Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Μεταπτυχιακό Πρόγραμμα Σπουδών
Διδακτική της Τεχνολογίας & Ψηφιακά Συστήματα
Κατεύθυνση Δικτυοκεντρικών Συστημάτων

Διπλωματική εργασία

# Σχεδιασμός, μελέτη και ανάπτυξη υπηρεσιοστρεφούς πλατφόρμας εποπτείας υποδομής, εποπτείας εκτέλεσης υπηρεσιών και παραγωγής γεγονότων σε υπολογιστικά νέφη

Καψάλης Ανδρέας ΜΕ10086

Πειραιάς 2013

February 27, 2013

# Περίληψη

Τα υπολογιστικά νέφη σήμερα αποτελούν μία αναδυόμενη τεχνολογία για τη παροχή υπηρεσιών και λύσεων, προσφέροντας πρόσβαση σε πόρους. Υλοποιούν τεχνολογιές και αρχιτεκτονικές εικονοποιήσης για να προσφέρουν στους τελικούς χρήστες 3 βασικά είδη υπηρεσιών: Υποδομή ως υπηρεσία (IaaS), Πλατφόρμα ως υπηρεσία (PaaS) και Εφαρμογή ως υπηρεσία (SaaS). Κάθε υπηρεσία προσφέρει στους χρήστες πρόσβαση σε εικονοποιημένα περιβάλλοντα, επιτρέποντας τους να εγκαθιστούν και να δημοσιεύουν εφαρμογές και υπηρεσίες. Η χαρακτηριστική ελαστικότητα των υποδομών υπολογιστικών νεφών, προσφέρει μία υψηλή ποιότητα υπηρεσίας (QoS) καθώς εγγυάται τη προσβασιμότητα των υπηρεσιών ή εφαρμογών των χρηστών ανά πάσα στιγμή.Με τη βοήθεια προκαθορισμένων SLAs (ServiceLevelAgreement), οι χρήστες μπορούν να καθορίσουν τον αριθμό και το είδος των υπηρεσιών που παρέχονται από τους παρόχους υπολογιστικών νεφών.

Για τη ικανοποίηση των όλο και αυξανόμενων αναγκών για ελαστικότητα στις υποδομές υπολογιστικών νεφών, αλλά και για τη διατήρηση του προκαθορισμένου επιπέδου παρεχόμενης υπηρεσίας,κάθε πάροχος θα πρέπει να υλοποιεί έναν εξελιγμένο μηχανισμό εποπτείας και παραγωγής γεγονότων ανάλογα με τις ανάγκες του. Ο μηχανισμός αυτός (ή πλατφόρμα) εγγυάται την ομαλή λειτουργία μίας υποδομής υπολογιστικού νέφους, ελέγχοντας τους πόρους, την υγεία των εικονοποιημένων πόρων, τη κατάσταση συγκεκριμένων υπηρεσιών και εφαρμογών, σε περίπτωση που αυτό χρειαστεί, και επιπρόσθετα παράγει τα ανάλογα γεγονότα προς τη υποδομή για να εκτελεστούν όλες οι απαραίτητες ρουτίνες. Τα γεγονότα αυτά περιγράφουν καταστάσεις των πόρων ή υπηρεσιών για κάποια δεδομένη στιγμή. Ως τέτοιες καταστάσεις μπορούν να λογιστούν καταστάσεις υπερβολικού κινδύνου, αποτυχίας καθώς και συγκεκριμένες καταστάσεις γενικού περιεχομένου για σκοπούς ιστορικότητας.

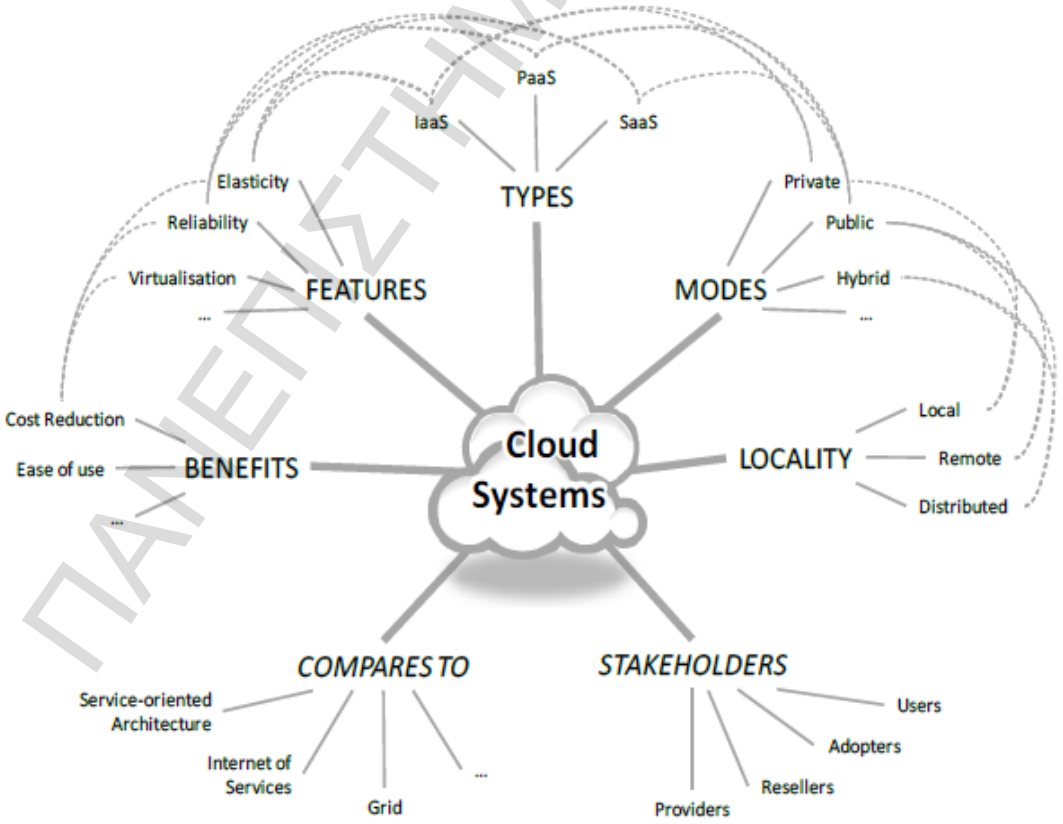
Η παρούσα εργασία επικεντρώνεταιστην εποπτεία λειτουργείας της υποδομής, αλλα και στη εποπτεία υπηρεσιών σε υπολογιστικά νέφη. Επίσης, περιγράφονται συγκεκριμένες πειραματικές διαδικασίες που είχαν σκοπό να ελέγξουν και να περιγράψουν τη λειτουργία της πλατφόρμας σε διάφορες συνθήκες φόρτου.

# Περιεχόμενα

# 1. Εισαγωγή

Το Υπολογιστικό Νέφος παρέχει την δυνατότητα να μειωθεί δραματικά το κόστος των υπηρεσιών λογισμικού τόσο μέσω της εμπορευματοποίησης των στοιχείων της τεχνολογίας του διαδικτύου όπως επίσης και με την υλοποίηση επιχειρηματικών μοντέλων βασισμένα στην χρήση (on-demand). Σύμφωνα με τον αναλυτή Ben Pring της Gartner, το «Νέφος είναι η φράση της ημέρας» ("It's become the phrasedu jour"). Η εικονικοποίηση του υλικού (virtualization), η πρόβλεψη υπηρεσιών, η ελαστικότητα, η επεκτασιμότητα, η ευελιξία των μοντέλων κοστολόγησης και χρέωσης επιτρέπουν στις τεχνολογίες του Υπολογιστικού Νέφους να παρέχουν την δυνατότητα αποτελεσματικής προσαρμογής των διαφόρων πόρων στις απαιτήσεις των χρηστών. Ερευνητικά αποτελέσματα και αρχιτεκτονικά πρότυπα όπως οι υπηρεσιοστρεφείς υποδομές, η εικονικοποίηση, τα πλέγματα υπολογιστών (Grids) έχουν ενσωματωθεί στο Υπολογιστικό Νέφος και οδηγούν σε τρεις κύριες κατηγορίες στην δομή υπηρεσιών Νεφών που είναι γενικά αναγνωρισμένες σε [3]:

- Υποδομή ως υπηρεσία (IaaS), η οποία αναφέρεται στην παροχή πόρων (κεντρικοί υπολογιστές, σκληρών δίσκων, δίκτυα υπολογιστών και άλλες συσκευές) στις οποίες οι καταναλωτές υπηρεσιών εγκαθιστούν το λογισμικό τους (συνήθως ως στιγμιότυπα εικονικών μηχανών – Virtual Machines).
- Πλατφόρμα ως υπηρεσία (PaaS), που αναφέρεται στην παροχή μιας πλατφόρμας και περιβάλλοντος ανάπτυξης που παρέχουν διάφορες υπηρεσίες όπως και αποθηκευτικό χώρο τα οποία φιλοξενούνται στο Νέφος.
- Λογισμικό ως υπηρεσία (SaaS), η οποία αναφέρεται στην παροχή μιας εφαρμογής ως υπηρεσία για το διαθέσιμη στο διαδίκτυο ή ένα κατανεμημένο περιβάλλον.




Εικόνα 1: Υπολογιστικά νέφη[4]

Επιπρόσθετα, κάποιος πρέπει να λάβει υπόψη ότι οι εφαρμογές του Μελλοντικού Διαδικτύου (*Future Internet*) αυξάνουν την ανάγκη για την ύπαρξη περιβαλλόντων που μπορούν να διευκολύνουν το συναγωνισμό και την αλληλεπίδραση και θέτουν έτσι συγκεκριμένες απαιτήσεις στην υπάρχουσα υποδομή, η οποία πρέπει να είναι σε θέση να προσαρμοστεί αποτελεσματικά στην πρόβλεψη των πόρων και στις απαιτήσεις σε ποιότητα των υπηρεσιών (QoS) των εφαρμογών. Οι εφαρμογές αυτές έχουν αυστηρές χρονικές και ποιοτικές προϋποθέσεις λειτουργίας οι οποίες αν παραβιαστούν μπορούν να οδηγήσουν σε υποβάθμιση της παρεχόμενης ποιότητας της υπηρεσίας. Αντιπροσωπευτικά παραδείγματα τέτοιων εφαρμογών είναι ο σχεδιασμός και η απεικόνιση στον τομέα της εφαρμοσμένης μηχανικής, η παραγωγή οπτικοακουστικού υλικού στις δημιουργικές βιομηχανίες, και τα εικονικά περιβάλλοντα πολλών χρηστών στην εκπαίδευση και τα ηλεκτρονικά παιχνίδια.

Η εποπτεία της εκτέλεσης εφαρμογών και της χρήσης των πόρων σε κατανεμημένα περιβάλλοντα αποτελεί μια από τις βασικότερες λειτουργίες των σύγχρονων συστημάτων με άμεσο αντίκτυπο τόσο στην απόδοση των εφαρμογών και των υπηρεσιών όσο και στην διαχείριση των πόρων της υποδομής. Η ανάπτυξη μηχανισμών λήψης αποφάσεων και διορθωτικών κινήσεων μιας εγκατάστασης συστήματος βασίζεται αποκλείστηκα στα δεδομένα που συγκεντρώνονται από την διαδικασία εποπτείας. Από την άλλη, οι δυναμικές αρχιτεκτονικές των υπηρεσιοστρεφών συστημάτων και των περιβαλλόντων Νεφών, εγείρουν σημαντικές προκλήσεις στον σχεδιασμό και λειτουργία των διαδικασιών εποπτείας.

# 2. Υπολογιστικά νέφη

## 2.1. Γενικές πληροφορίες για τα υπολογιστικά νέφη

Το Υπολογιστικό Νέφος είναι ένα μοντέλο για την ενεργοποίηση ευέλικτης, κατόπιν αιτήματος και διαρκούς πρόσβασης σε ένα σύνολο διαμοιρασμένων και διαμορφώσιμων υπολογιστικών πόρων (πχ. δίκτυα, εξυπηρετητές, αποθηκευτικοί χώροι, εφαρμογές και υπηρεσίες) οι οποίοι μπορούν πολύ γρήγορα να προβλεφθούν και διατεθούν με ελάχιστο διαχειριστικό κόστος ή αλληλεπίδραση χρήστη-παρόχου.

Αυτό το μοντέλο Νέφους αποτελείται από πέντε βασικά χαρακτηριστικά, τρία μοντέλα υπηρεσιών και τέσσερα μοντέλα ανάπτυξης τα οποία παρουσιάζουμε και αναλύουμε παρακάτω.


Εικόνα 2: Τυπικό διάγραμμα υπολογιστικού Νέφους

### Βασικά χαρακτηριστικά:

***Αυτο-εξυπηρέτηση κατόπιν αιτήματος***

Ένας χρήστης μπορεί μονομερώς να εφοδιαστεί και να "καταναλώσει" υπολογιστικές υπηρεσίες, όπως χρόνο χρήσης σε εξυπηρετητές και πρόσβαση σε δικτυακό αποθηκευτικό χώρο, αυτοματοποιημένα χωρίς να απαιτείται προσωπική αλληλεπίδραση με τον κάθε πάροχο αυτών των υπηρεσιών ξεχωριστά.

***Ευρεία δικτυακή πρόσβαση***

Οι δυνατότητες και οι υπηρεσίες αυτές είναι διαθέσιμες και προσβάσιμες μέσω του δικτύου χρησιμοποιώντας μηχανισμούς που προάγουν την χρήση ετερογενών προγραμμάτων/πλατφόρμων πελάτη (κινητά τηλέφωνα, ταμπλέτες, φορητοί υπολογιστές, σταθμούς εργασίας κ.α.).

***Συγκέντρωση πόρων***

Οι υπολογιστικοί πόροι ενός παρόχου είναι συγκεντρωμένα έτσι ώστε να εξυπηρετούν πολλαπλούς χρήστες/καταναλωτές, χρησιμοποιώντας μοντέλα πολλαπλής μίσθωσης (multi-tenancy), ο καθένας από τους οποίους έχει διαφορετικούς φυσικούς ή εικονικούς πόρους δυναμικά εκχωρημένους σε αυτόν σύμφωνα με τις δικές του απαιτήσεις. Η έννοια της ανεξαρτησίας από την τοποθεσία, όσον αφορά τους υπολογιστικούς πόρους, έχει την σημασία ότι ο πελάτης δεν έχει γνώση ή έλεγχο σχετικά με την ακριβή θέση των παρεχομένων πόρων αλλά μπορεί να θέσει σχετικούς περιορισμούς σε υψηλότερο επίπεδο αοριστίας (πχ. χώρα, πολιτεία, κέντρο δεδομένων). Παραδείγματα τέτοιων πόρων συμπεριλαμβάνουν αποθηκευτικό χώρο, υπολογιστική ισχύ, μνήμη και χρήση δικτύου.

***Άμεση ελαστικότητα***

Οι υπηρεσίες αυτές των Νεφών μπορούν να εκχωρηθούν και διατεθούν με έναν ελαστικό και μερικές φορές αυτόματο τρόπο. Αυτό σημαίνει ότι μπορούν να αυξήσουν ή μειώσουν τις δυνατότητες τους και την χωρητικότητα τους αναλόγως με την ζήτηση. Για τον καταναλωτή, οι δυνατότητες που του διατίθενται συχνά φαίνονται να είναι απεριόριστες και μπορούν να πιστωθούν σε οποιαδήποτε ποσότητα ανά πάσα στιγμή.

***Υπηρεσία μετρήσεων***

Τα συστήματα Νεφών μπορούν αυτόματα να ελέγχουν και να βελτιστοποιούν την χρήση τον πόρων αξιοποιώντας δυνατότητες μετρήσεων σε κάποιο επίπεδο αφαίρεσης (abstraction) κατάλληλο στις διάφορες παρεχόμενες υπηρεσίες. Η χρήση των παρεχομένων πόρων παρακολουθείται, ελέγχεται και αναφέρεται, παρέχοντας έτσι διαφάνεια τόσο για τον πάροχο όσο και για τον καταναλωτή.


Εικόνα 3: Η στοίβα του υπολογιστικού Νέφους

## Μοντέλα Υπηρεσιών:

***Λογισμικό ως Υπηρεσία (Software as a Service - SaaS)***

Η δυνατότητα που παρέχεται σε έναν καταναλωτή να χρησιμοποιήσει τις εφαρμογές ενός παρόχου που εκτελούνται σε μια υποδομή Νέφους. Οι εφαρμογές αυτές είναι προσβάσιμες από διάφορες συσκευές πελατών είτε μέσω διεπαφών, όπως ένα πρόγραμμα περιήγησης διαδικτύου (thin client), είτε μέσω ολοκληρωμένων προγραμμάτων. Ο καταναλωτής δεν διαχειρίζεται ούτε ελέγχει την υποδομή Νέφους συμπεριλαμβανομένων του δικτύου, εξυπηρετητών, λειτουργικών συστημάτων αλλά ούτε και τις δυνατότητες των εφαρμογών, με πιθανή εξαίρεση μόνο περιορισμένες ρυθμίσεις των εφαρμογών αυτών που σχετίζονται με τον ίδιο τον χρήστη.

***Πλατφόρμα ως Υπηρεσία (Platform as a Service - PaaS)***

Παρέχονται δυνατότητες στο χρήστη να εγκαθιστά στην υποδομή ενός Νέφους εφαρμογές δικής του δημιουργίας είτε άλλες εφαρμογές, χρησιμοποιώντας γλώσσες προγραμματισμού, βιβλιοθήκες και εργαλεία που διατίθενται από τον πάροχο. Ο χρήστης δεν διαχειρίζεται ή ελέγχει την υπάρχουσα υποδομή Νέφους συμπεριλαμβανομένων του δικτύου, των εξυπηρετητών, των λειτουργικών συστημάτων ή του χώρου αποθήκευσης. Παρόλα αυτά μπορεί να έχει έλεγχο πάνω στις εγκατεστημένες εφαρμογές όπως και σε πιθανές ρυθμίσεις του περιβάλλοντος φιλοξενίας των εφαρμογών αυτών.

***Υποδομή ως Υπηρεσία (Infrastructure as a Service – IaaS)***

Η δυνατότητα που παρέχεται στον χρήστη να έχει πρόσβαση σε υπολογιστικούς, αποθηκευτικούς, δικτυακούς και άλλους στοιχειώδους πόρους, πάνω στους οποίους μπορεί να εγκαθιστά και να εκτελεί λογισμικό, το οποίο περιλαμβάνει τόσο εφαρμογές όσο και λειτουργικά συστήματα. Ο χρήστης δεν διαχειρίζεται ή ελέγχει την υπάρχουσα υποδομή Νέφους συμπεριλαμβανομένων του δικτύου, των εξυπηρετητών, των λειτουργικών συστημάτων ή του χώρου αποθήκευσης. Μπορεί παρόλα αυτά να έχει περιορισμένο έλεγχο πάνω σε δικτυακά στοιχεία (π.χ. τοίχος προστασίας, τρόπο πρόσβασης).

## Μοντέλα ανάπτυξης:

***Ιδιωτικό Νέφος (PrivateCloud)***

Η υποδομή του Νέφους παρέχεται για αποκλειστική χρήση από ένα μοναδικό οργανισμό που μπορεί να αποτελείται από πολλαπλούς χρήστες (π.χ. διαφορετικά τμήματα μιας επιχείρησης). Μπορεί επίσης να είναι ιδιοκτησία, να διαχειρίζεται και να λειτουργείται από την επιχείρηση, έναν εξωτερικό οργανισμό είτε ένα συνδυασμό αυτών. Τέλος μπορεί να βρίσκεται εντός ή και εκτός από τις εγκαταστάσεις του οργανισμού αυτού.

***Κοινοτικό Νέφος (Communitycloud)***

Η υποδομή του Νέφους παρέχεται για αποκλειστική χρήση από τους χρήστες ενός οργανισμού οι οποίοι έχουν κοινά ενδιαφέρονται και ανησυχίες. Μπορεί να είναι ιδιοκτησία και να διαχειρίζεται από έναν ή περισσότερους οργανισμούς μιας κοινότητας, έναν εξωτερικό οργανισμό ή συνδυασμό αυτών. Επίσης, η υποδομή μπορεί να βρίσκεται εντός ή και εκτός από τις εγκαταστάσεις του οργανισμού αυτού.

***Δημόσιο Νέφος (Public Cloud)***

Η υποδομή του Νέφους παρέχεται για δημόσια χρήση από το κοινό. Μπορεί να είναι ιδιοκτησία και να διαχειρίζεται από έναν επιχειρηματικό,ακαδημαϊκό, κυβερνητικό οργανισμό ή συνδυασμό των παραπάνω. Η υποδομή βρίσκεται στις εγκαταστάσεις του παρόχου αυτού.

***Υβριδικό Νέφος (Hybrid Cloud)***

Η υποδομή του Νέφους αποτελείται από έναν συνδυασμό δύο ή περισσοτέρων μοντέλων Νεφών (ιδιωτικό, κοινοτικό ή δημόσιο) τα οποία υπάρχουν και λειτουργούν αυτόνομα αλλά είναι διασυνδεδεμένα μέσω τυποποιημένων ή ιδιόκτητων τεχνολογιών που επιτρέπουν την μεταφορά δεδομένων και εφαρμογών (πχ. ενεργοποίηση δευτερεύοντος Νέφους με σκοπό την εξισορρόπηση φόρτου εργασίας – Cloud bursting).

