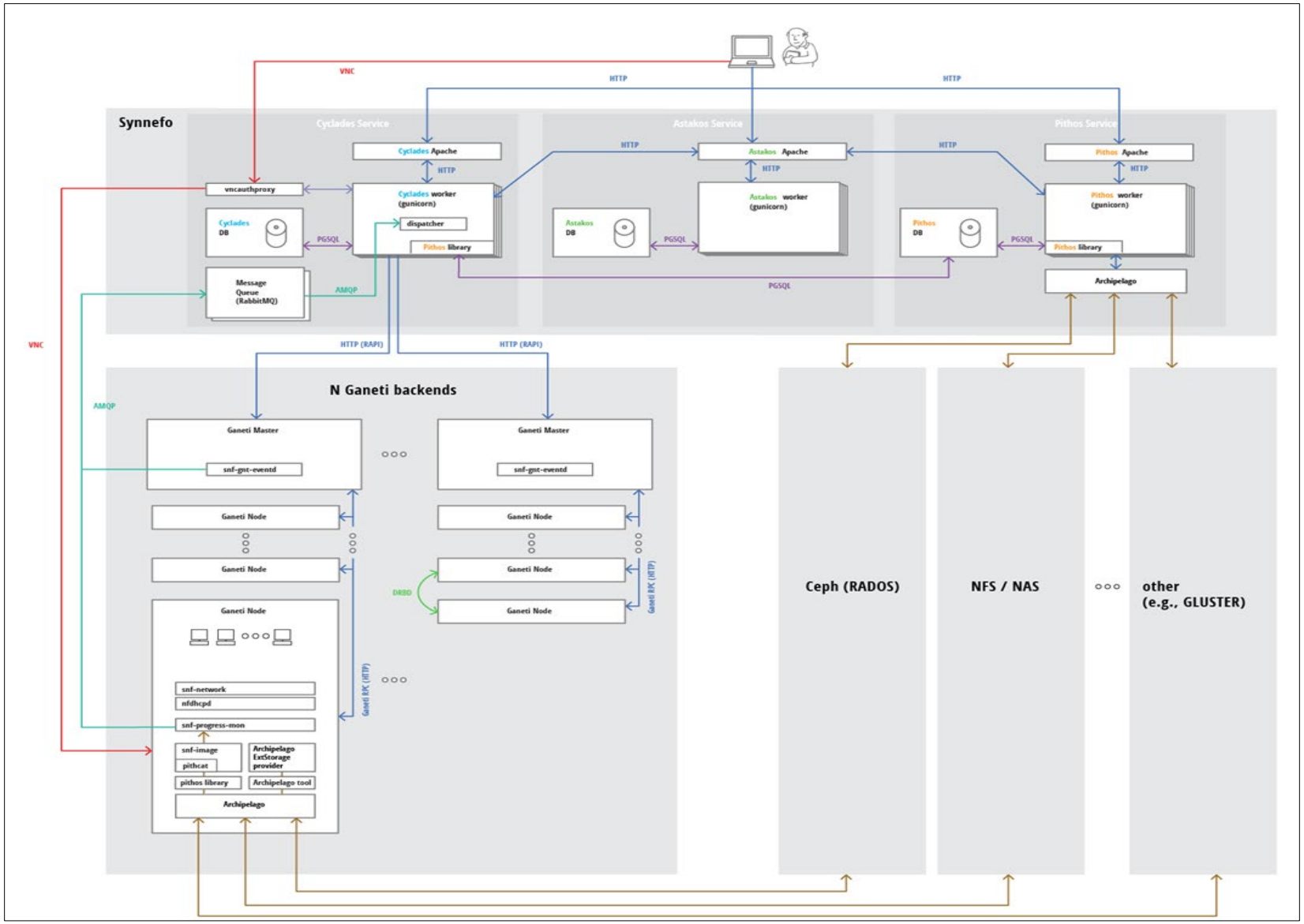
Το Ganeti της Google είναι λογισμικό διαχείρισης εικονικών πόρων σε συστάδες (clusters). Είναι ανοιχτού κώδικα με άδεια GPL v2, έχει αναπτυχθεί σε Haskell και Python και λειτουργεί πάνω από κατάλληλο λογισμικό hypervisor XEN ή KVM. Κάθε μονάδα hypervisor συνιστά έναν κόμβο Ganeti. Οι κόμβοι αυτοί ομαδοποιούνται και συντονίζονται από αρχικοποιήσεις (instances) της εφαρμογής Ganeti Master που εκκινείται στους διαχειριστικούς φυσικούς πόρους. Η επικοινωνία των κόμβων με τους masters πραγματοποιείται με κλήσεις απομακρυσμένων διαδικασιών (RPC) μέσω HTTP. Στον Ωκεανό, οι φυσικοί κόμβοι εκκινούν λειτουργικό σύστημα Debian Wheezy με λογισμικό hypervisor KVM.

Το Ceph RADOS είναι ένα κατανεμημένο σύστημα αποθήκευσης που επιτρέπει την ολοκληρωμένη διαχείριση κοινών (commodity) αποθηκευτικών πόρων, ακόμα κι όταν βρίσκονται σε διαφορετικά ικριώματα. Οι δυνατότητες του RADOS περιλαμβάνουν τη διαχείριση αντικειμένων (objects) και blocks, έλεγχο (monitoring), καθώς και μία RESTful διεπαφή. Στον Ωκεανό χρησιμοποιείται το DRBD, ένας κατανεμημένος μηχανισμός για διαχείριση δεδομένων ως blocks. Οι πόροι του DRBD μπορούν να χρησιμοποιηθούν απ' ευθείας από το Ganeti (volumes) και τον Pithos+ (object storage) ή να παρεμβληθεί ένα ενδιάμεσο επίπεδο που ελέγχεται από το λογισμικό Archipelago. Στον Ωκεανό έχει επιλεχθεί η δεύτερη λύση.

Στους διαχειριστικούς φυσικούς πόρους που αναφέρθηκαν πριν, αρχικοποιούνται οι διάφορες υπηρεσίες που προσφέρονται από τον ~okeanos: Cyclades, Archipelago, Pithos+ που αξιοποιούν τους εικονικούς πόρους, όπως αυτοί προσφέρονται από το Ganeti και το Ceph (RADOS). Οι Cyclades διαχειρίζονται πολλαπλά συμπλέγματα Ganeti.

Το λογισμικό Archipelago ενοποιεί το φυσικό αποθηκευτικό χώρο των υπηρεσιών του νέφους. Έτσι, οι σχετικοί αποθηκευτικοί πόροι χρησιμοποιούνται τόσο για τις υπολογιστικές υπηρεσίες (Cyclades Volumes), όσο και για τις υπηρεσίες δεδομένων χρηστών (Pithos+). Το Archipelago μπορεί να λειτουργήσει πάνω από διαφορετικές λύσεις κατανεμημένης αποθήκευσης, χωρίς αυτό να επηρεάσει το υπόλοιπο σύστημα.

Η αρχιτεκτονική του συστήματος ~okeanos παρουσιάζεται στο παρακάτω σχήμα:



1.3.1.2. Δρομολόγηση (Scheduling)

Η κύρια συνεισφορά της πλατφόρμας ΕΝΕΔΗ είναι η έξυπνη τοποθέτηση των φορτίων. Οι μετρικές που έχουν συλλεχθεί μπορούν να βελτιώσουν δυναμικά το αλγόριθμο δρομολόγησης των εικονικών μηχανών και να οδηγήσουν στην εξισορρόπηση του φορτίου. Κατά τη δημιουργία μίας εικονικής μηχανής η δρομολόγηση πρέπει να βασίζεται στην πραγματική χρήση και στα πρότυπα κατανάλωσης και όχι στους δεσμευμένους πόρους. Οι διαχειριστές πρέπει δυναμικά να αξιολογούν την πληροφορία των μετρικών χρήσης και να επανατοποθετούν τις εικονικές μηχανές βάσει του φόρτου μέσα στη συστάδα μηχανών (cluster). Η εξισορρόπηση μπορεί να πραγματοποιηθεί είτε χειροκίνητα από τον ίδιο το διαχειριστή, είτε αυτοματοποιημένα εφόσον δεν τηρούνται συγκεκριμένα κατώφλια μετρικών ή ακόμα και με τη χρήση προκαθορισμένων ρυθμίσεων.

Το συγκεκριμένο σενάριο χρήσης δεν απαιτεί μεγάλη περίοδο διατήρησης των δεδομένων, ίσως κάποιων ημερών και τα ερωτήματα που θα πραγματοποιηθούν μπορεί να είναι προκαθορισμένα και προϋπολογισμένα.

1.3.1.3. Διαχειριστές (Admins)

Οι διαχειριστές μπορούν να επωφεληθούν από τις μετρικές και να λαμβάνουν αποτελεσματικότερες αποφάσεις για την πολιτική διαχείρισης της συνολικής υποδομής. Εξετάζοντας τα πραγματικά δεδομένα χρήσης, μπορούν να αποφασίζουν για την κατανομή των πόρων σε νέα έργα καθώς και την ανάκτηση υποχρησιμοποιούμενων από ήδη υπάρχοντα. Επίσης, υπάρχει η δυνατότητα εντοπισμού ασυνήθιστων μετρήσεων και η άμεση αντιμετώπιση τους, περιορίζοντας έτσι το εύρος επιρροής της αποτυχίας και στη συνέχεια η δυνατότητα διερεύνησης της αιτίας του περιστατικού.

Στο συγκεκριμένο σενάριο, συνήθως χρησιμοποιούνται τα πιο πρόσφατα δεδομένα, επομένως ο χρόνος διατήρησης είναι μικρός και τα ιστορικά δεδομένα συγκρατούνται με μικρότερη ακρίβεια. Τα ερωτήματα επίσης γίνονται σε όχι τόσο συχνή βάση και ο συνδυασμός των ερωτημάτων ποικίλει χωρίς ωστόσο να απουσιάζουν ερωτήματα τα οποία επαναλαμβάνονται και μπορεί να έχουν προϋπολογιστεί και βελτιστοποιηθεί.

1.3.1.4 Γραφική Αναπαράσταση

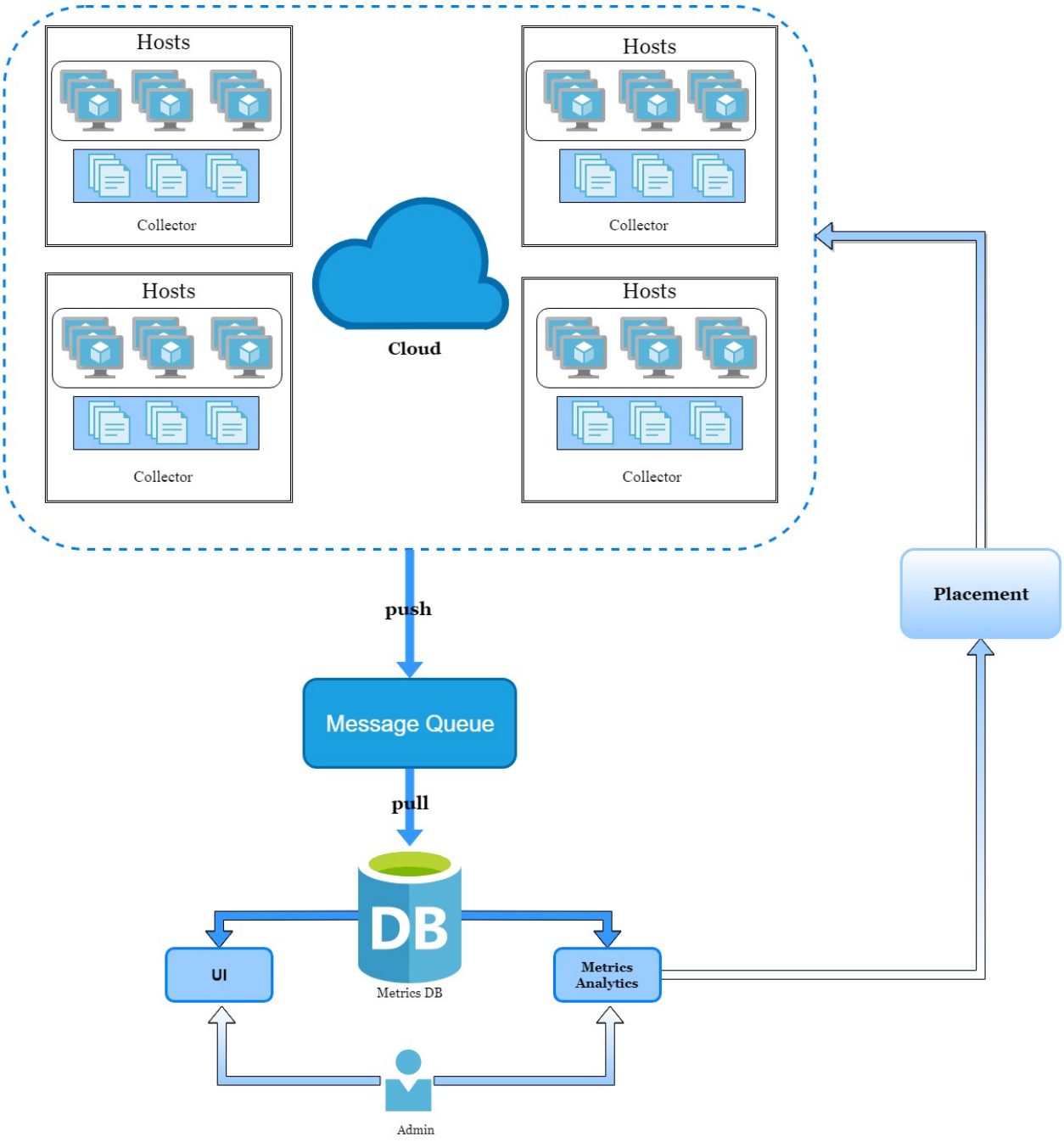
Η οπτικοποίηση των μετρικών σε διαγράμματα για τους διαχειριστές και τους χρήστες της υποδομής αποτελεί βασικό εργαλείο για την εποπτεία της ορθής λειτουργίας της υποδομής, της αξιολόγησης της κίνησης των δεδομένων, της κατανόησης των πραγματικών αναγκών καθώς και την αντιμετώπιση και διερεύνηση σφαλμάτων. Η πλατφόρμα μετρικών ΕΝΕΔΗ προσφέρει πλήθος διαθέσιμων γραφημάτων από τη χρήση διάφορων υπολογιστικών πόρων της υποδομής.

Για τη συγκεκριμένη λειτουργικότητα είναι απαραίτητο να υπάρχουν διαθέσιμα τόσο τα πρόσφατα δεδομένα όσο και ιστορικά δεδομένα σε επαρκή προς ανάλυση μορφή για την καλύτερη εμπειρία του χρήστη, ο οποίος εξετάζει τα γραφήματα για κάποιο χρονικό διάστημα. Στην περίπτωση αυτή υπάρχει η δυνατότητα προετοιμασίας και υπολογισμού σύνθετων ερωτημάτων για τη διευκόλυνση της λειτουργίας. Τέλος, παρουσιάζεται η ανάγκη προσαρμογής του συστήματος για την αποτροπή προβολής των δεδομένων από άλλους χρήστες της υπηρεσίας.

1.3.1.5. Χρέωση (Billing)

Η γνώση των πραγματικών πόρων που χρησιμοποιούνται από τις εικονικές μηχανές επιτρέπουν στον πάροχο της νεφοϋπολογιστικής υποδομής να βελτιώσει τις πολιτικές χρεώσεις. Αξιοποιώντας τις μετρικές μπορεί να δεσμεύει πόρους βάσει των πραγματικών δεδομένων και όχι όπως τυπικά συμβαίνει με τη χρήση πλασματικών ονομαστικών δεδομένων. Εφικτή θα ήταν επίσης, η επιβολή ποινών σε χρήστες οι οποίοι δεσμεύουν πόρους οι οποίοι τελικά παραμένουν σε αδράνεια, στερώντας την αξιοποίησή τους από άλλους χρήστες.

1.3.2. Αρχιτεκτονική Συστήματος



Εικόνα 2 Αρχιτεκτονική συστήματος πλατφόρμας συλλογής μετρικών ΕΝΕΔΗ

Στη συγκεκριμένη ενότητα παρουσιάζουμε την αρχιτεκτονική της πλατφόρμας συλλογής μετρικών η οποία είναι σύμφωνη με τις προδιαγραφές της παραγράφου 3.1. Απαιτήσεις και περιπτώσεις χρήσης.

Τα τρία συστατικά στοιχεία της αρχιτεκτονικής είναι:

- Οι συλλέκτες μετρικών (metrics collectors)
- Η ενδιάμεση ουρά μηνυμάτων (message queue)
- Η βάση δεδομένων μετρικών για μελλοντική ανάλυση (metrics database)

Κάθε στοιχείο της αρχιτεκτονικής αποτελείται από επιμέρους υποσυστήματα που επιτελούν συγκεκριμένες λειτουργικότητες. Όλα τα στοιχεία συνδέονται κατάλληλα και σχηματίζουν μία ροή δεδομένων που συμβάλλει στη συλλογή και αποθήκευσή τους. Ο συνδυασμός όλων των παραπάνω αποτελεί την υποδομή μετρικών του ΕΝΕΔΗ όπως φαίνεται στην Εικόνα 2.

1.3.2.1. Συλλέκτης (Collector)

Ο συλλέκτης είναι ένα λογισμικό deamon το οποίο είναι υπεύθυνο για τη συλλογή των μετρικών από τους εξυπηρετητές και την προώθησή τους στην ενδιάμεση ουρά μηνυμάτων (message queue). Πλήθος μετρικών συλλέγονται σε προκαθορισμένα χρονικά διαστήματα και αρχικά αποθηκεύονται σε μία εσωτερική ουρά για να διασφαλιστεί ότι δεν θα χαθούν δεδομένα σε περίπτωση σφάλματος στην ενδιάμεση ουρά μηνυμάτων (message queue) όπου και προωθούνται (push) στη συνέχεια.

Τα βασικά χαρακτηριστικά του συλλέκτη είναι τα παρακάτω:

- **Επεκτασιμότητα (Extensibility)**: Η δυνατότητα προσθήκης plugins εισόδου/εξόδου με σκοπό την δυναμική βελτιστοποίηση των αναγκών εποπτείας.

- **Ανθεκτικότητα (Robustness)**: Η ομαλή λειτουργία του συστήματος προϋποθέτει την ανεξαρτησία των plugins που χρησιμοποιούνται ώστε σε περίπτωση σφάλματος ενός, τα υπόλοιπα να είναι λειτουργικά έως την επαναφορά του.

- **Διατήρηση (Persistence)**: Σε περίπτωση σφάλματος, ο συλλέκτης πρέπει να χάσει όσο το δυνατόν λιγότερα δεδομένα. Για παράδειγμα, σε περίπτωση που η ενδιάμεση ουρά αποθήκευσης δεν λειτουργεί (transient failure), ο συλλέκτης πρέπει να διατηρήσει εσωτερικά το μέγιστο δυνατό αριθμό δεδομένων.

- **Οικονομία πόρων (Lightweight)**: Ο συλλέκτης πρέπει να χρησιμοποιεί τους λιγότερους δυνατούς πόρους του εξυπηρετητή ώστε να μην επηρεάζει τη λειτουργία των εικονικών μηχανών.

1.3.2.2. Ενδιάμεση ουρά μηνυμάτων (message queue)

Η πλατφόρμα εποπτείας χειρίζεται χιλιάδες μετρικές ανά δευτερόλεπτο, όπου κάθε μία μπορεί να έχει μέγεθος κάποιων εκατοντάδων bytes. Με βάση τις πραγματικές συνθήκες λειτουργίας η πλατφόρμα πρέπει να είναι ασφαλής και αξιόπιστη, στοιχεία που επιτυγχάνονται με την προσθήκη ενός ενδιάμεσου προσωρινού σταδίου αποθήκευσης, της ενδιάμεσης ουράς μηνυμάτων, όπου διοχετεύονται οι μετρικές πριν αποθηκευτούν στη μόνιμη βάση δεδομένων.

Στην ενδιάμεση ουρά μηνυμάτων αποτυπώνονται οι διάφορες πτυχές της αρχιτεκτονικής μας που αποτελείται από το μέρος της συλλογής και το μέρος αποθήκευσης και επεξεργασίας. Τα πλεονεκτήματα της συγκεκριμένης αρχιτεκτονικής παρουσιάζονται παρακάτω:

- Το τμήμα συλλογής (collection part) αλληλεπιδρά με την ενδιάμεση ουρά μηνυμάτων που ενεργεί ως ενδιάμεσο μέσο αποθήκευσης και διευκολύνει τη βάση δεδομένων μόνιμης αποθήκευσης, η οποία μπορεί να έχει αυξημένες κλήσεις όπως για παράδειγμα σε κάποιο σύνθετο ερώτημα ή ως εναλλακτική αποθήκευση σε περίπτωση που η βάση δεδομένων δεν λειτουργεί.

- Δημιουργείται μία ροή δεδομένων και πολλά ερωτήματα μπορούν να απαντηθούν σε πραγματικό χρόνο από τα δεδομένα της ενδιάμεσης ουράς μηνυμάτων αποφορτίζοντας τη μόνιμη βάση δεδομένων και μειώνοντας το χρόνο απόκρισης στο χρήστη.

Η ουρά ενδιάμεσης αποθήκευσης είναι μία δομή ουράς FIFO (First in – First out) δηλαδή τα δεδομένα που καταχωρίζονται πρώτα είναι τα δεδομένα που εξάγονται πρώτα. Οι μετρικές αποθηκεύονται σε συνεχόμενες θέσεις μνήμης γεγονός που επιταχύνει την ανάγνωση συνεχόμενων δεδομένων αλλά παρουσιάζει μικρότερες επιδόσεις στην ανάγνωση δεδομένων από τυχαίες θέσεις.

Τα χαρακτηριστικά της ενδιάμεσης ουράς μηνυμάτων είναι:

- **Διατήρηση (Persistence)**: Η δομή αποθηκεύει τα δεδομένα αξιόπιστα και στη συνέχεια γνωστοποιεί στο συλλέκτη την παραλαβή τους.
- **Αποτελεσματικότητα (Efficiency)**: Η δομή έχει την ικανότητα χειρισμού μεγάλου όγκου δεδομένων χωρίς καθυστερήσεις ενώ ταυτόχρονα ικανοποιούνται όλες οι απαιτήσεις του συστήματος.
- **Υψηλή διαθεσιμότητα (High availability)**: Ο κομβικός ρόλος της ενδιάμεσης ουράς αποθήκευσης καθιστά βασική προϋπόθεση στο σχεδιασμό της, τη μεγαλύτερη δυνατή αντοχή σε σφάλματα.
- **Κλιμακωσιμότητα (Scalability)**: Η ουρά πρέπει να μπορεί να λαμβάνει και να αποθηκεύει δεδομένα ακόμα και σε περιπτώσεις που η υποδομή μεγαλώνει και περισσότερες εικονικές μηχανές αλληλεπιδρούν με αυτή.

1.3.2.3. Βάση δεδομένων μετρικών

Στη βάση δεδομένων μετρικών τα δεδομένα αποθηκεύονται μόνιμα για μελλοντική χρήση, ανάλυση και επεξεργασία. Μία κοινή βάση δεδομένων δεν θα ήταν κατάλληλη για την αποθήκευση χρονοσειρών και για το λόγο αυτό προτείνεται να χρησιμοποιηθεί non-SQL βάση influxDB, που είναι κατάλληλη για την αποθήκευση τέτοιων δεδομένων.

Οι προδιαγραφές της βάσης δεδομένων είναι οι παρακάτω:

- **Χρονοσειρές (Time Series)**: Όλες οι μετρικές περιλαμβάνουν χρονικούς συσχετισμούς με αποτέλεσμα η χρονική ετικέτα να είναι απαραίτητη για την ανάλυση των δεδομένων.

- **Ανθεκτικότητα (Robustness)**: Η απώλεια δεδομένων απαγορεύεται και η βάση πρέπει να διασφαλίζει την ακεραιότητα των δεδομένων.