## 2.2. Κύριοι πάροχοι υπηρεσιών υπολογιστικών νεφών

Στο σημείο αυτό θα επισημάνουμε τα δημοφιλέστερα συστήματα δημιουργίας και διαχείρισης Νεφών όπως και τις διεπαφές χρήσης (APIs) που παρέχουν. Εκτός από τους μεγάλους παρόχους εμπορικών Νεφών, έχουν περιγραφεί συστήματα ανοιχτού κώδικα που επιτρέπουν την δημιουργία και διαχείριση εικονικών υποδομών αλλά και την δια-λειτουργικότητα με άλλα εμπορικά Νέφη.

***Amazon EC2 API***

Η Amazon θεωρείται ως ο πρωτοπόρος πάροχος υπηρεσιών Υπολογιστικού Νέφους στην παρούσα αγορά. Η σχετική υπηρεσία που παρέχει ονομάζεται Amazon Web Services (AWS)[9] και παρουσιάστηκε αρχικά το 2002. Στην πράξη η AWS είναι ένα περιβάλλον διαδικτυακών υπηρεσιών οι οποίες στο σύνολό τους διαμορφώνουν την πλατφόρμα Νέφους της Amazon. Η εταιρία παρέχει επίσης μια πληθώρα σχετικών και εξειδικευμένων υπηρεσιών όπως: Elastic Compute Cloud (Amazon EC2), Simple Storage Service (S3), Simple Queue Service (SQS), CloudFront, Simple DB και άλλες. Σε αυτό το κεφάλαιο δεν θα επεκταθούμε περισσότερο σε όλες υπηρεσίες και τα προϊόντα της Amazon αλλά θα επικεντρωθούμε στα βασικά χαρακτηριστικά των δύο βασικών υπηρεσιών Νέφους: EC2[10] (Υπολογιστικό Νέφος) και S3 (Νέφος αποθήκευσης δεδομένων).

Το EC2 είναι μια διαδικτυακή υπηρεσία η οποία επιτρέπει την εγκατάσταση και διαχείριση στιγμιότυπων εξυπηρετητών στα κέντρα δεδομένων και υπολογιστών της Amazon. Η υπηρεσία αυτή παρέχει συγκεκριμένες διεπαφές χρήσης και προγραμματισμού (API) έτσι ώστε κάποιος να μπορεί να ελέγχει τους υπολογιστικούς πόρους, τους περιορισμούς πρόσβασης και γενικά να μπορεί να δημιουργήσει ένα προσωποποιημένο περιβάλλον εφαρμογών. Εκτός από τον πλήρη έλεγχο που παρέχεται, ένα δεύτερο σημαντικό χαρακτηριστικό της EC2 υπηρεσίας είναι η επεκτασιμότητα (*scalability*). Όπως και το ίδιο όνομα της υπηρεσίας υπονοεί (EC2: Elastic Cloud), η χωρητικότητα του κάθε πόρου μπορεί να προσαρμοστεί δυναμικά, είτε χειροκίνητα είτε αυτόματα από την εφαρμογή μέσω του παρεχόμενου API.

Η υπηρεσία S3 της Amazon παρέχει διαδικτυακό αποθηκευτικό χώρο που μπορεί να είναι προσβάσιμος από οποιοδήποτε, είτε άτομο είτε εφαρμογή στο διαδίκτυο. Η υπηρεσία αυτή εμφανίστηκε ως μια εναλλακτική λύση στα συμβατικά, τοπικά συστήματα αποθήκευσης δεδομένων και αποτελεί μια από τις βασικότερες υπηρεσίες της συλλογής AWS της Amazon. Παρόμοια με το την EC, η S3 παρέχει ένα API μέσω του οποίου μπορεί κάποιος να αποθηκεύσει και επανακτήσει δεδομένα από τα κέντρα δεδομένων και υπολογιστών της Amazon. Επιπρόσθετα, η εταιρία εμφανίζεται να διαφημίζει ότι μπορεί να παρέχει 99.99% διαθεσιμότητα και αξιοπιστία στα αποθηκευμένα δεδομένα, όπως αυτό περιγράφεται και στο συμβόλαιο υπηρεσίας (Amazon S3 Service Level Agreement).

Για να κατανοήσουμε καλύτερα την EC2 υπηρεσία, θα εξηγήσουμε σε αυτό το σημείο την ορολογία που η Amazon χρησιμοποιεί για την προδιαγραφή του σχετικού API. Στιγμιότυπο *(instance)* είναι ένας εικονικός εξυπηρετητής που εκτελείται σε ένα φυσικό πόρο στις εγκαταστάσεις της Amazon. Τα πρότυπα πάνω στα οποία η δημιουργία των στιγμιότυπων βασίζεται ονομάζονται Εικόνες Μηχανής της Amazon *(Amazon Machine Images* - AMIs*)* και συμπεριλαμβάνουν επιλογές του λειτουργικού συστήματος ή άλλες ρυθμίσεις που μπορεί κάποιος να επιλέξει για να προσδιορίσει το συγκεκριμένο στιγμιότυπο που θέλει να χρησιμοποιήσει.

Για να έχει κάποιος πρόσβαση σε ένα στιγμιότυπο, μπορεί να χρησιμοποιήσει τις ελαστικές IP διευθύνσεις της Amazon *(Amazon's Elastic IP Addresses)* οι οποίες είναι στατικές IP διευθύνσεις σχεδιασμένες για δυναμικά υπολογιστικά Νέφη. Αυτή η διεύθυνση είναι συσχετισμένη με το λογαριασμό του στην Amazon και μπορεί να επαναπροσδιοριστεί και να συνδεθεί με διαφορετικό στιγμιότυπο. *Elastic Block Storage* (EBS) είναι ο αποθηκευτικός χώρος για κάθε EC2 στιγμιότυπο στον οποίο κρατείται η κατάσταση (*state*) του κάθε στιγμιότυπου.

Επιπρόσθετα, μπορεί κάποιος να δημιουργήσει μια απεικόνιση της κατάστασης (*snapshot*) μιας συγκεκριμένης στιγμής η οποία θα αποθηκευθεί σε μια S3 υπηρεσία για μεγαλύτερη αντοχή στο χρόνο. Όπως παρουσιάζεται στο Σχήμα 11, η ροή εργασίας όταν χρησιμοποιούμε την EC2 ξεκινά από την αναζήτηση ενός διαθέσιμου δημόσια AMI και την προσαρμογή του στις ανάγκες μας είτε στην δημιουργία ενός από την αρχή. Το επόμενο βήμα είναι η δημιουργία ενός στιγμιότυπου του AMI χρησιμοποιώντας τις προγραμματιστικές διεπαφές (API) της EC2. Το αποτέλεσμα αυτής της διαδικασίας είναι ένα AMI ID που μπορεί να χρησιμοποιηθεί έτσι ώστε να εκτελέσουμε όσα στιγμιότυπα θέλουμε από το επιλεγμένο AMI. Τέλος, μέσω των παρεχομένων από το API εργαλείων, μπορούμε να διαχειριστούμε και να χρησιμοποιήσουμε τα στιγμιότυπα όπως επιθυμούμε σε οποιονδήποτε εξυπηρετητή.

Το API της Amazon EC2 παρέχει πρόσβαση στην διαδικτυακή υπηρεσία είτε μέσω SOAP API είτε μέσω Query API. Στην πρώτη περίπτωση, οι διεπαφές ορίζονται μέσω ενός XML κειμένου περιγραφής υπηρεσίας (*Web Service Description Language* - WSDL). Αφού τα αιτήματα και οι απαντήσεις σε SOAP, στην Amazon EC2, ακολουθούν τα προδιαγεγραμμένα πρότυπα οποιαδήποτε γλώσσα προγραμματισμού (π.χ. Java, C++, C#, Python, Perl και Ruby) μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών που θα εκτελούνται στο Νέφος της Amazon. Το Query API είναι μια διεπαφή βασισμένη στο μοντέλο REST [3] που υποστηρίζει λειτουργίες GET και POST κατά την πραγματοποίηση αιτημάτων. Η τεχνολογία αυτή φαίνεται να προτιμάται από τους προγραμματιστές της Amazon και παρέχεται σχεδόν από όλες τις υπηρεσίες AWS.

***VMware vCloud***

Η VMware ως πρωτοπόρος εταιρία στις τεχνολογίες εικονικοποίησης (*virtualization*), εξέδωσε το 2009 το προϊόν vCloud [37], μια πλατφόρμα Υπολογιστικού Νέφους βασισμένο στο πρότυπο OVF 1.0 ως το προγραμματιστικό περιβάλλον διεπαφής (API) για την διαχείριση εικονικών πόρων στο Νέφος. Το API του vCloud ήταν αποτέλεσμα συνδυαστικής προσπάθειας της VMware και των συνεργατών της με σκοπό να παραδώσουν μια εύκολη στη χρήση διεπαφή χρήσης υπηρεσιών Νέφους, επεκτάσιμο και βασισμένο σε διάφορα καθιερωμένα πρότυπα όπως XML, HTTP, OVF κ.α. Πιο λεπτομερώς, το vCloud API μπορεί να διαχωριστεί σε δύο μέρη: το διαχειριστικό (*Administrative*) API και το API χρηστών (*Users API*). Το πρώτο χρησιμοποιείται για την δημιουργία, διαχείριση και παρακολούθηση πόρων, χρηστών και ρόλων σε ένα περιβάλλον vCloud, ενώ το δεύτερο παρέχει λειτουργίες περιήγησης και αναζήτησης πόρων όπως και δημιουργίας, μετατροπής και εγκατάστασης λειτουργικών εικονικών συσκευών. Το VMware vCloud API επιτρέπει στους προγραμματιστές εφαρμογών να δημιουργούν προγράμματα-πελάτες για υπηρεσίες vCloud εκμεταλλευόμενοι το *RESTful* προγραμματιστικό παράδειγμα. Έτσι, οι εφαρμογές αυτές επικοινωνούν μεταξύ τους ανταλλάσσοντας μηνύματα σε XML που αναπαριστούν οντότητες του vCloud.

***Google App Engine***

Η είσοδος της Google στον χώρο των τεχνολογιών υπολογιστικού Νέφους έγινε μέσω της υπηρεσίας Google App Engine[14]. Σε αντίθεση με άλλους παρόχους και λύσεις οι οποίες υλοποιούν το παράδειγμα IaaS (π.χ. Amazon AWS), η υπηρεσία App Engine είναι ένα σύστημα Πλατφόρμα-ως-Υπηρεσία (PaaS). Σε αυτή τη βάση, η Google App Engine είναι μια πλατφόρμα μέσω της οποίας ένας προγραμματιστής μπορεί να εγκαταστήσει και εκτελέσει ένα διαδικτυακή εφαρμογή στα κέντρα δεδομένων της Google. Αυτή τη στιγμή η πλατφόρμα υποστηρίζει ανάπτυξη εφαρμογών γραμμένες σε Java και Python άλλες γλώσσες βασισμένες σε JVM όπως Groovy, JRuby και Scala. Εκτός από το περιβάλλον ανάπτυξης (SDK) που παρέχεται, υπάρχουν διαθέσιμες προεκτάσεις προγραμμάτων (*plugins*) για σουίτες όπως το Eclipse.Η διαχείριση δεδομένων στην πλατφόρμα Νέφους της Google υλοποιείται μέσω του Datastore API. Στην περίπτωση ανάπτυξης εφαρμογών σε Java, το Datastore API αποθηκεύει και εκτελεί ερωτήματα (*queries*) πάνω σε αντικείμενα δεδομένων γνωστά ως *οντότητες*. Κάθε οντότητα έχει ένα μοναδικό χαρακτηριστικό κωδικό (*key*) και μια ή περισσότερες ιδιότητες, ως τιμές συγκεκριμένων τύπων δεδομένων. Το Datastore υποστηρίζει τις Java Data Objects (JDO) 2.3 και Java Persistent API (JPA) 1.0 πρότυπες διεπαφές.

Στην περίπτωση ανάπτυξης εφαρμογών με την χρήση Python, το Datastore υλοποιείται μέσω μιας γλώσσας παρόμοιας με SQL η οποία ονομάζεται QGL. Η γλώσσα αυτή δεν υποστηρίζει δηλώσεις Ένωσης (*Join*) καθώς είναι ανέφικτη η εφαρμογή τέτοιων λειτουργιών όταν τα δεδομένα είναι τοποθετημένα σε πολλαπλά μηχανήματα. Υπό αυτές τις συνθήκες, η γλώσσα αυτή δεν είναι μια σχεσιακή βάση δεδομένων αλλά παρόμοιες λειτουργίες μπορούν να υλοποιηθούν μέσω συγκεκριμένων μηχανισμών που παρέχονται από το API της QGL.

Όπως αναφέραμε, η Google App Engine πλατφόρμα παρέχει ένα API και συνολικά ένα ολοκληρωμένο περιβάλλων ανάπτυξης εφαρμογών είτε μέσω Java είτε μέσω Python. Το περιβάλλον αυτό προσομοιώνει όλες τις λειτουργίες της Google App Engine και σου δίνει την δυνατότητα να εγκαταστήσεις κατευθείαν την εφαρμογή στο περιβάλλον Νέφους. Η όλη διαδικασία περιγράφεται με τα παρακάτω βήματα:

- Ανάπτυξη της εφαρμογής χρησιμοποιώντας τα διαθέσιμα εργαλεία και APIs
- Καταχώρηση ενός Κωδικού Εφαρμογής (Application ID) στην υπηρεσία Google App Engine μέσω της Κονσόλας Διαχείρισης (Administration Console)
- Μεταφόρτωση των αρχείων της εφαρμογής
- Πρόσβαση στην εφαρμογή μέσω του διαδικτύου, χρησιμοποιώντας συγκεκριμένη διεύθυνση βασισμένη στον Κωδικό Εφαρμογής

Η Google παρέχει δωρεάν χώρο χρήσης στους εξυπηρετητές και κέντρα δεδομένων της, με κάποιους περιορισμούς. Οι προγραμματιστές μπορούν πάντα να επεκτείνουν την χρήση ενεργοποιώντας την χρέωση στον λογαριασμό τους. Ενδεικτικά, στον παρακάτω πίνακα (Πίνακας 1) παρουσιάζονται τα χαρακτηριστικά δωρεάν χρήσης.

| Quota | Limit |
|---|---|
| Apps per developer | 10 |
| Time per request | 30 sec |
| Blobstore size (total file size per app) | 1 GB |
| HTTP response size | 10 MB |
| Datastore item size | 1 MB |
| Application code size | 150 MB |

Πίνακας1: Google App Engine free quota

***Microsoft Azure***

Η γρήγορη εξέλιξη του Υπολογιστικού Νέφους και των σχετικών τεχνολογιών δεν θα μπορούσε να αφήσει την Microsoft εκτός του κλάδου αυτού. Ως αποτέλεσμα, η Microsoft εξέδωσε την λύση Windows Azure [15] ως ένα λειτουργικό σύστημα που παρέχει υπηρεσίες Νέφους. Στην πραγματικότητα το Azure είναι μια πλατφόρμα υπηρεσιών που επιτρέπει την ανάπτυξη, εγκατάσταση και εκτέλεση εφαρμογών Windows στα κέντρα δεδομένων και υπολογιστών της Microsoft. Οι προγραμματιστές μπορούν να αναπτύσσουν τις εφαρμογές χρησιμοποιώντας τις συνηθισμένες γλώσσες προγραμματισμού των Windows (C#, C++, VMκλπ) ή και άλλες γλώσσες που υποστηρίζονται (π.χ. Java, Ruby, PHP, Python) μέσω της σουίτας Microsoft Visual Studio.

Η πλατφόρμα Azure αποτελείται από τρία βασικά στοιχεία: Υπηρεσία Υπολογισμού (Compute Service), Υπηρεσία Αποθήκευσης (Storage Service) και Στρώμα Εφαρμογών (Application Fabric). Το πρώτο παρέχει τις απαραίτητες διεπαφές και υποστήριξη για την εκτέλεση των εφαρμογών οι οποίες μπορούν να έχουν πολλαπλά εγκατεστημένα στιγμιότυπα. Όλες οι εφαρμογές μπορούν να έχουν πρόσβαση σε πόρους δεδομένων κάνοντας χρήση της Υπηρεσίας Αποθήκευσης μέσω διεπαφών REST. Η υπηρεσία αυτή παρέχει αντικείμενα BLOB για την αποθήκευση μεγάλων δυαδικών αντικειμένων, πινάκων και ουρών για την διαχείριση δεδομένων. Για εφαρμογές πιο απαιτητικές όσον αφορά την διαχείριση δεδομένων, η πλατφόρμα παρέχει το SQL Azure Database, ένα σύστημα Νέφους για διαχείριση δεδομένων (DBMS). Το σύστημα αυτό βασίζεται στο παλιότερο Microsoft SQL Server αλλά υποστηρίζει ένα διαχειριστικό περιβάλλον στο Νέφος. Η πρόσβαση των δεδομένων μπορεί να πραγματοποιηθεί μέσω των διεπαφών ADO.NET ή άλλων διεπαφών πρόσβασης των Windows.

Η υπηρεσία για την υποστήριξη της υποδομής του Νέφους στο Windows Azure πραγματοποιείται μέσω του Στρώματος Εφαρμογών (*Application Fabric*). Κάθε εφαρμογή μπορεί να δημιουργήσει διεπαφές πρόσβασης (*endpoints*) χρησιμοποιώντας το εργαλείο Service Bus του Στρώματος Εφαρμογών έτσι ώστε να μπορούν να χρησιμοποιηθούν από άλλες εφαρμογές του Νέφους ή και ανεξάρτητες εφαρμογές. Η διασύνδεση ενός προγράμματος πελάτη REST προς μία εφαρμογή ελέγχεται από το εργαλείο *Access Control* του Στρώματος Εφαρμογών. Οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές είτε με ρόλο *Web* είτε με ρόλο *Worker*, και να προσδιορίσουν έτσι πόσα αντίγραφα στιγμιότυπων θέλουν να εκτελέσουν στις εικονικές μηχανές των Windows. Αυτές οι εικονικές μηχανές (VMs) δεν δημιουργούνται από τον προγραμματιστή αλλά παρέχονται από το Azure σύστημα (συγκεκριμένα το *hypervisor* υποσύστημα) το οποίο είναι ειδικά σχεδιασμένο για λειτουργία σε περιβάλλον Νέφους. Οι εφαρμογές Web είναι συνήθως υλοποιημένες σε ASP.NET περιβάλλον ενώ τα στιγμιότυπα *Worker* είναι εργασίες που αλληλεπιδρούν με τις διαδικτυακές εφαρμογές μέσω της Υπηρεσίας Αποθήκευσης.

***Sun Cloud***

Η Sun Microsystems ενεπλάκη με τις τεχνολογίες Νέφους το 2009 όταν παρουσίασε την δική της υποδομή Υπολογιστικού Νέφους και το αντίστοιχο API με την ονομασία Sun Open Cloud [16]. Όπως και οι λέξεις υπονοούν, το σύστημα Νέφους της Sun είναι μια λύση ανοιχτού κώδικα (*open source*) με το API να δημοσιεύεται υπό την άδεια της *Creative Common*, το οποίο ουσιαστικά επιτρέπει σε οποιονδήποτε να το χρησιμοποιήσει με κάθε τρόπο.

Το Open Cloud αποτελείται από δύο κύρια μέρη: την υπηρεσία αποθήκευσης *Sun Cloud Storage Service* και την υπηρεσία υπολογισμού *Sun Cloud Compute Service*. Η πρώτη είναι ένα σύνολο διαδικτυακών υπηρεσιών και πρωτοκόλλων *WebDAV* και παρέχουν δυνατότητες αποθήκευσης δεδομένων και ανάκτησής τους σε διάφορες μορφές. Η υπηρεσία αυτή είναι επίσης συμβατή με την αντίστοιχη υπηρεσία δεδομένων της Amazon, το Amazon S3 API. Η Sun Cloud Compute Service παρέχει στον προγραμματιστή όλα τα απαραίτητα εργαλεία και διεπαφές για να αναπτύξει και να λειτουργήσει ένα κέντρο δεδομένων στο Νέφος, ή χρησιμοποιώντας την ορολογία της Sun, ένα Εικονικό Κέντρο Δεδομένων (*Virtual Data Center* – VDC). Το VDC παρέχει ένα φιλικό προς τον χρήστη περιβάλλον εργασίας, προσβάσιμο μέσω ενός απλού περιηγητή διαδικτύου μέσω του οποίου κάποιος μπορεί να σχεδιάσει μια εφαρμογή, με διαφορετικές απαιτήσεις λειτουργικού συστήματος (Windows, Solaris, Linux κ.α.) και που θα εκτελείται σε ένα Νέφος.

Το Sun Cloud API είναι μια διεπαφή προγραμματισμού βασισμένη στο παράδειγμα REST, ορίζοντας, λοιπόν, κάθε οντότητα ως ένα πόρο στο Νέφος (υπολογιστικοί, δικτυακοί, πόροι αποθήκευσης δεδομένων κ.α.). Η χρήση του API αυτού γίνεται με το HTTP πρωτόκολλο και μέσω των διαδεδομένων αιτημάτων GET, PUT, POST και DELETE. Το API λειτουργεί βασισμένο στους παρακάτω τύπους πόρων:

- **Νέφος (Cloud):** Μια δομή υψηλού επιπέδου η οποία συγκεντρώνει όλα τα Εικονικά Κέντρα Δεδομένων στα οποία έχει πρόσβαση ένας χρήστης
- **Εικονικό Κέντρο Δεδομένων (Virtual Data Center - VDC):** Ένας απομονωμένος χώρος ο οποίος εμπεριέχει συστάδες υπολογιστών (Clusters), εικονικά ιδιωτικά δίκτυα (Private Virtual Networks), δημόσιες διευθύνσεις (Public Addresses), αποθηκευτικές μονάδες (Storage Volumes) κ.α.
- **Συστάδες Υπολογιστών (Cluster):** Μια διαχειριστική ομαδοποίηση Εικονικών Μηχανών (Virtual Machines), χρήσιμες για έλεγχο πρόσβασης, αντιγραφή, γεωγραφική απομόνωση κ.α.
- **Εικονική Μηχανή (Virtual Machine - VM**): Ένας εξυπηρετητής
- **Εικονικά Ιδιωτικό Δίκτυο (Private Virtual Network - VNet):** Ένα υποδίκτυο, που δεν είναι συνδεδεμένο με το διαδίκτυο, και μπορεί να χρησιμοποιηθεί για να συνδεθούν οι Εικονικές Μηχανές (VMs) με ένα Εικονικό Κέντρο Δεδομένων (VDC).
- **Δημόσια Διεύθυνση (Public Address):** Η σύνδεση με το διαδίκτυο.
- **Μονάδα Αποθήκευσης (Storage Volume):** Ένας πόρος αποθήκευσης ο οποίος μπορεί να προσπελαστεί μέσω WebDAV ή κάποιο άλλο πρωτόκολλο αποθήκευσης.

***Eucalyptus***

Το Eucalyptus[17] ήταν μια από τις πρώτες εφαρμογές ανοιχτού κώδικα που επικεντρώθηκε στην δημιουργία IaaS Νεφών. Δημιουργήθηκε έτσι ώστε να παρέχει λειτουργίες παρόμοιες με το Amazon Web Services API αλλά διατίθεται ως εφαρμογή open source. Οι χρήστες μπορούν να αλληλεπιδρούν με το Νέφος του Eucalyptus χρησιμοποιώντας τα ίδια εργαλεία που χρησιμοποιούν για να έχουν πρόσβαση στο Amazon EC2. Επιπλέον όμως, παρέχεται ένα Νέφος αποθήκευσης API για την αποθήκευση των γενικών δεδομένων των χρηστών και των VM εικόνων. Συνοψίζοντας, το Eucalyptus παρέχει τα ακόλουθα συστατικά:

- Linux-based controller
- EC2-compatible (SOAP, Query) , S3-compatible (SOAP, REST) CLI και Web portal interfaces
- Xen, KVM, και VMWare backends
- Amazon EBS-compatible virtual storage devices
- Διεπαφή προς το Amazon EC2 public cloud
- Εικονικά δίκτυα (virtual networks)

Η αρχιτεκτονική του Eucalyptus, αποτελεί συστατικό στοιχείο κάθε συστήματος σε υψηλό επίπεδο ως αυτόνομη υπηρεσία Web με τα ακόλουθα στοιχεία ελέγχου:

- **Node controller (NC):** Ελέγχει την εκτέλεση, επιθεώρηση και τον τερματισμό των στιγμιότυπων των εικονικών μηχανών (VMs) στον χώρο που φιλοξενούνται και εκτελούνται.
- **Cluster controller (CC):** Συλλέγουν πληροφορίες σχετικά με τις εκτελέσεις των VMs αλλά και τις προγραμματίζουν σε συγκεκριμένους node controllers (NC), τόσο καλά όσο διαχειρίζονται τις περιπτώσεις εικονικών δικτύων.
- **Storage controller (SC):** Είναι μια υπηρεσία λήψης/αποθήκευσης που εφαρμόζει την διεπαφή του Amazon S3 και παρέχει τον τρόπο αποθήκευσης και πρόσβασης της πληροφορίας που έχει χρησιμοποιηθεί από τον χρήστη.
- **Cloud controller (CLC):** Είναι το σημείο εισόδου για το Νέφος για απλούς χρήστες και διαχειριστές. Θέτει ερωτήματα στους διαχειριστές κόμβων για πληροφορίες και πόρους, παίρνει αποφάσεις προγραμματισμού ενεργειών υψηλού επιπέδου, και εφαρμόζει όλα αυτά κάνοντας αιτήματα (*requests*) στου ελεγκτές συμπλεγμάτων (*cluster controllers*).
- **Walrus (W):** Είναι το εξάρτημα του ελεγκτή που διαχειρίζεται την πρόσβαση στις υπηρεσίες αποθήκευσης μέσα στον Eucalyptus. Τα αιτήματα αποστέλλονται στον Walrus χρησιμοποιώντας διεπαφές SOAP ή REST.

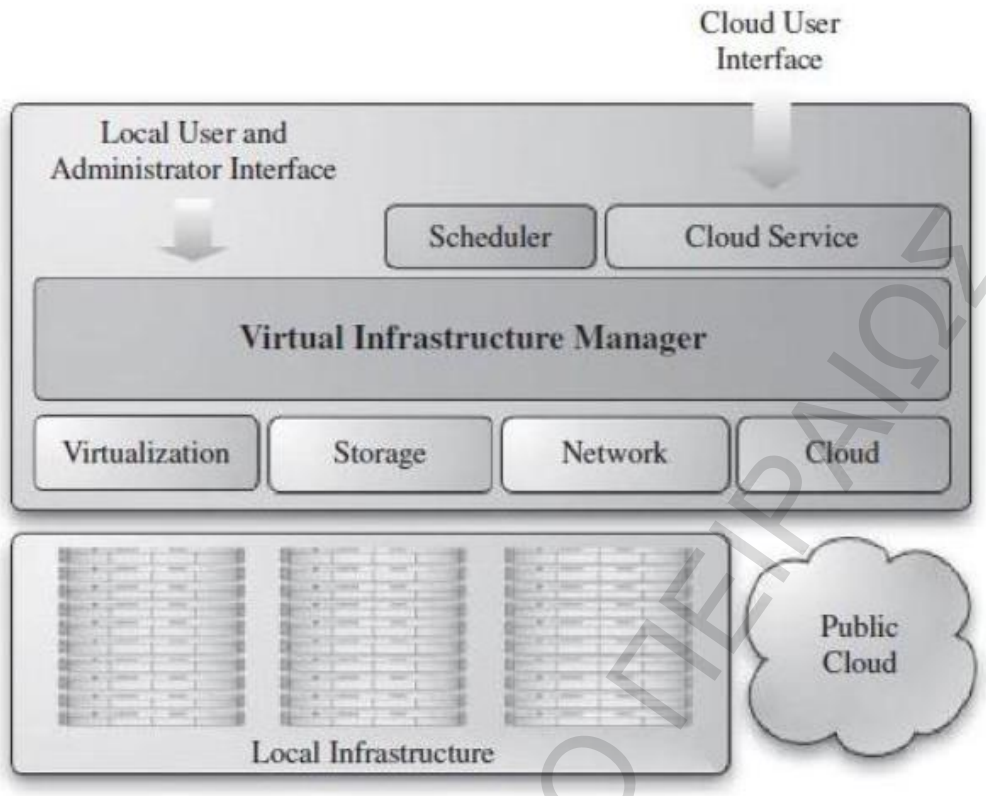
***Open Nebula***

Το Open Nebula[6] είναι μία από τις πιο πλούσιες εφαρμογές ανοιχτού κώδικα για υλοποίηση IaaS. Αρχικά είχε σχεδιαστεί για την διαχείριση εικονικών υποδομών και περιλάμβανε απομακρυσμένες διεπαφές που καθιστούσαν υλοποιήσιμη την κατασκευή δημόσιων Νεφών. Συνολικά τέσσερα APIs είναι διαθέσιμα:

- XML-RPC
- Libvirt
- EC2 (Query) APIs
- OpenNebula Cloud API (OCA)

Η αρχιτεκτονική του περιλαμβάνει διάφορα εξειδικευμένα δομικά συστατικά. Η κύρια ενότητα της αρχιτεκτονικής του περιλαμβάνει φυσικούς εξυπηρετητές και τα *hypervisors* τους, τους κόμβους αποθήκευσης και τα στρώμα δικτύου. Η διαχείριση των εργασιών πραγματοποιείται από οδηγούς που αλληλεπιδρούν με τα APIs των *hypervisors*, με τις συσκευές αποθήκευσης και τις τεχνολογίες δικτύων των δημόσιων Νεφών. Συνοψίζοντας, το Open Nebula έχει τις ακόλουθες δυνατότητες:

- CLI, XML-RPC, EC2-compatible Query και OCA interfaces
- Xen, KVM, και VMware backend
- Διεπαφές σε δημόσια Νέφη (AmazonEC2, ElasticHosts)
- Εικονικά δίκτυα
- Δυναμική ανάθεση πόρων
- Προληπτική δέσμευση πόρων




Εικόνα 4: Αρχιτεκτονική υψηλού επιπέδου Open Nebula[6]

**Δικτύωση**: Γενικά, οι υπηρεσίες που αναπτύσσονται στο Νέφος, από μια συστάδα υπολογιστών (*cluster*) προς την κλασική three-tier επαγγελματική εφαρμογή, απαιτούνται πολλαπλές αλληλένδετες εικονικές μηχανές (VMs) με ένα εικονικό δίκτυο εφαρμογών (VAN) να είναι ο συνδετικός κρίκος μεταξύ τους. Το Open Nebula δημιουργεί δυναμικά αυτά τα εικονικά δίκτυα εφαρμογών και ακολουθεί τις MAC διευθύνσεις που χρησιμοποιήθηκαν στο δίκτυο για τις υπηρεσίες των VMs.

***OpenStack***

ToOpenStack [7] είναι ένας πάροχος υποδομής υπολογιστικού νέφους ανοικτού κώδικα. Δημιουργήθηκε από τη RackSpace και τη NASA το 2010. Το OpenStack αποτελείται από τα ακόλουθα ξεχωριστά υπό-προγράμματα:

- Το **Nova,** για τη δημιουργία υπολογιστικών πόρων.
- Το **Swift,** που αποτελεί μία υπηρεσία αποθηκευτικών πόρων αρχείων.
- Το **Keystone,** που αποτελεί τη κύρια υπηρεσία αυθεντικοποίησης χρηστών.
- Το **Glance,** για την αποθήκευση στιγμιοτύπων (images) λειτουργικών συστημάτων.
- Το **Horizon,** που αποτελεί μία δικτυακή εφαρμογή για τη διαχείρηση της υποδομής.

Σήμερα το OpenStackχρησιμοποιείται ευρέως από μεγάλους οργανισμούς, παρόχους υπηρεσιών, ερευνητές, μικρές και μεσαίες επιχειρήσεις που θέλουν να εγκαταστήσουν μεγαλού μεγέθους υποδομές υπολογιστικών νεφών. Ολόκληρος ο κώδικας είναι διαθέσιμος (Apache 2.0 license).

***Rackspace cloud***

Το Rackspace Cloud[24] αποτελεί ένα εμπορικό πάροχο υποδομής υπολογιστικού νέφους (IaaS) από το 2006, θέτοντας τον από τους πρώτους στο είδος του. Προς το παρόν το Rackspaceδιατήρει κέντρα δεδομένων μόνο στις Η.Π.Α. Σχεδιάζεται να δημιουργηθούν περαιτέρω κέντρα εξυπηρετητών στο Ηνωμένο Βασίλειο. Παρέχονται 3 βασικές υπηρεσίες:

- **CloudServer**, που είναι μία υπηρεσία υποδομής (IaaS) που προσφέρεται για τη λειτουργία εικονικών μηχανημάτων.
- **CloudFIles**, που είναι υπηρεσίας παροχής αποθηκευτικού χώρου για αρχεία και διαμοιρασμού περιεχομένου (CDN).
- **CloudSite**, που αποτελεί υπηρεσία παροχής πλατφόρμας (PaaS), για τη λειτουργία εφαρμογών που βασίζονται στη στοίβα εργαλιών LAMP (Linux,Apache,MySQL,PHP) ή στη πλατφόρμα .NET.

***FlexiScale***

Η Flexiant είναι μία εταιρία υπηρεσιών και λογισμικού και μία ανεξάρτητη πάροχος δημόσιου υπολογιστικού νέφους. Παρέχει υπηρεσίες και λογισμικό υποδομής υπολογιστικού νέφους και υπηρεσίες για παρόχους φιλοξενίας, κατόχους εξυπηρετητών δεδομένων και χειριστών τηλεπικοινωνιών. Το προιόν Exilitytης εταιρίας κυκλοφόρησε το Μάρτιο του 2010.

Το δημόσιο νέφος, FlexiScale [18], είναι η πρώτη πλατφόρμα υπολογιστικού νέφους στην Ευρώπη και ξεκίνησε τη λειτουργία της για πρώτη φόρα το 2007. Επιτρέπει στους τελικούς χρήστες να αγοράσουν υπηρεσίες υπολογιστικών πόρων σε μία ευέλικτη, επεκτάσιμη και αυτοματοποιήμενη υποδομή. Οι πελάτες μπορούν να διαμορφώσουν τις απαιτήσεις τους κατά βούληση και να πληρώνουν μόνο για τις υπηρεσίες που ουσιαστικά χρησιμοποιούν.

***Joyent Smartmachines***

Η εταιρία Joyent προσφέρει υπηρεσίες νέφους στο ιδιωτικό της δίκτυο[19]. Το λογισμικό το οποίο στήριξε το δημόσιο νέφος της Joyenteίναι διαθέσιμο σε παρόχους υπηρεσιών ως μία ολοκληρωμένη λύση υπολογιστικού νέφους. Η λύση Joyent Smart Data Center[20] περιέχει τα πάντα, τα οποία οι πάροχοι χρειάζονται για να προσφέρουν μία υπηρεσία υπολογιστικού νέφους, στους πελάτες τους. Έχει σχεδιαστεί έτσι ώστε να μεγιστοποιήσει τη δυνατότητα των διαχειρηστών να ανταγωνιστούν τους υπόλοιπους παρόχους (AmazonEC2, Google, Microsoft). Τα κύρια χαρακτηριστικά που παρέχονται συνοψίζονται ως εξής:

- **Αποδεδειγμένη Τεχονολογία Νέφους.** Το SmartDataCenterέχει κτιστεί πάνω στο SmartOS, το οποίο είναι ένα γρήγορο, ασφαλές, επεκτάσιμο και σταθερό λειτουργικό σύστημα νέφους το οποίο περιέχει το ισχυρό λογισμικό που χρειάζεται για να λειτουργήσει μία υπηρεσία υπολογιστικού νέφους.
- **Αποδεδειγμένο επιχειρησιακό μοντέλο.** Το SmartDataCentereίναι βασισμένο σε 5+ χρόνια εμπειρίας σε ανάπτυξη και υποστήριξη λύσεις υπολογιστικών νεφών, ικανές να υποστηρίζουν μαζική επέκταση σε αριθμό χρηστών. Η λύση περιέχει διαδικασίες οι οποίες βοηθούν τους παρόχους νεφών στη γρήγορη εγκατάσταση.
- **Γρήγορη εκκίνηση και υποστήριξη υλοποίησης.** Η Joyent συνεργάζεται με εταιρίες ολοκλήρωσης συστημάτων, όπως η DellServicesκαι η Data Craft Asia, και προγραμματιστές για να προσφέρει ολοκληρωμένες υπηρεσίες, όπως παραμετροποίηση, ολοκλήρωση, διαχείρηση, υποστήριξη και παροχή τεχνικών συμβουλών, με χρόνο παράδοσης λιγότερο από 3 μήνες.

Το προιόν αποτελεί έναν άμεσο ανταγωνιστή προς τα κορυφαία προιόντα στο χώρο. Επιπρόσθετα, προσφέρει τη δυνατότητα να παραμετροποιείται ανά πελάτη η λύση υπολογιστικού νέφους. Για το σκοπό αυτό προσφέρουν υπηρεσίες συμβουλών με επικέντρωση στις επιδόσεις εφαρμογών και την επεκτασιμότητα των υπηρεσιών. Οι επιδόσεις μπορούν να αυξηθούν μέσω συγκεκριμένων προιόντων που χαρακτηρίζουν το είδος του εικονικού μηχανήματος.

- Joyent Smart OS (προιόν Unixυπολογιστικού νέφουςβασισμένο σε Solaris)
- MySQL Accelerator
- Riak
- Zeus Traffic manager

Η ονομασία REC αντιστοιχεί στα SmartMachines, τα οποία προωθούνται ως επικρατέστερα των κοινών εικονικών μηχανημάτων. Τα SmartMachines είναι και αυτά εικονικά μηχανήματα, βελτιστοποιημένα για εφαρμογές λογισμικού, αντί να είναι σχεδιασμένα για την αντιγραφή υλικού. Οι ιδιωτικοί εικονικοί εξυπηρετητές είναι ελαφριοί σε καταλάλωση πόρων από τα κοινά εικονικά μηχανήματα και προσφέρουν μεγαλύτερη διαφάνεια στην εκτέλεση των SmartOS. Ενώ τα SmartOSμπορούν να παρέχουν και εικονικά μηχανήματα αλλά και SmartMachines, γενικώς προτείνονται τα SmartMachinesγια τις περισσότερες εφαρμογές. Ένα αξιοσημείωτο χαρακτηριστικό είναι οι εκρηκτικές παράμετροι τους. Η διαδικτυακή κίνηση δεν είναι ομαλή, καθώς έχει καθημερινά, ωριαία, εβδομαδιαία, ακόμα και μηνιαία μέγιστα. Ο αποκλειστικός εικονικός εξυπηρετητής του, μπορεί να χειριστεί αυτό τοδιακυμενόμενο φόρτο με το ναχρησιμοποιείπρόσθετες μονάδες επεξεργαστών (bursting), οι οποίες δεν χρησιμοποιούνται από άλλες εφαρμογές. Δεν υπάρχει καθυστέρηση ή ανάγκη για τα μηχανήματα να εκκινούν (boot). Πολλές εφαρμογές στα SmartMachinesυπό κανονικές συνθήκες μπορούν να εξυπηρετήσουν 10 φορές πιο πολύ διαδικτυακό φορτίο. Η πλατφόρμα SmartPlatformπαρέχει ένα JavaScriptAPI, για τους χρήστες που επιθυμούν να αναπτύξουν εφαρμογές επάνω σε αυτό.

Το σημαντικότερο προιόν της Joyenteίναι το SmartDataCenter (πρώην CloudControl). Το Smart Data Center διαχειρίζεται μία συλλογή από SmartMachinesγια να εξασφαλίσει τη βέλτιστη χρησιμοποίηση πόρων, ενώ παρέχει ένα API το οποίο επιτρέπει πλήρη έλεγχο και επεκτασιμότητα των πειραμάτων. Με τον έλεγχο των υπολογιστικών, αποθηκευτικών και δικτυακών πόρων, η αρχιτεκτονική νέφους του SmartDataCenterδίνει στους χρήστες ένα εικονικό κέντρο ελέγχου και μία πλατφόρμα ανάπτυξης εφαρμογών η οποία κάνει το κέντρο ελέγχου να δουλεύει όπως σε ένα κεντρικό υπολογιστή.

Το SmartDataCenterπροσφέρει τα ακόλουθα:

- **Πλήρη αυτοματοποίηση των κέντρων δεδομένων**. Η εικονοποίηση των κέντρων δεδομένων υποστηρίζει πολύπλοκες, πολλαπλών-συσκευών αρχιτεκτονικές, εικονικά τοπικά δίκτυα (VLANs), εξισορρόπιση φόρτου, δρομολόγιση, VPNsκ.α.
- **Διαχείρηση πελατών**. Παρέχει μία διεπαφή φιλική προς το χρήστη για να διαχειρίζεται ολόκληρη τη συλλογή από SmartMachinesπου διαθέτει.
- **Πιστοποιημένη επεκτασιμότητα**. Η πλατφόρμα SmartDataCenterπαρέχει επεκτασιμότητα σε ένα περιβάλλον πολλαπλών «ενοίκων» (mutli-tenant), με πολλούς φυσικούς πόρους.
- **Ευέλικτες επιλογές εγκατάστασης**. Περίεχει τις υπηρεσίες φιλοξενίας της Joyent, δημόσιους ή ιδιωτικούς εξυπηρετητές από τρίτους παρόχους.

***Cross Platform Cloud APIs***

Εκτός από τους παρόχους Νεφών με συγκεκριμένα APIs υπάρχουν διεπαφές προγραμματισμού για Υπολογιστικά Νέφη ανεξάρτητα πλατφόρμας. Το Jclouds [21] είναι πλαίσιο ανάπτυξης εφαρμογών για Νέφη ανοιχτού κώδικα και βασισμένο στην Java. Χρησιμοποιώντας τις βιβλιοθήκες αυτού του συστήματος, κάποιος μπορεί να αναπτύξει προγράμματα και εφαρμογές συμβατά με διάφορους παρόχους Νέφους όπως Amazon, VMWare, Azure και άλλους. Το Deltacloud[22] είναι ένα επίσης API ανοιχτού κώδικα βασισμένο στο REST, διατίθεται από την RedHat και είναι συμβατό με το EC2, Rackspace, GoGRID και άλλους παρόχους. Στην ίδια λογική, το libcloud[23] είναι μια βιβλιοθήκη της Python, που επιτρέπει στις εφαρμογές να αλληλεπιδρούν με διαφορετικές υλοποιήσεις Υπολογιστικών Νεφών. Διατίθεται υπό την άδεια λογισμικού *Apache Software License* και εξυπηρετεί μέσω συγκεκριμένων οδηγών (*drivers*) τα περισσότερα εμπορικά Νέφη.