- **Διατήρηση (Retention)**: Τα ιστορικά δεδομένα θα πρέπει να διατηρούνται μέχρι κάποιο χρονικό προκαθορισμένο κατώφλι.

- **Υποδειγματοληψία (Downsampling)**: Οι περιορισμοί στο υλικό καθιστούν απαραίτητη τη δειγματοληψία ιστορικών δεδομένων με χρήση είτε ενσωματωμένων μηχανισμών είτε εξωτερικών υπηρεσιών.

- **Ερωτήματα (Querying)**: Η εκφραστικότητα της γλώσσας της βάσης δεδομένων πρέπει να είναι ικανή για σύνθετα ερωτήματα επί των δεδομένων.

- **Κλιμακωσιμότητα (Scalability)**: Η βάση δεδομένων πρέπει να χειρίζεται και να απαντά αποτελεσματικά σε ερωτήματα που αφορούν δεδομένα μεγάλης κλίμακας.

1.3.3. Ανάλυση μετρικών

Βασικός στόχος του βρόχου ανάδρασης είναι, όπως παρουσιάστηκε στις προηγούμενες ενότητες, η επεξεργασία της πληροφορίας που συλλέχθηκε από το σύστημα και η λήψη κατάλληλων αποφάσεων βάσει της πραγματικής κατάστασης της νεφοϋπολογιστικής υποδομής.

Η ανάλυση των μετρικών συμβαίνει στο αριστερό πλαίσιο στο βρόχο ανάδρασης στην Εικόνα 1 και αποτελείται από τα παρακάτω στοιχεία:

- Ερωτήματα άμεσα στη βάση δεδομένων μετρικών και ανάκτηση συνόλου δεδομένων.
- Ανάλυση των δεδομένων παράλληλα ενώ βρίσκονται στην ενδιάμεση ουρά αποθήκευσης.

Οι δύο παραπάνω διαδικασίες δίνουν τη δυνατότητα πρόσβασης στα δεδομένα. Οι ερωτήσεις στη βάση δεδομένων και η ανάκτηση των δεδομένων είναι αποτελεσματική εκτός από τις περιπτώσεις που τα ερωτήματα είναι συχνά και απαιτούν αυξημένο όγκο δεδομένων, περιπτώσεις στις οποίες η αμεσότητα των ερωτήσεων στην ενδιάμεση ουρά αποθήκευσης βελτιστοποιεί σημαντικά το χρόνο απόκρισης.

Η επιλογή του τρόπου με τον οποίο θα αποκτήσουμε πρόσβαση στα δεδομένα εξαρτάται από το σενάριο χρήσης που εξετάζουμε εκείνη τη στιγμή και απαιτεί την αξιολόγηση επιπλέον παραμέτρων όπως η πολυπλοκότητα υλοποίησης, τα κατώφλια του συστήματος και η αποθηκευτική δυνατότητα.

Ένα πιθανό σενάριο χρήσης με ερωτήματα στη βάση δεδομένων είναι η περιοδική εξισορρόπηση των μηχανών της ίδιας συστάδας. Η οπτικοποίηση μέσω γραφημάτων των μετρικών μεγάλου όγκου δεδομένων, της κατάστασης του συστήματος και της επίδοσης των εικονικών μηχανών αποτελούν τους υπό αξιολόγηση συντελεστές. Αντίστοιχα, η ανάλυση των χρονοσειρών σε πραγματικό χρόνο απαιτείται στις περιπτώσεις που υπάρχει η ανάγκη εντοπισμού σφαλμάτων, τα οποία θα γίνονται αντιληπτά από ερωτήματα στην ενδιάμεση ουρά αποθήκευσης και θα οδηγούν σε διορθωτικές ενέργειες σε περίπτωση υπέρβασης των προκαθορισμένων κατωφλίων χρήσης.

1.4. Διαχείριση υπολογιστικών πόρων (μελέτη υπερδέσμευσης κεντρικής μονάδας επεξεργασίας CPU)

1.4.1. Ορισμός προβλήματος

Η πλατφόρμα ΕΝΕΔΗ έχει ως κύριο στόχο την εξοικονόμηση ενέργειας μέσω της αποτελεσματικής διαχείρισης της νεφοϋπολογιστικής υποδομής, αυξάνοντας τις δυνατότητες χρήσης σε διαφορετικούς τύπους φορτίων ενώ ταυτόχρονα διατηρεί της ποιότητα της υπηρεσίας. Στη συγκεκριμένη ενότητα γίνεται ανάλυση του τμήματος διαχείρισης των πόρων στην υποδομή όπως περιγράφεται στην Εικόνα 1.

Η υπερδέσμευση πόρων αποτελεί τυπική τακτική στις περιπτώσεις διαμοιρασμού πεπερασμένων πόρων μεταξύ πολλών χρηστών. Με αυτό τον τρόπο επιτυγχάνεται καλύτερη χρήση των πόρων και αυξημένος αριθμός χρηστών σε αντίθεση με πολιτικές αποκλειστικής δέσμευσης πόρων από συγκεκριμένους χρήστες, γεγονός που μειώνει τη διαθεσιμότητα τους. Η συγκεκριμένη απόφαση βασίζεται στην αρχή ότι οι περισσότεροι χρήστες δεν χρειάζονται όλους του δεσμευμένους πόρους αλλά ακόμα και όταν τους χρειάζονται δεν είναι ταυτόχρονα.

Η τεχνική της υπερδέσμευσης μπορεί να εφαρμοστεί σε διάφορους τύπους πόρων ωστόσο είναι αποδεκτό να ενσωματώνεται σε αυτούς που δεν προσεγγίζουν και τείνουν να ξεπεράσουν τα όρια ορθής λειτουργίας και τελικά να επηρεάσουν τη λειτουργικότητα της υποδομής. Η μελέτη επικεντρώνεται στη CPU και την παρουσίαση των μοντέλων υπερδέσμευσης αυτής.

Όπως αναφέρθηκε και προηγουμένως, υπάρχει μεγάλη ανάγκη για υπολογιστική ισχύ, ωστόσο ο περιορισμένος αριθμός φυσικών CPU (pCPU) καθιστά απαραίτητη τη χρησιμοποίηση εικονικών κεντρικών μονάδων επεξεργασίας (vCPU). Συγκέντρωση ακόμα των φορτίων του χρήστη στην ίδια εικονική μηχανή δεν είναι εφικτή διότι ο χρήστης χρειάζεται την απομόνωση των διαφορετικών φορτίων και επομένως η προσαρμογή εικονικών μηχανών με πολλές εικονικές vCPU αποτελεί μία αποτελεσματική τεχνική διαχείρισης της υπολογιστικής ισχύος.

Η ανάγκη για βελτίωση της συνολικής CPU με την ενσωμάτωση των φορτίων του χρήστη στον εξυπηρετητή μέσω μίας εικονικής μηχανής πρέπει να πραγματοποιείται διασφαλίζοντας το επιθυμητό επίπεδο ποιότητας της υπηρεσίας. Η δρομολόγηση πολλαπλών vCPU στην ίδια pCPU μπορεί να προκαλέσει υποβάθμιση της επίδοσης αν οι vCPU απαιτήσουν περισσότερη χωρητικότητα από αυτή που είναι διαθέσιμη στη pCPU. Επίσης, η υποβάθμιση είναι μη ντετερμινιστική με αποτέλεσμα να μην υπάρχει η δυνατότητα διασφάλισης της υπηρεσίας στους «σημαντικούς» χρήστες της υπηρεσίας.

1.4.2. Υπολογιστικές μονάδες (Compute Units - CU)

Στοχεύοντας στην επίλυση των παραπάνω προβλημάτων, εισάγουμε την έννοια των υπολογιστικών μονάδων (CU), μίας αφαιρετικής έννοιας στο πλαίσιο των vCPU, για τη δημιουργία ξεχωριστών κλάσεων vCPU και τη διαβάθμισή τους ανάλογα με τις υπολογιστικές τους ανάγκες.

Η προτεινόμενη λύση βασίζεται στις εξής παραδοχές :

- Διαφορετικοί τύποι χρηστών έχουν διαφορετικές ανάγκες και διαφορετικές απαιτήσεις σε CPU.
- Οι περισσότεροι χρήστες δε χρησιμοποιούν τις δεσμευμένες τους CPU διαρκώς, με αποτέλεσμα να παρουσιάζονται περιπτώσεις αδρανούς υλικού (hardware).

Η ιδέα για τη δημιουργία των CU προέρχεται από την ανάγκη διαχωρισμού των vCPU σε κλάσεις με διαφορετικές εγγυήσεις επίδοσης. Υψηλότερης κλάσης vCPU θα έχουν υψηλότερη προτεραιότητα στις περιπτώσεις, που οι πόροι είναι περιορισμένοι. Επιπλέον, με αυτό το σχεδιαστικό σχήμα, οι διαθέσιμοι κύκλοι των CPU θα χρησιμοποιούνται από χαμηλότερης κλάσης vCPU όταν θα υπάρχει απαίτηση για την αξιοποίηση τους.

Οι υπολογιστικές μονάδες CU μπορούν είτε να θεωρηθούν ως ένα ανάλογο της φυσικής pCPU στην οποία η εικονική μηχανή τρέχει, είτε ως ανάλογο μιας τυπικής CPU, με διαφορετικά δεσμευμένα ποσοστά της CPU στην οποία η εικονική μηχανή τρέχει. Το πλεονέκτημα της πρώτης μεθόδου είναι η απλότητα της υλοποίησης δεδομένου ότι κάθε vCPU είναι ποσοστό της pCPU σε ένα ετερογενές περιβάλλον εξυπηρετητών ωστόσο τα CU δεν αντιπροσωπεύουν άμεσα δείκτη υπολογιστικής ισχύος. Στη δεύτερη μέθοδο τα CU απεικονίζονται άμεσα σε έναν δείκτη υπολογιστικής ισχύος, ωστόσο υπάρχει υψηλή πολυπλοκότητα στην υλοποίηση του συστήματος αφού είναι αναγκαίος ο υπολογισμός και το ταίριασμα της υπολογιστικής ισχύος μεταξύ διαφορετικών pCPU.

Οι υπολογιστικές μονάδες CU αντιστοιχίζονται σε κάθε vCPU και εγγυώνται ότι σε κάθε μία εκχωρείται ένα ελάχιστο μέρος της CPU ισχύος. Ενώ το κατώφλι της CPU είναι εγγυημένο, οι αδρανείς πόροι CPU του εξυπηρετητή μπορούν να χρησιμοποιηθούν από τις vCPU πέρα από τις ήδη δεσμευμένες υπολογιστικές τους μονάδες.

Ενώ εξ’ ορισμού οι υπολογιστικές μονάδες CU δεσμεύονται ανά vCPU, υπάρχει η δυνατότητα προσαρμογής και ανά εικονική μηχανή. Με άλλα λόγια, υπάρχει η δυνατότητα διαμοιρασμού εφεδρικών CU μεταξύ vCPU της ίδιας εικονικής μηχανής, δημιουργώντας μία πολύ χρήσιμη και ευέλικτη επέκταση του μοντέλου για την αποτελεσματικότερη κάλυψη των αναγκών των χρηστών.

Η υλοποίηση του μοντέλου απαιτεί τους παρακάτω μηχανισμούς:

- Δημιουργία μηχανισμού τοποθέτησης για τη δέσμευση και την καταγραφή των υπολογιστικών μονάδων ανά εξυπηρετητή τόσο κατά την δημιουργία των εικονικών μηχανών όσο και κατά τις ενέργειες εξισορρόπησης και μεταφοράς αυτών.

- Δημιουργία μηχανισμού κατανομής του χρόνου χρήσης της CPU τοπικά στον εξυπηρετητή, ανάλογα με τις δεσμευμένες υπολογιστικές μονάδες της εικονικής μηχανής. Η ανάγκη για εφαρμογή του μηχανισμού παρουσιάζεται σε περιπτώσεις πληρότητας δηλαδή όταν οι αιτούμενοι κύκλοι CPU είναι περισσότεροι από αυτούς που έχει διαθέσιμους.

Το παραπάνω μοντέλο θα μπορούσε να ενισχυθεί με ένα μηχανισμό εποπτείας και εξισορρόπησης στη συστάδα, με σκοπό τη βελτιστοποίηση της χρήσης της CPU. Η επέκταση αυτή προβάλλει τη δυνατότητα βελτιστοποίησης της ποιότητας της υπηρεσίας ακόμα και σε περιπτώσεις που θεωρητικά τα φορτία κατανέμονται δίκαια και εντός των προκαθορισμένων προδιαγραφών.

1.4.3. Σενάρια χρήσης (Use cases)

Ένα παράδειγμα εξισορρόπησης που βασίζεται στην πραγματική χρήση της CPU, θα μπορούσε να μεταφέρει τα burstable φορτία που έχουν μικρότερες εγγυήσεις, σε εξυπηρετητές που έχουν αδρανείς πόρους CPU. Παρόλο που και οι δύο μηχανισμοί τοποθέτησης και κατανομής που αναφέρουμε διασφαλίζουν τις ελάχιστες απαιτήσεις χρήσης, μπορούμε να βελτιώσουμε την απόδοση κατανέμοντας κατάλληλα τους μη εγγυημένους πόρους. Για παράδειγμα, να προσφέρουμε μη εγγυημένους πόρους σε έναν κόμβο του οποίο οι εξυπηρετητές ζητούν διαρκώς περισσότερους πόρους από αυτούς που έχουν δεσμεύσει έναντι εξυπηρετητών που υποχρησιμοποιούν τους πόρους τους.

Ένα ακόμα σενάριο ανακατανομής του φορτίου της συστάδας με σκοπό την ενεργειακή εξοικονόμηση είναι η ομαδοποίηση των φορτίων των εικονικών μηχανών σε λιγότερους κόμβους με σκοπό την απενεργοποίηση των αδρανών. Η εκτίμηση της μελέτης μας είναι ότι η αυξημένη ενέργεια στους νέους ομαδοποιημένους κόμβους δεν υπερβαίνει την παραγόμενη ενέργεια από τους κόμβους που κάνουν μέτρια χρήση των πόρων τους και είναι αυξημένοι σε αριθμό. Εκμεταλλευόμενοι τον ορισμό των υπολογιστικών μονάδων CU, μπορούμε να συγκεντρώσουμε τα υπολογιστικά φορτία των εικονικών μηχανών χωρίς να μειώσουμε την ποιότητα της υπηρεσίας και σε συνδυασμό με την εποπτεία της πραγματικής χρήσης των πόρων, να ανακατανέμουμε και να απελευθερώσουμε πολύτιμους για το σύνολο της υποδομής πόρους.

1.4.4. Συνεκτιμήσεις

Πέρα από τη θεωρητική προσέγγιση της διαίρεσης των CPU των εξυπηρετητών σε vCPU, υπάρχουν και πρακτικοί παράγοντες που πρέπει να αξιολογηθούν. Οι παράγοντες αυτοί κυρίως αφορούν τη δυνατότητα εικονοποίησης του συστήματος, τη συμπεριφορά του λειτουργικού συστήματος του εξυπηρετητή αλλά και καταστάσεις που καταναλώνουν υπολογιστικούς κύκλους όπως η μέθοδος δρομολόγησης κατά τη χρονική διαίρεση της CPU σε μικρότερα τμήματα.

Επιγραμματικά, οι πραγματικοί παράγοντες κατανάλωσης κύκλων μηχανής, που πρέπει να συνεκτιμηθούν κατά το σχεδιασμό της αρχιτεκτονικής είναι:

- Περιορισμοί του συστήματος επίβλεψης και εικονοποίησης (λογισμικό QEMU).
- Περιορισμοί του συστήματος (vmenter/vmexit, χρονικοί περιορισμοί του πυρήνα σε ενέργειες εισόδου/εξόδου).
- Άλλοι περιορισμοί του συστήματος όπως υπηρεσίες που τρέχουν στο υπόβαθρο του εξυπηρετητή και μπορεί να απαιτούν αυξημένο αριθμό πόρων.

Μία ακόμα σημαντική παράμετρος για το σχεδιασμό της αρχιτεκτονική της υπηρεσίας είναι η παροχή της δυνατότητας αναπροσαρμογής, από τον ίδιο το διαχειριστή της υπηρεσίας, των υπολογιστικών μονάδων CU σε vCPU είτε χειροκίνητα είτε με κάποιο αυτοματοποιημένο μηχανισμό ανατροφοδότησης. Η συγκεκριμένη δυνατότητα δίνει ευελιξία πολιτικών διαχείρισης αφού με αυτό τον τρόπο ο διαχειριστής μπορεί να μειώνει τη διαθεσιμότητα υπολογιστικών μονάδων των χρηστών που συστηματικά τους υποχρησιμοποιούν, εξοικονομώντας έτσι πόρους και αυξάνοντας τη διαθεσιμότητα της υπηρεσίας.

2. Πειραματική λειτουργία συστήματος έξυπνης ενεργειακής διαχείρισης

2.1 Περίληψη κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάζονται τα συμπεράσματα από τη μελέτη και αξιολόγηση των επιλεγμένων προδιαγραφών, όπως παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Συγκεκριμένα, παρουσιάζεται η υλοποίηση των κατάλληλων τεχνικών και εργαλείων, η αξιοποίησή τους στην περίπτωση της κατανομής και τοποθέτησης φορτίων, καθώς και η προσαρμογή των εργαλείων αυτών σε ένα περιβάλλον αναφοράς, το οποίο θα εξομοιώνει ένα MVP για την παρούσα μελέτη.

Οι γενικές παρατηρήσεις που απορρέουν από την υψηλού επιπέδου αρχιτεκτονική του ΕΝΕΔΗ είναι ο διαχωρισμός σε δύο υποσυστήματα, το σύστημα Εποπτείας και το σύστημα Διαχείρισης. Το σύστημα Εποπτείας είναι υπεύθυνο για τη συλλογή μετρικών για τις εικονικές μηχανές, την αποθήκευσή τους, τη μεταφορά τους όπου κρίνεται απαραίτητη καθώς και τη δυνατότητα υποβολής ερωτημάτων για την εξαγωγή συμπερασμάτων. Στη συνέχεια, μέσω του συστήματος Διαχείρισης, οι διαχειριστές ορίζουν την πολιτική κατανομής των φορτίων στη νεφοϋπολογιστική υποδομή βελτιστοποιώντας την απόδοσή της με τη χρήση σύγχρονων εργαλείων αυτοματισμού.

2.1.1 Δομή Κεφαλαίου

Το κεφάλαιο δομείται σε τρεις βασικές ενότητες:

- Ορισμός και ανάλυση των μετρικών αξιολόγησης της αρχιτεκτονικής
- Παρουσίαση των βασικών τεχνολογιών στις οποίες βασίστηκε η υλοποίηση των συστημάτων
- Περιγραφή των ρυθμίσεων και της λειτουργίας των επιμέρους μερών

2.2 Ορισμός και ανάλυση μετρικών

Στη συγκεκριμένη ενότητα, παρουσιάζονται οι μετρικές που συλλέγονται από την αρχιτεκτονική ΕΝΕΔΗ, καθώς και η μέθοδος συλλογής τους. Στη υλοποίησή μας, τόσο οι εικονικές μηχανές, όσο και οι εξυπηρετητές τους, εποπτεύονται από το σύστημα ΕΝΕΔΗ. Η βασική διαφορά είναι ότι σε αντίθεση με την πληθώρα εργαλείων για την εποπτεία των εξυπηρετητών, τα αντίστοιχα εργαλεία για τις εικονικές μηχανές αναπτύχθηκαν ειδικά για τη συγκεκριμένη περίπτωση χρήσης.

2.2.1 Μετρικές αξιολόγησης εικονικών μηχανών (VM)

Όπως παρουσιάστηκαν στο προηγούμενο κεφάλαιο, υπάρχουν 4 κύριες κατηγορίες πόρων προς αξιολόγηση σε μία εικονική μηχανή. Για κάθε τέτοια κατηγορία μπορεί να συλλέγονται διαφορετικές μετρικές, οι οποίες εκθέτουν διαφορετικές οπτικές για μία συγκεκριμένη κατηγορία πόρων, ενώ ταυτόχρονα, μπορεί να παρέχουν αξιόλογη πληροφορία για την κατανάλωση κάποιων άλλων πόρων και κατ΄επέκταση για τη μείωση της κατανάλωσης.

Για την επιπλέον επεξήγηση των μετρικών και της ανάλυσής τους, παρουσιάζουμε το Synnefo, το λογισμικό για την ανάπτυξη νεφοϋπολογιστικών υποδομών, το οποίο αποτελεί τη βάση της πλατφόρμας ~okeanos για την ανάπτυξη εικονικών μηχανών. Το ~okeanos αξιοποιεί το QEMU/KVM ως επιβλέπων της εικονικής μηχανής για να παρέχει την υπηρεσία. Το QEMU είναι ένα γενικευμένο, ανοιχτού λογισμικού εργαλείο κατάλληλο για προσομοίωση και εικονοποίηση συστημάτων. Η υψηλή απόδοση του συγκεκριμένου λογισμικού οφείλεται στη δυνατότητα εκτέλεσης του κώδικα της εικονικής μηχανής απευθείας στη CPU του εξυπηρετητή χρησιμοποιώντας το KVM kernel module αλλά και τη δυνατότητα εικονοποίησης του hardware.

Από την οπτική του λειτουργικού συστήματος του εξυπηρετητή, μία εικονική μηχανή που υποστηρίζει QEMU/KVM αποτελεί μία πολυνηματική διεργασία. Χρησιμοποιούμε το στοιχείο αυτό για να συλλέξουμε τις μετρικές μας αφού η εποπτεία της εικονικής μηχανής προκύπτει σχεδόν άμεσα από την κατανάλωση των πόρων της αντίστοιχης διεργασίας. Δεδομένου του λειτουργικού συστήματος του εξυπηρετητή το οποίο είναι βασισμένο σε linux OS (Debian), χρησιμοποιούμε το `/proc` εικονικό filesystem για την εξαγωγή της πληροφορίας που σχετίζεται με τη χρήση των πόρων σε επίπεδο διεργασιών. Ως εικονικό σύστημα αρχείων, δεν βρίσκεται στο δίσκο, αλλά ο πυρήνας του λειτουργικού συστήματος το δημιουργεί στη μνήμη και παρέχει πληροφορίες σχετικά με το σύστημα, με έμφαση στις διεργασίες. Συγκεκριμένα, η διεύθυνση `/proc/<PID>/` περιλαμβάνει πληροφορίες αναφορικά με την κατάσταση της μνήμης της διεργασίας, το χρόνο της διεργασίας στον επεξεργαστή αλλά και τους file descriptors της.

Τα QEMU instances ελέγχονται από το λογισμικό Google Ganeti. Το Ganeti διατηρεί πληροφορίες για τις εικονικές μηχανές που φιλοξενούνται σε κάθε κόμβο στη διεύθυνση `/var/run/ganeti/`. Το Ganeti διατηρεί πολλές πληροφορίες στη συγκεκριμένη διεύθυνση, ωστόσο μας ενδιαφέρει κυρίως η αντιστοίχιση των εικονικών μηχανών με τις αντίστοιχες διεργασίες. Το συγκεκριμένο στοιχείο αποτυπώνεται στη διεύθυνση `/var/run/ganeti/kvn-hypervisor/pid/`, το οποίο περιέχει ένα αρχείο για κάθε εικονική μηχανή. Το όνομα του αρχείου υποδεικνύει αυτό της εικονικής μηχανής στο Ganeti, ενώ το περιεχόμενο του αρχείου είναι το PID της διεργασίας QEMU για το συγκεκριμένο κόμβο. Με αυτή την αντιστοίχιση είναι δυνατή η εξαγωγή της απαραίτητης πληροφορίας από τη διεπαφή του πυρήνα στη διεύθυνση `/proc`.

Στη επόμενη ενότητα παρουσιάζουμε τις τρεις μετρικές που συλλέγονται από το σύστημα (χρήση Κεντρικής Μονάδας Επεξεργασίας – CPU usage, κατανάλωση Μνήμης - Memory consumption, χρήση Δικτύου - Network usage). Μελλοντικά υπάρχει η πρόθεση για εποπτεία επιπλέον μετρικών όπως οι λειτουργίες εισόδου/εξόδου στο δίσκο, το εύρος ζώνης και το ποσοστό δέσμευσης του δίσκου.

2.2.1.1 Κεντρική Μονάδα Επεξεργασίας (CPU)

Για την συγκέντρωση των μετρικών της CPU, λαμβάνουμε όλα τα αναγνωριστικά των εικονικών μηχανών και τα PIDs τους από τη διεύθυνση `/var/run/ganeti/kvm-hypervisor/pid/`. Στη συνέχεια μέσω του αρχείου `/proc/<PID>/stat` συλλέγουμε την απαραίτητη πληροφορία. Στο συγκεκριμένο αρχείο καταγράφεται η κατάσταση της CPU για τη προς εξέταση διεργασία ενώ η ακριβής διατύπωση μπορεί να διαφέρει ανά αρχιτεκτονική. Οι πληροφορίες αποθηκεύονται σε μία γραμμή και χωρίζονται με κενά. Οι μετρικές, τις οποίες θα χρησιμοποιήσουμε για τη συγκεκριμένη περίπτωση χρήσης βρίσκονται στις λέξεις με αριθμό 14 (user CPU time), 15 (system CPU time) και 16 (guest CPU time). Η ανάλυση των πεδίων παρουσιάζεται παρακάτω:

- Η παράμετρος system CPU time της διεργασίας υποδεικνύει το χρόνο τον οποίο η διεργασία βρίσκεται σε kernel mode, για παράδειγμα όταν το QEMU κάνει κλήσεις συστήματος.
- Η παράμετρος guest CPU time υποδεικνύει το χρόνο που η διεργασία καταναλώνει τρέχοντας μία εικονική CPU για το φιλοξενούμενο λειτουργικό σύστημα, το οποίο αποτελεί τον πραγματικό χρόνο που η εικονική μηχανή δαπανά στη CPU.
- Τέλος, η παράμετρος user CPU time αναφέρεται στο χρόνο όπου η διεργασία βρίσκεται σε user mode. Περιλαμβάνει επομένως, την guest CPU time αλλά και το χρόνο στον οποίο η QEMU βρίσκεται σε user space.

Οι χρόνοι που συλλέγονται για τις επιμέρους παραμέτρους περιέχουν πάντα μία μικρή απόκλιση, στοιχείο που πρέπει να αντισταθμίζεται βάσει της ακρίβειας και της επίδοσης του συστήματος. Οι μετρητές που χρησιμοποιούνται, συλλέγουν το συνολικό χρόνο τον οποίο η διεργασία καταναλώνει για κάθε επιμέρους κατηγορία από τη στιγμή που δημιουργείται. Εφόσον οι τιμές που αποθηκεύονται είναι σωρευτικές , η τρέχουσα τιμή αφαιρείται από την τελευταία υπολογισμένη τιμή για κάθε μετρική, ώστε να προκύψει η πραγματική χρήση της CPU για κάθε λειτουργία.

Μία επιπλέον σημαντική παράμετρος για τη σωστή αξιολόγηση των δεδομένων είναι ότι η τιμή που αποθηκεύεται αφορά τη συνολικό χρόνο όπου η διεργασία εκτελείται σε όλες τις CPUs. Επομένως, ο χρόνος πρέπει να εξεταστεί σε συνδυασμό με τον αριθμό των vCPU, που έχει η κάθε εικονική μηχανή.

Για την καλύτερη επεξήγηση παρουσιάζουμε το παρακάτω παράδειγμα:

Δεδομένου ενός διαστήματος 10 δευτερολέπτων, μία μέτρηση του guest time με ένδειξη 10 δευτερόλεπτα, υποδεικνύει ότι η εικονική μηχανή έχει δεσμεύσει το αντίστοιχο μίας φυσικής CPU σε CPU time. Στην περίπτωση που η εικονική μηχανή έχει μία vCPU, η εικονική μηχανή χρησιμοποίησε το 100% της μέγιστης διαθέσιμης CPU. Εάν όμως η εικονική μηχανή έχει 4 vCPUs, έχει καταναλώσει το 25% της μέγιστης διαθέσιμης CPU κατανεμημένη στις 4 vCPUs.

Επομένως, για την αποτελεσματική τεκμηρίωση των στοιχείων, ο αριθμός των vCPUs που ανατίθενται σε κάθε μία εικονική μηχανή αποθηκεύεται κατά τη διάρκεια των μετρήσεων.

Τελικά, τέσσερις μετρικές συλλέγονται για κάθε μία από τις λειτουργίες (user, guest, system) που αναφέρθηκαν:

2.2.1.1.1 Total CPU time

Ο συνολικός χρόνος που η διεργασία καταναλώνει σε κάθε λειτουργία στο συγκεκριμένο διάστημα. Αυτό μπορεί να είναι μεγαλύτερο από το διάστημα αφού ισούται με το άθροισμα του χρόνου που δαπανάται σε κάθε CPU.

2.2.1.1.2 VM utilization

Υπολογίζεται από τη σχέση total_cpu_time * 100 / interval, όπου η μεταβλητή total_cpu_time λαμβάνεται από την προηγούμενη μετρική. Αυτή η μετρική υποδηλώνει τον αριθμό των πυρήνων που χρησιμοποίησε η εικονική μηχανή και μπορεί να φτάσει έως και 100 * vCPUs.

2.2.1.1.3 VM utilization normalized

Υπολογίζεται από τη σχέση vm_utilization / vCPUs, όπου η μεταβλητή vm_utilization λαμβάνεται από την προηγούμενη μετρική. Από τη συγκεκριμένη μετρική υπολογίζεται τη χρήση των πόρων των εικονικών μηχανών και αποτελεί έναν καλό δείκτη για την αποτελεσματικότητα της διαχείρισης των πόρων που ο χρήστης αιτήθηκε. Το εύρος τιμών της μετρικής κυμαίνεται από 0 έως 100.

2.2.1.1.4 Host resources utilization

Για τον υπολογισμό της χρησιμοποιούμε τη σχέση vm_utilization / pCPUs, όπου οι pCPUs είναι ο αριθμός των πραγματικών CPU του εξυπηρετητή. Αποτελεί δείκτη για τους πόρους του εξυπηρετητή που χρησιμοποιήθηκαν από την εικονική μηχανή και κυμαίνεται από 0 έως 100. Στη συνέχεια, το άθροισμα όλων των καταναλισκόμενων πόρων μπορούν να χρησιμοποιηθούν για τη διάκριση των πόρων που χρησιμοποιήθηκαν από την εικονική μηχανή και από άλλες διεργασίες. Αυτή η μετρική είναι χρήσιμη για την εξισορρόπηση της χρήσης της CPU, κατανέμοντας το φορτίο στις εικονικές μηχανές ώστε να επιτευχθεί παρόμοια χρήση ακόμα και σε διαφορετικούς και ετερογενείς εξυπηρετητές.

Για την πληρότητα των μετρικών που σχετίζονται με την CPU, κάθε μέτρηση περιλαμβάνει ποικιλία μεταδεδομένων όπως οι pCPUs του εξυπηρετητή, οι vCPUs της εικονικής μηχανής καθώς και ο συνολικός χρόνος που ο εξυπηρετητής χρησιμοποίησε τις CPUs του σε αυτό το διάστημα.

2.2.1.2 Μνήμη

Για την καταγραφή των μετρικών για τη μνήμη, αποθηκεύουμε τη μνήμη RSS που χρησιμοποιείται από τη διεργασία. Η RSS είναι το ποσοστό της συνολικής μνήμης που δεσμεύεται από μία διεργασία και διατηρείται στην κύρια μνήμη. Το συγκεκριμένο ποσοστό της μνήμης περιλαμβάνει ακόμα και τη μνήμη του QEMU, χωρίς ωστόσο να αποτελεί σημαντική πληροφορία για τον κάτοχο της εικονικής μηχανής και δεδομένου του μικρού της μεγέθους σε σχέση με αυτό της εικονικής μηχανής, μπορεί να θεωρηθεί ως ένα οριακό σφάλμα.

Πρέπει επίσης να σημειωθεί ότι στην πλατφόρμα ~okeanos εκμεταλλευόμαστε το KSM kernel feature (kernel same page merging) με το οποίο είναι δυνατός για το σύστημα εποπτείας, ο διαμοιρασμός πανομοιότυπων σελίδων μνήμης μεταξύ διαφορετικών διεργασιών. Με βάση αυτό πολλές διεργασίες QEMU σε κάθε εξυπηρετητή μπορούν να μοιράζονται πολλές σελίδες μνήμης, με τις σελίδες αυτές να συμπεριλαμβάνονται στις τιμές RSS που υπολογίζουμε. Επομένως, το άθροισμα όλων των τιμών RSS των διεργασιών QEMU σε έναν εξυπηρετητή είναι συνήθως μεγαλύτερο από τη μνήμη που χρησιμοποίησε η συγκεκριμένη διεργασία. Ο υπολογισμός μετρικών που σχετίζονται με το KSM θα μπορούσε να πραγματοποιηθεί σε μελλοντικές προσεγγίσεις για βελτιωμένη εποπτεία της μνήμης. Για την περίπτωση που εξετάζουμε, μόνο οι τιμές της RSS μνήμης αποθηκεύονται, αφού για λόγους επίδοσης, δεν επιτρέπεται στις εικονικές μηχανές να ανταλλάσσουν μνήμη.

Η μνήμη RSS συνοδεύεται με τη συνολική μνήμη του εξυπηρετητή καθώς και με τη συνολική μνήμη της εικονικής μηχανής. Ο τρόπος καταγραφής των τιμών αυτών περιγράφεται παρακάτω:

- Για τη μέτρηση της RSS μνήμης χρησιμοποιούμε το `/proc` filesystem και συγκεκριμένα το αρχείο `/proc/<PID>/statm` περιέχει επτά διαφορετικά στοιχεία. Η δεύτερη λέξη περιέχει την τιμή της RSS της διεργασίας σε σελίδες ενώ στη συνέχεια τη μετατρέπουμε σε bytes πολλαπλασιάζοντας με την προκαθορισμένη μεταβλητή του συστήματος “SC_PAGE_SIZE”.
- Η καταγραφή της συνολικής μνήμης του εξυπηρετητή βρίσκεται στη διεύθυνση `/proc/meminfo` όπου κάθε γραμμή περιλαμβάνει το όνομα της παραμέτρου, την τιμή της και τη μονάδα μέτρησης που χρησιμοποιείται (KB). Η παράμετρος που μας ενδιαφέρει είναι η “Total Memory”, η οποία αφού μετατραπεί σε bytes αποθηκεύεται στη μνήμη.
- Η συνολική μνήμη της εικονικής μηχανής παρέχεται σαν όρισμα στο QEMU κατά τη δημιουργία της με τη “-m” flag. Με σκοπό τον υπολογισμό της συνολικής μνήμης της εικονικής μηχανής ελέγχουμε το αρχείο `/var/run/ganeti/kvm-hypervisor/conf/<VM_ID>.runtime`, το οποίο περιλαμβάνει μία λίστα με ορίσματα για τη διεργασία QEMU.

Από τις παραπάνω επιλογές μετρικών, επιλέχθηκε να αποθηκευτούν οι παρακάτω τρεις τιμές:

2.2.1.2.1 RSS Memory

Περιέχει τη συνολική μνήμη που χρησιμοποιήθηκε από την εικονική μηχανή.

2.2.1.2.2 VM memory utilization

Εκφράζει το ποσοστό της χρησιμοποιούμενης μνήμης από το σύνολο της RAM που αποδόθηκε στην εικονική μηχανή. Υπολογίζεται από τη σχέση `rss memory` * 100 / 'flavor's RAM`, όπου η `rss_memory` υπολογίζεται όπως αναφέρεται παραπάνω και η `flavor's RAM` είναι η συνολική RAM που αντιστοιχίζεται στην εικονική μηχανή.

2.2.1.2.3 VM host memory utilization

Εκφράζει το ποσοστό της μνήμης του εξυπηρετητή που χρησιμοποιήθηκε από κάθε εικονική μηχανή. Υπολογίζεται από τη σχέση `rss_memory` * 100 / `host_memory`, όπου η μεταβλητή `host_memory` είναι η συνολική μνήμη του εξυπηρετητή.

2.2.1.3 Δίκτυο

Για την καταγραφή των μετρικών του δικτύου εποπτεύονται για κάθε εικονική μηχανή τα bytes που αποστέλλονται, τα bytes που λαμβάνονται, τα πακέτα που στέλνονται καθώς και αυτά που λαμβάνονται.

Για την κατανόηση αυτών των μετρικών, προϋπόθεση αποτελεί η γνώση του τρόπου λειτουργίας του Ganeti σε θέματα δικτύου. Για κάθε διεπαφή εικονικής μηχανής, το Ganeti δημιουργεί μία αντίστοιχη διεπαφή στον εξυπηρετητή και όλη η κίνηση της εικονικής μηχανής περνάει μέσα από αυτή. Αυτές οι διεπαφές εντοπίζονται στη διεύθυνση `/var/run/ganeti/kvm-hypervisor/nic/`, όπου υπάρχουν ξεχωριστές διευθύνσεις για κάθε μία εικονική μηχανή του συγκεκριμένου κόμβου. Περιλαμβάνει ένα αρχείο για κάθε διεπαφή της εικονικής μηχανής και το περιεχόμενο αυτών των αρχείων περιέχει το όνομα της διεπαφής του εξυπηρετητή που αντιστοιχεί σε αυτή την εικονική μηχανή. Αφού αντλήσουμε τα στοιχεία της αντίστοιχης διεπαφής συλλέγουμε τα στοιχεία που βρίσκονται στη διεύθυνση `sys/class/net/<interface>/statistics`.

Πιο συγκεκριμένα, για κάθε μετρική λαμβάνουμε:

2.2.1.3.1 Bytes received

Τα bytes που λαμβάνονται αποθηκεύονται στη διεύθυνση `/sys/class/net/<interface>/statistics/rx_bytes`.

2.2.1.3.2 Bytes transmitted

Τα bytes που αποστέλλονται αποθηκεύονται στη διεύθυνση `/sys/class/net/<interface>/statistics/tx_bytes`.

2.2.1.3.3 Packets received

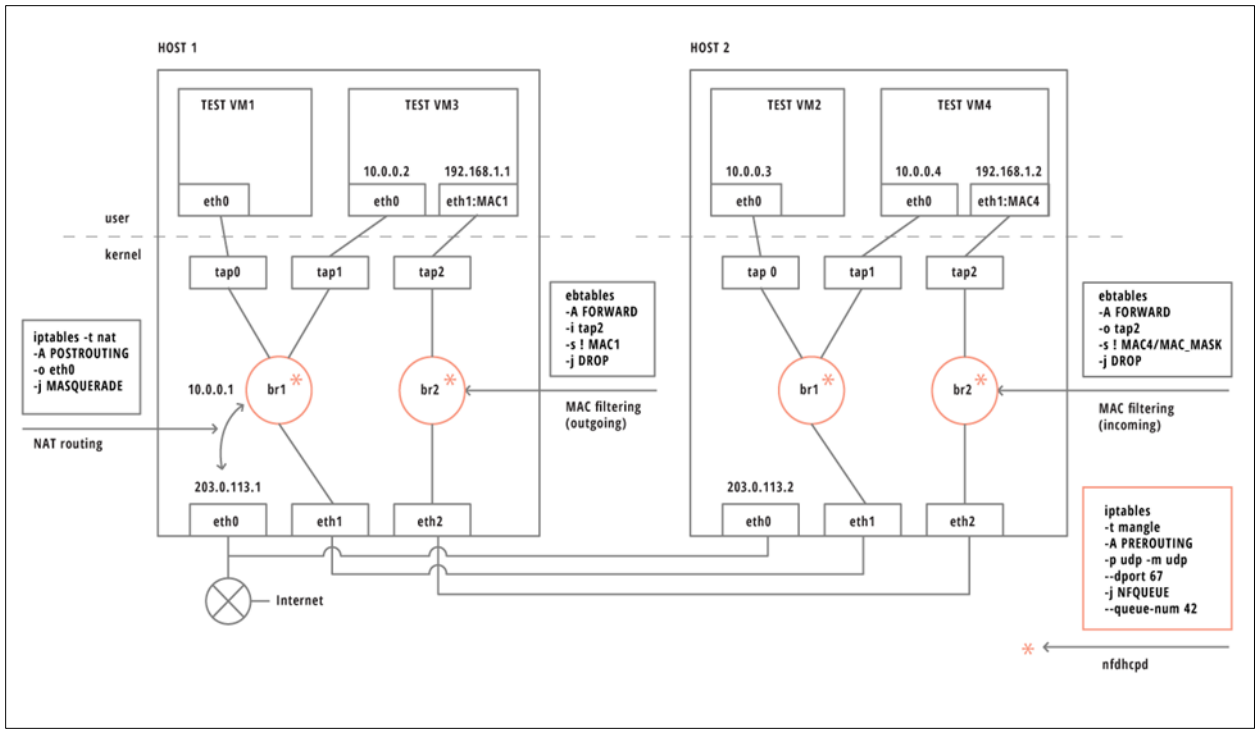
Η καταγραφή των πακέτων που λαμβάνονται γίνεται στη διεύθυνση `/sys/class/net/<interface>/statistics/rx_packets`.

2.2.1.3.4 Packets transmitted

Η καταγραφή των πακέτων που αποστέλλονται γίνεται στη διεύθυνση `/sys/class/net/<interface>/statistics/tx_packets`

Σημαντική παρατήρηση σε αυτό το σημείο είναι ότι η διεπαφή του εξυπηρετητή λειτουργεί ως γέφυρα μεταξύ της NIC's της εικονικής μηχανής και της NIC's του εξυπηρετητή με αποτέλεσμα τα ονόματα των μετρικών να εκφράζουν το αντίστροφο από αυτό που τα ονόματά τους υποδεικνύουν, δηλαδή στα σημεία που αναφέρεται αποστολή στη δική μας ανάλυση πρόκειται για λήψη, σε δεδομένα και πακέτα αντίστοιχα.

Οι τιμές που συλλέγονται συνδυάζονται με το εύρος ζώνης του NIC του εξυπηρετητή και περνούν ως ρυθμίσεις στο μηχανισμό συλλογής, αφού αποθηκευτούν στη βάση δεδομένων του συστήματος.



2.2.1.3 Μετρικές Εξυπηρετητή

Η συλλογή των μετρικών των εικονικών μηχανών είναι επαρκής για την εποπτεία της κατανάλωσης των πόρων από τις εικονικές μηχανές και ικανοποιούν τα κριτήρια για την περίπτωση της έξυπνης τοποθέτησης των φορτίων. Με σκοπό την πληρότητα των στοιχείων, η πλατφόρμα ΕΝΕΔΗ εποπτεύει συμπληρωματικά και τις μετρικές των ίδιων των εξυπηρετητών. Με αυτό τον τρόπο τα συνολικά δεδομένα εμπλουτίζουν τα στοιχεία που αντιστοιχούν στις εικονικές μηχανές και δίνουν τη δυνατότητα καλύτερης διερεύνησης αναφορικά με τη επίδοσή τους και τους αναγκαίους πόρους.

Οι μετρικές των εξυπηρετητών που συγκεντρώνονται είναι οι παρακάτω:

- Ο χρόνος που καταναλώνει η CPU σε διάφορες καταστάσεις, όπως η εκτέλεση κώδικα χρήστη, η εκτέλεση κώδικα συστήματος, ο χρόνος αναμονής για λειτουργίες εισόδου/εξόδου και ο χρόνος αδράνειας
- Το δίκτυο και συγκεκριμένα πληροφορίες για την κίνηση, τα πακέτα ανά δευτερόλεπτο καθώς και τα σφάλματα των διεπαφών ανά δευτερόλεπτο
- Στατιστικά απόδοσης των σκληρών δίσκων καθώς και επιμέρους partition όπου συναντώνται
- Η χρήση της φυσικής μνήμης του εξυπηρετητή

2.3 Αρχιτεκτονική Πλατφόρμας ΕΝΕΔΗ – Σχεδιασμός και Υλοποίηση

Η αρχιτεκτονική της πλατφόρμας ΕΝΕΔΗ σχεδιάστηκε ώστε να είναι ανθεκτική σε σφάλματα και εύκολα κλιμακώσιμη σε περίπτωση που παρουσιαστεί η ανάγκη. Τα κύρια συστατικά μέρη της είναι τα παρακάτω:

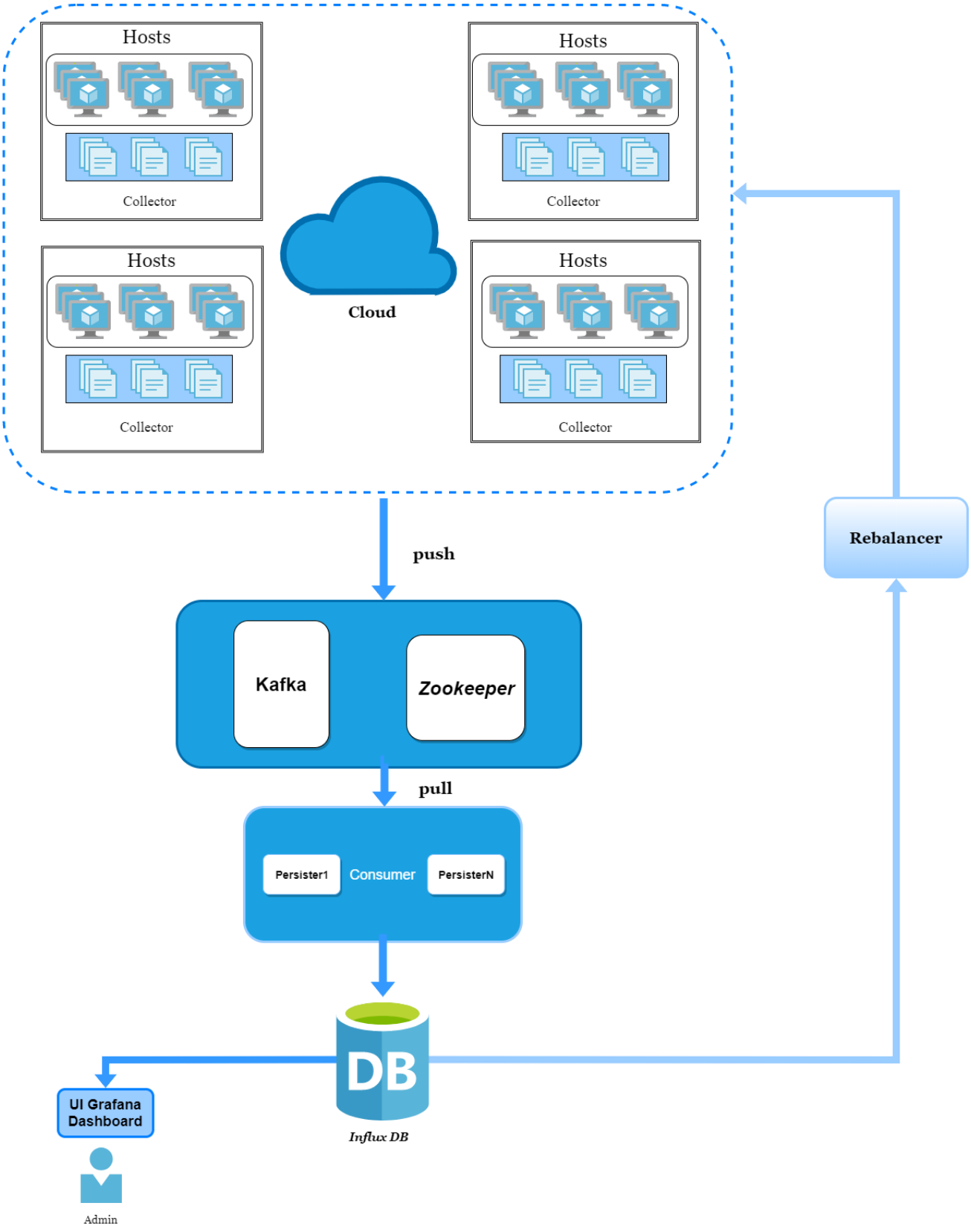
- Metrics Interface
- Collector
- Producer
- Message Queue
- Consumer
- Time series Database

2.3.1. Metrics Interface

Για την ομοιόμορφη κατανομή των δεδομένων σε όλα τα επίπεδα της πλατφόρμας, δημιουργήσαμε μία κλάση σε γλώσσα python με την ονομασία Metric. Η κλάση αυτή σχεδιάστηκε ώστε να είναι ευέλικτη και εύκολα επεκτάσιμη ανάλογα με τις ανάγκες που προκύπτουν από τις καταγεγραμμένες τιμές και τις επιμέρους αρχιτεκτονικές. Για την επίτευξη αυτών, η κλάση περιλαμβάνει τα παρακάτω πεδία:

1. **Value:** η τιμή της μέτρησης, χωρίς κάποιο περιορισμό στον τύπο
2. **Timestamp:** το χρονικό αποτύπωμα κατά την καταγραφή της μέτρησης, χωρίς και πάλι κάποιο περιορισμό στον τύπο
3. **Metric_name:** το όνομα της μετρικής ώστε να αποθηκευθεί στο αντίστοιχο πεδίο στη βάση δεδομένων
4. **_aggr_type:** το πεδίο υποδεικνύει τη μέθοδο υποδειγματοληψίας των μετρικών. Οι υποστηριζόμενες μέχρι αυτή τη στιγμή είναι η ‘mean` και `min`
5. **_metric_type:** το πεδίο αυτό περιλαμβάνει τον τύπο της μετρικής, όπως για παράδειγμα αν είναι ανεξάρτητη από τις υπόλοιπες μετρικές με το ίδιο metric_name, ή αν αποτελεί συνέχεια κάποιας ακολουθίας (π.χ. μετρητής)
6. **Tags**: αποτελείται από ένα python dictionary με ζεύγη της μορφής κλειδιού/τιμής. Το κλειδί είναι το όνομα του tag, το οποίο περιγράφει το περιεχόμενό του, ενώ η τιμή του ζεύγους είναι η τιμή του συγκεκριμένου tag. Τα δύο αυτά συστατικά στοιχεία έχουν τους ίδιους περιορισμούς όπως και το metric_name.