## 2.3. Πρότυπα

***Open Cloud Computing Interface (OCCI)***

Κατά τη διάρκεια της OGF 25 τον Μάρτιο του 2009, ο Ignacio M. Llorente του πανεπιστημίου της Μαδρίτης (UCM – OpenNebula )και ο Thijs Metsch (Sun Microsystems - RESERVOIR project) ίδρυσαν την ομάδα εργασίας Open Cloud Computing Interface Working Group (OCCI)[24]μετα αρχικό όνομα Cloud API (CAPI). Στις επόμενες συναντήσεις OGF26 και OGF27 η ομάδα μετονομάστηκε σε OCCI και τα πρώτα αποτελέσματα παρουσιάστηκαν. Ο σκοπός της ομάδας είναι η ανάπτυξη ενός "καθαρού" και ανοιχτού API για την υλοποίηση του παραδείγματος Υποδομή-ως-Υπηρεσία (IaaS) για Νέφη. Η ενεργή συμμετοχή ξεπερνά τα 200 μέλη και καθοδηγείται από τέσσερις προεδρεύοντες από την βιομηχανία, τον ακαδημαϊκό τομέα, παρόχους υπηρεσιών και χρήστες.

Το OCCI θα παρέχει ένα "λεπτό" αλλά επεκτάσιμο API βασισμένο στο REST παράδειγμα. Κάθε πόρος που προσδιορίζεται μέσω του OCCI θα κατέχει μια μοναδική διεύθυνση, χρησιμοποιώντας ένα URI (*Uniform Resource Identifier*). Το API αυτό υλοποιεί όλες τις CRUD λειτουργίες (*Create, Retrieve, Update* και *Delete*), μέσω των μεθόδων/ρημάτων POST, GET, PUT και DELETE αντιστοίχως. Οι τύποι των πόρων που υποστηρίζονται αφορούν την αποθήκευση δεδομένων, δικτυακούς και υπολογιστικούς πόρους, και μπορούν να συνδεθούν μεταξύ τους έτσι ώστε να δημιουργήσουν μια εικονική μηχανή με συγκεκριμένες δυνατότητες.

Το πρότυπο OCCI είναι σχεδιασμένο να είναι δυναμικό και να βασίζεται σε ξεχωριστές μονάδες. Τα χαρακτηριστικά και οι δυνατότητές του περιγράφονται σε μια σειρά παραδοτέων κειμένων και η ομάδα εργασίας συνεχίζει το έργο εξέλιξής και βελτίωσης του προτύπου αυτού (OCCI Core & Models, OCCI Infrastructure Models, OCCI XHTML5 rendering, OCCI HTTP Header rendering).

***WebServices***

Μία υπηρεσία διαδικτύου είναι διεπαφή προγραμματισμού εφαρμογών (API) η οποία είναι προσπελάσιμη μέσω του πρωτοκόλλου HTTPκαι εκτελείται σε από ένα απομακρυσμένο σύστημα. Χρησιμοποιούνται για να συνδέσουν εξωτερικές ή τοπικές εφαρμογές σε μία υποδομή νέφους. Υπάρχει μία ευρεία γκάμα προτύπων υπηρεσίων διαδικτύου, η οποίες υποστηρίζονται από τους δύο κύριους οργανισμούς OASIS[26]και W3C[27].

***Universal Description, Discovery and Integration (UDDI)***

Η γλώσσα περιγραφής, ανακάλυψης και ολοκλήρωσης (UDDI)[25]είναι μία γλώσσας σήμανσης (XML). Χρησιμοποιείται από επιχειρήσεις από όλα το κόσμος για να δημοσιεύσουν τις υπηρεσίες διαδικτύου που έχουν υλοποιήσει. Είναι επίσης ένας μηχανισμός για την εύρεση τέτοιων υπηρεσιών. Το UDDIήταν μία πρωτοβουλία της βιομηχανίας, και υποστηρίχθηκες απο τον οργανισμό για τη πρόοδο προτύπων δομημένης πληροφορίας (OASIS). Επιπρόσθετα με το ευρετήριο υπηρεσίων, το UDDIπροσδιορίζει το τρόπο με τον οποίο οι υπηρεσίες και οι εφαρμογές διαδικτύου αλληλεπιδρούν μέσα στο διαδίκτυο.

Αρχικά αναπτύχθηκε για να «ανακρίνεται» από τα μηνύματα SOAPκαι να προσφέρει πρόσβαση στα αρχεία περιγραφής υπηρεσιών (WSDL), που περιγράφουν τα προαπαιτούμενα και τις προδιαγραφές που πρέπει να ικανοποιούν οι υπηρεσίες για να αλληλεπιδρούν μεταξύ τους.

***Open Virtualization Format***

Αποτελεί ένα πρότυπο ανοικτού κώδικα για τη ομαδοποίηση και διανομή ενός ή περισσότερων εικονικών μηχανημάτων. Τα κύρια χαρακτηριστικά και πλεονεκτήματα του OVF[28]είναι:

- **Βελτιστοποιημένη διανομή.** Τοπρότυπο OVFεπιτρέπει τη φορητότητα και τη διανομή εικονικών πόρων. Επιπρόσθετα, εκτός από τη συμπίεση για πιο αποδοτική μεταφορά πακέτων, το πρότυπο υποστηρίζει πιστοποίηση περιεχομένου και έλεγχο ακεραιότητας και παρέχει ένα βασικό σχήμα για τη διαχείρηση αδειών λογισμικού.
- **Απλή και αυτοματοποιημένη χρήση.** Το πρότυπο προσφέρει μία ευρωστη και φιλική προς το χρήστη προσέγγιση για τον εξορθολογισμό της διαδικασίας εγκατάστασης. Κατά τη διάρκεια της εγκατάστασης, μεταδεδομένα σε ένα αρχείο OVFμπορούν να χρησιμοποιηθούν από τη υποδομή ενός χρήστη για να επικυρώσουν το συνολικό πακέτο και να αποφασίσει εάν ένα νεό εικονικό μηχάνημα θα πρέπει να εγκατασταθεί.
- **Υποστήριξη διαμόρφωσης για ένα ή πολλαπλά εικονικά μηχανήματα.**Με το OVF, οι χρήστες μπορούν να διαμορφώσουν πολύπλοκες και πολυεπίπεδες υπηρεσίες αποτελούμενες από πολλαπλά εικονικά μηχανήματα.
- **Φορητή πακετοποίηση εικονικών μηχανημάτων.**Το πρότυπο υποστηρίζει ένα εύρος από πρότυπα εικονικών σκληρών δίσκων τα οποία (vmdk, vdkκ.α), τα οποία χρησιμοποιούνται από εικονικά μηχανήματα.
- **Ανεξαρτησία από πλατφόρμες και κατασκευαστές.**Το πρότυπο OVFδεν βασίζεται στη χρήση συγκεκριμένου φυσικού εξυπηρετητή, πλατφόρμας εικονοποίησης ή λειτουργικού συστήματος.
- **Υποστήριξη πολλαπλών γλωσσών.** Το πρότυπο OVFυποστηρίζει περιγραφές σε διάφορες γλώσσες και υποστηρίζει την τοπικοποίηση των διαδραστικών διαδικασιών κατά τη διάρκεια της εγκατάστασης ενός εικονικού πόρου. Έτσι ένα συμπιεσμένο πακέτο εικονικών πόρων μπορεί να εξυπηρετήσει ευκαιρίες σε πολλαπλές αγορές ταυτόχρονα.
- **Μελλοντική επεκτασιμότητα.** Το πρότυπο OVFείναι επεκτάσιμο και είναι σχεδιασμένο για να επεκτείνεται καθώς η τεχνολογία των εικονικών πόρων εξελίσσεται ραγδαία.

Η εταιρία VMWare[37]χρησιμοποίησε αρχικά το πρότυπο VMDK. Το VMDKείναι ένα πρότυπο αρχείου, το οποίο κωδικοποιεί ένα εικονικό σκληρό δίσκο που είναι εγκατεστημένος σε ένα εικονικό μηχάνημα. Ένα αρχείο VMDKδεν περιέχει πληροφορία σχετικά σχετικά με τις προδιαγραφές του μηχανήματος, όπως ο επεξεργαστής, η RAMή το δίκτυο. Ένα εικονικό μηχάνημα μπορεί να περιέχει πολλαπλά εικονικούς δίσκους, δηλαδή VMDKαρχεία.

Αντίθετα, το πρότυπο OVFπαρέχει μία πλήρη περιγραφή του εικονικού μηχανήματος. Η περιγραφή αυτή περιέχει τη πλήρη λίστα των απαιτούμενων εικονικών σκληρών δίσκων, μαζί με τη απαιτήσεις συστήματος (CPU, RAM, networking,storage). Με αυτό το τρόπο, ο διαχειριστής συστήματος μπορεί γρήγορα να εισάγει ένα νέο εικονικό μηχάνημα σε μία διάταξη από εικονικά μηχανήματα, χωρίς να παρέμβει σχεδόν καθόλου. Επιπρόσθετα, το πρότυπο OVFείναι προτυποιημένο, φορητό και επιτρέπει στο χρήστη να εγκαθιστά ένα εικονικό μηχάνημα σε οποιοδήποτε επόπτη (hypervisor) που υποστηρίζει OVF.

# 3. Διαχείρηση εποπτείας σε υπολογιστικά νέφη

Στο κεφάλαιο αυτό παρουσιάζονται οι σχετικές τεχνολογίες και εργαλεία όσον αφορά την εποπτεία και παρακολούθηση πόρων σε περιβάλλοντα Νεφών και άλλες αρχιτεκτονικές. Η μεθοδολογία έρευνας και ανάλυσης βασίστηκε στον διαχωρισμό των υπαρχουσών τεχνολογιών σε: (i) τεχνικές εποπτείας εμπορικών Νεφών, (ii) συστήματα εποπτείας ανοιχτού κώδικα (με δυνατότητα χρήσης και επέκταση τους), (iii) καταγραφή και ανάλυση λοιπών μηχανισμών εποπτείας και σχετικών ερευνητικών δραστηριοτήτων. Επιπρόσθετα, η μελέτη των συστημάτων αυτών καταλήγει στον προσδιορισμό των προδιαγραφών και των απαιτήσεων σχεδιασμού ενός σύγχρονου μηχανισμού εποπτείας και παρακολούθησης για υπολογιστικά Νέφη και υπηρεσιοστρεφείς υποδομές. Σε αυτό το πλαίσιο, συζητάμε και οριοθετούμε τις συνθήκες λειτουργίας ενός συστήματος εποπτείας σε περιβάλλον Νέφους, παρουσιάζουμε τη ροή της πληροφορίας στα διάφορα στρώματα και καταγράφουμε τους ρόλους των διαφόρων οντοτήτων που λαμβάνουν μέρος σε αυτή την διαδικασία.

## 3.1. Εποπτεία και διαχείρηση

### 3.1.1. Προδιαγραφές

Η εγκατάσταση και η εκτέλεση εφαρμογών σε έντονα δυναμικές υποδομές, όπως τα υπολογιστικά Νέφη, εισαγάγουν ένα καινούριο σύνολο απαιτήσεων όσον αφορά την εποπτεία το οποίο είναι απαραίτητο να ληφθεί υπόψη από τους προγραμματιστές και τους παρόχους των εφαρμογών και υπηρεσιών αυτών. Έτσι λοιπόν, ένα σύστημα εποπτείας και παρακολούθησης θα πρέπει να είναι ικανό να ακολουθεί τις δυναμικές αλλαγές μιας εφαρμογής ή της υποδομής όταν αυτές αυξομειώνουν το μέγεθος και την χωρητικότητα τους. Επιπρόσθετα, καθώς η πλειονότητα των σύγχρονων βιομηχανικών εφαρμογών είναι κατά βάση ροές λειτουργίας ξεχωριστών μονάδων λογισμικού (application workflows) που εκτελούνται σε διαφορετικούς κόμβους, θα πρέπει και ο μηχανισμός παρακολούθησης να λειτουργεί στο πλαίσιο ενός κατανεμημένου συστήματος. Θα πρέπει, έτσι, να είναι προσαρμοστικός σε γρήγορες αλλαγές της εφαρμογής, ρυθμιζόμενος και παραμετροποιήσιμος όσον αφορά την συχνότητα συλλογής της πληροφορίας, την ποσότητα των δεδομένων όπως και τον τύπο των δεδομένων. Επιπλέον προσαρμοστικότα είναι απαραίτητη όταν έχουμε να κάνουμε με εφαρμογές πολλαπλών χρηστών πελατών (multitenant). Οι περιπτώσεις πολλαπλής χρήσης υπηρεσιών, εικονικών μηχανών και πόρων εισαγάγουν επιπρόσθετη πολυπλοκότητα στα περιβάλλοντα Νεφών, εγείροντας ιδιαίτερες απαιτήσεις και προδιαγραφές στα συστήματα διαχείρισης και παρακολούθησης. Από την άλλη μεριά, ένας χρήστης ενδιαφέρεται κυρίως για την αποδοτική λειτουργία της εφαρμογής του και ως εκ τούτου η πολυπλοκότητα αυτή θα πρέπει να είναι όσο το δυνατόν αόρατη. Επίσης, ο προγραμματιστής μιας εφαρμογής που θα εγκατασταθεί σε ένα Νέφος, είναι αυτός

που θα πρέπει να προσδιορίσει και τους Δείκτες Απόδοσης (Key Performance Indicators -KPIs) όπως και να συμβάλλει στην ρύθμιση και προσαρμογή του μηχανισμού παρακολούθησης σχετικά με τις παραμέτρους εποπτείας, την συχνότητα συλλογής αυτών αλλά και στις διορθωτικές κινήσεις που θα πρέπει να ληφθούν.

Εκτός από τις τεχνικές απαιτήσεις που προκύπτουν από τον ίδιο τον χρήστη ή της εφαρμογή, υπάρχουν και περιορισμοί που προβάλλονται από τις τάσεις των καινούριων τεχνολογιών και τις αρχές που αυτές ορίζουν. Όπως παρουσιάστηκε στο [42], μια ενδιαφέρουσα σύγκριση Πλεγμάτων και Νεφών, τα δεύτερα είναι περισσότερο υπηρεσιοστρεφή παρά καθοδηγούμενα από την εφαρμογή, όσον αφορά την αρχιτεκτονική προσέγγιση. Βασισμένοι σε αυτό, ένα σύστημα εποπτείας και παρακολούθησης για Νέφη θα πρέπει να ακολουθεί τις αρχές σχεδίασης των Νεφών και να εφαρμόζει έναν υπηρεσιοστρεφή σχεδιασμό. Η

ύπαρξη πολλαπλών λογικών στρωμάτων (layers), καθένα από τα οποία σχετίζεται απόλυτα με τα υπόλοιπα, δημιουργεί την ανάγκη συλλογής και συνάθροισης (aggregation) της πληροφορίας με στόχο έναν αποδοτικό μηχανισμό λήψης αποφάσεων. Επιπρόσθετα, η επαναχρησιμοποίηση της πληροφορίας είναι μέγιστης σημασίας, και ως εκ τούτου αποτελεσματικά μοντέλα δεδομένων θα πρέπει να υιοθετηθούν έτσι ώστε χρήστες και "καταναλωτές" των πληροφοριών αυτών να μπορούν να εκμεταλλευτούν στο έπακρο τα δεδομένα και να προβούν σε περαιτέρω ανάλυση και επεξεργασία. [43][44]

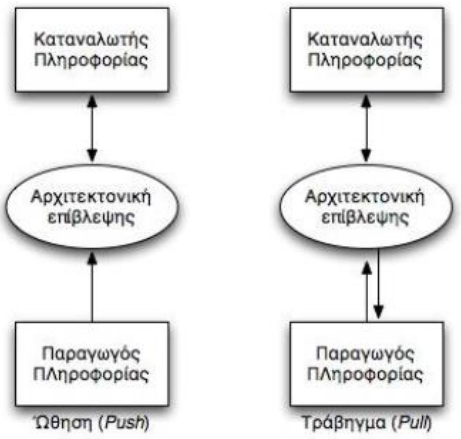
Μια ακόμη παράμετρος που θα πρέπει να ληφθεί υπόψη είναι η απόδοση του μηχανισμού εποπτείας όταν αυτός λειτουργεί σε περιορισμένο χρονικά περιβάλλον (όλο και περισσότερες εφαρμογές πολυμέσων εγκαθίστανται σε εικονικές υποδομές εκμεταλλευόμενες τις δυνατότητες του υπολογιστικού Νέφους, προβάλλοντας χρονικούς περιορισμούς λειτουργίας [29]). Τέλος, είναι πολύ σημαντικό ο μηχανισμός αυτός να είναι όσον το δυνατό μη-παρεμβατικός. Αυτό σημαίνει ότι θα πρέπει να επηρεάζει ελάχιστα (ή και καθόλου) το σύστημα το οποίο παρακολουθεί, αφήνοντας ένα αμελητέο αποτύπωμα λειτουργίας στο περιβάλλον εκτέλεσης [30]. Φυσικά, είναι αδύνατο για οποιοδήποτε σύστημα εποπτείας να επιτύχει μηδενική κατανάλωση υπολογιστικών πόρων (φαινόμενο παρατηρητή: η πιθανή επίδραση στο σύστημα όταν παρατηρούμε το αποτέλεσμα μιας διαδικασίας καθόσον η διαδικασία εκτελείται) αλλά ο στόχος είναι να ελαχιστοποιήσουμε την κατανάλωση αυτή σε σχέση με τους πόρους που χρειάζεται η ίδια εγκατεστημένη εφαρμογή που επιβλέπουμε.

### 3.1.2. Ροή πληροφορίας και ρόλοι

Κάθε υπολογιστικό (και όχι μόνο) σύστημα παράγει πληροφορίες οι οποίες πρέπει να συλλεχθούν, αποθηκευθούν και αξιολογηθούν έτσι ώστε να ανιχνευθούν τυχών σφάλματα λειτουργίας, να βελτιωθεί η λειτουργία της υποδομής αλλά και για να παρακολουθηθεί η χρήση μιας προσφερόμενης υπηρεσίας. Σε αυτό το μοντέλο λειτουργίας μπορούν να εντοπιστούν κάποιες διακριτές οντότητες με ξεχωριστό ρόλο στην διαδικασία αυτή. Οι δύο κύριες οντότητες είναι ο παραγωγός πληροφοριών και ο καταναλωτής πληροφοριών. Ο πρώτος μπορεί να είναι η φυσική υποδομή, η εικονική υποδομή, μια εφαρμογή ή υπηρεσία. Οι πληροφορίες που παράγονται είναι δεδομένα που καταγράφοντα από αυτούς του πόρους και αφορούν την χρήση/κατανάλωση πόρων σε χαμηλό επίπεδο (π.χ. της ενέργειας, της υπολογιστικής ισχύος, του αποθηκευτικού χώρου) αλλά και παραμέτρους υψηλού επιπέδου όπως η απόδοση της εφαρμογής ή ο χρόνος ανταπόκρισης της υπηρεσίας. Ως καταναλωτής πληροφορίας μπορεί να είναι οποιοσδήποτε χρειάζεται πρόσβαση στα συλλεχθέντα δεδομένα.

Τον ρόλο αυτό μπορεί να έχει ο ίδιος ο τελικός χρήστης ή κάποιο συστατικό του συστήματος/πλατφόρμας που χρησιμοποιεί και επεξεργάζεται τα δεδομένα με σκοπό την αναγνώριση αλλά και αποφυγή σφαλμάτων στην λειτουργία ή την βελτιστοποίηση της χρήσης της υποδομής.

Όσον αφορά την εποπτεία υπολογιστικών συστημάτων με στόχο την συλλογή δεδομένων υπάρχουν δύο βασικές τεχνικές: ώθηση της πληροφορίας (push) και εξαγωγή της πληροφορίας (pull). Το κάθε πρότυπο έχει τα δικά του χαρακτηριστικά που του δίνουν συγκεκριμένα πλεονεκτήματα ή μειονεκτήματα κατά περίπτωση λειτουργίας.




Εικόνα 5: Τεχνικές ώθησης και τραβήγματος πληροφορίας

Στο πρότυπο ώθησης, αυτός που ενεργοποιεί την διαδικασία είναι ο παραγωγός. Ο παραγωγός στέλνει τις πληροφορίες κατάστασης είτε όταν ανιχνεύει αλλαγές μεγαλύτερες από ένα κατώτατο όριο, είτε σεπροκαθορισμένες χρονικές στιγμές. Είναι ιδανικό για την κράτηση μιας καλής ισορροπίας στην επικοινωνία μεταξύ του παραγωγού και του καταναλωτή εάν το κατώτατο όριο είναι κατάλληλα ορισμένο. Εντούτοις, εάν το κατώτατο όριο είναι μικρό, οι ελάχιστες αλλαγές οδηγούν σε πάρα πολύ συχνή μετάδοση πληροφοριών, η οποία μπορεί να επιβαρύνει το δίκτυο. Από την άλλη, όπως φαίνεται και στο Σχήμα 21 το πρότυπο ώθησης λειτουργεί με μονομερή επικοινωνία προς τον καταναλωτή κάτι το οποίο μειώνει εν γένει την χρησιμοποίηση του δικτύου. Επιπρόσθετα, η τοποθέτηση του ενεργού σκέλους τις εποπτείας στην πλευρά του παραγωγού έχει ένα διπλό αντίκτυπο: δενεπιβαρύνεται υπολογιστικά και λειτουργικά η μεριά του καταναλωτή, αλλά από την άλλη ηεποπτεία της λειτουργίας εποπτείας είναι κατανεμημένος στους διαφόρους παραγωγούς. Ως εκ τούτου, μια εσφαλμένη λειτουργία του μηχανισμού συλλογής και αποστολής δεδομένων θα μπορούσε να μην αναγνωριστεί. Τελικά, μπορούμε νασχολιάσουμε, ότι το συγκεκριμένο πρότυπο είναι κατάλληλο σε συστήματα εποπτείας στα οποία ο όγκοςδεδομένων και η συχνότητα παρακολούθησης δεν θα επιβαρύνουν σημαντικά την δικτυακή υποδομή και επίσης μια εσφαλμένη λειτουργία του μηχανισμού εποπτείας δεν θα επιφέρει κρίσιμες επιπτώσεις στην λειτουργία του συστήματος συνολικά.