Τα tags αποτελούν συνήθως στατική πληροφορία και χαρακτηρίζουν τόσο την εικονική μηχανή όσο και τον ίδιο τον εξυπηρετητή.



Εικόνα 3 Αρχιτεκτονική της πλατφόρμας ΕΝΕΔΗ

Τα πιο συνηθισμένα tags που συναντώνται είναι:

a. Monitored_instance_type: δηλώνει αν πρόκειται για εικονική μηχανή ή εξυπηρετητή
b. Monitored_instance_id: περιέχει το αναγνωριστικό id, όπως snf-2016, ok0-00
c. Resource: υποδεικνύει τον τύπο της μετρικής (cpu, μνήμη)
d. Hostname: το όνομα του εξυπηρετητή της εικονικής μηχανής
e. Cluster: το όνομα του cluster στο οποίο ανήκει η μηχανή
f. Resource_metric: περιέχει το συγκεκριμένο όνομα της μετρικής (π.χ. χρήση cpu, ελεύθερη μνήμη κτλ)

Για την πληρότητα των μετρούμενων ποσοτήτων προστέθηκαν επιπλέον tags ανάλογα με τον πόρο που εξετάζουμε:

- Για τις μετρικές που αφορούν την CPU προστέθηκαν tags με τις pCPU των εξυπηρετητών και τις vCPU των εικονικών μηχανών
- Για τις μετρικές δικτύου σημειώνεται το όνομα της διεπαφής που εποπτεύεται
- Για τις μετρικές μνήμης αποθηκεύουμε τη συνολική μνήμη του εξυπηρετητή αλλά και των εικονικών μηχανών

Συμπληρωματικά με την κλάση Metric, δημιουργήθηκε μία μετακλάση AbstractEncoder, η οποία παρέχει ένα API για κωδικοποίηση/ αποκωδικοποίηση των μετρικών από και προς το πρωτόκολλο επικοινωνίας αλλά και για την αποκωδικοποίησή τους σε μορφή κατάλληλη για αποθήκευση στη βάση δεδομένων.

2.3.2 Κωδικοποιητής (Encoder)

Για την αποτελεσματική κωδικοποίηση/αποκωδικοποίηση των μετρικών υλοποιήθηκε η κλάση Abstract Base Class AbstractEncoder.

Οι τρείς βασικές συναρτήσεις που υλοποιεί ο κωδικοποιητής είναι:

- **Encode:** δέχεται ως είσοδο δύο ορίσματα, μία λίστα με μετρικές και ένα dictionary με τα κοινά tags. Τα tags αυτά λειτουργούν ως βάση για την κωδικοποίηση των μετρικών.
- **Decode_to_metrics:** λαμβάνει ως είσοδο ένα κωδικοποιημένο μήνυμα από τη συνάρτηση encode και επιστρέφει την αρχική λίστα των μετρικών, τροποποιημένων βάσει των κοινών tags.
- **Decode_to_influx:** λαμβάνει ως είσοδο ένα κωδικοποιημένο μήνυμα από τη συνάρτηση encode και επιστρέφει την αρχική λίστα των μετρικών σε μορφή κατάλληλη για εισαγωγή στην InluxDB.

Οι λειτουργίες κωδικοποίησης και αποκωδικοποίησης πρέπει να είναι αντίστροφες και κατά τη μετάβαση από τη μία στην άλλη να μην αλλάζει το περιεχόμενο των μετρικών παρά μόνο να προστίθενται κάποια tags.

Τα δεδομένα κωδικοποιούνται πριν σταλούν στη message queue και αποκωδικοποιούνται πριν γίνει η εισαγωγή τους στη βάση δεδομένων. Οι κλάσεις Metric και AbstractEncoder είναι πολύ σημαντικές για την λειτουργικότητα της αρχιτεκτονικής διότι απομονώνουν τα διακριτά στοιχεία της: το συλλέκτη (collector), την ουρά μηνυμάτων και την βάση δεδομένων. Με αυτό τον τρόπο η ανάπτυξη και ο έλεγχος γίνονται πιο εύκολα, αφού οι αλλαγές σε οποιοδήποτε στοιχείο δεν επηρεάζουν τα υπόλοιπα, εξαιτίας αυτής της σχέσης κωδικοποίησης/ αποκωδικοποίησης. Οποιαδήποτε επομένως τροποποίηση ή ακόμα και αντικατάσταση των επιμέρους στοιχείων δεν επηρεάζει καθόλου τη συνολική αρχιτεκτονική.

Στη συγκεκριμένη υλοποίηση χρησιμοποιούμε το JsonEncoder το οποίο μετατρέπει τις μετρικές σε JSON format:

```
{
   value: …,
   timestamp: …,
   metric_name: …,
   _aggr_type: …,
   _metric_type: …,
   tags: {key: value, ...}
}
```

Η συγκεκριμένη μορφή είναι βοηθητική για την υλοποίηση του πρωτοκόλλου, διότι είναι ευέλικτη και επεκτάσιμη, όπως στην περίπτωση προσθήκης/ αφαίρεσης tags.

2.3.3 Συλλέκτης (Collectd)

Ο συλλέκτης υλοποιήθηκε ως ένας collector deamon, όπως χρησιμοποιείται και στη νεφοϋπολογιστική υποδομή ~okeanos. Επομένως, επαναχρησιμοποιήθηκε ο βασικός κώδικας αλλά και τα προ-εγκαταστημένα πακέτα. Αναπτύχθηκαν plugins για τη συλλογή των μετρικών από τις εικονικές μηχανές αλλά και για την εγγραφή τους στο Apache Kafka Queue. Χρησιμοποιήθηκε το collectd python plugin το οποίο ενσωματώνει έναν python-interpreter και παρέχει περιβάλλον διεπαφής για το συγκεκριμένο σύστημα. Υλοποιήθηκαν τρία scripts για την συλλογή των μετρικών (cpu, μνήμη, δίκτυο) και επιπλέον για την εγγραφή τους στο Apache Kafka.

Στη συνέχεια ακολουθεί μία σύντομη περιγραφή της λειτουργίας του collectd με τα plugins:

1. Κατά την αρχικοποίηση του plugin εισόδου, λαμβάνονται όλες οι ρυθμίσεις από το configuration file, από όπου παρέχεται το script με τις στατικές πληροφορίες όπως το όνομα του cluster. Με αντίστοιχο τρόπο δημιουργείται ένα Kafka Producer Object, βάσει των προδιαγραφών του configuration file.
2. Τα plugins εισόδου έχουν προγραμματιστεί να τρέχουν σε διαστήματα των δέκα δευτερολέπτων.
3. Τα plugins εισόδου συλλέγουν τις μετρικές και τις αποθηκεύουν σε Metric objects, ενώ στη συνέχεια μετατρέπονται σε collectd values.
4. Οι τιμές αυτές μπαίνουν σε μία εσωτερική collectd queue το μέγεθος της οποίας προσδιορίζεται από τις παραμέτρους WriteQueueLimitHigh/ WriteQueueLimitLow, οι οποίες διασφαλίζουν ότι σε περίπτωση σφάλματος δεν θα δεσμευτεί άσκοπα μεγάλο μέρος της μνήμης του εξυπηρετητή.
5. Το εργαλείο collectd's Filter Chains χρησιμοποιείται για το διαχωρισμό των μετρικών που αφορούν την πλατφόρμα ΕΝΕΔΗ και αποστέλλονται για εγγραφή στο kafka plugin. Το Filter Chains δίνει τη δυνατότητα φιλτραρίσματος επιλεγμένων μετρικών και προώθησής τους σε επόμενα στάδια εξόδου.
6. Το plugin στην έξοδο διαβάζει τις τιμές από την ουρά, τις μετατρέπει από collectd values σε Metrics και τις εμπλουτίζει με νέα tags. Στη συνέχεια, κάθε Metric μετατρέπεται σε μορφή κατάλληλη για μεταφορά χρησιμοποιώντας την AbstractEncoder_API class και διοχετεύεται στην KafkaProducer class για την εγγραφή της στην Kafka Queue.

Για τη συλλογή μετρικών για τους εξυπηρετητές χρησιμοποιούμε ήδη υπάρχουσες υλοποιήσεις, όπως είναι τα cpu, interface, disk, df, memory plugins.

2.3.4 Producer (Παραγωγός)

Ο Kafka Producer είναι ένας client υπεύθυνος για τη διοχέτευση των μηνυμάτων στην Kafka Queue. Υπάρχουν πολλές υλοποιήσεις του συγκεκριμένου πρωτοκόλλου, ωστόσο στην παρούσα μελέτη, έγινε επέκταση του Confluent Kafka python producer. Εφόσον το Apache Kafka βρίσκεται στον πυρήνα της πλατφόρμας Conluent, διασφαλίζουμε τη συμβατότητα με τα νέα εργαλεία του Apache Kafka.

Η Confluent Kafka python είναι μία βιβλιοθήκη η οποία λειτουργεί ως ένας wrapper της librdkafka, βιβλιοθήκης σε C για την αξιόπιστη και αποδοτική μεταφορά μηνυμάτων.

Οι μετρικές αποθηκεύονται σε μία εσωτερική ουρά μέχρι να ολοκληρωθεί η εγγραφή στο Kafka. Το μέγιστο μέγεθος της ουράς ορίζεται από το Collectd output plugin. Δεν υλοποιήθηκε κάποια στρατηγική partitioning ενώ επιλέχθηκε τα μηνύματα να «καθαρίζονται» μετά από συγκεκριμένο χρονικό διάστημα. Τα μηνύματα αποστέλλονται σε συμπιεσμένα τμήματα και θεωρούνται ως υποβληθέντα, μόνο όταν αποθηκευτούν σε όλους τους kafka brokers, οι οποίοι περιέχουν τα αντίγραφα του και έχουν ρυθμιστεί από το collectd plugin. Στην περίπτωση που η αποστολή ενός μηνύματος αποτύχει, το μήνυμα

απορρίπτεται. Εάν η εσωτερική ουρά του παραγωγού είναι πλήρης, όλα τα νέα μηνύματα απορρίπτονται μέχρι κάποιο από τα υπάρχοντα μηνύματα παραδοθεί ή αν περατωθεί το ανώτατο χρονικό όριο αποστολής (timeout).

2.3.5 Message Queue (ουρά μηνυμάτων)

Για την περίπτωση που εξετάζεται, επιλέχθηκε το Apache Kafka ως Message Queue το οποίο είναι μία κατανεμημένη ουρά μηνυμάτων. Ο συντονισμός των brokers του kafka επιτυγχάνεται με τη χρήση του Apache Zookeeper, υπηρεσία κατάλληλη για το συντονισμό κατανεμημένων εφαρμογών.

2.3.5.1 Zookeeper

Το Zookeeper είναι ένα εργαλείο κατάλληλο για την ανάπτυξη κατανεμημένων εφαρμογών. Δίνεται η δυνατότητα ανάπτυξης υπηρεσιών σε υψηλότερο επίπεδο με έμφαση στη συντήρηση και στο συγχρονισμό. Το μοντέλο δεδομένων βρίσκεται μετά τη δενδρική δομή του συστήματος αρχείων και είναι σχεδιασμένο με τρόπο κατάλληλο για προγραμματισμό.

Το Zookeeper χρησιμοποιείται για την αποδέσμευση κατανεμημένων εφαρμογών, όπως το Kafka, από τη δημιουργία νέων μηχανισμών για το συντονισμό των υπηρεσιών. Επιπλέον, παρέχει όπως ένα σύστημα αρχείων, ένα ιεραρχικό ονοματοχώρο στον οποίο κάθε κόμβος είναι προσβάσιμος με ένα μονοπάτι που τον χαρακτηρίζει.

Η διαφορά με ένα σύστημα αρχείων, στο οποίο υπάρχει η διάκριση μεταξύ διευθύνσεων και αρχείων, είναι ότι ο ονοματοχώρος του μπορεί να έχει τόσο δεδομένα όσο και παιδιά. Οι κόμβοι αυτοί ονομάζονται znodes και διατηρούν μία δομή με τις διάφορες εκδόσεις των υποσυστημάτων τους για ελέγχους στην cache αλλά και τη δυνατότητα αναβάθμισης (update). Τα δεδομένα αποθηκεύονται σε αυτά και μπορούν να ανακτηθούν είτε ατομικά είτε ως σύνολο. Κάθε znode έχει ένα Access Control List (ACL), το οποίο ελέγχει τις απαραίτητες αδειοδοτήσεις.

Όλοι οι εξυπηρετητές Zookeeper είναι συμβατοί με τους clients, ωστόσο κάθε φορά μόνο ένας μπορεί να δέχεται αιτήματα προς εξυπηρέτηση. Τα αιτήματα ανάγνωσης εξυπηρετούνται από το τοπικό αντίγραφο κάθε database server, ενώ ο χειρισμός των αιτημάτων εγγραφής ακολουθεί συγκεκριμένο πρωτόκολλο επικοινωνίας.

Τέλος, το Zookeeper λειτουργεί για το kafka ως μέρος αποθήκευσης των consumer offsets.

2.3.5.2 Apache Kafka

Το Apache Kafka είναι μία πλατφόρμα streaming, η οποία προσφέρει τρεις βασικές δυνατότητες:

- Εγγραφή και κοινοποίηση εγγραφών σε μορφή stream, όμοια με μία ουρά μηνυμάτων ή ένα σύστημα ανταλλαγής μηνυμάτων
- Αποθήκευση των streams με ένα ασφαλή και ανθεκτικό τρόπο
- Δυνατότητα επεξεργασίας των εγγραφών του stream

Στη συγκεκριμένη μελέτη, γίνεται αξιοποίηση της πρώτης δυνατότητας, αφού το Kafka ενεργεί ως cluster σε ένα ή περισσότερες εξυπηρετητές που μπορεί να βρίσκονται ακόμα και σε διαφορετικά datacenters. Τα streams αποθηκεύονται σε κατηγορίες που ονομάζονται topics και κάθε εγγραφή συνοδεύεται με ένα κλειδί, μία τιμή και μία χρονική στιγμή.

2.3.5.2.1 Guarantees (Εγγυήσεις)

Το Apache Kafka παρέχει τις παρακάτω εγγυήσεις:

- Τα μηνύματα που αποστέλλονται από κάποιον παραγωγό σε κάποιο συγκεκριμένο topic partition θα τοποθετηθούν με τη σειρά με την οποία εστάλησαν. Για παράδειγμα, αν μία εγγραφή M1 σταλεί από τον ίδιο παραγωγό ως M2 και το M1 έχει σταλεί πρώτο, το M1 θα έχει μικρότερο offset και θα εμφανίζεται αργότερα στα αρχεία log.
- Ο καταναλωτής βλέπει τις εγγραφές με τη σειρά που είναι αποθηκευμένα στο log.
- Για ένα topic με N αντίγραφα, το Kafka αντέχει έως N-1 αποτυχίες του εξυπηρετητή χωρίς καμία απώλεια δεδομένων από το log.

2.3.5.2.2 Brokers

Broker ονομάζεται ένα στιγμιότυπο του Kafka. Ένα Kafka cluster περιέχει πολλούς brokers, οι οποίοι συντονίζονται μέσω του Zookeeper όπως αναφέρθηκε και προηγουμένως.

2.3.5.2.3 Topics και Logs

Ένα topic είναι μία κατηγορία της οποίας οι εγγραφές κοινοποιούνται. Ένα topic στο Kafka μπορεί να έχει κανέναν, έναν ή περισσότερους καταναλωτές και να περιλαμβάνει ένα ή περισσότερα partitions.

Ένα partition είναι μία ταξινομημένη, αμετάβλητη ακολουθία εγγραφών τα δεδομένα της οποία προστίθενται συνεχόμενα στα logs. Ένα partition log αποθηκεύεται στο Kafka cluster για κάθε topic.

Οι εγγραφές σε κάθε partition διαθέτουν ένα μοναδικό αναγνωριστικό που προκύπτει από έναν αύξων αριθμό και ονομάζεται offset. Οι εγγραφές διατηρούνται με ασφάλεια στο Kafka cluster, ανεξάρτητα με το αν καταναλώθηκαν, για κάποιο καθορισμένο χρονικό διάστημα.

Το διάστημα συγκράτησης όπως και ο χώρος μπορεί να είναι περιορισμένα, ωστόσο υπάρχει η δυνατότητα συγκράτησης των δεδομένων για μεγάλο χρονικό διάστημα χωρίς σημαντική μείωση της απόδοσης του Kafka.

2.3.5.2.4 Distribution

Τα log's partitions είναι κατανεμημένα στο Kafka cluster. Κάθε εξυπηρετητής χειρίζεται αιτήματα και δεδομένα για ένα μέρος των partitions, για τα οποία μπορεί να υπάρχουν αντίγραφα για την περίπτωση απώλειας δεδομένων, ο αριθμός των οποίων είναι προκαθορισμένος. Για κάθε partition ένας συγκεκριμένος εξυπηρετητής επιλέγεται ως «αρχηγός» ενώ οι υπόλοιποι λειτουργούν ως «ακόλουθοι».

2.3.5.2.5 Producer

Οι παραγωγοί κοινοποιούν τα δεδομένα στα topics της επιλογής τους και είναι υπεύθυνοι για την ανάθεση εγγραφών στα partitions.

2.3.5.2.6 Consumer

Οι καταναλωτές αποδίδουν στον εαυτό τους ένα group name. Κάθε εγγραφή ενός topic αποστέλλεται σε έναν καταναλωτή του group και το μόνο μετα-δεδομένο που διατηρείται για αυτόν είναι ένα offset ή η θέση του στα αρχεία log.

Τα στιγμιότυπα καταναλωτών μπορεί να βρίσκονται σε διαφορετικές διεργασίες ή ακόμα και διαφορετικές μηχανές. Στην περίπτωση που εξετάζουμε οι καταναλωτές τρέχουν στις ίδιες μηχανές όπως και τα στιγμιότυπα του Kafka. Το Kafka υλοποιεί τη διαδικασία της κατανάλωσης, μέσω της διαίρεσης των partition στο αρχείο log, κατανέμοντας με όσο το δυνατό πιο ομοιόμορφο τρόπο τα partition στους καταναλωτές.

Το Kafka δυναμικά χειρίζεται τη συντήρηση των μελών στα διάφορα groups. Όταν ένα νέο στιγμιότυπο εισέρχεται στο group ή κάποιο υπάρχον «πεθαίνει», τα partitions ανακατανέμονται ώστε να διατηρηθεί η ομοιομορφία στην κατανομή τους.

2.3.5.2.7 Compression

Το Kafka προσφέρει τη δυνατότητα καθολικής συμπίεσης των blocks, η οποία όταν είναι ενεργοποιημένη, τα δεδομένα συμπιέζονται από τον παραγωγό και αφού αποθηκευτούν σε αυτή τη μορφή, αποσυμπιέζονται από τον καταναλωτή. Τρεις είναι οι αλγόριθμοι που υποστηρίζονται για συμπίεση και είναι οι gzip, snappy, lz4. Στη συγκεκριμένη περίπτωση, τα μηνύματα περιέχουν μία μεγάλη ποσότητα επαναλαμβανόμενης πληροφορίας διότι τα Metrics είναι του ίδιου τύπου, με κοινά tags keys και διαμοιραζόμενα tag values. Τα μηνύματα συμπιέζονται τόσο στον παραγωγό όσο και στον broker, με αποτέλεσμα να μειώνεται η δικτυακή κίνηση στον παραγωγό καθώς ο απαιτούμενος αποθηκευτικός χώρος. Η συμπίεση αυτή βέβαια, απαιτεί επιπλέον CPU, στοιχείο που οδηγεί στην ανάγκη για επιλογή της καλύτερης μεθόδου για αυτό το σκοπό. Μετά από πειραματική αξιολόγηση, καταλήξαμε ότι ο αλγόριθμος snappy είναι ο καταλληλότερος αφού παρουσιάζει το καλύτερο trade-off μεταξύ απαιτούμενης ισχύος και ποσοστού συμπίεσης κοντά στο 30%. Το ποσοστό συμπίεσης αποτελεί πολύ σημαντική παράμετρο για το συνολικό σύστημα, διότι επηρεάζει τη διάρκεια συγκράτησης των μηνυμάτων και συνεπακόλουθα τη ανθεκτικότητά του σε σφάλματα.

2.3.5.2.8 Protocol/API

(Πηγή: https://cwiki.apache.org/confluence/display/KAFKA/A+Guide+To+The+Kafka+Protocol)

API

Το πρωτόκολλο του Kafka έχει έξι κύρια client requests APIs:

- Metadata – Περιγράφει τους διαθέσιμους brokers, τους εξυπηρετητές και τις πόρτες επικοινωνίας. Παρέχει ακόμα πληροφορίες σχετικά με τα partitions του broker.
- Send – Στέλνει μηνύματα σε ένα broker, ενώ επιτρέπει την αποστολή συνόλων μηνυμάτων με πολλά topic partition στην ίδια κλήση.
- Fetch – Λαμβάνει μηνύματα από ένα broker, ανακτά τα δεδομένα, λαμβάνει τα μετα-δεδομένα του cluster και παίρνει το offset του topic.
- Offset - Λαμβάνει τις πληροφορίες σχετικά με τα διαθέσιμα offsets για κάθε διαθέσιμο topic partition.
- Offset Commit – Δεσμεύει ένα σύνολο από offsets για ένα group καταναλωτών.
- Offset Fetch – Ανακτά ένα σύνολο από offsets για ένα group καταναλωτών.

Network

Το Kafka χρησιμοποιεί ένα δυαδικό πρωτόκολλο πάνω από το TCP. Το πρωτόκολλο ορίζει όλα τα apis σαν ζευγάρια μηνυμάτων κλήσης/απάντησης. Ο client αρχικοποιεί ένα socket για την επικοινωνία, έπειτα γράφει μία ακολουθία από μηνύματα κλήσεων και στη συνέχεια διαβάζει τις απαντήσεις.

Το Kafka είναι ένα σύστημα partitioning οπότε σε καμία χρονική στιγμή όλοι οι εξυπηρετητές δεν έχουν τα πλήρη δεδομένα. Σε τέτοια συστήματα προκύπτει το εύλογο ερώτημα της επιλογής μεθόδου για κατανομή των δεδομένων. Στο Kafka οι clients άμεσα ελέγχουν την επιλογή, αφού αντί να κοινοποιούν τα μηνύματα, αυτοί αναθέτουν τα μηνύματα σε συγκεκριμένα partitions και τα ανακτούν από συγκεκριμένα partitions. Αυτά τα αιτήματα απευθύνονται απευθείας στον broker που δρα ως «αρχηγός» στο partition.

Κάθε broker μπορεί να απαντήσει σε ένα ερώτημα αναφορικά με την παρούσα κατάσταση του cluster:

- Ποια topics υπάρχουν,
- Ποια partitions έχουν αυτά τα topics,
- Ποιος broker είναι ο «αρχηγός» για αυτά τα partitions,
- Ποιοι είναι οι εξυπηρετητές και οι πόρτες επικοινωνίας για αυτούς τους brokers.

Ο client δεν χρειάζεται να ελέγχει διαρκώς την κατάσταση του cluster, αφού μετά τη λήψη των στοιχείων κατά την αρχικοποίηση, θα λάβει ένδειξη σφάλματος σε περίπτωση που η κατάσταση αλλάξει.

Partitioning

Η ανάθεση μηνυμάτων σε partitions είναι κάτι το οποίο ο παραγωγός ελέγχει. Το partitioning στο Kafka ικανοποιεί τους παρακάτω σκοπούς:

- Εξισορροπεί τα δεδομένα και το φορτίο των κλήσεων στους brokers
- Υλοποιεί το partitioning, αφού κατακερματίζει τη συνολική επεξεργασία στις διεργασίες καταναλωτές, ενώ διατηρεί την τοπική κατάσταση και τη διάταξη μέσα στο partition.

Στην παρούσα υλοποίηση δεν υλοποιήθηκε κάποιος ξεχωριστός μηχανισμός partitioning, αλλά χρησιμοποιήθηκε η προκαθορισμένη βιβλιοθήκη librdkafka, η οποία παράγει τυχαία partition επιστρέφοντας το ακέραιο πολλαπλάσιο της συνάρτησης rand της C με τον αριθμό των partitions.

Batching

Μία πολύ σημαντική αύξηση στην επίδοση του συστήματος επιτυγχάνεται εξαιτίας του Kafka message's batching.Τα APIs αποστολής και λήψης μηνυμάτων δουλεύουν πάντα για μία ακολουθία μηνυμάτων και όχι για ένα μήνυμα μειώνοντας έτσι το χρόνο απόκρισης.

Versioning

Το πρωτόκολλο είναι σχεδιασμένο ώστε να επιτρέπει τη συμβατότητα με προηγούμενες εκδόσεις. Η συμβατότητα υποστηρίζεται για κάθε ξεχωριστό API και κάθε έκδοση αποτελείται από ένα ζεύγος κλήσης/απάντησης. Κάθε κλήση περιέχει ένα API key το οποίο αναγνωρίζει το ζητούμενο API και τον αριθμό της έκδοσης που υποδεικνύει τη μορφή της κλήσης και της αναμενόμενης απάντησης. Ο εξυπηρετητής απορρίπτει μηνύματα τα οποία η έκδοση δεν υποστηρίζεται και απαντά πάντα στη μορφή που ορίζει το συγκεκριμένο πρωτόκολλο.

2.3.5.2.9 Configuration

Η πολυπλοκότητα του Apache Kafka, καθιστά δύσκολη τη ρύθμισή του και αποτελεί ξεχωριστή διαδικασία για κάθε περίπτωση χρήσης. Στη συνέχεια παρουσιάζονται οι ρυθμίσεις για τη παρούσα περίπτωση χρήσης.

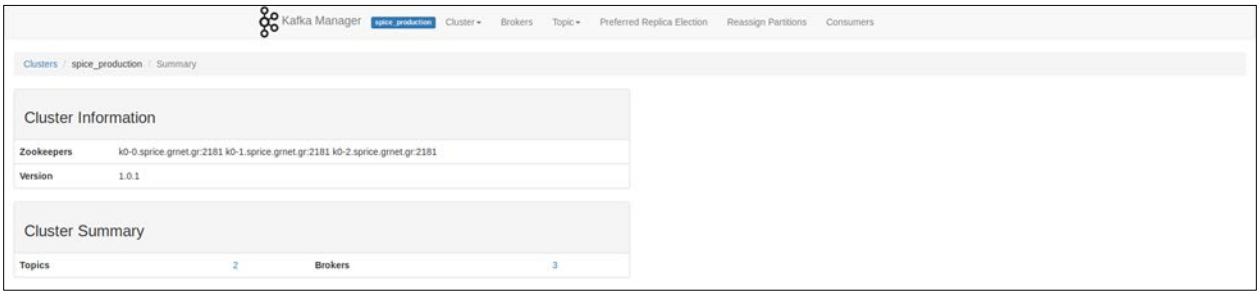
Το kafka cluster αποτελείται από 3 κόμβους. Η συγκράτηση των δεδομένων επιτυγχάνεται με την αντιγραφή των δεδομένων σε όλους τους brokers θέτοντας ενεργή την επιλογή "default.replication.factor" και ορίζοντας τη παράμετρο "acks" ίση με "all". Θέτοντας τη "min.isync.replicas" ίση με δύο, τουλάχιστον δύο brokers είναι συγχρονισμένοι υπό την έννοια ότι τα partitions τους βρίσκονται στο ίδιο offset . Έτσι, εάν ένας broker αποτύχει, δεν θα υπάρχει απώλεια δεδομένων, ενώ αν περισσότεροι αποτύχουν τα νέα μηνύματα θα απορριφθούν, επιτυγχάνοντας μία ισορροπία μεταξύ συνέπειας και διαθεσιμότητας. Τέλος, η επιλογή "log.retention.bytes" χρησιμοποιείται για τη ρύθμιση της πολιτικής συγκράτησης, επιλογή η οποία εφαρμόζεται ανά partition και είναι ίση με DISK_SIZE / NUMBER_OF_PARTITIONS.

2.3.5.3. Kafka Manager

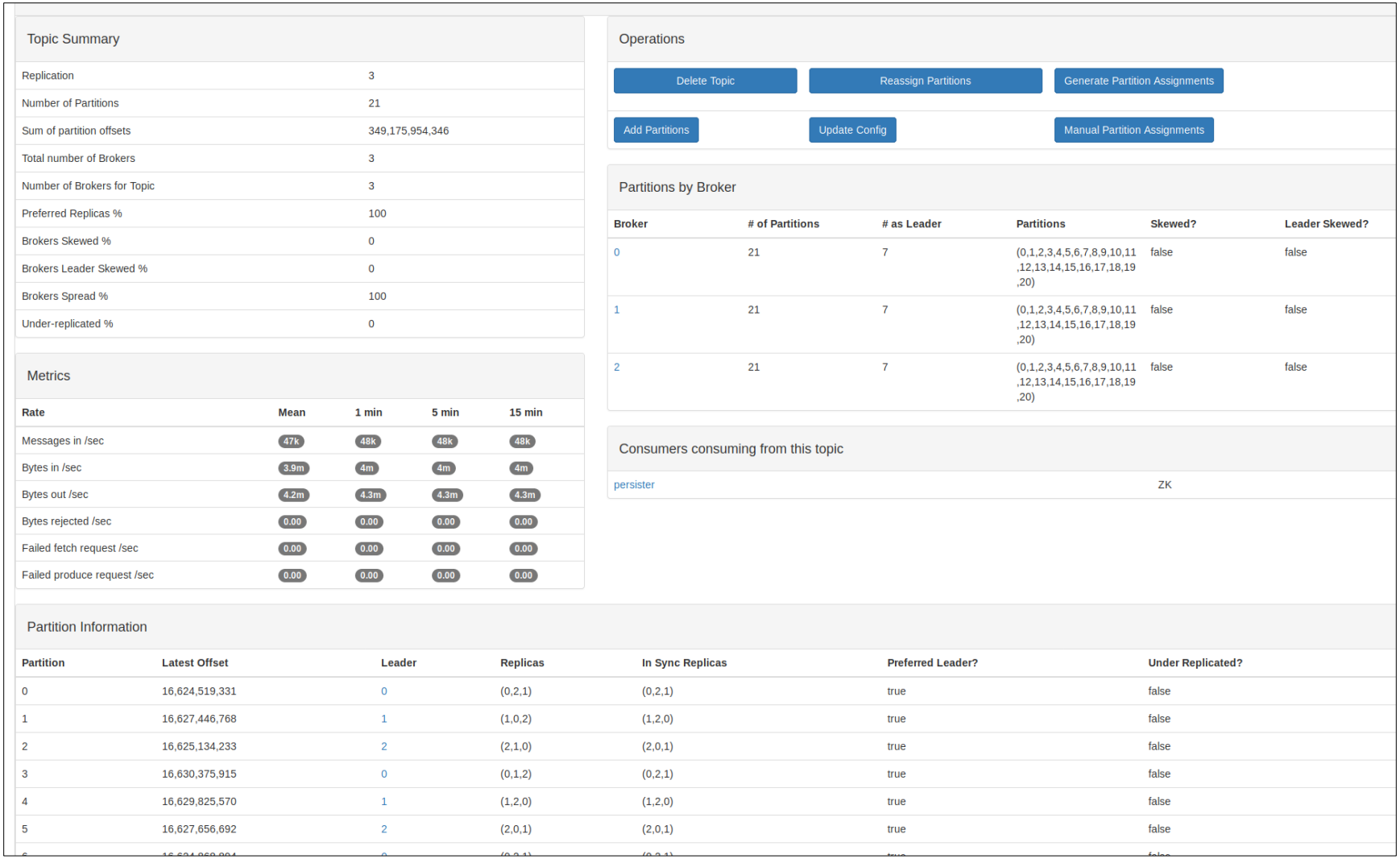
Για τον έλεγχο και τη διαχείριση της Kafka queue χρησιμοποιήθηκε το εργαλείο Kafka Manager. Το συγκεκριμένο εργαλείο παρέχει μία διεπαφή χρήστη (web UI), με το οποίο υπάρχει η δυνατότητα διαχείρισης πολλών kafka cluster. Συγκεκριμένα, υπάρχει η δυνατότητα προβολής της κατάστασης του cluster, των topics, των καταναλωτών καθώς και τα offsets τους. Επιπλέον, υπάρχει η δυνατότητα δημιουργίας ή διαγραφής topic, η αλλαγή γενικών ή τοπικών ρυθμίσεων, η προσθήκη partition και η εκ νέου εξισορρόπηση των υπολοίπων. Τέλος, ενεργοποιώντας την επιλογή JMX στο Kafka brokers Kafka Manager παρέχονται μετρικές σε επίπεδο broker και topic, όπως είναι τα ποσοστά παραγωγής και κατανάλωσης.

Στη συνέχεια παρουσιάζονται κάποιες εικόνες από το Kafka Manager που χρησιμοποιήθηκε για το παραγωγικό μας Kafka cluster.

Εικόνα 4 Αρχική Σελίδα



Εικόνα 5 Πληροφορίες για τα clusters



Topic Summary

Replication	3
Number of Partitions	21
Sum of partition offsets	349,175,954,346
Total number of Brokers	3
Number of Brokers for Topic	3
Preferred Replicas %	100
Brokers Skewed %	0
Brokers Leader Skewed %	0
Brokers Spread %	100
Under-replicated %	0

Metrics

Rate	Mean	1 min	5 min	15 min
Messages in /sec	47k	48k	48k	48k
Bytes in /sec	3.9m	4m	4m	4m
Bytes out /sec	4.2m	4.3m	4.3m	4.3m
Bytes rejected /sec	0.00	0.00	0.00	0.00
Failed fetch request /sec	0.00	0.00	0.00	0.00
Failed produce request /sec	0.00	0.00	0.00	0.00

Operations

Delete Topic | Reassign Partitions | Generate Partition Assignments

Add Partitions | Update Config | Manual Partition Assignments

Partitions by Broker

Broker	# of Partitions	# as Leader	Partitions	Skewed?	Leader Skewed?
0	21	7	(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)	false	false
1	21	7	(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)	false	false
2	21	7	(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)	false	false

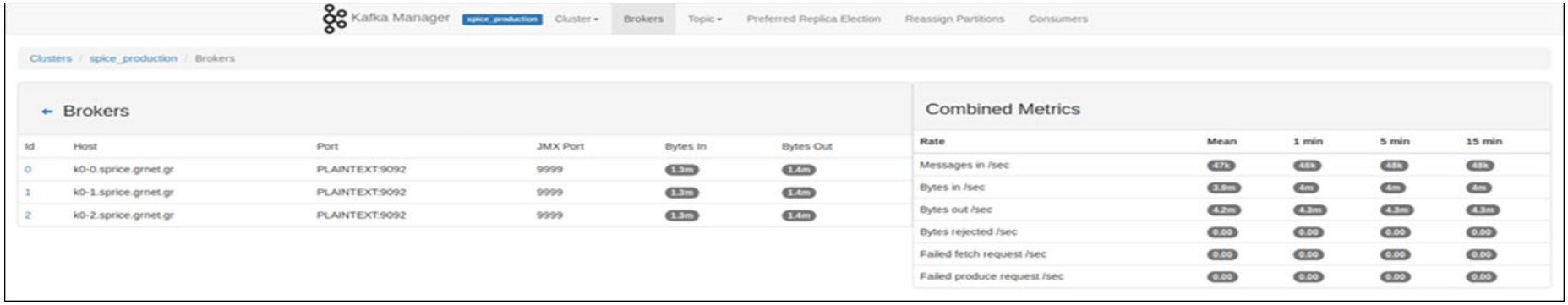
Consumers consuming from this topic

persister	ZK

Partition Information

Partition	Latest Offset	Leader	Replicas	In Sync Replicas	Preferred Leader?	Under Replicated?
0	16,624,519,331	0	(0,2,1)	(0,2,1)	true	false
1	16,627,446,768	1	(1,0,2)	(1,2,0)	true	false
2	16,625,134,233	2	(2,1,0)	(2,0,1)	true	false
3	16,630,375,915	0	(0,1,2)	(0,2,1)	true	false
4	16,629,825,570	1	(1,2,0)	(1,2,0)	true	false
5	16,627,656,692	2	(2,0,1)	(2,0,1)	true	false

Εικόνα 6 Πληροφορίες που αναφέρονται στην κατάσταση των brokers



Εικόνα 7 Πληροφορίες για τους brokers όπως τα bytes αποστολής και λήψης

2.3.6 Consumers (Καταναλωτές)

Οι καταναλωτές διαβάζουν τις μετρικές από την Message queue και τα αποθηκεύουν στη βάση δεδομένων. Για την βελτίωση της επίδοσης του συστήματος, η αποθήκευση γίνεται μόνο όταν σωρευθούν αρκετά δεδομένα. Για τη λειτουργία αυτή χρησιμοποιείται το Openstack Monasca Persister, το οποίο έχει τη δυνατότητα να διαβάζει από διάφορες ουρές και να γράφει αντίστοιχα σε πλήθος διαφορετικών βάσεων δεδομένων. Δημιουργήθηκε κλάση που υλοποιεί την AbstractInfluxdbRepository, ένα API που μετατρέπει τα μηνύματα από τη message queue σε μορφή κατάλληλη για εισαγωγή στη βάση inluxdb. Το SpriceMetricInfluxdbRepository επίσης συλλέγει δεδομένα σχετικά με τα μηνύματα που καταχωρήθηκαν στη βάση δεδομένων.

Για να διασφαλιστεί ότι δεν χάθηκε κάποιο μήνυμα, το Kafka ενημερώνεται για την ολοκλήρωση της μεταφοράς μόνο όταν τα δεδομένα εγγραφούν στη βάση δεδομένων. Για την οριζόντια κλιμάκωση της διαδικασίας, πολλά Persister instances ξεκινούν και το κάθε ένα διαβάζει από τουλάχιστον ένα partition από το ίδιο consumer group. Στη περίπτωσή μας δημιουργήσαμε τόσους καταναλωτές όσα και τα διαθέσιμα partitions και τοποθετήθηκαν προς εκτέλεση στις ίδιες μηχανές όπως το Kafka και το Zookeeper.

2.3.7 InfluxDB

Η InfluxDB είναι μία βάση δεδομένων κατάλληλη για την αποθήκευση χρονοσειρών, δηλαδή μεγάλου όγκου δεδομένα μαζί με το χρονικό τους αποτύπωμα. Κάποια βασικά χαρακτηριστικά της influxDB είναι τα παρακάτω:

- SQL-like querying language
- HTTP API
- data sharding
- retention policies
- Continuous queries

Τα δεδομένα σε μία InfluxDB αποτελούνται από ένα κλειδί, μία χρονική στιγμή και τα tags. Στο πεδίο του κλειδιού βρίσκονται τα δεδομένα που πρέπει να αποθηκευτούν και είναι πάντα συνδεδεμένα με μία χρονική στιγμή. Τα tags έχουν ένα κλειδί και μία τιμή που χρησιμοποιούνται για την καταγραφή μετα-δεδομένων. Τα δεδομένα οργανώνονται σε μετρήσεις, όπου η κάθε μέτρηση ενεργεί ως container για ένα δεδομένο. Κάθε μέτρηση έχει ένα χαρακτηριστικό όνομα, ενώ μπορεί να ισχύουν διαφορετικές πολιτικές συγκράτησης των δεδομένων. Μία πολιτική συγκράτησης περιγράφει το χρονικό διάστημα για το οποίο διατηρούνται τα δεδομένα στη βάση καθώς και για τον αριθμό των αντιγράφων στο cluster.

Στην παρούσα υλοποίηση, ένα απλό στιγμιότυπο της InfluxDB χρησιμοποιείται, με μία σχετικά μικρή περίοδο συγκράτησης (μία εβδομάδα). Οι μετρήσεις δημιουργούνται δυναμικά βάσει των Metrics, μετατρέπονται σε influx format με το metric_name, διαβάζουν την τιμή και μετατρέπουν όλα τα πεδία (_aggr_type,_metric_type και tags) σε influx tags. Χρησιμοποιώντας μόνο μία βάση δεδομένων η διαδικασία εγγραφής γίνεται πιο απλή καλύπτοντας ταυτόχρονα τις ανάγκες της συγκεκριμένης μελέτης.

Σε περίπτωση που απαιτείται κλιμάκωση της πλατφόρμας, υπάρχει η δυνατότητα χρησιμοποίησης περισσότερων βάσεων δεδομένων. Κάθε βάση μπορεί να αποθηκεύει διαφορετικές μετρικές, για παράδειγμα μία βάση να αποθηκεύει τις μετρικές που αφορούν την CPU και άλλη αυτές του δικτύου. Μία άλλη στρατηγική θα ήταν η κατανομή των μετρικών βάσει των αναγνωριστικών των στιγμιότυπων, ώστε κάθε βάση δεδομένων να περιέχει μετρικές που αφορούν την ίδια μηχανή. Με αυτό τον τρόπο θα είναι αποδοτικότερη η ανάκτηση των δεδομένων που αφορούν την ίδια εικονική μηχανή ή cluster.

Οι αρχικές μετρικές αποθηκεύονται για μία μικρή χρονική περίοδο διότι λόγω της ποιότητάς έχουν μεγάλες απαιτήσεις σε δίσκο. Με σκοπό να το περιορίσουμε διενεργείται δειγματοληψία στα δεδομένα ώστε να μπορούν να αποθηκευτούν για μεγαλύτερο χρονικό διάστημα. Για τον ορισμό της μεθόδου δειγματοληψίας για κάθε μέτρηση χρησιμοποιείται το “_aggr_type” tag, το οποίο περιέχει το όνομα της InluxDB συνάρτησης που χρησιμοποιείται για την υποδειγματοληψία συγκεκριμένων μετρικών. Σημειώνεται ότι εξαιτίας της αρχιτεκτονικής , τα δεδομένα δεν καταγράφονται σειριακά, όπως για παράδειγμα συμβαίνει στην περίπτωση που κάποιος καταναλωτής περιμένει να ολοκληρώσει την

εγγραφή κάποιος άλλος με αποτέλεσμα τα δεδομένα να αποθηκεύονται πολύ αργότερα από τη στιγμή της παραγωγής τους. Για την αποφυγή επομένως της απώλειας δεδομένων υποδειγματοληπτούμε τα πιο πρόσφατα δεδομένα συχνότερα και περιοδικά επαναδειγματοληπτούμε όλα τα δεδομένα ανάλογα με τη πολιτική συγκράτησης.

Η InfluxDB παρέχει τη δυνατότητα υποβολής Continuous Queries (CQs), δηλαδή ερωτημάτων που τρέχουν περιοδικά, μετασχηματίζουν τα δεδομένα των μετρήσεων και τα αποθηκεύουν σε κάποια άλλη μέτρηση. Εκτελούνται ακολουθιακά όπως και τα υπόλοιπα ερωτήματα με αποτέλεσμα να παρουσιάζουν την ίδια συμπεριφορά και να μην μπορούν να κλιμακωθούν καλά.

Για την αντιμετώπιση αυτών των περιπτώσεων δημιουργήθηκε ένα εργαλείο το οποίο χρησιμοποιεί το InfluxDB ‘s HTTP API για την υποδειγματοληψία και εκτελεί τα ίδια ερωτήματα με τα αντίστοιχα CQs. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι τα ερωτήματα εκτελούνται ανά μέτρηση και το χρονικό εύρος στο οποίο λαμβάνει χώρα η δειγματοληψία χωρίζεται σε μικρότερα μέρη, διασφαλίζοντας ότι υπάρχει αρκετός χώρος στη μνήμη. Επιπλέον, προτού εκτελεστεί κάποιο ερώτημα δειγματοληψίας ελέγχεται αν τα νέα δεδομένα έχουν υποστεί δειγματοληψία σε κάποιο προηγούμενο διάστημα. Το συγκεκριμένο script εκτελείται τακτικά με *cron jobs* , προσφέροντας έτσι μεγαλύτερη ευελιξία σε σχέση με τα CQs.