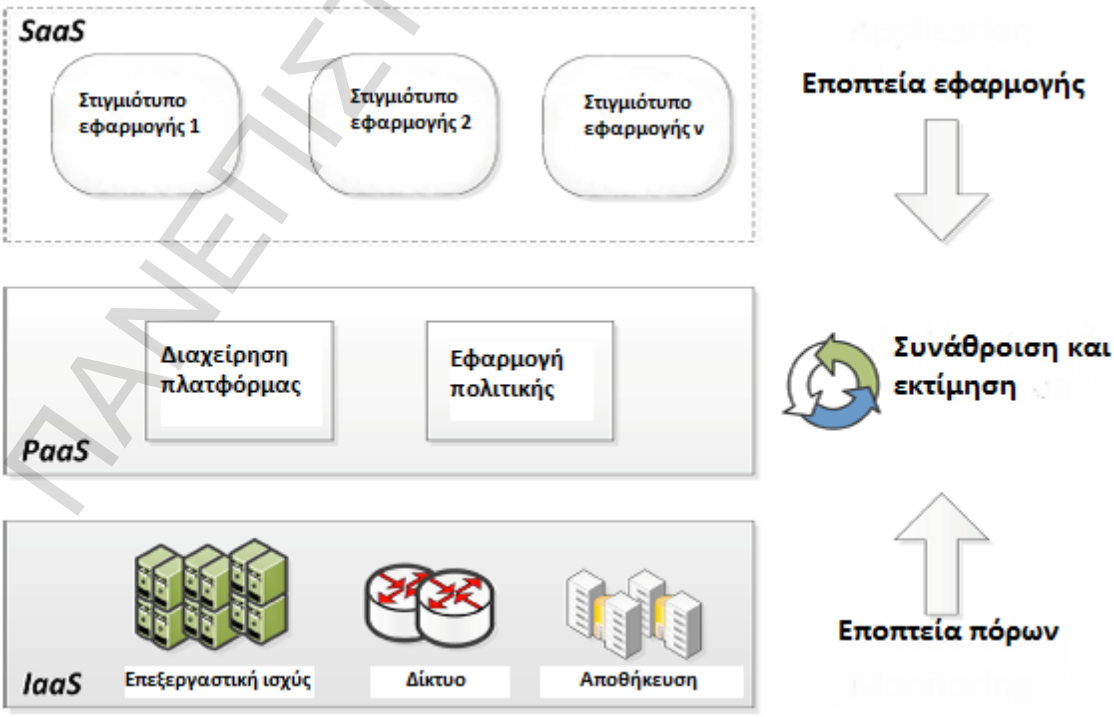
Στο πρότυπο τραβήγματος, ο καταναλωτής είναι αυτός που εκκινεί την διαδικασία μέσω μιας αίτησης. Υπό αυτήν τη μορφή, το χαμηλό ποσοστό τραβήγματος καταναλώνει λίγο εύρος ζώνης δικτύων, αλλά μπορεί να υπονοήσει απώλεια ενημέρωσης, το οποίο είναι ανεπιθύμητο για τους καταναλωτές. Εντούτοις, οι πληροφορίες στο υψηλό ποσοστό τραβήγματος είναι πιο «φρέσκιες» αλλά η όλη διαδικασία είναι βαριά παρεισφρητική στο αρχικό σύστημα. Η λειτουργία αυτού του προτύπου απαιτεί την ύπαρξη συστατικών του μηχανισμού εποπτείας και στις δύο μεριές (καταναλωτής, παραγωγός), κάτι που επίσης επιβαρύνει την όλη εγκατάσταση και λειτουργία. Από την άλλη, η αρχιτεκτονική αυτού του συστήματος εποπτείας δίνει τον έλεγχο στον καταναλωτή κάτι το οποίο κατά περίπτωση μπορεί να βελτιώσει την αξιοπιστία του συστήματος, τον εντοπισμό σφαλμάτων αλλά και με σωστή διαμόρφωση να περιορίσει (σε λογικά πλαίσια) την κατανάλωση του εύρους ζώνης όταν δεν υπάρχει λόγος λειτουργίας του συστήματος εποπτείας (π.χ. κατά τη διάρκεια της νύχτας).

Έχουν προταθεί και συνδυαστικές μέθοδοι (P&P)[45], στις οποίες εφαρμόζονται και οι δύο τεχνικές εποπτείας στο ίδιο σύστημα βάσει συγκεκριμένων συνθηκών λειτουργίας. Στην διατριβή αυτή, χρησιμοποιήσαμε και τα δύο πρότυπα συλλογής πληροφοριών στα συστήματα που υλοποιήσαμε, κάθε φορά λαμβάνοντας υπόψη τα χαρακτηριστικά των δεδομένων που συλλέγαμε και την λειτουργικότητα του συστήματος που επιβλέπαμε.

### 3.1.3. Εποπτεία σε περιβάλλον Νέφους

Στα μοντέρνα περιβάλλοντα Νεφών και πλατφορμών υπηρεσιών που συμπεριέχουν ένα στρώμα εικονικοποίησης, ένα σύνολο Μονάδων Εικονικών Μηχανών (*Virtual Machine Units* - VMUs) που φιλοξενεί διαφορετικές υπηρεσίες, αρχικοποιείται με σκοπό να καλύψει τις ανάγκες μιας συγκεκριμένης εφαρμογής. Σε αυτή τη βάση, μια μονολιθική εφαρμογή μπορεί να διαχωριστεί σε πολλαπλά Συστατικά Εφαρμογής (*Application Components*) καθένα από τα οποία αναπαριστά μια διακριτή λειτουργία μέσα στη ροή εργασίας της εφαρμογής. Για παράδειγμα, σε μια εφαρμογή ψηφιακής επεξεργασίας βίντεο η Μονάδα Επεξεργασίας Εικόνας (*Image Processing Unit*) και ο Ισορροπιστής Φορτίου (*Load Balancer*) είναι διαφορετικά συστατικά εφαρμογής της ίδιας ροής λειτουργίας (*workflow*). Για να επιτύχουμε υψηλό επίπεδο διαθεσιμότητας και λειτουργία στην συμφωνηθείσα ποιότητα υπηρεσίας, μια συνεχόμενη ροή δεδομένων παρακολούθησης απαιτείται από όλα τα στρώματα, όπως παρουσιάζεται στην Εικόνα 6.

Εξ ‘αιτίας της πληθώρας δεδομένων εποπτείας, το σχέδιο του μηχανισμού αυτού θα πρέπει να ακολουθεί μια ιεραρχική προσέγγιση βασισμένη στα τρία στρώματα του μοντέλου Νέφους. Στο SaaS επίπεδο ο μηχανισμός παρακολούθησης θα συλλέγει δεδομένα σχετικά με την εφαρμογή σε υψηλό επίπεδο (KPIs κ.α.), τα οποία είναι προσωποποιημένα για κάθε εφαρμογή τόσο στο περιεχόμενο όσο και στην τεχνική συλλογής τους. Φυσικά, τα δεδομένα που συλλέγονται από μια εφαρμογή δεν είναι αρκετά από μόνα τους για να κατανοήσουμε την λανθασμένη συμπεριφορά και λειτουργία μιας εφαρμογής, ούτε να εντοπίσουμε αποτυχίες, να τις απομονώσουμε και να επανέλθουμε σε φυσιολογική λειτουργία. Δεδομένα εποπτείας χαμηλού επιπέδου, σχετικά με τον επεξεργαστή, το δίκτυο, τον χώρο αποθήκευσης στο στρώμα IaaS του Νέφους, παρέχουν πληροφορίες για την κατανάλωση πόρων κάθε συστατικού της εφαρμογής που είναι εγκατεστημένο σε ένα εικονικό περιβάλλον. Ο συνδυασμός και συνάθροιση αυτών των δύο πηγών πληροφορίας στο στρώμα PaaS του Νέφους, μας επιτρέπει να έχουμε ένα αποδοτική και ολιστική διαχείριση όπως επίσης και άμεση εφαρμογή πολιτικών ανάκτησης για την διασφάλιση του απαιτούμενου επιπέδου ποιότητας.




Εικόνα 6: Ροή πληροφορίας εποπτείας

## 3.2. Υπάρχουσες τεχνολογίες & εργαλεία

### 3.2.1. Εποπτεία εμπορικών Νεφών

Όπως παρουσιάστηκε σε προηγούμενο κεφάλαιο, οι υπηρεσίες Νεφών της Amazon (AWS) είναι πιθανόν οι πιο διαδεδομένες και ευρέως χρησιμοποιούμενες. Οι βασικές υπηρεσίες που παρέχονται συμπεριλαμβάνουν υπολογιστική επεξεργασία (Elastic Compute Cloud - EC2) και αποθήκευση δεδομένων (Elastic Block Storage – EBS [25]) και κάποιος μπορεί να επιβλέψει και παρακολουθήσει την χρήση χρησιμοποιώντας την παρεχόμενη επίσης υπηρεσία της Amazon, CloudWatch [26]. Η υπηρεσία παρακολούθησης CloudWatch μπορεί να χρησιμοποιηθεί είτε μέσω μιας διαδικτυακής διεπαφής είτε προγραμματιστικά μέσω του διαθέσιμου API. Η βασική αρχή λειτουργίας είναι ότι οι επιβλέπουσες υπηρεσίες στέλνουν τις διάφορες μετρικές και παραμέτρους στο CloudWatch. Τα δεδομένα που παρέχει το σύστημα EBS αποτελούνται από τον αριθμό των bytes για ανάγνωση/εγγραφή, τον αριθμό των αιτημάτων ανάγνωσης/εγγραφής, τον συνολικό χρόνο που απαιτείται για την εκτέλεση αιτημάτων ανάγνωσης/εγγραφής, τον ανενεργό χρόνο της αποθήκευσης και τον αριθμό των αιτημάτων που εκκρεμούν. Τα δεδομένα που παρέχονται από το EC2 αποτελούνται από την κατανάλωση CPU, τον αριθμό των bytes που διαβάζονται/γράφονται στο δίσκο, τον αριθμό των λειτουργιών ανάγνωσης/εγγραφής και τον αριθμό των *bytes* που ελήφθησαν/απεστάλησαν στο δίκτυο. Τα δεδομένα αυτά αποστέλλονται από τις υπηρεσίες κάθε πέντε λεπτά στην περίπτωση του EBS και του EC2, ενώ στο τελευταίο η συχνότητα μπορεί να ρυθμιστεί και στο ένα λεπτό προσθέτοντας όμως επιπλέων κόστος.

Μερικές από τις υπόλοιπες υπηρεσίες που η Amazon προσφέρει, παρέχουν τις δικές τους συγκεκριμένες μετρικές και παραμέτρους. Παραδείγματατέτοιωνυπηρεσιώνείναιοι: Relational Database Service (RDS)[46], Auto Scaling Service και Elastic Load Balancing Service. Επιπρόσθετα, η Amazon έχει πρόσφατα εκδώσει ένα API το οποίο επιτρέπει σε εφαρμογές χρηστών να αποστέλλουν αυθαίρετες μετρικές στο CloudWatch. Αυτές οι παράμετροι αντιμετωπίζονται από το σύστημα όπως και οι υπόλοιπες που το CloudWatch παρακολουθεί και έτσι μπορεί κάποιος να αποκτήσει πρόσβαση σε αυτές χρησιμοποιώντας τις διεπαφές που περιγράφθηκαν προηγουμένως.

Ο μεγαλύτερος ανταγωνιστής του προϊόντος AWS είναι το Windows Azure, το οποίο παρέχει τους δικούς του μηχανισμούς εποπτείας και παρακολούθησης. Αυτοί συμπεριλαμβάνουν τον Azure Diagnostic Manager[47], που παρέχει μια κονσόλα διαχείρισης για υποδομές και εφαρμογές βασισμένες σε Azure και προορίζεται κυρίως για διαχειριστές και προγραμματιστές εφαρμογών. Ένα ακόμα παράδειγμα είναι ο Subscription Manager[48] για το Azure το οποίο επιτρέπει σε χρήστες και διαχειριστές Νεφών να αξιολογήσουν την χρησιμοποίηση των Azure πόρων τους.

Μια λύση εποπτείας εμπορικού Νέφους ανεξάρτητη από τον πάροχο είναι το CloudStatus[49]. Βασισμένο στο Hyperic HQ, ένα εργαλείο παρακολούθησης και διαχείρισης για διαδικτυακές υποδομές, παρέχει τα μέσα για την εποπτεία της απόδοσης και κατάστασης μιας πληθώρας υπηρεσιών Νεφών συμπεριλαμβανομένων AWS και Google App Engine υπηρεσίες. Μέχρι σήμερα, μόνο μια περιορισμένη λίστα υπηρεσιών είναι διαθέσιμη αλλά όλο και περισσότερες θα προστεθούν στο μέλλον.

### 3.2.2. Μηχανισμοί εποπτείας ανοιχτού κώδικα

Σε αυτό το υποκεφάλαιο παρουσιάζονται και αναλύονται επιλεγμένα εργαλεία και APIs που χρησιμοποιούνται ευρέως για την εποπτεία πόρων σε διάφορες υπολογιστικές υποδομές. Τα περισσότερα από αυτά έχουν σχεδιαστεί να εξυπηρετούν συστάδες υπολογιστών (*cluster*) και υποδομές Grid αλλά με μικρές μετατροπές και ρυθμίσεις μπορούν να χρησιμοποιηθούν και ως η βάση ενός συστήματος παρακολούθησης και εποπτείας υπολογιστικών Νεφών. Τέλος, όλοι οι μηχανισμοί που παρουσιάζονται είναι συστήματα ανοιχτού κώδικα (*open source*) και διατίθενται δωρεάν στο διαδίκτυο.

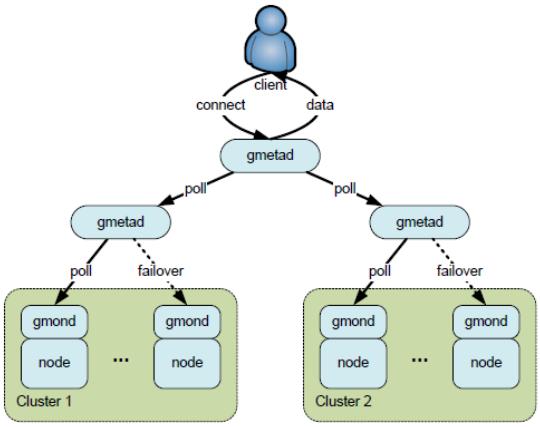
***Ganglia***

Το Ganglia[50] είναι ένας μηχανισμός εποπτείας πόρων που σχεδιάστηκε να εξυπηρετεί υπολογιστικά συστήματα υψηλής απόδοσης όπως συστάδες υπολογιστών και περιβάλλοντα πλέγματος. Είναι βασισμένο σε μία ιεραρχική αρχιτεκτονική η οποία παρέχει υψηλή επεκτασιμότητα και απόδοση, με δυνατότητα να υποστηρίζει μέχρι και 2000 υπολογιστικούς κόμβους. Η διαχείριση της πληροφορίας γίνεται μέσο XML, για την παροχή δεδομένων, και με XDR[51], για την μεταφορά δεδομένων, ενώ τα αποτελέσματα της εποπτείας αποθηκεύονται με μορφή γράφου μέσο ενός RRD[52] εργαλείου. Υποστηρίζεται εικονικοποίηση των δεδομένων απόδοσης, όπως επίσης πραγματοποίηση διαγνωστικών ελέγχων και διαμόρφωση ιστορικής τάσης. Το Ganglia έχει μόνο επίγνωση των παραμέτρων και μετρικών διαθέσιμων από τα εσωτερικά του στοιχεία παρακολούθησης, χωρίς να είναι δυνατόν να υποστηρίξει ενεργά την εποπτεία υπηρεσιών εξωτερικά του κόμβου. Ένα ακόμα αξιοσημείωτο χαρακτηριστικό είναι το μικρό υπολογιστικό κόστος λειτουργίας που εισαγάγει στο περιβάλλον εκτέλεσης.

Οι μετρικές που υποστηρίζονται από το σύστημα αυτό χωρίζονται σε δύο βασικές κατηγορίες: εσωτερικά ορισμένες (*build-in*) παράμετροι και παράμετροι ορισμένες από τον χρήστη. Μέσο των πρώτων είναι δυνατόν η συλλογή πληροφοριών από τους υπολογιστικούς κόμβους, ενώ οι δεύτερες αντιπροσωπεύουν καταστάσεις συγκεκριμένων εφαρμογών, επιτρέποντας έτσι την προσαρμογή και επεκτασιμότητα των δεδομένων εποπτείας. Κάθε εσωτερική (*build-in*) παράμετρος έχει ένα προκαθορισμένος όριο τιμής ως την βάση αναφοράς έτσι ώστε να αποφασιστεί εάν και εφόσον τα δεδομένα ενός τοπικού κόμβου θα συλλεχθούν και αποσταλούν στο δίκτυο μέσω της επικοινωνίας *multicast*. Αυτή η τιμή βάσης μπορεί να προσδιοριστεί κατά τη διάρκεια εγκατάστασης για να εξυπηρετεί διαφορετικά περιβάλλοντα. Το Ganglia διατίθεται υπό την άδεια λογισμικού BSD, ως ένα πρόγραμμα ελεύθερου λογισμικού και αναπτύχθηκε από το πανεπιστήμιο της Καλιφόρνιας, Μπέρκλεϋ.

Στην Εικόνα 7 παρουσιάζεται η ιεραρχική δενδροειδής αρχιτεκτονική του Ganglia, η οποία αποτελείται από τέσσερα δομικά συστατικά:

- Συλλέκτης δεδομένων gmond
- Συναθροιστής (aggregator) δεδομένων gmetad
- Εργαλείο RRD (round-robin database tool)
- Διαδικτυακή διεπαφή σε PHP




Εικόνα 7: Αρχιτεκτονική Ganglia

Σε μια συστάδα υπολογιστών, το *gmond* συλλέγει τις παραμέτρους εποπτείας για κάθε κόμβο και τις προωθεί στον συναθροιστή *gmetad.* Ο *Gmetad* συνδυάζει τις μετρικές ανά κόμβο και συντάσσει μια αναφορά σε XML μέσω διαδοχικών και περιοδικών ερωτημάτων σε κάθε κόμβο. Με αντίστοιχο τρόπο, σε ένα περιβάλλον Grid, η αναγνώριση της κατάστασης ενός κόμβου επιτυγχάνεται από το *gmetad* συνδυάζοντας τις καταστάσεις των συστάδων υπολογιστών του Grid μέσω ερωτημάτων. Τα δεδομένα απόδοσης που συλλέγονται, αποθηκεύονται σε μια RDD. Αυτά τα δεδομένα μπορούν κατόπιν να παρουσιαστούν μέσω ενός κώδικα PHP και να εικονικοποιηθούν χρησιμοποιώντας το γραφικό περιβάλλον που παρέχεται για τν παραγωγή γράφων δυναμικά.

***Nagios***

Το Nagios[53]είναι ένα ακόμα σύστημα εποπτείας πόρων σε υποδομές που διατίθεται δωρεάν. Λειτουργεί βασισμένο σε ελέγχους κατάστασης (*status checks*) και παρέχει επίσης μηχανισμό αυτόματης ειδοποίησης για να ενημερώνει για τυχόν προβλήματα. Ένα κύριο χαρακτηριστικό του Nagios είναι ότι ο σχεδιασμός του βασίζεται και λειτουργεί με προσθήκες λογισμικού (*plugins*) για να επιτύχει τις διάφορες διαδικασίες εποπτείας και παρακολούθησης της κατάστασης. Αυτή η δομή επιτρέπει την επεκτασιμότητα του μηχανισμού και τον δυναμικό χαρακτήρα της λειτουργίας.

Από μια λειτουργική σκοπιά, η αρχιτεκτονική του Nagios χωρίζεται σε δύο επίπεδα: το στρώμα λογικής της εποπτείας (*monitoring logic*) και το στρώμα λειτουργίας (*operation*). Τα δομικά στοιχεία της λειτουργίας του Nagios βρίσκονται στο πρώτο ενώ τα plugins στο δεύτερο.

Το ίδιο το Nagios δεν έχει εσωτερικό μηχανισμό για να πραγματοποιεί την εποπτεία πόρων, αλλά είναι τα plugins που υλοποιούν την συλλογή των πληροφοριών. Κάθε *plugin* είναι υπεύθυνο για ένα συγκεκριμένο έλεγχο κατάστασης και αποστέλλει κατόπιν τα αποτελέσματα στο στρώμα λογικής εποπτείας. Με αυτόν τον τρόπο, εάν υπάρχουν διαθέσιμα *plugins*, το Nagios είναι δυνατόν να επιβλέπει πόρους κάθε τύπου. Το βασικό API που παρέχεται, συμπεριλαμβάνει ορισμένα plugins αλλά κάποιος μπορεί να αναπτύξει και τα δικά του σύμφωνα με τις οδηγίες που είναι διαθέσιμες και μέσω του API[54].

Σχετικά με την αξιολόγηση των δεδομένων και την ειδοποίηση, ο μηχανισμός του Nagios συγκρίνει τις τιμές, που συλλέγονται από τα plugins, με τα αντίστοιχα όρια τιμών που έχουν ορισθεί εξαρχής. Βάσει των αποτελεσμάτων (*alert level OK, Warning, Critical or Unknown*) λαμβάνει τις απαραίτητες κινήσεις, όπως εκκίνηση μιας διαδικασίας ή τηναποστολή μιας ειδοποίησης, έχοντας πάντα ως σκοπό την διατήρησης της ομαλής λειτουργίας της υπολογιστικής υποδομής. Η ειδοποίηση μπορεί να αποσταλεί με διάφορους τρόπους όπως email, SMS, έτσι ώστε να ενημερωθεί το τεχνικό προσωπικό για το πιθανό πρόβλημα.

Το Nagios διατίθεται με την άδεια χρήσης ανοιχτού κώδικα GPL αλλά βασισμένα σε αυτήν την έκδοση υπάρχουν εμπορικές εκδοχές όπως το Nagios XI[55], που παρέχουν επιπρόσθετες δυνατότητες συμπεριλαμβανομένων ισχυρού διαδικτυακού περιβάλλοντος διαχείρισης και διαχείριση παραμέτρων για εξειδικευμένους χρήστες.

Παραθέτοντας λίγα λόγια ακόμα για την αρχιτεκτονική, το Nagios έχει υλοποιηθεί με το μοντέλο πελάτη/εξυπηρετητή (*client/server*) για να λειτουργεί και επιβλέπει κατανεμημένα συστήματα. Αποτελείται από τρία βασικά δομικά συστατικά:

- Nagios daemon
- Web-based GUI
- Plug-ins

Σε ένα κατανεμημένο περιβάλλον, ο Nagios *deamon* εκτελείται στην πλευρά του εξυπηρετητή, ο οποίος είναι και υπεύθυνος για την εκκίνηση της εποπτείας και παίρνει τις απαραίτητες αποφάσεις αξιολογώντας τα αποτελέσματα που συλλέχθηκαν. Τα *plugins* του Nagios εκτελούνται σε όλους τους απομακρυσμένους κόμβους που πρέπει να επιβλεφθούν. Έτσι, επικοινωνούν με τα στοιχεία του κάθε κόμβου και επιστρέφουν τις τιμές των παραμέτρων στον Nagios *deamon*.

***Hyperic HQ Open Source***

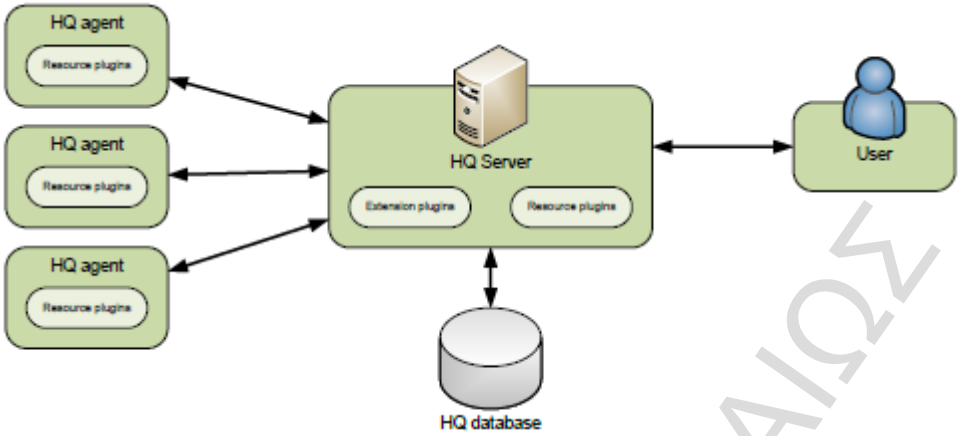
Το Hyperic HQ[56] είναι ένα λογισμικό εποπτείας εφαρμογών, σχεδιασμένο να διαχειρίζεται και επιβλέπει διαδικτυακές εφαρμογές και υποδομές σε ετερογενήπεριβάλλοντα. Είναι βασισμένο σε μια αρχιτεκτονική πράκτορα/εξυπηρετητή, με τον πράκτορα να πραγματοποιεί τις λειτουργίες παρακολούθησης και τον κεντρικό εξυπηρετητή να χρησιμοποιείται για την επεξεργασία των αποτελεσμάτων. Σε αντίθεση με πολλές λύσεις εποπτείας, οι οποίες δεν μπορούν εύκολα να παρακολουθήσουν εφαρμογές, το Hyperic HQ υποστηρίζει λεπτομερή εποπτεία όχι μόνο του λειτουργικού συστήματος αλλά και διαφόρων εκδόσεων διαδικτυακών εξυπηρετητών, σχεσιακών βάσεων δεδομένων, εξυπηρετητών εφαρμογών, εξυπηρετητών ηλεκτρονικού ταχυδρομείου κ.α. Μέσω του επεκτάσιμου πλαισίου λειτουργίας του, επιπρόσθετες μετρικές μπορούν να εύκολα να προστεθούν.

Οι δυνατότητες διαχείρισης του HQ βασίζονται στο μοντέλο απογραφής[57] το οποίο προσδιορίζει τις σχέσεις μεταξύ τον πόρων που διαχειρίζονται. Το HQ αυτόματα εντοπίζει το λογισμικό που εκτελείται σε ένα μηχάνημα και αποθηκεύει τις πληροφορίες στην βάση δεδομένων HQ σύμφωνα με το ιεραρχικό μοντέλο απογραφής. Επιπρόσθετα, το HQ υποστηρίζει αυτόματη ανακάλυψη και απομακρυσμένο έλεγχο πόρων, τα οποία διευκολύνουν τις διαχειριστικές διαδικασίες.

Υπάρχουν τρείς τύποι *plugins* πόρων, οι οποίοι διαμορφώνουν τις δυνατότητες εποπτείας: εσωτερικά *plugins* πόρων, *plugins* παρεχόμενα από την κοινότητα και *plugins* πόρων που αναπτύσσονται από τους πελάτες (γνωστά και ως *plugins* επέκτασης). Είναι δυνατόν λοιπόν κάποιος να επεκτείνει όχι μόνα τα εσωτερικά plugins αλλά και τα plugins πόρων που διατίθενται από την Hyperic κοινότητα[58]. Οι πράκτορες εξαρτώνται από τα plugins για να συλλέξουν πληροφορίες για την απόδοση, να δημιουργήσουν ειδοποιήσεις και να αναφέρουν στοιχεία αποδοτικότητας.

Το Hyperic HQ παρέχει επίσης ένα δομικό συστατικό ως πύλη (*portal*) διαχείρισης. Από το HQ Portal γίνεται παρουσίαση των δεδομένων εποπτείας μέσω ενός γραφικού περιβάλλοντος το οποίο μπορεί να προσαρμοστεί έτσι ώστε να παρέχει συγκεκριμένες αναφορές για την εποπτεία, την απόδοση και τα διαθέσιμα δεδομένα. Οι διαφορές της εμπορικής έκδοσης του Hyperic HQ σε σχέση με αυτήν που διατίθεται ελεύθερα είναι η πρόσθετες λειτουργίες όπως αυτοματισμοί εγκατάστασης, επιλογές ειδοποίησης για προχωρημένους, προγραμματισμός λειτουργιών υπηρεσιών (π.χ. επανεκκίνηση κ.α.).

Η αρχιτεκτονική του Hyperic HQ βασίζεται όπως αναφέραμε στην δομή πράκτορα/εξυπηρετητή (*agent/server*), κατά την οποία οι διαδικασίες εκτελούνται από τους πράκτορες οι οποίοι αναφέρουν πίσω στον κύριο εξυπηρετητή εποπτείας (Εικόνα 8). Ένας πράκτορας εκτελείται σε κάθε κόμβο που παρακολουθείται και εφαρμόζει διάφορα *plugins* έτσι ώστε να επιβλέψει κάθε πόρο όπως και τις εφαρμογές που εκτελούνται σε αυτόν. Ο εξυπηρετητής αποθηκεύει τα δεδομένα, που αναφέρθηκαν από τους πράκτορες, σε μια βάση δεδομένων και παρέχει ένα φιλικό προς τον χρήστη περιβάλλον παρουσίασης μέσω μια διαδικτυακής πύλης (*portal*).




Εικόνα 8: Αρχιτεκτονική Hyperic HQ

***Lattice monitoring Framework***

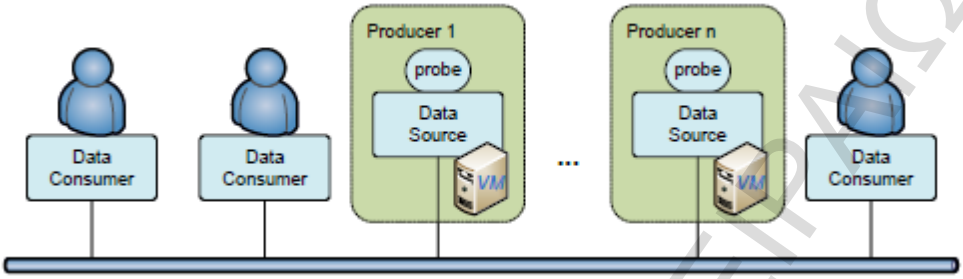
Το πλαίσιο εποπτείας Lattice[45] είναι ένα API ανοιχτού κώδικα για την ανάπτυξη συστημάτων εποπτείας και παρακολούθησης για φυσικά ή εικονικά περιβάλλοντα, εικονικά δίκτυα και υπηρεσίες. Μέσω του βασισμένο στην Java API που παρέχεται, κάποιος μπορεί να αναπτύξει ένα σύστημα διαχείρισης για την εποπτεία πόρων και υπηρεσιών όπως παρουσιάζεται στο [60]. Λειτουργώντας σαν εργαλειοθήκη (*toolbox*), το Lattice παρέχει διάφορα δομικά συστατικά και λειτουργίες για την δημιουργία ενός συστήματος σχεδιασμένο για τις ανάγκες του κάθε πελάτη/χρήστη. Υπό αυτήν την έννοια, το Lattice δεν είναι μια ολοκληρωμένη, έτοιμη προς χρήση λύση, αλλά ένα API που παρέχει όλες τις απαραίτητες βιβλιοθήκες για την υλοποίηση ενός προσωποποιημένου συστήματος παρακολούθησης. Στην πράξη το Lattice έχει χρησιμοποιηθεί επιτυχώς στο ευρωπαϊκό ερευνητικό πρόγραμμα AutoI[61] για την εποπτεία εικονικών δικτύων, όπως επίσης και στο RESERVOIR Project όπου χρησιμοποιήθηκε για την ανάπτυξη ενός ολοκληρωμένου συστήματος παρακολούθησης Νέφους υπηρεσιών.

Παρόλο που το Lattice Framework παρέχει δυνατότητες για παρακολούθηση φυσικών πόρων (CPU, σκληρούς δίσκους κ.α.), χρησιμοποιείται κυρίως για την συλλογή μετρήσεων σε εικονικές μηχανές και παροχή των αποτελεσμάτων στο επίπεδο διαχείρισης. Οι λειτουργίες του επικεντρώνονται στην συλλογή και διανομή των δεδομένων εποπτείας είτε μέσω multicast πρωτοκόλλου είτε μέσω UDP. Δεν διαθέτει έτσι δυνατότητα εικονικοποίησης, αξιολόγησης και αυτόματης ειδοποίησης. Η παρακολούθηση των πόρων στο εικονικό περιβάλλον πραγματοποιείται με αλληλεπίδραση με API της πλατφόρμας διαχείρισης εικονικών μηχανών (*VM hypervisor*), Libvirt API.

Όπως παρουσιάζεται στην Εικόνα 9, ο σχεδιασμός του Lattice βασίζεται στην αρχή ύπαρξης παραγωγών (*producers*) και καταναλωτών (*consumers*). Καθαυτή την έννοια το πλαίσιο αποτελείται από τα παρακάτω στοιχεία:

- Καταναλωτές (Consumers)
- Παραγωγοί (Producers)
- Πηγές Δεδομένων (Data Sources)
- Συλλέκτες (Probes)
- Πλαίσιο διανομής (Distribution Framework)

Στο σύστημα οι Παραγωγοί, Πηγές Δεδομένων και οι Συλλέκτες, συγκεντρώνουν τα δεδομένα και οι Καταναλωτές τα διαβάζουν και τα χρησιμοποιούν. Η επικοινωνία μεταξύ Παραγωγών και Καταναλωτών επιτυγχάνεται με την υποστήριξη του Πλαισίου Διανομής το οποίο μπορεί να διανέμει τις μετρήσεις στο σύστημα εποπτείας. Στην πλευρά της συλλογής δεδομένων βρίσκονται και οι Συλλέκτες οι οποίοι προσδιορίζουν τις παραμέτρους παρακολούθησης και συλλέγουν τα αποτελέσματα σε μια Πηγή Δεδομένων, η οποία συμπεριφέρεται ως ένα σημείο ελέγχου για το «δοχείο» (*container*) ενός ή περισσοτέρων Συλλεκτών.




Εικόνα 9: Λειτουργία Lattice

***Zenoss***

Το Zenoss Core[62] είναι ένα λογισμικό ελεύθερου κώδικα για τη διαχείριση και εποπτεία, βασισμένο στον εξυπηρετητή εφαρμογών Zope[63]. Για να οργανώσει και διαχειριστεί τους πόρους σε μεγάλα περιβάλλοντα αποδοτικά, το Zenoss χρησιμοποιεί μια προσέγγιση που ονομάζεται ZenModel για να ορίσει τους πόρους και τις μεταξύ τους σχέσεις. Βάσει του ZenModel, το Zenoss ενεργοποιεί αυτόματη εποπτεία και ειδοποίηση αφού πρώτα έχει εντοπίσει, επίσης αυτόματα, τις συσκευές. Για να διαχειριστεί πολύπλοκα δεδομένα εποπτείας αποδοτικά, το Zenoss χρησιμοποιεί τρείς ξεχωριστές βάσεις δεδομένων για την αποθήκευση τις σχετικής πληροφορίας: αποθηκεύει τα δεδομένα απόδοσης σε RRD αρχεία, τα δεδομένα συμβάντων σε MySQL βάση δεδομένων και χρησιμοποιεί μια αντικειμενοστραφή βάση δεδομένων (που αναπτύσσεται μέσα στο Zope εξυπηρετητή εφαρμογής) για την αποθήκευση δεδομένα ρυθμίσεων σχετικά με τις συσκευές.

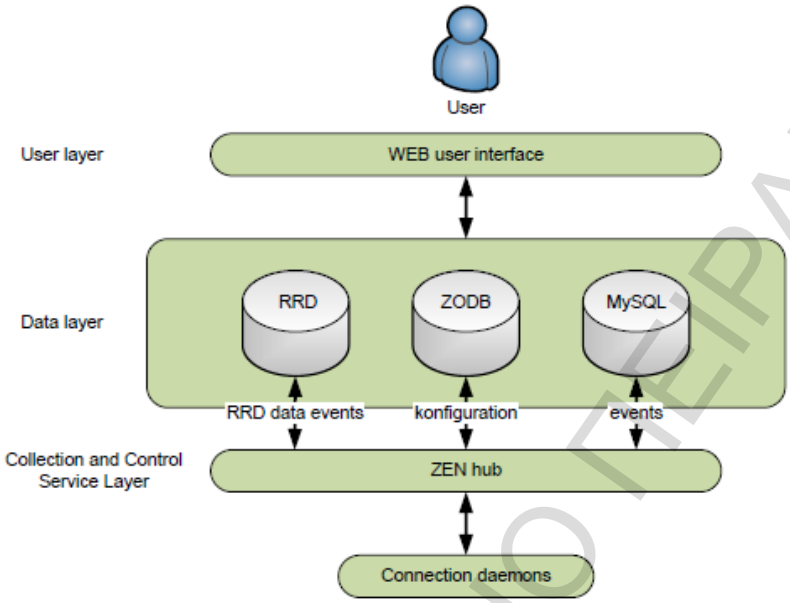
Το Zenoss είναι ικανό να επιβλέψει την διαθεσιμότητα όσο και την απόδοση και έχει πολλούς διαφορετικούς τρόπους να παρακολουθήσει μετρικές απόδοσης συσκευών και δομικών συστατικών (μέσω SNMP, εντολών ZenCommand ή RPC). Για την παραμετροποίηση και ρύθμιση του συστήματος χρησιμοποιούνται συγκεκριμένα πρότυπα που περιγράφουν αυτή την πληροφορία. Η διαδικασία εντολών (*ZenCommand*) επιτρέπει την εκτέλεση αρχείων κώδικα, και χρησιμοποιεί plugins για την συλλογή δεδομένων απόδοσης τόσο τοπικά όσο και απομακρυσμένα. Ο μηχανισμός για την αυτόματη ειδοποίηση βασίζεται στον Event Management System του Zenoss το οποίο μπορεί να αναφέρει τα αποτελέσματα της εποπτείας με γράφους ή με κάποιο προσωποποιημένο ταμπλό.

Υπάρχουν δύο εκδόσεις του συστήματος αυτού, το Zenoss Core και το Zenoss Enterprise. Το πρώτο διατίθεται δωρεάν υπό την άδεια GPL ενώ το δεύτερο είναι η εμπορική έκδοση, βασισμένη στην πρώτη, με επιπρόσθετες λειτουργίες.

Το Zenoss σύστημα μπορεί να χωριστεί σε τρία διαφορετικά στρώματα (Εικόνα 10):

- Στρώμα Χρηστών (User layer)
- Στρώμα Δεδομένων (Data layer)
- ΣτρώμαΥπηρεσιώνΣυλλογήςκαιΕλέγχου (Collection and Control Service layer )

Το Στρώμα Χρηστών αποτελείται από ένα διαδικτυακό GUI το οποίο επιτρέπει στον χρήστη να έχει πρόσβαση στα δεδομένα εποπτείας. Το Στρώμα Δεδομένων είναι η κεντρική τοποθεσία για τις τρείς βάσεις δεδομένων που αναφέρθηκαν προηγουμένως. Με το Στρώμα Υπηρεσιών Συλλογής και Ελέγχου συσχετίζονται διάφοροι τύποι προγραμμάτων συλλογής (collection daemons) δεδομένων. Το κεντρικό δομικό συστατικό αυτού του στρώματος είναι το ZenHub, ένα στοιχείο που διαχειρίζεται την πληροφορία μεταξύ του Στρώματος Δεδομένων και των προγραμμάτων συλλογής.




Εικόνα 10: Αρχιτεκτονική Zenoss

### 3.2.3. Λοιπά συστήματα εποπτείας και τεχνολογίες

Παρόλο που έχουν αρκετές διαφορές, το υπολογιστικό πλέγμα (*Grid*) επιδιώκει παρόμοιο στόχο με το Υπολογιστικό Νέφος (*Cloud*): την παροχή πόρων κατ' απαίτηση. Η εποπτεία και παρακολούθηση πόρων έχει υπάρξει ενεργή περιοχή έρευνας για αρκετό καιρό[64]. Στην ενότητα που ακολουθεί θα καταγράψουμε και ταξινομήσουμε τους διαθέσιμους μηχανισμούς και θα αναγνωρίσουμε τα χαρακτηριστικά τα οποία θα μπορούσαμε να χρησιμοποιήσουμε έτσι ώστε να σχεδιάσουμε και υλοποιήσουμε έναν ισχυρό και αποδοτικό μηχανισμό εποπτείας για Νέφη. Όπως καταγράφεται στο[65], ένας μεγάλος αριθμός διαφορετικών λύσεων εποπτείας έχουν αναπτυχθεί. Προσφέρουν διαφορετικές ικανότητες και χαρακτηριστικά γνωρίσματα, μερικές φορές επικαλυπτόμενα, μερικές φορές συμπληρωματικά. Όντας λύσεις για πλέγματα, είναι σε θέση να ενσωματώσουν τις πληροφορίες ελέγχου που προέρχονται από τις πολλαπλές πηγές σε μια ενιαία δομή. Πολλές λύσεις μπορούν να λειτουργήσουν με την αυθαίρετη μέτρηση στοιχείων στους πόρους, το δίκτυο, τις εφαρμογές, και άλλα.

Το GridICE[66] είναι ένα αυτόνομο σύστημα εποπτείας το οποίο υποστηρίζει ροή γεγονότων από αισθητήρες προς τους καταναλωτές της πληροφορίας. Παρέχει διαφορετικές συσχετίσεις και συνδυασμούς της συλλεχθείσας πληροφορίας βάση των συγκεκριμένων αναγκών κάθε κατηγορίας χρηστών με διαφορετικό επίπεδο αοριστίας στο Grid (*Virtual Organization level, Grid Operation Center level, Site Administration level and End-User level*), αλλά στερείται διεπαφής παραγωγού (*producer*). Σε γενικές γραμμές, παρέχει λειτουργικότητες χαμηλού επιπέδου, δύσκολες στην επέκταση και προσαρμογή σε ένα πολύπλοκο μηχανισμό λήψης αποφάσεων. Επιπρόσθετα, λειτουργεί με χρονική συχνότητα δεκάδων δευτερολέπτων, ή ακόμα και λεπτών κάτι το οποίο το καθιστά ακατάλληλο για εφαρμογές πραγματικού χρόνου (*real-time*).