2.3.8 Grafana

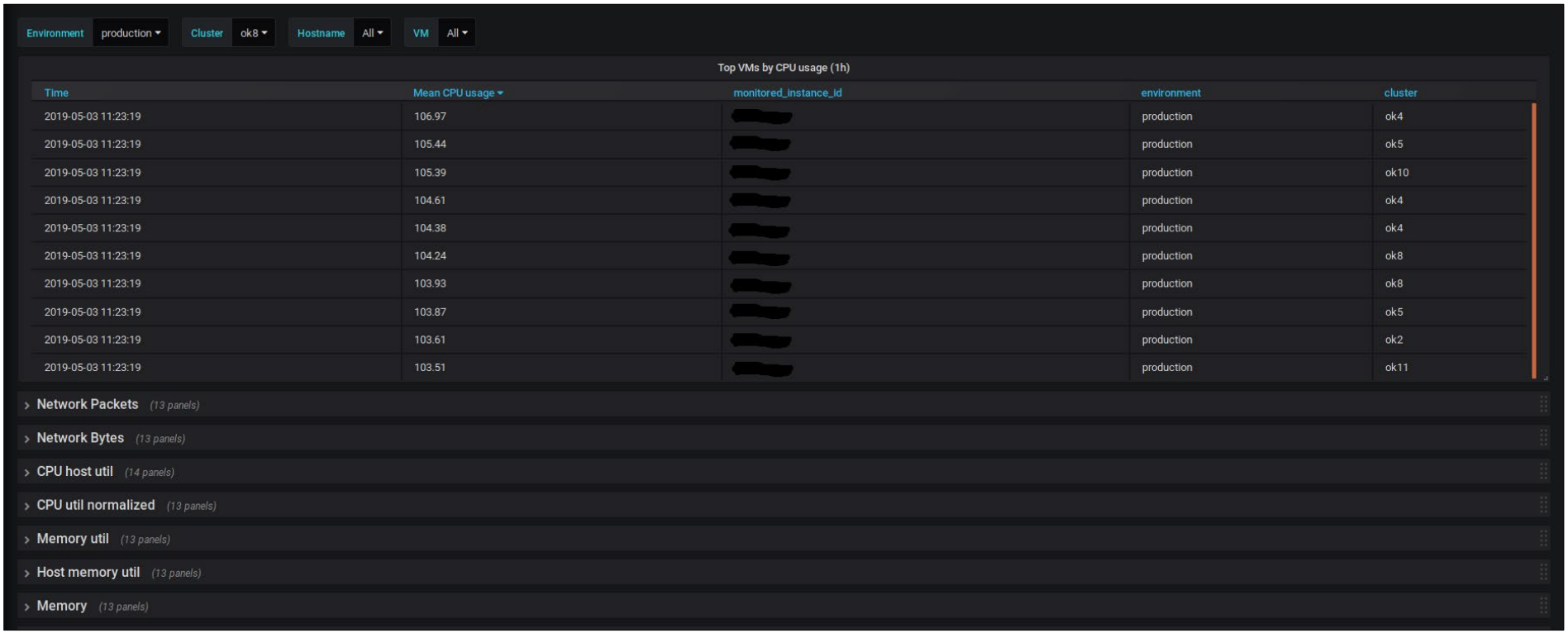
Για την οπτικοποίηση των δεδομένων της InfluxDB χρησιμοποιήθηκε το εργαλείο Grafana. Το Grafana είναι μία σουίτα ανοιχτού λογισμικού για αναλυτική και οπτικοποίηση μετρικών. Μπορεί να υποβάλλει ερωτήματα σε ένα backend και να δημιουργήσει γραφήματα από τα δεδομένα βάσει προκαθορισμένων template. Για κάθε ένα από τα υποστηριζόμενα backend, το Grafana προσφέρει εξειδικευμένο περιβάλλον υποβολής ερωτημάτων ανάλογα με τη γλώσσα. Παρέχει επίσης τη δυνατότητα δημιουργίας χρηστών και καταχώριση αδειών ανά χρήστη.

Η παρούσα υλοποίηση δίνει πρόσβαση στα γραφήματα μόνο για τους διαχειριστές, ενώ μελλοντικά υπάρχει η πρόθεση για προβολή των γραφημάτων με κάποιες μετρικές των εικονικών μηχανών στους ίδιους τους χρήστες.

Τα dashboard που είναι διαθέσιμα είναι ένα για την οπτικοποίηση των μετρικών ολόκληρου του cluster και ένα για τις μετρικές που αφορούν μία συγκεκριμένη εικονική μηχανή.

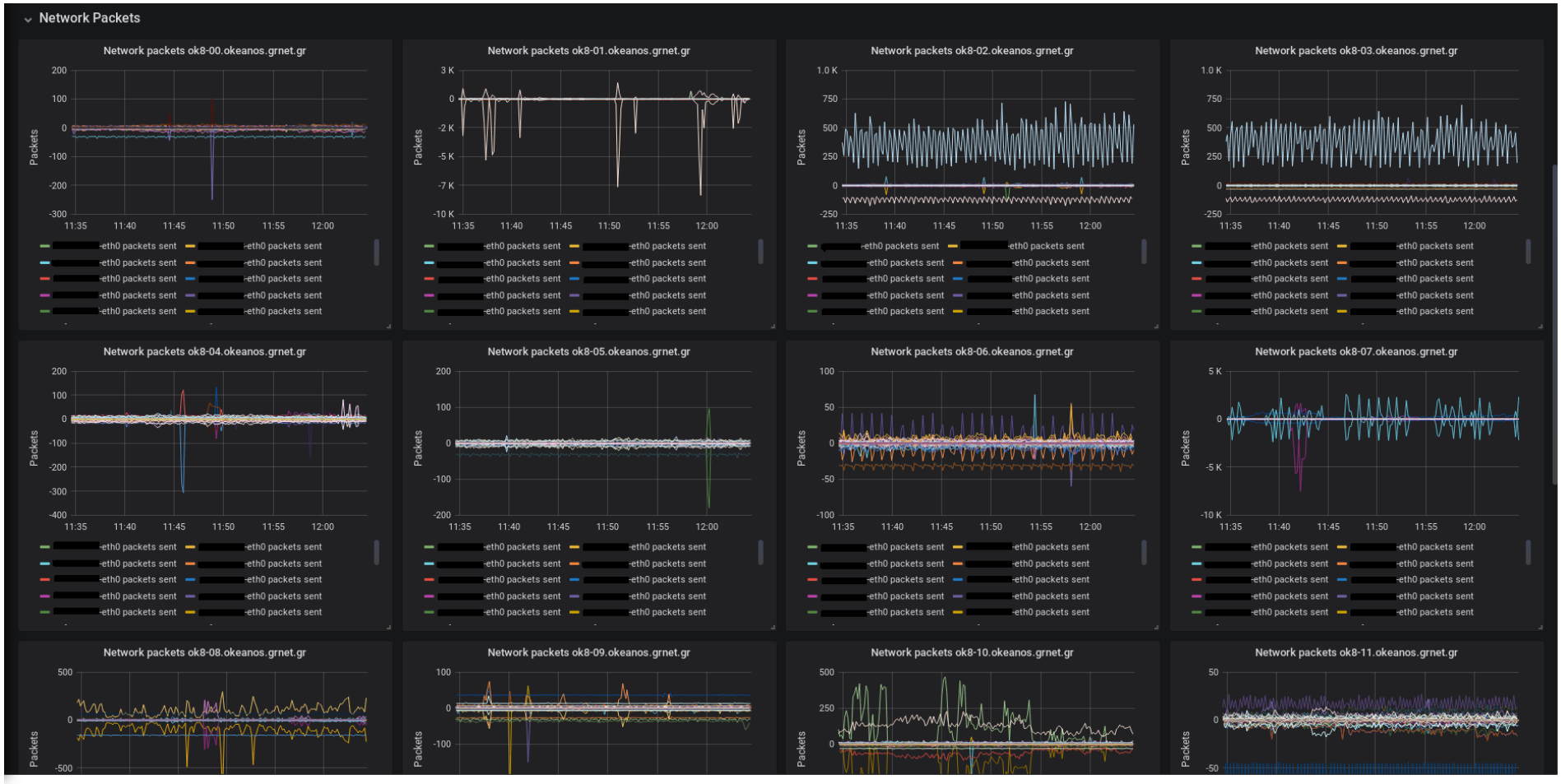
Το dashboard για το cluster προσφέρει ένα μενού που επιτρέπει στο διαχειριστή να διαλέξει το περιβάλλον και το cluster που επιθυμεί.

Μπορεί επίσης να επιλέξει ένα υποσύνολο εξυπηρετητών και εικονικών μηχανών ταξινομημένα με φθίνουσα κατανάλωση της CPU όπως φαίνεται παρακάτω:



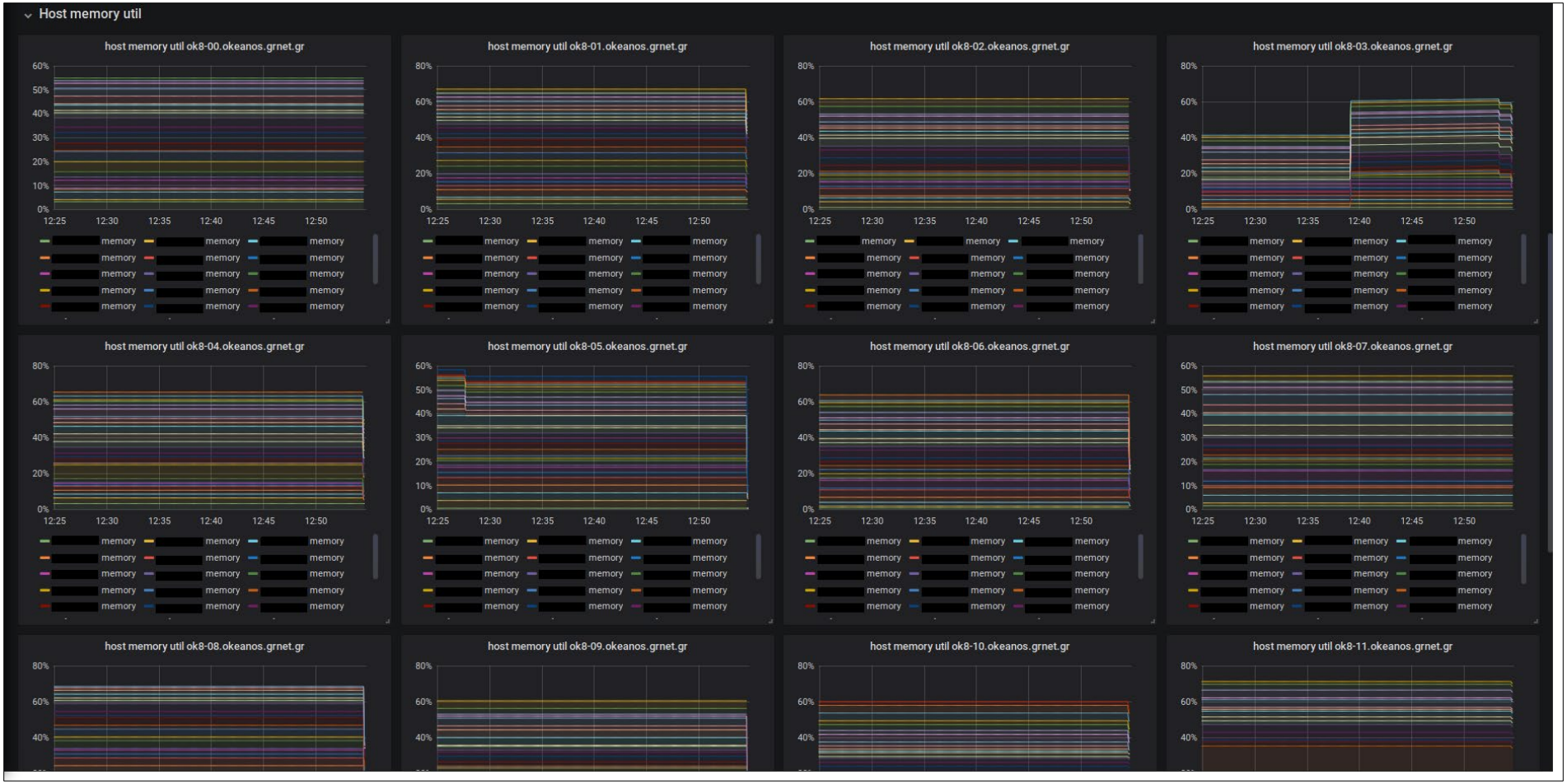
Εικόνα 8 Grafana - Μετρικές εξυπηρετητών και εικονικών μηχανών

Επιλέγοντας το cluster που θέλουμε να εξετάσουμε μπορούμε να δούμε τις μετρήσεις που αντιστοιχούν σε αυτό. Για πακέτα του δικτύου υπενθυμίζουμε ότι τα θετικά πακέτα υποδεικνύουν τα απεσταλμένα πακέτα ενώ τα αρνητικά αναφέρονται στα εισερχόμενα.



Εικόνα 9 Grafana - Μετρικές για την κίνηση στο δίκτυο (λήψη και αποστολή πακέτων)

Εικόνα 10 Grafana - Μετρικές για τη χρήση της CPU



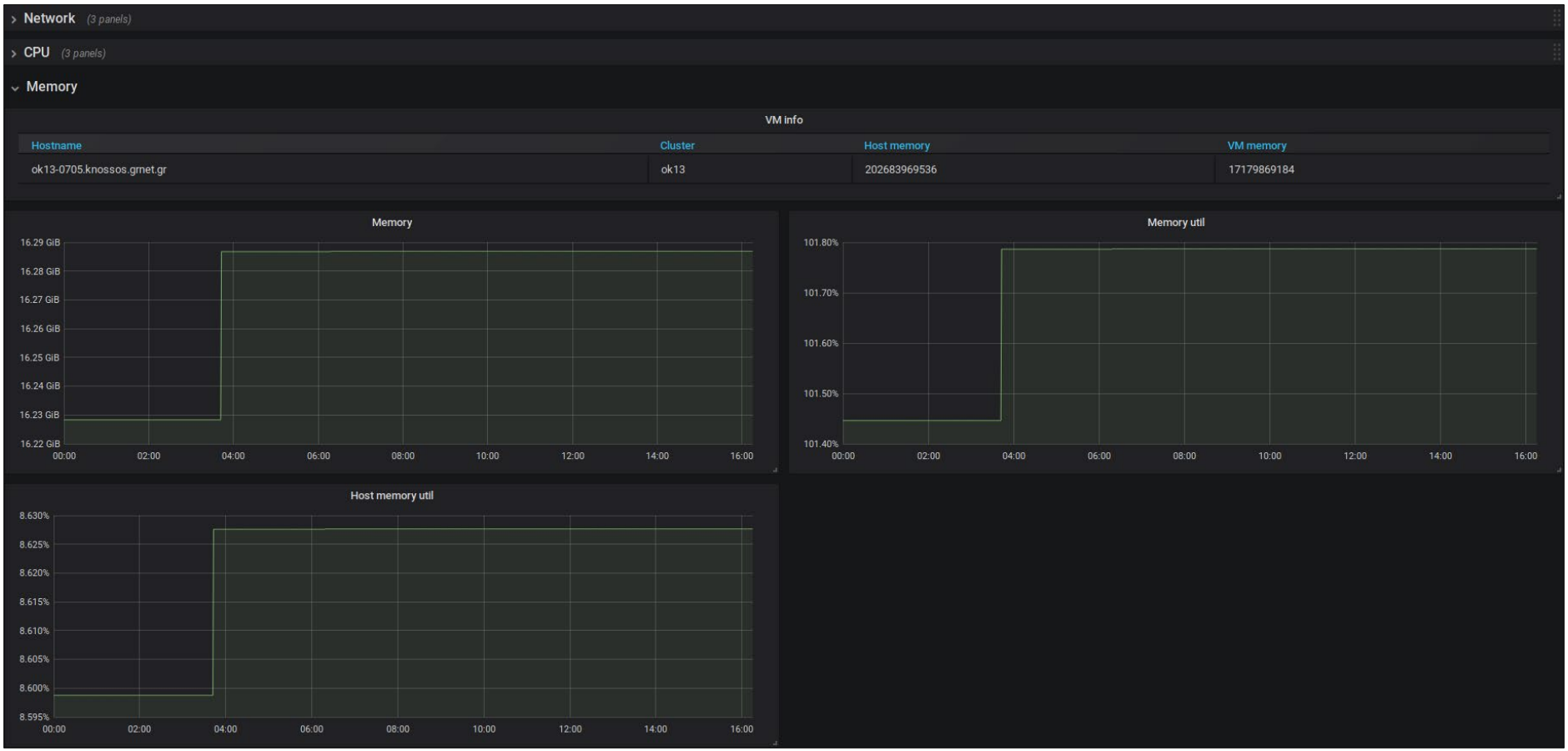
Εικόνα 11 Grafana - Μετρικές για τη χρήση της μνήμης

Το VM dashboard δίνει τη δυνατότητα προβολής μετρικών για συγκεκριμένο περιβάλλον και εικονική μηχανή. Παρέχονται γραφήματα για όλους τους πόρους που μελετήθηκαν, καθώς και επιπλέον πληροφορίες που αφορούν την εικονική μηχανή όπως το όνομα του εξυπηρετητή, το όνομα του cluster και οι vCPU της μηχανής.

Παραδείγματα τέτοιων γραφημάτων, που απεικονίζουν τα αποτελέσματα από την πειραματική λειτουργία, φαίνονται στις εικόνες που ακολουθούν:

Εικόνα 12 Grafana - Μετρικές δικτύου για μία συγκεκριμένη εικονική μηχανή

Εικόνα 13 Grafana - Μετρικές CPU για μία συγκεκριμένη εικονική μηχανή



Εικόνα 14 Μετρικές μνήμης για μία συγκεκριμένη εικονική μηχανή

3. Παραγωγική λειτουργία πλατφόρμας έξυπνης διαχείρισης ΕΝΕΔΗ

Στο συγκεκριμένο κεφάλαιο παρουσιάζονται οι μεθοδολογίες και οι αλγόριθμοι που υλοποιήθηκαν σε παραγωγικό openstack περιβάλλον. Παρουσιάζονται ακόμα, συγκριτικές μελέτες προσομοιώσεων και πραγματικών πειραμάτων, τα οποία πραγματοποιήθηκαν στην παραπάνω παραγωγική υποδομή. Στη συνέχεια, γίνεται περιγραφή του αλγορίθμου που υλοποιήθηκε, με σκοπό την δυναμική αποφόρτιση κόμβων και κατά συνέπεια τη μείωση του συνολικού ενεργειακού αποτυπώματος του συστήματος. Όπως σε όλα τα σύγχρονα συστήματα. βασική αρχή των μεθόδων για την ανακατανομή των υπολογιστικών πόρων είναι η διατήρηση του Quality of Service (QoS) σε όλα τα μεταβατικά στάδια της νεφοϋπολογιστική υποδομής.

3.1 Γνωστικό Υπόβαθρο

Με σκοπό τη διατήρηση του Quality of Service (QoS), έχουμε εισάγει την έννοια του Compute Unit (CU), τα οποία αναθέτουμε σε κάθε vCPUs. Τα CUs είναι εξασφαλισμένα για κάθε vCPU και αποτελούν το ελάχιστο ποσό υπολογιστικής (CPU) ισχύος, το οποίο μπορούν να απαιτήσουν. Η σπουδαιότερη συνεισφορά με την αναφερόμενη αρχιτεκτονική είναι ότι οι αδρανείς πόροι CPU μπορούν να χρησιμοποιηθούν από άλλες vCPUs, πέρα από τα δικά τους εγγυημένα CUs. Με αυτό τον τρόπο αξιοποιείται όλη η διαθέσιμη υπολογιστική ισχύ, όπου κρίνεται απαραίτητη από τους διαχειριστές της υποδομής.

Όπως αναφέρεται και στα προηγούμενα κεφάλαια, για την αποτελεσματικότερη διαχείριση των πόρων, τα φορτία διακρίνονται σε δύο κατηγορίες. Τα φορτία της πρώτης κατηγορίας πρέπει άμεσα να εξυπηρετηθούν, αφού παράγουν αποτελέσματα και επιστρέφουν δεδομένα σε κάποιο αίτημα. Παράδειγμα τέτοιων φορτίων αποτελούν τα αιτήματα σε εξυπηρετητές εφαρμογών ή βάσεων δεδομένων (application or database servers). Για τα φορτία αυτά, τα οποία ονομάζουμε reserved, η υποδομή αποδίδει άμεσα όλους τους απαιτούμενους πόρους που ζητούνται, σε όποιο κόμβο είναι διαθέσιμοι.

Τη δεύτερη κατηγορία αποτελούν τα φορτία τα οποία απαιτούν μεγάλη επεξεργασία και η άμεση εξυπηρέτηση δεν αποτελεί αυστηρή προϋπόθεση. Χαρακτηριστικό παράδειγμα τέτοιων φορτίων αποτελούν πειραματικές μελέτες και αναλύσεις που διενεργούνται σε μεγάλου όγκου δεδομένα. Τα φορτία αυτά ονομάζονται elastic, αφού χαρακτηρίζονται από την δυνατότητα εξυπηρέτησης με μικρότερη προτεραιότητα, σε χρόνο που οι υπολογιστικοί πόροι παρουσιάζουν μεγαλύτερη διαθεσιμότητα.

Η παραπάνω διάκριση των φορτίων δίνει τη δυνατότητα στην υποδομή να βελτιστοποιεί τη λειτουργία της, διασφαλίζοντας την άμεση εξυπηρέτηση των απαιτητικών reserved φορτίων και την συμπληρωματική εξυπηρέτηση των elastic φορτίων, όπου και σε χρόνο που η διαθεσιμότητα των διάφορων κόμβων το επιτρέπει.

3.2 Παράδειγμα παραγωγικής λειτουργίας

Το παραγωγικό περιβάλλον που εξετάστηκε αποτελείται από μία εγκατάσταση κόμβων openstack, αποτελούμενο από ένα ελεγκτή συστάδας, ο οποίος τρέχει όλες τις openstack υπηρεσίες και έξι κόμβους για τους υπολογισμούς. Οι έξι κόμβοι τρέχουν μόνο τις openstack worker υπηρεσίες.

Στη συνέχεια παρουσιάζεται η διαθεσιμότητα σε πόρους για τους έξι κόμβους υπολογισμού:

Name	CPU	CU	Memory(MB)	Disk (GB)
node-01	16	160	16045	236
node-02	16	160	16045	236
node-03	16	160	16045	236
node-04	16	160	16045	236
node-05	16	160	16045	236
node-06	16	160	16045	236

Στο openstack ένα flavor ορίζει την υπολογιστική χωρητικότητα, τη μνήμη και τον αποθηκευτικό χώρο των instances. Εναλλακτικά θα λέγαμε, ότι ορίζει το μέγεθος των εικονικών εξυπηρετητών που θα χρησιμοποιηθούν. Με αυτό το δεδομένο, ενσωματώνουμε στους πόρους τα CU, αντίστοιχα με την CPU, τη μνήμη και τον δίσκο, δημιουργώντας σαφώς ορισμένα flavors με προκαθορισμένα CUs. Για το συγκεκριμένο παράδειγμα χρησιμοποιήσαμε τις συγκεκριμένες κατηγορίες flavors:

Class	Name	CPU	CU	Memory(MB)	Disk (GB)
Reserved	c4u10m512	4	40	512	3
	c2u10m512	2	20	512	3
Elastic (50%)	c4u5m512	4	20	512	3
	c2u5m512	2	10	512	3
Elastic (20%)	c4u2m512	4	8	512	3
	c2u2m512	2	4	512	3

Για τα ελαστικά instances, υπολογίζουμε το ελάχιστο όριο εξυπηρέτησης (baseline) για τη CPU για κάθε flavor:

Flavor	Baseline CPU time	Maximum CPU time
c4u5m512	200	400
c2u5m512	100	200
c4u2m512	80	400
c2u2m512	40	200

Σε περίπτωση που ένας κόμβος υπολογισμού έχει υπερδεσμευτεί, τα elastic instances θα συμπιεστούν διατηρώντας όμως το κατώτερο όριο χρόνου CPU. Με την στρατηγική αυτή η υποδομή θα διατηρήσει το QoS και ταυτόχρονα δεν θα επηρεαστούν τα reserved φορτία που απαιτούν άμεση εξυπηρέτηση.

Διάρθρωση εγκατάστασης

Για την καλύτερη κατανόηση των αλγορίθμων για την αποδέσμευση των κόμβων και την εξοικονόμηση ενέργειας,, κατανέμουμε instances τυχαία σε όλους τους κόμβους υπολογισμού. Στη συνέχεια, για κάθε instance αποδίδουμε τυχαία κάποια πραγματικά φορτία, με γνώμονα τη διατήρηση μίας μέσης χρήσης της CPU σε κάθε ένα. Ο πίνακας που ακολουθεί περιγράφει λεπτομερώς τα αντίστοιχα ποσοστά χρήσης της CPU ανά κόμβο και instance.

node-01:	c2u5m512-61e7: 74.5%
	c2u2m512-76e6: 25.4%
	c4u10m512-442a: 7.4%
	c4u5m512-2865: 88.2%
	c4u5m512-0e89: 49.9%
	c4u5m512-0ad1: 48.7%
	c2u10m512-c42a: 1.4%
node-02:	c2u5m512-d3fe: 91.2%
	c2u5m512-d8b6: 49.8%
	c4u5m512-95a4: 98.1%
	c2u2m512-7918: 72.3%
	c4u5m512-5a90: 77.8%
	c2u2m512-9bc0: 93.4%
	c2u10m512-8000: 63.2%
	c2u5m512-2bdf: 68.8%
	c2u5m512-44dc: 5.1%
	c4u2m512-5a2e: 12.6%
	c4u10m512-ddbf: 18.7%

node-03:	c4u10m512-18ec: 47.0%
	c4u10m512-5818: 73.0%
	c4u2m512-d58b: 4.7%
	c2u5m512-d954: 4.2%
node-04:	c4u5m512-2468: 81.9%
	c2u2m512-2f5e: 5.8%
	c4u5m512-0842: 91.6%
	c4u5m512-846e: 41.1%
	c4u5m512-ea9d: 31.1%
	c2u5m512-9524: 30.1%
	c4u5m512-8b09: 43.5%
	c2u2m512-873a: 9.6%
	c4u2m512-5b0f: 43.3%
	c2u5m512-edf3: 94.3%
node-05:	c2u5m512-18b8: 88.7%
	c4u10m512-8bef: 35.2%
	c4u10m512-5b71: 19.7%
	c2u10m512-8913: 79.9%
	c4u5m512-3395: 68.9%
node-06:	c2u10m512-060b: 98.9%
	c4u10m512-1edf: 64.8%
	c2u5m512-e3b9: 23.7%
	c4u2m512-51c7: 59.9%
	c4u2m512-af6d: 91.5%
	c2u2m512-0c66: 87.4%
	c4u5m512-4f88: 19.6%
	c4u2m512-9533: 92.9%
	c2u2m512-f610: 56.5%
	c2u2m512-9cc1: 26.3%
	c4u2m512-4ffe: 77.5%

Συνολικά, για κάθε κόμβο προκύπτουν οι παρακάτω τιμές:

node-01[134] cpu: 975 (60.9%), idle: 545, excess: 211
node-02[116] cpu: 1637 (102.3%), idle: -117, excess: 672
node-03[138] cpu: 581 (36.3%), idle: 939, excess: 0
node-04[136] cpu: 1605 (100.3%), idle: -85, excess: 474
node-05[130] cpu: 827 (51.7%), idle: 693, excess: 152
node-06[134] cpu: 2205 (137.8%), idle: -685, excess: 1184

Το cpu εκφράζει το ποσοστό των πόρων που χρησιμοποιούνται στον κόμβο, ενώ η παράμετρος idle εκφράζει το αδρανές μέρος – απόθεμα του κάθε κόμβου σε CPU. Τέλος, το excess εκφράζει το συνολικό ποσοστό «συμπίεσης» που μπορούν να υποστούν τα elastic instances, χωρίς κάποιο από αυτά να περάσει το κάτω όριο. Διαφορετικά, θα μπορούσε να προσδιοριστεί ως το ποσοστό της CPU, το οποίο θα μπορούσε να αντικαταστήσει το idle time όταν τα instances μεταφέρονται σε ένα εξυπηρετητή. Όσο μεγαλύτερη είναι η τιμή του excess, τόσο περισσότερα elastic instances μπορούν να συμπιεστούν, καθώς νέα instances προστίθενται στον κόμβο.

Όπως παρατηρείται υπάρχουν κάποια instances στους κόμβους 02, 04 και 06, όπου αθροιστικά απαιτούν χρόνο στη CPU, μεγαλύτερο από τη χωρητικότητα του κόμβου (ποσοστό μεγαλύτερο του 100% ή αρνητικές τιμές του idle time). Τα instances σε αυτούς τους κόμβους πρέπει να συμπιεστούν ώστε να κανονικοποιηθεί η χρησιμοποιούμενη CPU. Μετά από αυτή την ενέργεια, η κατάσταση των κόμβων υπολογισμού παρουσιάζεται παρακάτω:

node-01[134] cpu: 975 (60.9%), idle: 545, excess: 211
node-02[116] cpu: 1519 (94.9%), idle: 1, excess: 554
node-03[138] cpu: 581 (36.3%), idle: 939, excess: 0
node-04[136] cpu: 1511 (94.4%), idle: 9, excess: 380
node-05[130] cpu: 827 (51.7%), idle: 693, excess: 152
node-06[134] cpu: 1517 (94.8%), idle: 3, excess: 496

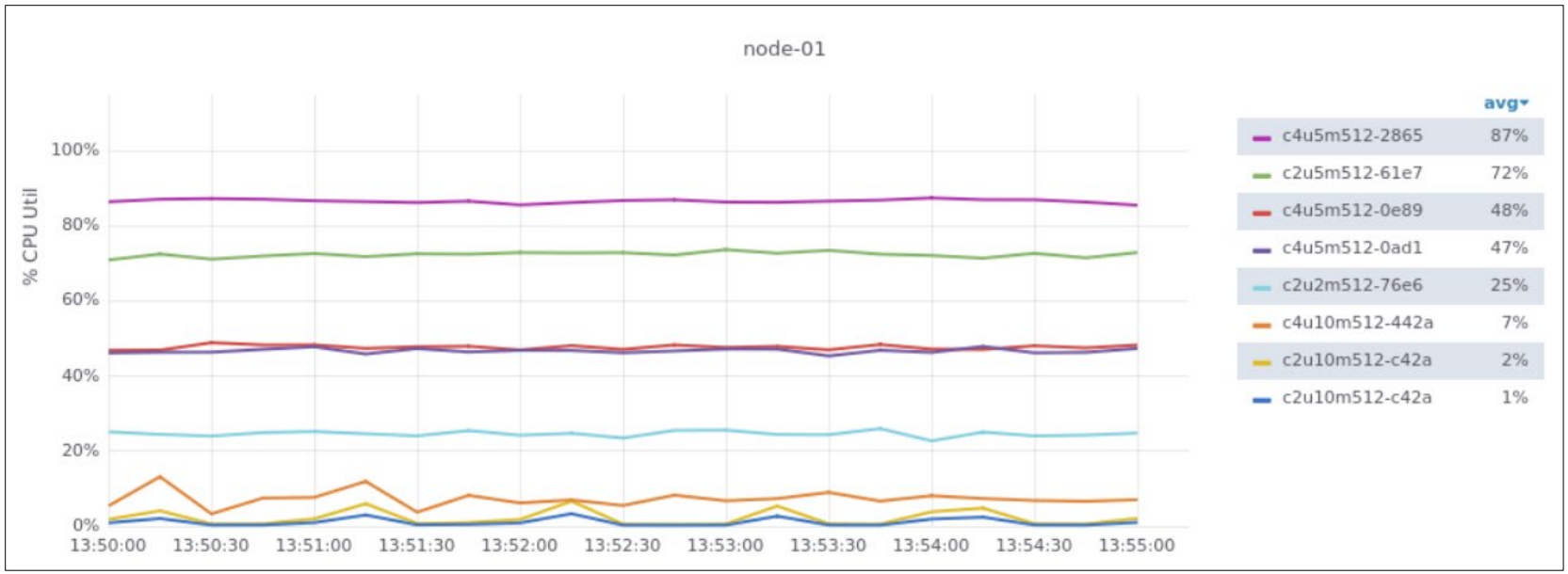
Εφαρμόζοντας την προηγούμενη μεθοδολογία στην υποδομή openstack, υπολογίζουμε τους παρακάτω μέσους χρόνους χρήσης της CPU σε κάθε κόμβο και επιβεβαιώνουμε τους αρχικούς μας υπολογισμούς:

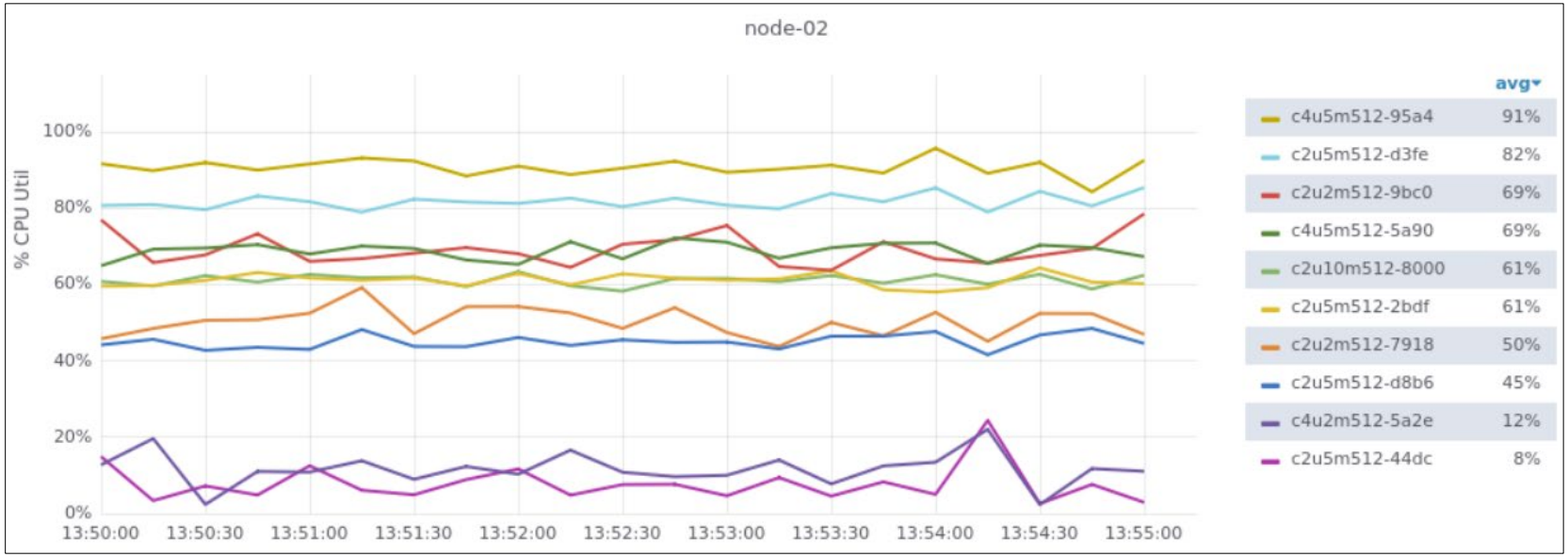
node-01[134] cpu: 942 (58.9%), idle: 578, excess: 196
node-02[116] cpu: 1420 (88.8%), idle: 100, excess: 476
node-03[138] cpu: 571 (35.7%), idle: 949, excess: 0
node-04[136] cpu: 1338 (83.6%), idle: 182, excess: 274
node-05[130] cpu: 801 (50.1%), idle: 719, excess: 137
node-06[134] cpu: 1491 (93.2%), idle: 29, excess: 509

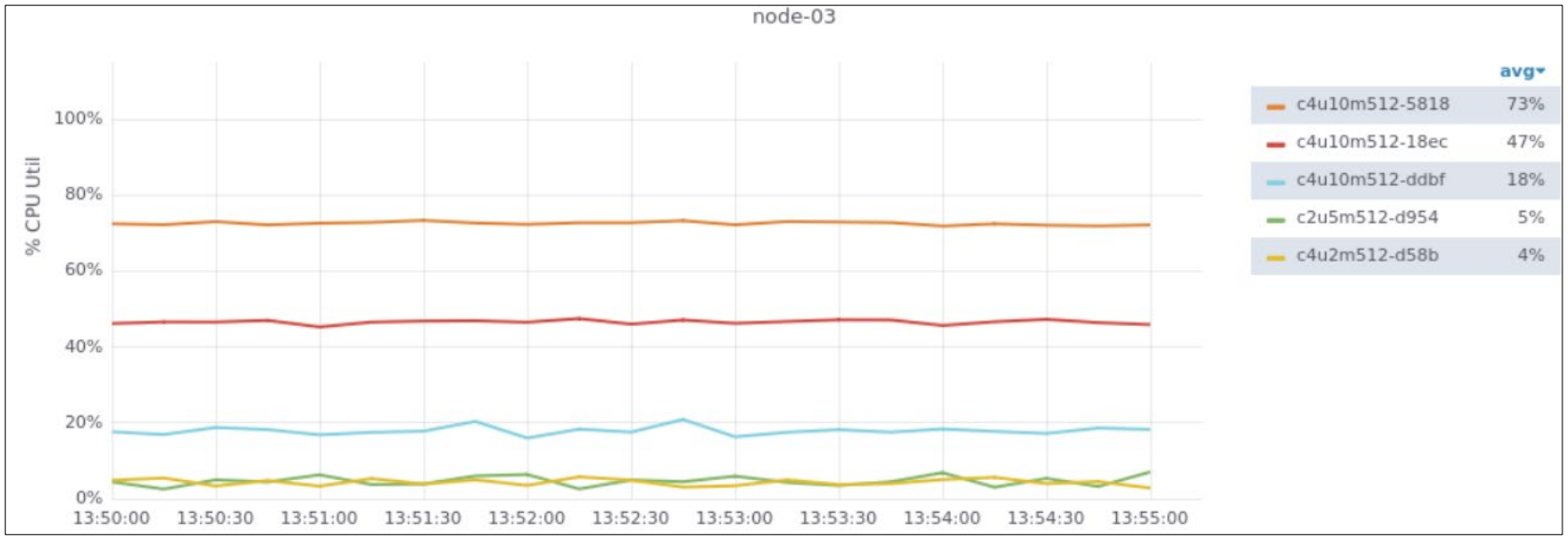
Οι κόμβοι 01 και 05 κυμαίνονται περίπου στο 55% της διαθέσιμης CPU, ενώ ο κόμβος 03 είναι πιο αδρανής, αφού χρησιμοποιείται κατά 35%. Ταυτόχρονα, οι κόμβοι 02 και 04 χρησιμοποιούνται σε πολύ μεγάλο ποσοστό κατά περίπου 85% και 93% αντίστοιχα.

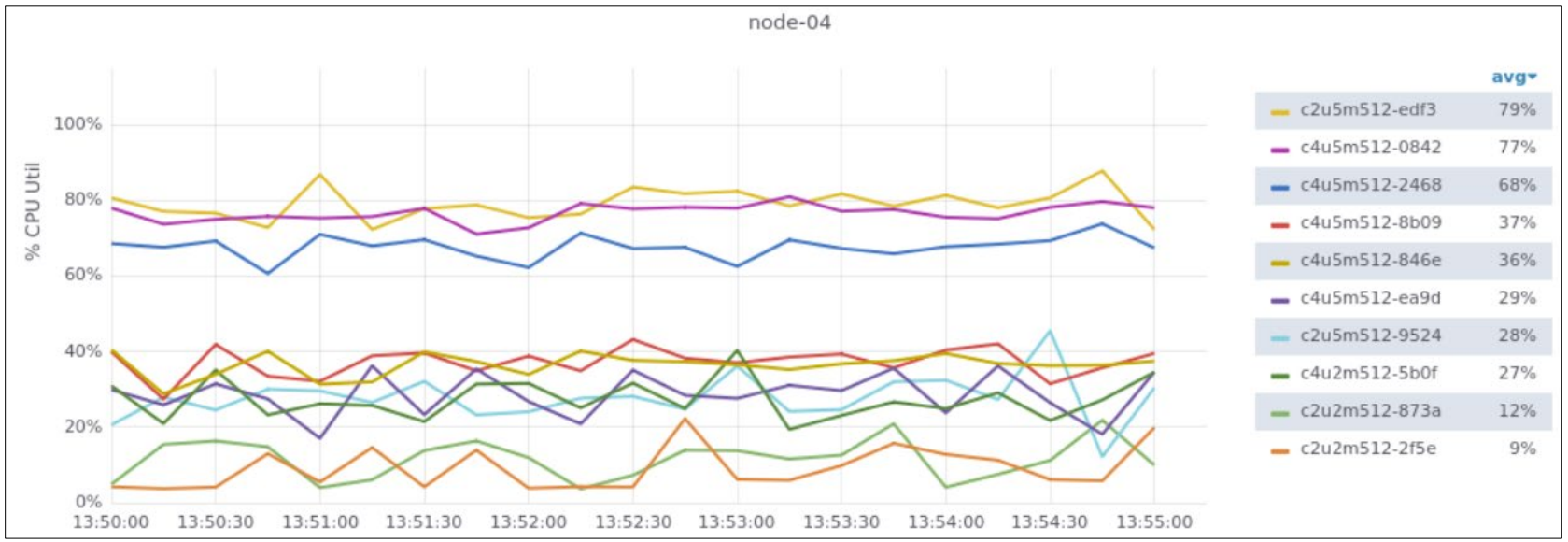
Η τυπική απόκλιση του χρόνου στη CPU στο σύνολο των κόμβων είναι 414.67, στοιχείο που αποδεικνύει την ανομοιόμορφη κατανομή των φορτίων.

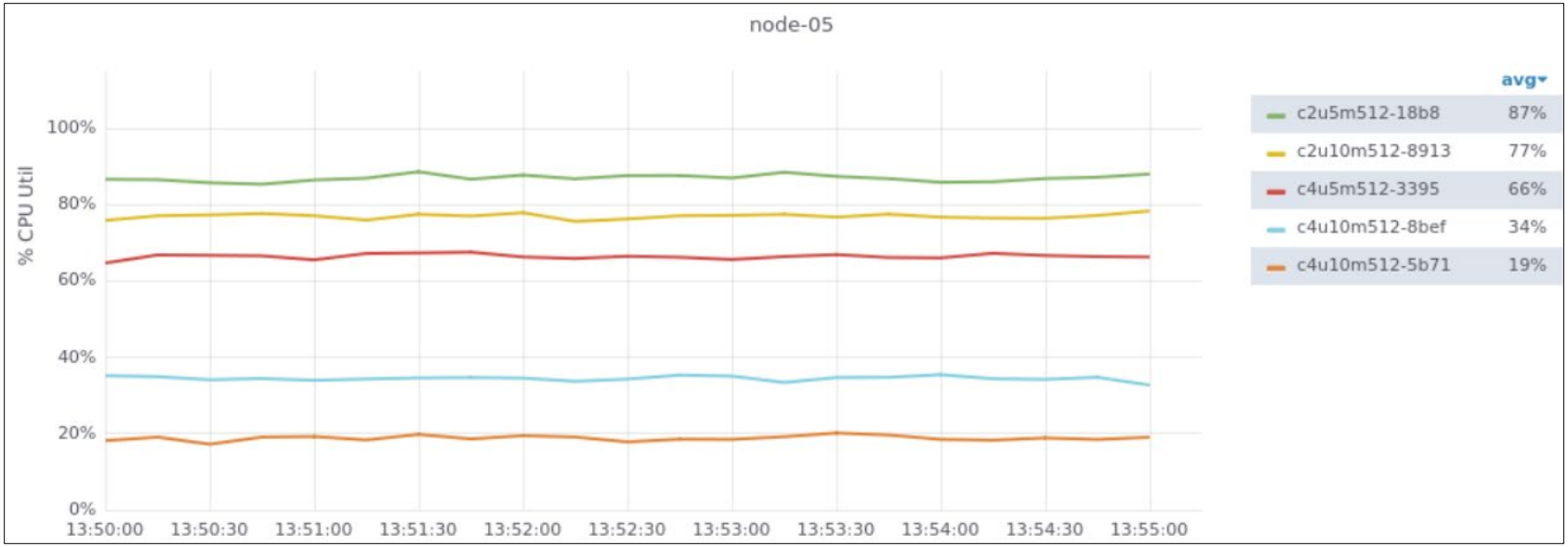
Τα γραφήματα που ακολουθούν παρουσιάζουν την χρησιμοποιούμενη CPU από κάθε instance σε κάθε έναν κόμβο υπολογισμού.

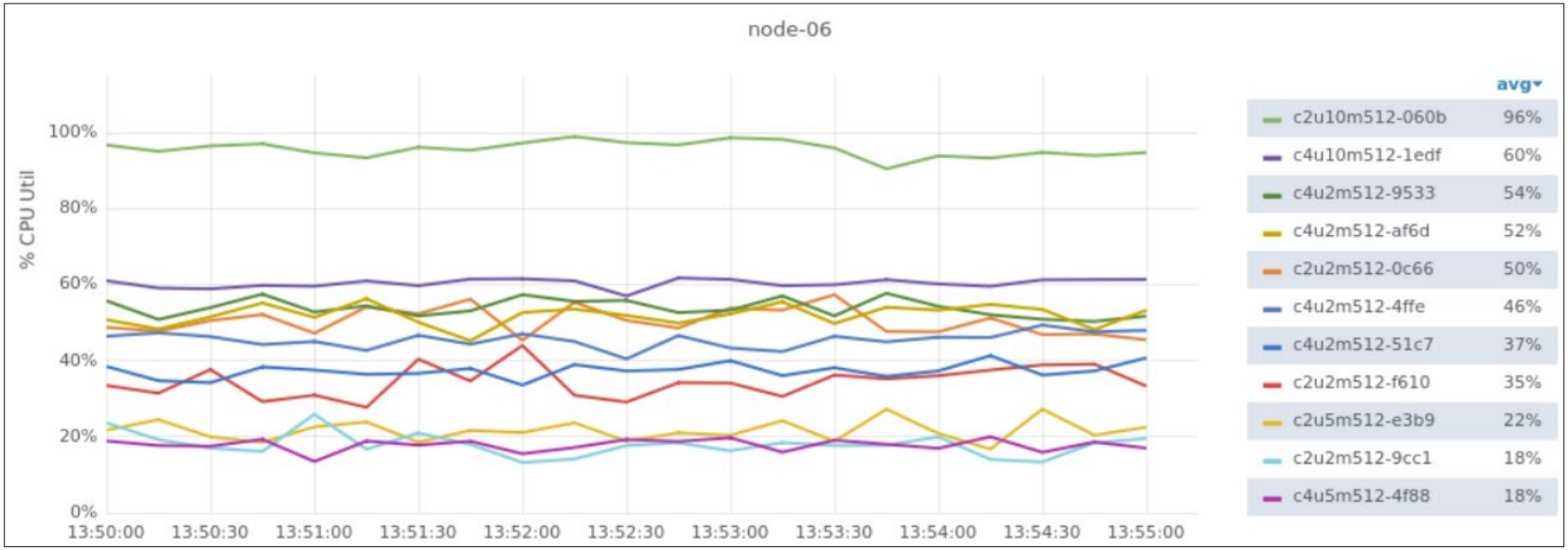

node-01
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c4u5m512-2865 87%
c2u5m512-61e7 72%
c4u5m512-0e89 48%
c4u5m512-0ad1 47%
c2u2m512-76e6 25%
c4u10m512-442a 7%
c2u10m512-c42a 2%
c2u10m512-c42a 1%


node-02
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c4u5m512-95a4 91%
c2u5m512-d3fe 82%
c2u2m512-9bc0 69%
c4u5m512-5a90 69%
c2u10m512-8000 61%
c2u5m512-2bdf 61%
c2u2m512-7918 50%
c2u5m512-d8b6 45%
c4u2m512-5a2e 12%
c2u5m512-44dc 8%


node-03
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c4u10m512-5818 73%
c4u10m512-18ec 47%
c4u10m512-ddbf 18%
c2u5m512-d954 5%
c4u2m512-d58b 4%


node-04
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c2u5m512-edf3 79%
c4u5m512-0842 77%
c4u5m512-2468 68%
c4u5m512-8b09 37%
c4u5m512-846e 36%
c4u5m512-ea9d 29%
c2u5m512-9524 28%
c4u2m512-5b0f 27%
c2u2m512-873a 12%
c2u2m512-2f5e 9%


node-05
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c2u5m512-18b8 87%
c2u10m512-8913 77%
c4u5m512-3395 66%
c4u10m512-8bef 34%
c4u10m512-5b71 19%


node-06
% CPU Util
100%
80%
60%
40%
20%
0%
13:50:00
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
avg
c2u10m512-060b 96%
c4u10m512-1edf 60%
c4u2m512-9533 54%
c4u2m512-af6d 52%
c2u2m512-0c66 50%
c4u2m512-4ffe 46%
c4u2m512-51c7 37%
c2u2m512-f610 35%
c2u5m512-e3b9 22%
c2u2m512-9cc1 18%
c4u5m512-4f88 18%

3.3 Αλγόριθμος αποφόρτισης ΕΝΕΔΗ (offloading algorithm)

Ο αλγόριθμος αποφόρτισης που υλοποιήθηκε έχεις ως στόχο την ανακατανομή των φορτίων των χρηστών με τρόπο, ώστε ένας ή περισσότεροι κόμβοι να μην φιλοξενούν πλέον φορτία. Μετά την εφαρμογή του αλγορίθμου, ο διαχειριστής της υποδομής μπορεί να απενεργοποιήσει τους αδρανείς κόμβους, εξοικονομώντας έτσι ενέργεια.