Λύσεις όπως η MonALISA (MONitoring Agents using a Large Integrated Services Architecture)[67] και το Globus MDS[68] είναι έντονα ευέλικτα συστήματα τα οποία προσφέρουν μηχανισμούς παρακολούθησης για υποδομές clusters και Grids. Το MonALISA είναι ένα πλαίσιο υπηρεσιών που βασίζεται στο Jini API[69] και σε τεχνολογίες δικτυακών υπηρεσιών (*Web Services*) για να επιβλέπουν υπολογιστικούς κόμβους και δίκτυα σε μεγάλης κλίμακας κατανεμημένα συστήματα. Το Globus Monitoring and Discovery Service (MDS) είναι μια σουίτα από δομικά συστατικά για εποπτεία και ανακάλυψη πόρων και υπηρεσιών. Παρέχεται μαζί με το Globus Toolkit σύστημα και διαθέτει τυποποιημένες διεπαφές που βασίζονται στα WS-Resource Framework (WSRF)[70] και WS-Notification (WS-N)[71]. Από τα δύο προαναφερθέντα APIs το πιο ανεπτυγμένο και ευρέως χρησιμοποιημένο είναι το Globus MDS. Υποστηρίζεται από την πολύ ενεργή κοινότητα του Globus η οποία είναι από τις πιο σημαντικές στον χώρο των τεχνολογιών Grid. Η επιτυχία του επίσης βασίστηκε στο γεγονός ότι υιοθέτησε ευρέως γνωστά πρότυπα (standards) όπως GLUE [72] και WSRF, ενισχύοντας με αυτόν τον τρόπο την διαλειτουργικότητα. Όσον αφορά την απόδοση, τα αρχικά στοιχεία που παρουσιάζονται στην βιβλιογραφία δείχνουν ότι συμπεριφέρεται σταθερά ακόμα και σε συχνότητα λειτουργίας κοντά στο ένα δευτερόλεπτο[73][74]. Το API παρέχει τις βασικές λειτουργικότητες αλλά μπορεί εύκολα να επεκταθεί υλοποιώντας διαδικτυακές υπηρεσίες σε Java που εγκαθίστανται στον Globus Container.

Ένα ακόμα σημαντικό παράδειγμα είναι το Grid Monitoring Architecture (GMA) όπως αυτό προσδιορίστηκε από το Open Grid Forum (OGF)[75]. Στην βιβλιογραφική αυτή αναφορά περιγράφεται ένα αρχιτεκτονικό σχέδιο το οποίο μπορεί να υλοποιηθεί με διάφορες τεχνολογίες και ρυθμίσεις. Η αρχιτεκτονική αυτή περιέχει μια υπηρεσία καταλόγου που επιτρέπει παραγωγούς και καταναλωτές πληροφοριών εποπτείας να εντοπίζει ο ένας τον άλλον. Τα δεδομένα κατόπιν, μεταφέρονται απευθείας μεταξύ των δύο οντοτήτων. Είναι όμως εφικτή η τοποθέτηση ενδιάμεσου δομικού συστατικού που θα αναλαμβάνει την συσχέτιση και τον συνδυασμό των δεδομένων.

Εκτός από τα διαθέσιμα APIs και συστήματα που κάποιος μπορεί να χρησιμοποιήσει για να υλοποιήσει μια λύση για εποπτεία, υπάρχουν διάφορες ερευνητικές πρωτοβουλίες οι οποίες είναι άξιες αναφοράς. Στο [76] παρουσιάζεται ένα πλήρες πλαίσιο διαχείρισης πληροφορίας, εποπτείας, διαχείρισης λογαριασμών και χρέωσης για πλατφόρμες Νεφών. Παρόλο που παρουσιάζουν μια καλή ανάλυση προδιαγραφών όσον αφορά την εποπτεία σε Υπολογιστικά Νέφη, στερούνται πληροφοριών υλοποίησης και στοιχείων για την απόδοση. Στο [77] αναλύεται ένα πολύ ενδιαφέρον σύστημα παρακολούθησης το οποίο συμπεριφέρεται αποδοτικά ακόμα και με μεγάλο όγκο αιτήσεων. Το μειονέκτημά του όμως είναι η στατικότητα του μηχανισμού σχετικά με την περίοδο λειτουργίας με αποτέλεσμα έλλειψη δυναμικότητας και προσαρμογής. Τέλος, η ανάλυση που παρουσιάζεται στο [78] αποδεικνύει ότι στις αντικειμενοστρεφείς αρχιτεκτονικές (SOA) η υπολογιστική επιβάρυνση που εισαγάγετε από το σύστημα εποπτείας μπορεί να είναι σημαντικό και υπολογίσιμο και εν τέλει θα μπορούσε να επηρεάσει την λειτουργία της εφαρμογής/υπηρεσίας.

Το κύριο πρόβλημα όταν μεταφέρουμε τις λύσεις εποπτείας πλέγματος στα Υπολογιστικά Νέφη είναι ότι τα συστήματα ελέγχου για Grids σχεδιάστηκαν με υποθέσεις διαφορετικές από εκείνους των Νεφών. Μια κύρια αρχή των πλεγμάτων είναι η συνεργασία μεταξύ των διαφορετικών περιοχών προμηθευτών και χρηστών. Υπό αυτήν τη μορφή, η κύρια εστίασή τους είναι στην ενσωμάτωση των πληροφοριών ελέγχου από πολλαπλές περιοχές σε ένα ενιαίο σύστημα ελέγχου. Για μεμονωμένα Νέφη, αυτή η λειτουργία δεν απαιτείται. Για συνενωμένα σε ομοσπονδία Νέφη, αυτή η λειτουργία απαιτείται και οι έννοιες και οι αρχιτεκτονικές αναπτυγμένες από την κοινότητα Grid, π.χ. η Grid Monitoring Architecture, μπορεί να μεταφερθεί στη συνενωμένη σε ομοσπονδία περιοχή Νεφών. Ένα σημαντικό εμπόδιο στη χρησιμοποίηση των συστημάτων εποπτείας πλέγματος στα περιβάλλοντα Νεφών είναι ότι η διανομή των πληροφοριών στα πλέγματα είναι δεδομένη, και δεν θεωρείται ως παραβίαση ασφάλειας όπως θα ήταν στα περιβάλλοντα Νεφών. Τα δικαιώματα πρόσβασης στα δεδομένα εποπτείας στα πλέγματα υπάρχουν μόνο μερικές φορές και εάν υπάρχουν, έχουν συνήθως τη χονδροειδή κοκκοποίηση (*granularity*) (π.χ., συστάδες) και χωρίζονται κυρίως από τις εικονικές οργανώσεις (*virtual organizations*), όχι από τα άτομα. Συγκεκριμένα, οι πληροφορίες για την κατάσταση των πόρων που χρησιμοποιούνται από έναν προμηθευτή πλέγματος διανέμονται συνήθως στους χρήστες του πλέγματος. Οι προμηθευτές Νεφών επεξεργάζονται τέτοιες πληροφορίες όπως τα εταιρικά μυστικά και δεν θα έδιναν αυτές τις πληροφορίες έξω στους πελάτες[79].

Μια άλλη διαφορά μεταξύ των πλεγμάτων και των Νεφών είναι η κοινή υπόθεση στα πλέγματα ότι υπολογιστικοί πόροι διανέμονται με έναν μη-εικονικοποιημένο τρόπο. Συνδεδεμένη με αυτή την προσέγγιση είναι ότι πληροφορίες εποπτείας συσχετίζονται με τους φυσικούς πόρους και έρχονται στο επίπεδο των φυσικών μηχανών, όχι των εικονικών μηχανών. Στα Νέφη, οι πληροφορίες εποπτείας για τους φυσικούς πόρους είναι μόνο χρήσιμες στους προμηθευτές των πόρων, όχι στους πελάτες. Επιπλέον, ένας κύριος σκοπός πολλών συστημάτων εποπτείας και παρακολούθησης πλεγμάτων είναι να ενισχυθούν οι αποφάσεις σχετικά με το σχεδιασμό εργασίας. Ο χρονοπρογραμματιστής πλέγματος προσπαθεί να στείλει τις εργασίες σε μια περιοχή που θεωρεί βέλτιστη όσον αφορά μερικά κριτήρια, συμπεριλαμβανομένων των διαθέσιμων πόρων υλικού και του εγκατεστημένου λογισμικού. Η απόφαση σχετικά με το πού να τοποθετηθεί ένα ιδιαίτερο VM μέσα σε ένα Νέφος είναι πάντα η απόφαση του συγκεκριμένου προμηθευτή Νεφών. Η απόφαση του χρήστη για το ποιό Νέφος να χρησιμοποιήσει, εξαρτάται από τα κριτήρια που ο μεμονωμένος χρήστης επιλέγει, συνήθως συμπεριλαμβανομένης της τιμής και της φήμης. Η διαθεσιμότητα των πόρων υποτίθεται πάντα και δεν είναι επομένως ένα ζήτημα.

# 4. Υπηρεσιοστρεφής πλατφόρμα εποπτείας υποδομής, εποπτείας εκτέλεσης υπηρεσιών και παραγωγής γεγονότων σε υπολογιστικά νέφη

Σε αυτό το κεφάλαιο θα περιγράψουμε την αρχιτεκτονική του πειράματος που διενεργήσαμε. Αρχικά θα περιγραφεί η πλατφόρμα και το περιβάλλον του πειράματος. Στη συνέχεια θα γίνει αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη των επιμέρους τμημάτων του λογισμικού και των υπηρεσιών που χρησιμοποιήθηκαν. Αναλυτική αναφορά θα γίνει στα μέρη του λογισμικού που αναπτύχθηκαν για να δημιουργηθεί ο μηχανισμός ελέγχου και παρακολούθησης.

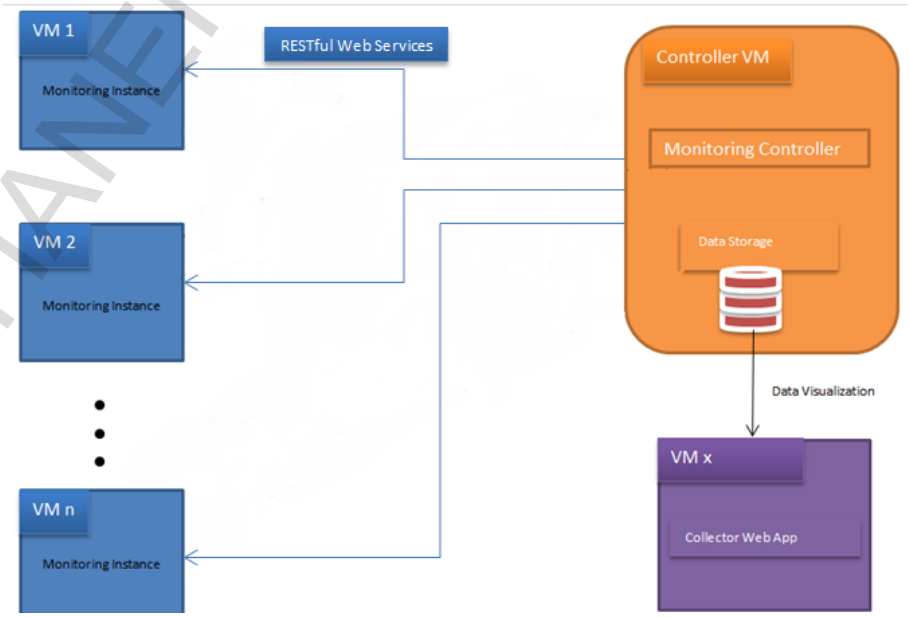
Τέλος θα παρουσιαστούν τα αποτελέσματα του πειράματος που διενεργήθηκε, και θα αξιολογηθεί το κατά πόσο ο μηχανισμός ελέγχου και παρακολούθησης σε συνεργασία με το μηχανισμό διαχείρησης ροής εργασιών, ικανοποίησαν τις ανάγκες της υποδομής για να προσφέρει μία προκαθορισμένη ποιότητα υπηρεσίας (QoS), για μία επιλεγμένη δοκιμαστική εφαρμογή που θα εγκατασταθεί σε κάθε εικονικό μηχάνημα.

## 4.1. Περιγραφή αρχιτεκτονικής

Στο παρόν υποκεφάλαιο θα παρουσιάσουμε την αρχιτεκτονική του μηχανισμού ελέγχου και παρακολούθησης για το πείραμα που αναφέρθηκε στο προηγούμενο υποκεφάλαιο. Η συνολική υλοποίηση διασπάται στα εξής επιμέρους τμήματα λογισμικού:

- Στιγμιότυπο τοπικού ελέγχου και εποπτείας (**System Monitoring**).
- Κεντρικός ελεγκτής ελέγχου και εποπτείας (**Monitoring Controller**).
- Μία κεντρική βάση δεδομένων(**vmmanager db**).
- Μία διαδικτυακή εφαρμογή(**Collector Web App**), η οποία προσφέρει μία διεπαφή στο χρήση ώστε να έχει εύκολη πρόσβαση στα συγκεντρωτικά δεδομένα από όλα τα εικονικά μηχανήματα.

Περιλιπτικά, στο παρακάτω σχήμα[Εικόνα 11] παρουσιάζεται ο ρόλος του κάθε ενός από τα παραπάνω κομμάτια της αρχιτεκτονικής στο ολικό μηχανισμό ελέγχου και εποπτείας:




Εικόνα 11: Αρχιτεκτονική μηχανισμού ελέγχου και εποπτείας

Παρακάτω θα αναλύσουμε το ρόλο του κάθε μέρους καθώς επίσης το πως ο συνολικός μηχανισμός αλληλεπιδρά με την εφαρμογή, αλλά και με τον διαχειρηστή ροής εργασιών.

### 4.1.1. Στιγμιότυπο τοπικού ελέγχου και εποπτείας (SystemMonitoring)

Η εφαρμογή τοπικού ελέγχου και εποπτείας είναι εγκατεστημένη σε κάθε εικονικό μηχάνημα ξεχωριστά και έχει ως ρόλο το να «ρωτά» (query) το λειτουργικό σύστημα του μηχανήματος για την κατάσταση του. Για να το πετύχει αυτό εκτελεί κάθε φορά εντολές UNIX, τουφλοιού του λειτουργικού συστήματος (shellcommands). Η κατάσταση του μηχανήματος καθορίζεται από 3 διαφορετικές παραμέτρους συστήματος. Αυτές είναι:

- Χρησιμοποίηση συνολικής επεξεργαστικής ισχύος (CPU utilization %).
- Δεσμευμένη μνήμη τυχαίας προσπέλασης (used RAM %).
- Χρησιμοποιούμενος αποθηκευτικός χώρος (used disk space %).

Εκτός από τα παραπάνω, η εφαρμογή είναι υποχρεωμένη, κατά την εκκίνηση της, να δημοσιεύει τη ταυτότητα του μηχανήματος, αποθηκεύοντας συγκεκριμένη πληροφορία στη κεντρική βάση δεδομένων, η οποία είναι διαθέσιμη από τα υπόλοιπα μέρη του συστήματος. Η πληροφορία περιέχει πληροφορίες σχετικά με:

- Τη διεύθυνση IP του μηχανήματος.
- Τον αριθμό επεξεργαστικών πυρήνων του μηχανήματος.
- Το μέγεθος της μνήμης τυχαίας προσπέλασης σε MB.
- Το μέγεθος του χώρου αποθήκευσης σε MB.
- Την ημερομηνία και ώρα εκκίνησης του μηχανήματος.
- Την ημερομηνία και ώρα τερματισμού του μηχανήματος.

Η εφαρμογή υλοποιεί μία υπηρεσία διαδικτύου. Κάθε φορά όπου μία μέθοδος της υπηρεσίας καλείται, η αντίστοιχη εντολή φλοιού εκτελείται στο σύστημα, η εφαρμογή επεξεργάζεται τα δεδομένα και τα αποστέλλει στον κεντρικό διαχειρηστή σε σειριοποιημένη μορφή. Τέλος, όταν ο χρήστης ή ο διαχειρηστής εργασιών επιχειρήσει να τερματίσει το μηχάνημα, η εφαρμογή αναλαμβάνει να ενημερώσει τη κεντρική βάση δεδομένων με τη ημέρα και ώρα τερματισμού του μηχανήματος και να ειδοποιήσει το κεντρικό ελεγκτή για το συμβάν.

### 4.1.2. Κεντρικός ελεγκτής μηχανισμού ελέγχου και εποπτείας (Monitoring Manager)

Η εφαρμογή του κεντρικού ελεγκτή είναι υπεύθυνη να συγκεντρώνει πληροφορίες σχετικά με τη κατάσταση των ενεργών εικονικών μηχανημάτων και να ελέγχει εάν κάποια παράμετρος συστήματος έχει ξεπεράσει κάποια προκαθορισμένα όρια. Η εφαρμογή διαβάζει μία λίστα από κανόνες από ένα αρχείο μορφής XML. Το αρχείο ορίζει 3 διαφορετικές καταστάσεις κρισιμότητας: 1) ομαλή (normal), 2) προειδοποιητική (warning), 3) κρίσιμη (alarm).

Για κάθε κατάσταση όριζονται οι ανώτερες επιτρεπτές τιμές για κάθε παράμετρο συστήματος των εικονικών μηχανημάτων, καθώς επίσης και το χρονικό διάστημα το οποίο πρέπει να μεσολαβεί ανάμεσα σε δύο επερωτήσεις κατάστασης των εικονικών μηχανημάτων. Μία τυπική μορφή ενός XMLαρχείου κανόνων είναι η παρακάτω:

```
<?xml version="1.0" encoding="UTF-8"?>
- <rules>
    <pull-data-interval-sec>30</pull-data-interval-sec>
  - <alarm-rules>
      <pull-data-interval-sec>10</pull-data-interval-sec>
      <max-cpu-usage>0.4</max-cpu-usage>
      <max-memory-usage>0.5</max-memory-usage>
      <max-disk-usage>0.4</max-disk-usage>
    </alarm-rules>
  - <critical-rules>
      <pull-data-interval-sec>5</pull-data-interval-sec>
      <max-cpu-usage>0.6</max-cpu-usage>
      <max-memory-usage>0.8</max-memory-usage>
      <max-disk-usage>0.6</max-disk-usage>
```

Εικόνα 12: Κανόνες με ανώτερα επιτρέπτα όρια για τις παραμέτρους συστήματος

Όπως βλέπουμε και στην εικόνα 12 το σύνολο των κανόνων χωρίζεται σε 3 βασικές καταστάσεις κρισιμότητας. Στην ομαλή κατάσταση, η οποία δεν ορίζει τίποτε άλλο εκτός από το προκαθορισμένο χρονικό διάστημα μεταξύ δύο διαδοχικών κλήσεων του ελεγκτή προς την υπηρεσία ενός εικονικού μηχανήματος. Η κατάσταση προειδοποίησης (alarmstate), ορίζει ποιες (cpu,memory,disk) είναι εκείνες που θα μεταβάλουν την κατάσταση του μηχανήματος από ομαλή σε προειδοποιητική. Στο συγκεκριμένο παράδειγμα, η τιμή για το μέγιστο επιτρεπόμενο φόρτο επεξεργαστών είναι 0.5 (50%). Αντίστοιχα για την μνήμη είναι επίσης 0.5 (50%) και για το δίσκο είναι 0.4 (40%). Τέλος, όταν το σύστημα μεταβεί σε προειδοποιητική κατάσταση το χρονικό διάστημα μεταξύ δύο διαδοχικών κλήσεων θα πρέπει να μειωθεί ώστε να υπάρχει πιο εντατικήεποπτεία του συστήματος. Στη παραπάνω εικόνα βλέπουμε ότι αυτό το χρονικό διάστημα μειώνεται από 30 σε 10 sec.

Τέλος, παρατηρούμε ότι όταν οι τιμές του συστήματος αυξηθούν πάνω ένα όριο που ορίζεται στην ομάδα κρισιμότητας critical-rules, τότε το εικονικό μηχάνημα θεωρούμε ότι έχει μεταβεί σε κρίσιμη κατάσταση, καθώς κάποια/ες παράμετροι είναι στο όριο, και σημαίνει ότι υπάρχει κίνδυνος το εν λόγω εικονικό μηχάνημα να πάψει να αποκρίνεται. Επιπρόσθετα, σε ένα τέτοιο σενάριο ο χρόνος μεταξύ δύο διαδοχικών κλήσεων μειώνεται ακόμα περισσότερο, από 10 σε 5 sec. Το παραπάνω παράδειγμα μας δείχνει το τρόπο που λειτουργεί ο ελεγκτής όταν διαπιστώσει μη-ομαλές καταστασείς για ένα ή περισσότερα ενεργά εικονικά μηχανήματα.