Η φιλοσοφία του αλγορίθμου είναι η επιλογή κόμβων πηγής, προορισμού και φορτίου στόχου, το οποίο θα μεταφερθεί μεταξύ αυτών. Αφού τοποθετηθούν τα reserved φορτία, ξεκινάει η ανακατανομή των elastic στον εναπομείναν διαθέσιμο χώρο. Ξεκινώντας από τον πιο αδρανή κόμβο, επιλέγουμε το πιο απαιτητικό φορτίο του και δοκιμάζουμε αν μπορεί να εξυπηρετηθεί από τον αμέσως πιο αδρανή κόμβο, βάσει του idle και excess time. Αν δεν υπάρχει η δυνατότητα, δηλαδή δεν μπορούν τα συμπιεστούν τα ήδη υπάρχοντα φορτία για να δεχτούν το νέο, διατηρώντας το κατώτατο όριο εξυπηρέτησης, δοκιμάζουμε στον αμέσως λιγότερο κ.ο.κ. Μόλις ολοκληρωθεί η διαδικασία σημειώνουμε το συγκεκριμένο φορτίο ως ολοκληρωμένο, υπό την έννοια ότι δεν μπορεί να μεταφερθεί εκ νέου και βρίσκουμε τον νέο κόμβο με την μεγαλύτερη αδράνεια. Επιλέξουμε ξανά το πιο απαιτητικό του φορτίου και επαναλαμβάνουμε τη διαδικασία. Η αναδρομή σταματάει μόλις όλα τα φορτία σημειωθούν ή μετά από συγκεκριμένο αριθμό επαναλήψεων. Μετά την ολοκλήρωση του αλγορίθμου ελέγχουμε πόσοι κόμβοι εμφανίζουν CU = 0. Οι συγκεκριμένοι κόμβοι δεν έχουν κάποιο φορτίο να εξυπηρετήσουν και μπορούν να απενεργοποιηθούν.

Στη συνέχεια ακολουθεί μία προσομοίωση του αλγορίθμου, που υποδεικνύει τις μεταφορές των φορτίων στους διάφορους κόμβους βάσει του αλγορίθμου:

c4u5m512-3395 node-05 -> node-03
c4u5m512-2865 node-01 -> node-05
c4u5m512-95a4 node-02 -> node-01
c4u5m512-5a90 node-02 -> node-05
c4u10m512-1edf node-06 -> node-02
c4u5m512-0842 node-04 -> node-06
c4u5m512-2468 node-04 -> node-01
c2u5m512-edf3 node-04 -> node-02
c4u2m512-5b0f node-04 -> node-05
c4u5m512-8b09 node-04 -> node-02
c4u5m512-846e node-04 -> node-06
c2u2m512-873a node-04 -> node-01
c2u5m512-9524 node-04 -> node-02
c4u5m512-ea9d node-04 -> node-06
c2u2m512-2f5e node-04 -> node-02

3.3.1 Προσομοίωση αλγορίθμου

Η αρχική προσομοίωση δίνει τα παρακάτω αποτελέσματα για τις υπολογισμένες μεταβάσεις:

node-01[158] cpu: 1361 (85.1%), idle: 159, excess: 378
node-02[160] cpu: 1519 (94.9%), idle: 1, excess: 351
node-03[158] cpu: 856 (53.5%), idle: 664, excess: 75
node-04[0] cpu: 0 (0.0%), idle: 1520, excess: 0
node-05[158] cpu: 1388 (86.8%), idle: 132, excess: 433
node-06[154] cpu: 1507 (94.2%), idle: 13, excess: 257

Παρατηρούμε ότι μετά τις μετακινήσεις που υπολόγισε ο αλγόριθμος, ο κόμβος 04 πλέον στο 0%, δηλαδή δεν έχει κάποιο instance και μπορεί επομένως να απενεργοποιηθεί.

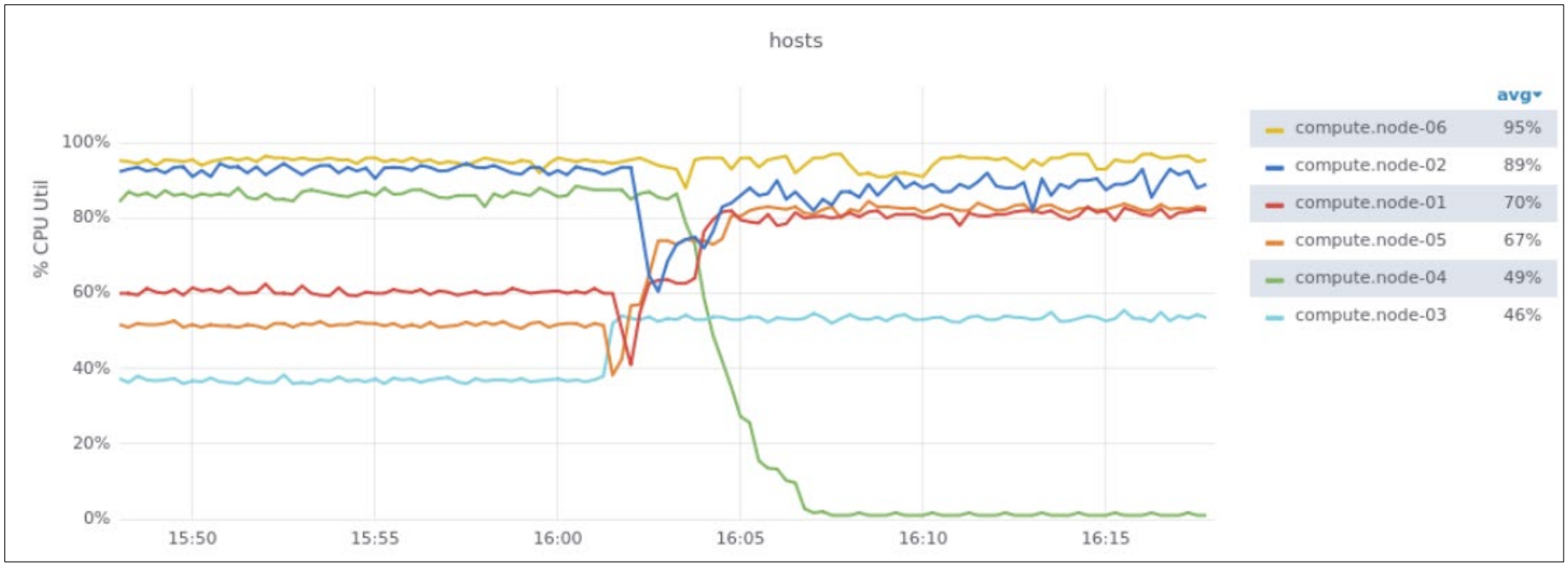
3.3.2 Πειραματικά αποτελέσματα αλγορίθμου σε παραγωγικό περιβάλλον

Εκτελώντας τώρα τις μετακινήσεις στο παραγωγικό περιβάλλον, παρατηρούμε τις παρακάτω καταστάσεις για κάθε κόμβο:

node-01[158] cpu: 1290 (80.6%), idle: 310, excess: 325
node-02[160] cpu: 1397 (87.3%), idle: 203, excess: 282
node-03[158] cpu: 831 (51.9%), idle: 769, excess: 64
node-04[0] cpu: 0 (0.0%), idle: 1600, excess: 0
node-05[158] cpu: 1311 (81.9%), idle: 289, excess: 370
node-06[154] cpu: 1530 (95.6%), idle: 70, excess: 362

Σε αντιστοιχία με την προσομοίωση, ο κόμβος 04 δεν έχει κάποια φορτία να εξυπηρετήσει, αφού όλα μετακινήθηκαν σε άλλους κόμβους που είχαν την απαιτούμενη χωρητικότητα και μπορεί να κλείσει, εξοικονομώντας ενέργεια για την υποδομή.

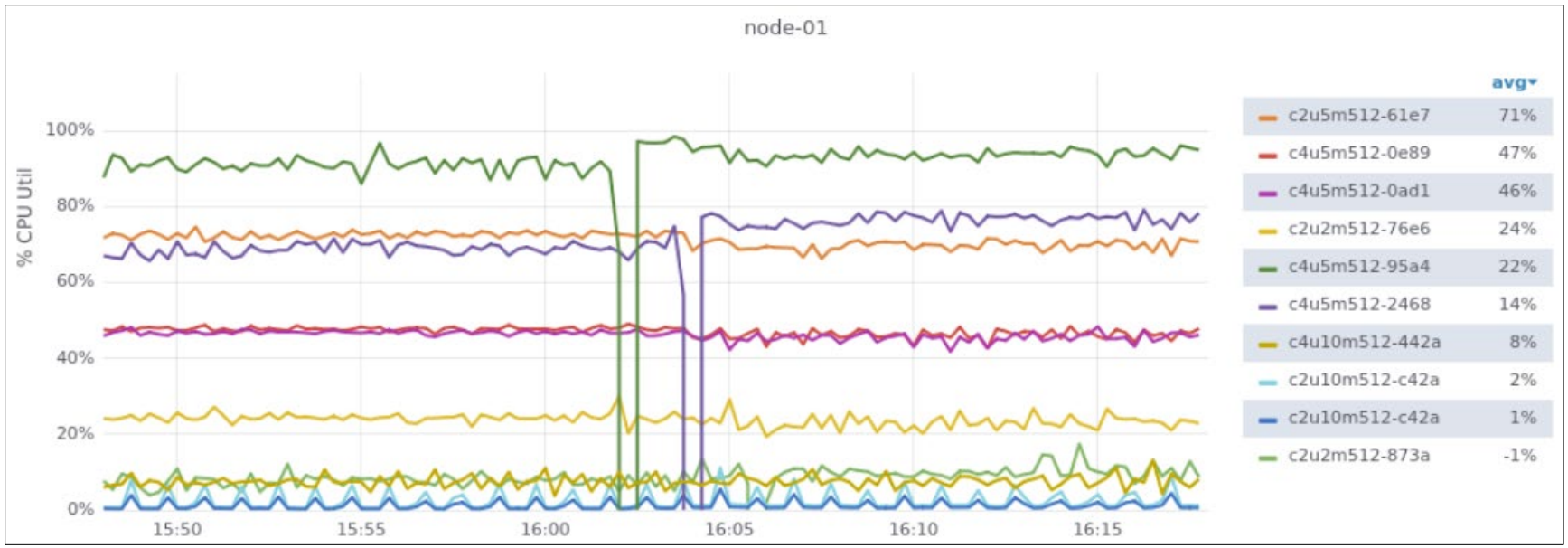
Το διάγραμμα που ακολουθεί παρουσιάζει την κατάσταση των κόμβων μετά την εφαρμογή των αντίστοιχων μετακινήσεων φορτίων όπως προσδιορίστηκαν από τον αλγόριθμο αποφόρτισης ΕΝΕΔΗ:



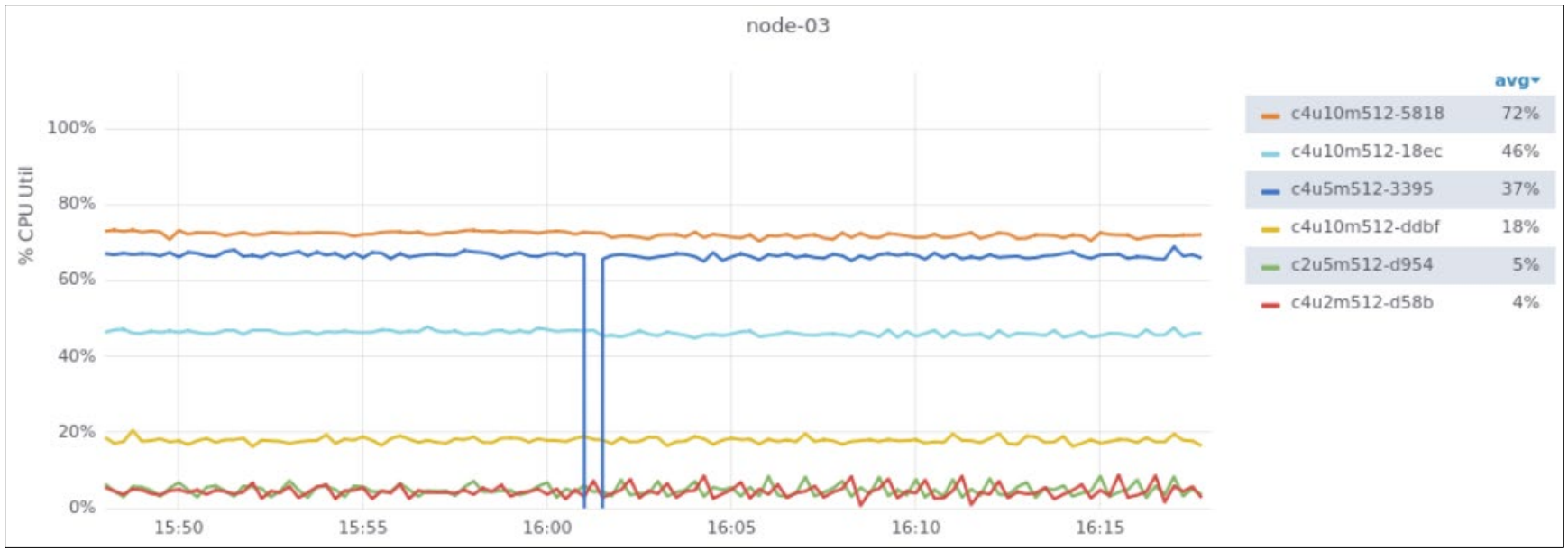
Εικόνα 15 Απελευθέρωση του κόμβου 04 μετά την εφαρμογή του αλγορίθμου αποφόρτισης ΕΝΕΔΗ

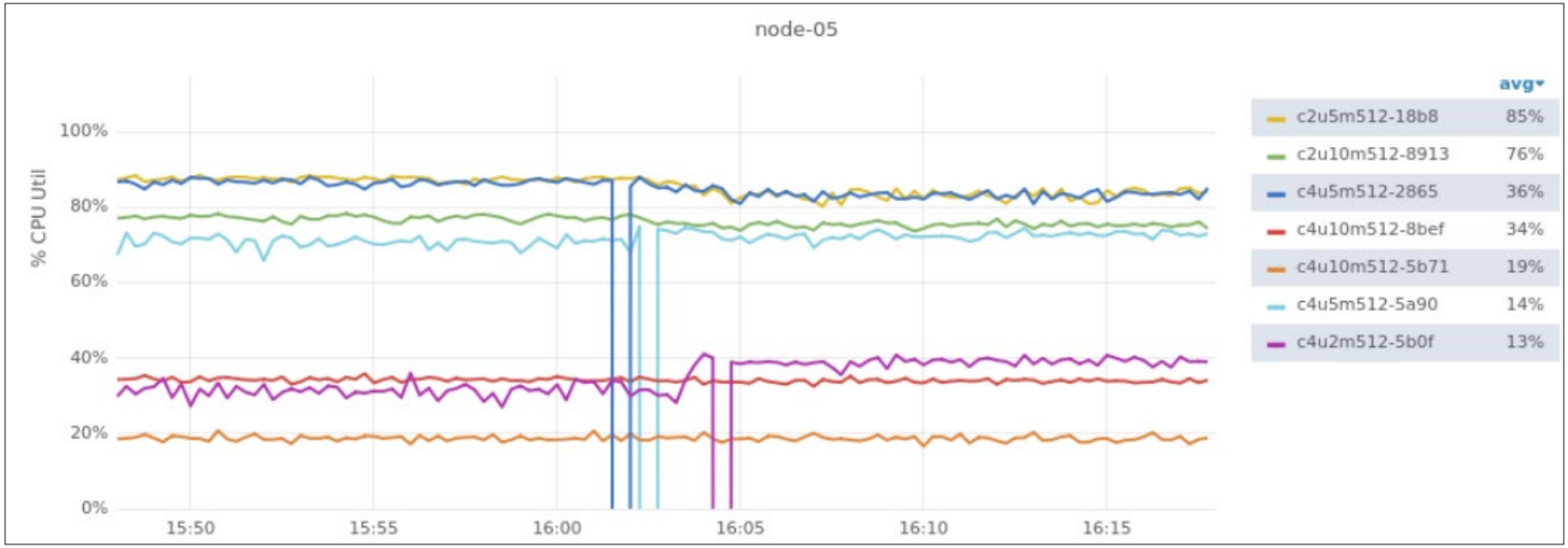
Κατά την εφαρμογή των μετακινήσεων των φορτίων στους διάφορους κόμβους, πολλοί κόμβοι δέχονται νέα φορτία, είτε συμπιέζουν ήδη περιλαμβανόμενα φορτία με σκοπό να εκμεταλλευτούν με τον καλύτερο δυνατό τρόπο τη διαθέσιμη υπολογιστική τους ισχύ. Με αυτό τον τρόπο καθίσταται δυνατή η συνολική αποδέσμευση ολόκληρων κόμβων από την συστάδα είτε για να χρησιμοποιηθούν σε κάποια άλλη εργασία, είτε στη δική μας περίπτωση να απενεργοποιηθούν μειώνοντας το ενεργειακό κόστος της υποδομής.

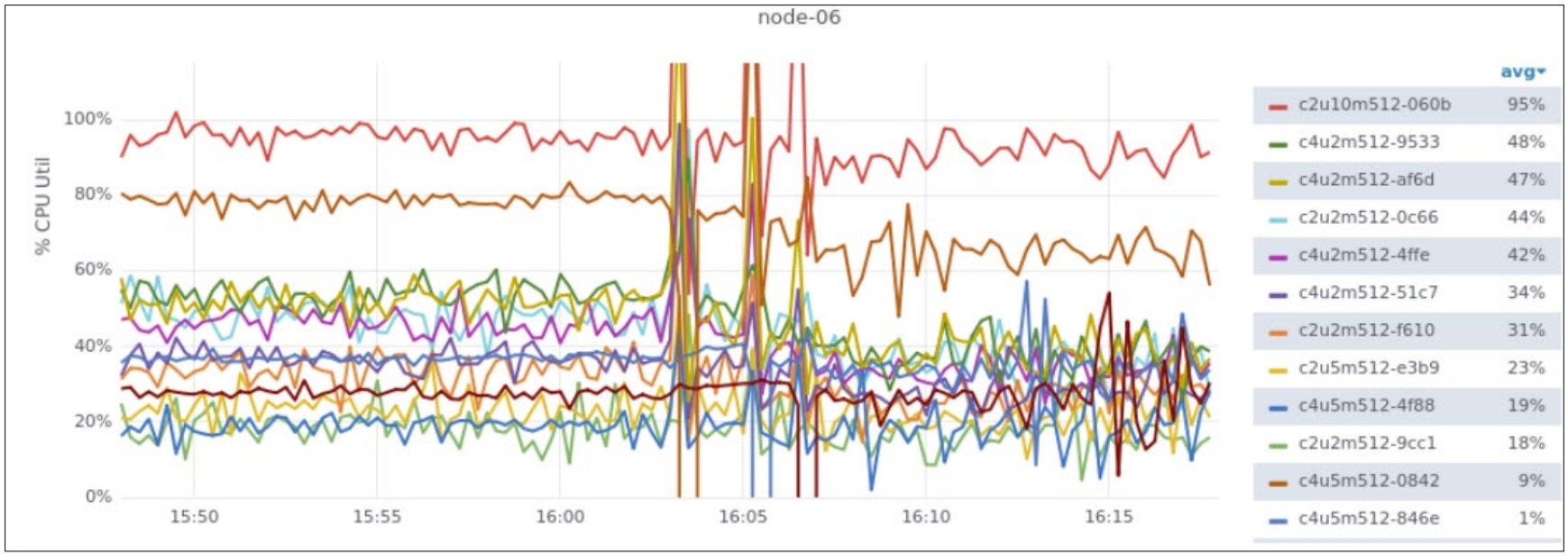
Η αναλυτική κατάσταση των κόμβων που απέμειναν παρουσιάζεται στα διαγράμματα που ακολουθούν:


node-01
% CPU Util
100%
80%
60%
40%
20%
0%
15:50
15:55
16:00
16:05
16:10
16:15
avg
c2u5m512-61e7 71%
c4u5m512-0e89 47%
c4u5m512-0ad1 46%
c2u2m512-76e6 24%
c4u5m512-95a4 22%
c4u5m512-2468 14%
c4u10m512-442a 8%
c2u10m512-c42a 2%
c2u10m512-c42a 1%
c2u2m512-873a -1%


node-02
% CPU Util
100%
80%
60%
40%
20%
0%
15:50
15:55
16:00
16:05
16:10
16:15
avg
c2u5m512-d3fe 82%
c2u2m512-9bc0 72%
c2u10m512-8000 62%
c2u5m512-2bdf 62%
c2u2m512-7918 55%
c2u5m512-d8b6 45%
c2u5m512-edf3 29%
c4u5m512-8b09 12%
c4u2m512-5a2e 11%
c2u2m512-2f5e 11%
c2u5m512-44dc 8%
c4u10m512-1edf 7%


node-03
% CPU Util
100%
80%
60%
40%
20%
0%
15:50
15:55
16:00
16:05
16:10
16:15
avg
c4u10m512-5818 72%
c4u10m512-18ec 46%
c4u5m512-3395 37%
c4u10m512-ddbf 18%
c2u5m512-d954 5%
c4u2m512-d58b 4%


node-05
% CPU Util
100%
80%
60%
40%
20%
0%
15:50
15:55
16:00
16:05
16:10
16:15
avg
c2u5m512-18b8 85%
c2u10m512-8913 76%
c4u5m512-2865 36%
c4u10m512-8bef 34%
c4u10m512-5b71 19%
c4u5m512-5a90 14%
c4u2m512-5b0f 13%


node-06
% CPU Util
100%
80%
60%
40%
20%
0%
15:50
15:55
16:00
16:05
16:10
16:15
avg
c2u10m512-060b 95%
c4u2m512-9533 48%
c4u2m512-af6d 47%
c2u2m512-0c66 44%
c4u2m512-4ffe 42%
c4u2m512-51c7 34%
c2u2m512-f610 31%
c2u5m512-e3b9 23%
c4u5m512-4f88 19%
c2u2m512-9cc1 18%
c4u5m512-0842 9%
c4u5m512-846e 1%

4. Συμπεράσματα

Στη παρούσα μελέτη παρουσιάστηκε ένα εξελιγμένο σύστημα έξυπνης διαχείρισης φορτίων για χρήση σε σύγχρονα κέντρα δεδομένων. Η υλοποίηση του αλγορίθμου με την εισαγωγή της έννοιας των Compute Units δίνει τη δυνατότητα απενεργοποίησης ενός ή περισσοτέρων κόμβων μίας υποδομής αξιοποιώντας καλύτερα τους υπολοίπους.

Η συγκεκριμένη υλοποίηση είναι κλιμακώσιμη, επομένως μπορεί να εφαρμοστεί σε όλες τις υποδομές ανεξαρτήτως μεγέθους, βελτιώνοντας και αυτοματοποιώντας εργασίες του διαχειριστή και εξοικονομώντας ταυτόχρονα χρόνο. Ο αλγόριθμος αποφόρτισης σε συνδυασμό με την χρήση ανανεώσιμων πηγών ενέργειας για την λειτουργία των κέντρων δεδομένων, μπορεί να αποτελέσει μία συνολική λύση για σημαντική μείωση των ρύπων στη σημερινή εποχή των μεγάλων δεδομένων και της διαρκούς ροής, ανταλλαγής και επεξεργασίας αυτών.