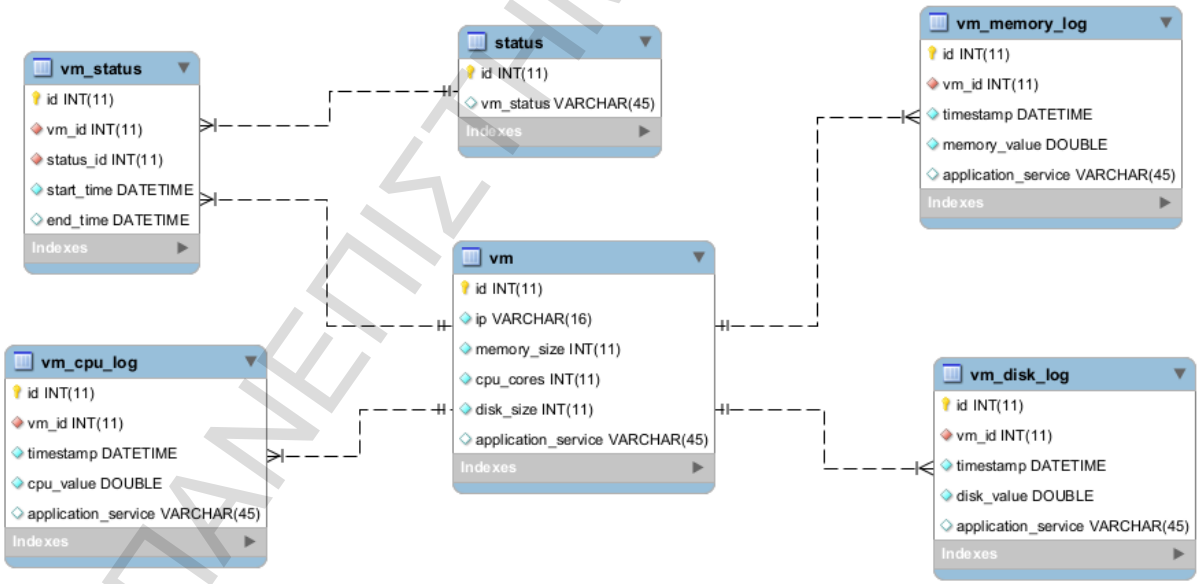
Αρχικά, ο ελεγκτής ανατρέχει στη βάση δεδομένων ώστε να ανακτήσει τη λίστα με όλα τα ενεργά εικονικά μηχανήματα. Για κάθε εικονικό μηχανήμα ο ελεγκτής καλεί ξεχωριστά την υπήρεσια του στιγμιοτύπου ελέγχου, για κάθε μία από τις 3 παραμέτρους. Για παράδειγμα, αν μία δεδομένη στιγμή, υπάρχουν 4 ενεργά εικονικά μηχανήματα, τότε ο ελεγκτής θα πρέπει να πραγματοποιεί 4X3=12 κλήσεις στις επιμέρους υπηρεσίες των στιγμιοτύπων της εφαρμογής που είναι εγκατεστημένες στα εικονικά

μηχανήματα, ανά 30 sec σε ομάλες συνθήκες. Ο κύριος ρόλος του ελεγκτή είναι να ελέγχει κάθε τιμή που λαμβάνει και, αφού πρώτα την αποθηκεύσει στη βάση δεδομένων (logging), να τη συγκρίνει με τους κανόνες που έχουν οριστεί. Σε περίπτωση που διαπιστώσει αλλαγή κατάστασης για ένα εικονικό μηχάνημα, ο ελεγκτής θα πρέπει να ενημερώσει τον διαχειρηστή ροής εργασίας, καλώντας μία υπηρεσία στην οποία στέλνει το συμβάν περιγράφοντας το είδος του συμβάντος, τη τιμή, τη χρονική στιγμή και τέλος το βαθμό κρισιμότητας του συμβάντος. Με αυτό το τρόπο επιτυγχάνεται σε μεγάλο βαθμό η ελαστικότητα του πειράματος, αφού ο ελεγκτής διασφαλίζει ότι κάθε αλλαγή στη συμπεριφορά ενός εικονικού μηχανήματος μπορεί να εντοπιστεί έγκαιρα και να ενημερωθεί ο διαχειρηστής ροής εργασιών. Έτσι εάν ένα μηχάνημα έχει καταναλώσει ένα μεγάλο μέρος των πόρων του, μπορεί να του παρεσχηθούν επιπλέον πόροι ώστε να διατηρηθεί η ίδια απόδοση του κατά τη διάρκεια του πειράματος, ή να ενεργοποιηθεί ένα καινούργιο εικονικό μηχάνημα, ώστε να γίνει πιο ομαλή η κατανομή του φόρτου εργασίας.

### 4.1.3. Κεντρική βάση δεδομένων

Η βάση δεδομένων διατήρει όλη τη απαραίτητη πληροφορία, που χρειάζονται τόσο ο διαχειρηστής ροής εργασιών, όσο και ο διαχειρηστής ελέγχου και εποπτείας. Η βάση δεδομένων ανατρέχεται από όλα τα αρχιτεκτονικά τμήματα του μηχανισμού ελέγχου και εποπτείας. Διατηρεί πληροφορία σχετικά με:

- Τη ταυτότητα των εικονικών μηχανημάτων και των τεχνικών χαρακτηριστικών τους.
- Τη κατάσταση των μηχανημάτων (ενεργά/ανενεργά).
- Το ιστορικό των μετρήσεων όλων των εικονικών μηχανημάτων για κάθε παράμετρο ξεχωριστά (CPU,RAM,disk).




Εικόνα 13: Σχήμα της βάσης δεδομένων

Παρακάτω αναλύουμε τα πεδία κάθε πίνακα ξεχωριστά και τι έιδους πληροφορία διατηρούν:

***Status***

Ένας παραμετρικός πίνακας που περιγράφει τις 2 πιθανές καταστάσεις των εικονικών μηχανημάτων. Περιέχει το πεδίο κλειδί id. Το κύριο πεδίο μπορεί να πάρει τιμές “ACTIVE” “INACTIVE”.

***VM***

Ο πίνακας που διατηρεί τη ταυτότητα των εικονικών μηχανημάτων. Εκτός από το πεδίο κλειδί, περιέχει τα εξής πεδία:

- Το πεδίο Ip που περιέχει την IPδιεύθυνση του εικονικού μηχανήματος.
- Το πεδίο memory_size που περιέχει το συνολικό μέγεθος της μνήμης RAM του εικονικού μηχανήματος.
- Το πεδίο cpu_cores που περιέχει τον αριθμό των πυρήνων επεξεργαστή που διαθέτει το εικονικό μηχάνημα.
- Το πεδίο disk_size που περιέχει το μέγεθος του σκληρού δίσκου σε MB, που διαθέτει το εικονικό μηχάνημα.
- Το πεδίο application_service το οποίο είναι προεραιτικό και περιέχει τη διεύθυνση της υπηρεσία μίας οποιασδήποτε εφαρμογής που μπορεί να λειτουργεί στο εικονικό μηχάνημα και να χρειάζεται να ελέγχεται από το μηχανησμό.

***Vm_Status***

Οπίνακας περιέχει πληροφορίες σχετικά με τη κατάσταση των εικονικών μηχανημάτων. Ο πίνακας κάνει αναφορά στο πίνακα vm (1..1) μέσω του πεδίου vm_id, και στο πίνακα status (1...1) μέσω του πεδίου status_id. Επίσης περίεχει τα πεδία start_timestamp και end_timestamp. Το πεδίο start_timestamp περιγράφει τη χρονική στιγμή την οποία το εικονικό μηχάνημα εκκίνησε τη λειτουργία του, ενώ το πεδίο end_timestampπεριέχει τη χρονική στιγμή την οποία το εικονικό μηχάνημα τερμάτισε τη λειτουργία του.

***Vm_cpu_log***

Ο πίνακας περιέχει τις μετρήσεις επεξεργαστικού φόρτου των εικονικών μηχανημάτων. Η πληροφορία αυτή διατηρείται στο πεδίο cpu_value. Τοπεδίο vm_id αποτελεί αναφορά στο πίνακα vm (1…*). Τέλος το πεδίο timestampπεριέχει τη χρονική στιγμή κατά την οποία έγινε η μέτρηση.

***Vm_memory_log***

Ο πίνακας περιέχει τις μετρήσεις σχετικά με τη χρησιμοποίηση της μνήμης RAM των εικονικών μηχανημάτων. Η πληροφορία αυτή διατηρείται στο πεδίο memory_value. Τοπεδίο vm_idαποτελεί αναφορά στο πίνακα vm (1…*). Τέλος το πεδίο timestamp περιέχει τη χρονική στιγμή κατά την οποία έγινε η μέτρηση.

***Vm_disk_log***

Ο πίνακας περιέχει τις μετρήσεις σχετικά με τη χρησιμοποίηση του αποθηκευτικού χώρου των εικονικών μηχανημάτων. Η πληροφορία αυτή διατηρείται στο πεδίο disk_value. Τοπεδίο vm_id αποτελεί αναφορά στο πίνακα vm (1…*). Τέλος το πεδίο timestampπεριέχει τη χρονική στιγμή κατά την οποία έγινε η μέτρηση.

### 4.1.4. Διαδικτυακή εφαρμογή (Collector Web Application)

Πρόκειται για μία εφαρμογή διαδικτύου η οποία έχει σκοπό να παρουσιάσει με ένα φιλικό προς το χρήστη τρόπο τα δεδομένα και τις μετρήσεις τα οποία έχει συλλέξει ο μηχανισμός ελέγχου και εποπτείας. Η κεντρική του σελίδα παρουσιάζει τη λίστα με τα ενεργά εικονικά μηχανήματα. Ο χρήστης μπορεί να επιλέξει ένα εικονικό μηχάνημα και να δει τα στοχεία και τη κατάσταση του. Επιπρόσθετα ο χρήστης μπορεί να με τη βοήθεια ενός γραφήματος να δει το ιστορικό των μετρήσεων για κάθε μηχάνημα.

## 4.2. Περιγραφή πειράματος

### 4.2.1. Υποδομή

Το πείραμα διενεργήθηκε στη πλατφόρμα Ομόσπονδης Υποδομής Νέφους, η οποία προσέφερε την υποδομή και τους αναγκαίους πόρους για να γίνει δυνατή η υλοποίηση του πειράματος. Επιπρόσθετα, έδινε την δυνατότητα στους χρήστες να υλοποιούν πειράματα δημιουργόντας εικονικά μηχανήματα που είναι εγκατεστημένα σε φυσικούς εξυπηρετητές διαφορετικών τοποθεσιών. Αυτή η ετερογένια, δίνει τη δυνατότητα δημιουργίας πολύπολοκων σεναρίων εκτέλεσης πειραμάτων, και προσφέρει ένα πεδίο για πολύ χρήσιμη ερευνητική εργασία.

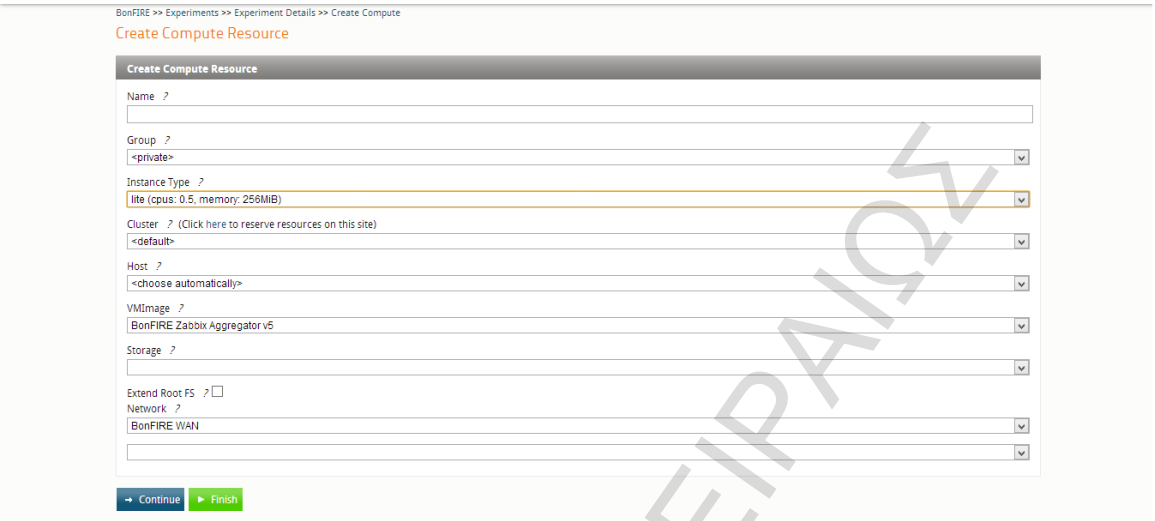
Επίσης, η πλατφόρμα υποστηριζόταν από μία μεγάλη και πολύ καταρτισμένη κοινότητα ανθρώπων, η οποία ήταν διαθέσιμη κάθε στιγμή για προσφέρει βοήθεια στους χρήστες. Σημαντική επίσης είναι η διαδικτυακή εφαρμογή που προσφέρει μία διεπαφή, με την οποία ο χρήστης μπορεί να δημιουργεί πειράματα και πόρους με τη βοήθεια ενός πολύ φιλικού γραφικού περιβάλλοντος.


**Εικόνα 14: Η διαδικτυακή διαπεφή της πλατφόρμας Υποδομής νέφους**

Η σύνδεση και ο χειρισμός των μηχανημάτων παρέχεται από ένα σύνολο εργαλειών και πρωτοκόλλων, όπως το SSH. Ο χρήστης μπορεί πολύ εύκολα να συνδεθεί σε ένα μηχάνημα γνωρίζοντας την διεύθυνση δικτύου του, μέσα από οποιαδήποτε γραμμή εντολών του λειτουργικού του συστήματος. Η αυθεντικοποίηση του χρήστη γίνεται με την ανταλλαγή ενός δημόσιου κλειδιού που ανταλλάσεται μεταξύ πελάτη και μίας «πύλης» SSH. Κάθε φυσική τοποθεσία που απαρτίζει την υποδομή του Υποδομής Νέφους, παρέχει στους χρήστες από μία SSHπύλη. Όταν ο χρήστης συνδεθεί στη πύλη, από εκεί μπορεί να έχει πρόσβαση σε οποιδήποτε εικονικό μηχάνημα έχει δημιουργήσει για το πείραμα του, είτε αυτό έχει δημιουργηθεί στην ίδια τοποθεσία, είτε σε άλλη.

Ο χρήστης μπορεί πολύ εύκολα να δημιουργεί πόρους (υπολογιστικούς,αποθηκευτικούς,δικτύου). Παρέχεται επίσης τη δυνατότητα, μέσα από ένα εύρος επιλογών, να ρυθμίσει ο ίδιος τις παραμέτρους του μηχανήματος όπως εκείνος χρειάζεται.

Κάθε τοποθεσία φυσικών υποδομών, παρέχει το δικό της υποδίκτυο δημόσιων και τοπικών διευθύνσεων IP, και έτσι είναι δυνατή η επικοινωνία πόρων μεταξύ διαφορετικών υποδομών, επιτρέποντας έτσι τη διενέργηση *cross-site*πειραμάτων.

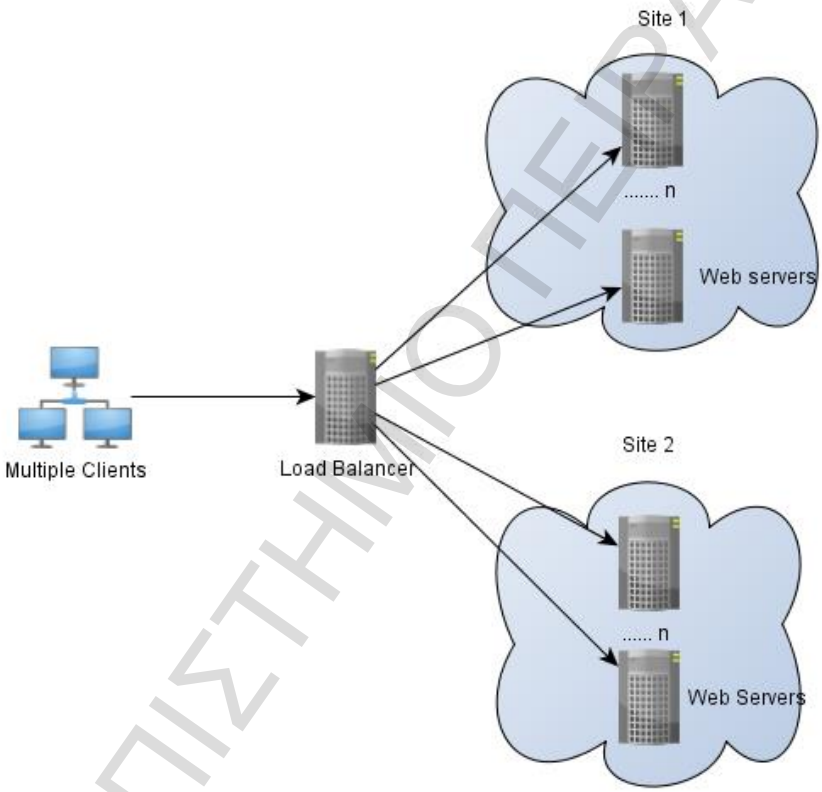



**Εικόνα 15: Σελίδα δημιουργίας εικονικού μηχανήματος**

### 4.2.2. Περιγραφή πειραματικής διαδικασίας

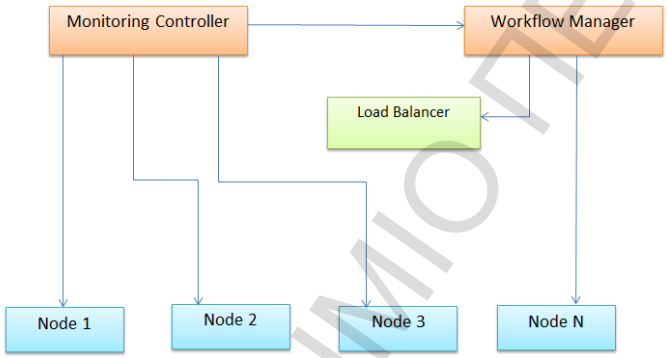
***Τοπολογία πειράματος***

Το πείραμα που διεξήχθη είχε ως σκοπό να παρατηρήσει την συμπεριφορά μίας εφαρμογής ιστού (στην προκειμένη περίπτωση τον έλεγχο εγκυρότητας ενός αριθμού φορολογικού μητρώου) όταν υπάρχει αρκετά μεγάλος φόρτος αιτημάτων προς την εφαρμογή αυτή. Αναλυτικότερα όπως φαίνεται και στην εικόνα[Εικόνα 16] προσομοιόθηκαν πολλαπλοί πελάτες που στέλνουν τα αιτήματα τους σε έναν εξισορροπητή φορτίου (load balancer) ο οποίος με την σειρά του τα μοιράζει σε πολλαπλούς εξυπηρετητές που φέρουν ένα στιγμιότυπο της εφαρμογής. Οι εξυπηρετητές μπορούν να βρίσκονται στην ίδια τοποθεσία στο σύννεφο ή σε διαφορετικές.




**Εικόνα 16: Τοπολογία πειράματος**

Αναλυτικότερα όπως φαίνεται και στην εικόνα[Εικόνα 17] υπάρχει ένα στιγμιότυπο του MonitoringControllerστη τοποθεσία το οποίο πραγματοποιεί εποπτεία στους κόμβους 1 έως 3. Σε περίπτωση που διαπιστώσει ότι κάποιος από τους ορισμένους κανόνες έχει παραβιασθεί, τότε ενημερώνει το στιγμιότυπο του Workflowmanager. Εκείνος ανάλογα με το είδος ειδοποίησης που έχει λάβει από το MonitoringControllerμπορεί να δημιουργήσει ένα νέο εικονικό μηχάνημα. Έαν πράξει κάτι τέτοιο τότε θα πρέπει να ενημερώσει το στιγμιότυπο του LoadBalancer, ώστε να συμπεριλάβει και το νέο μηχάνημα στην εξισορρόπηση. Αυτόματα ενημερώνεται καιοMonitoringController, και συμπεριλαμβάνει και το νέο εικονικό μηχάνημα στη λίστα μηχανημάτων που εποπτεύει. Αξίζει να σημειωθεί ότι για τη συγκεκριμένη πειραματική διαδικασία, ο MonitoringControllerδεν πραγματοποιεί εποπτεία για το LoadBalancer, παρά μόνο για τους εκάτοστε κόμβους που χρησιμοποιεί ο LoadBalancer. Κάθε επιμέρους κόμβος είναι εικονικό μηχάνημα λειτουργεί ένα στιγμιότυπο του Monitoring Instance, του μηχανισμού δηλαδή που αναλαμβάνει να συλλέγει, με τις κατάλληλες εντολές φλοιού συστήματος, τις πληροφορίες για τη κατάσταση του συστήματος καθώς και πληροφορίες σχετικά με τον αριθμό των αιτήσεων που εξηπυρετεί ο κάθε κόμβος. Ο MonitoringInstanceεπιστρέφει τις πληροφορίες αυτές μέσω υπηρεσιών RESTful, στο στιγμιότυπο του MonitoringController.



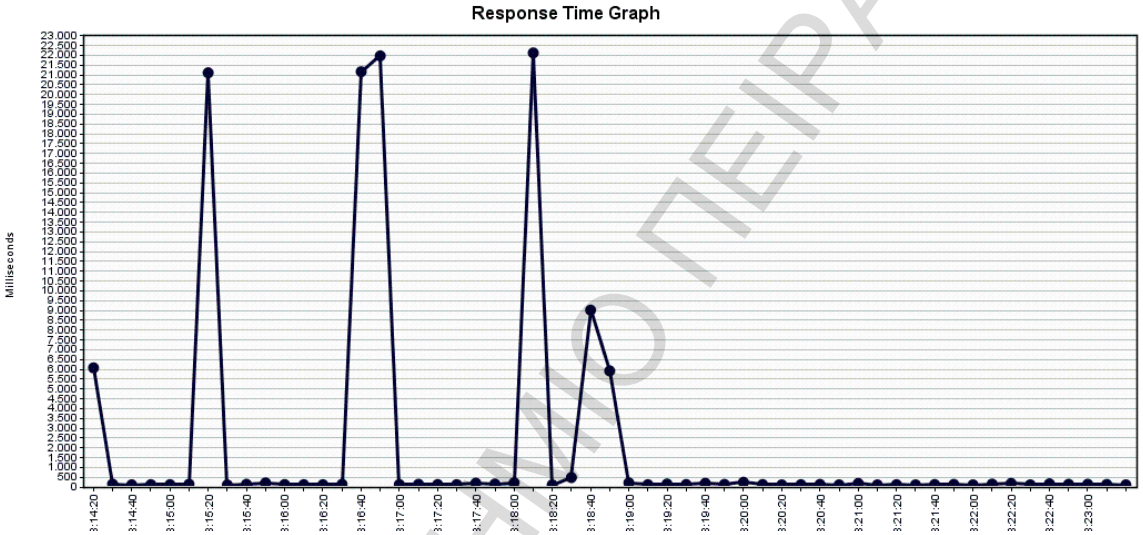

**Εικόνα 17: Σενάριο τοπολογίας**

***Προσομοιώσεις***

Για το συγκεκριμένο πείραμα έγιναν δύο προσομοιώσεις. Στην πρώτη προσομοίωση χροισιμοποιήθηκαν:

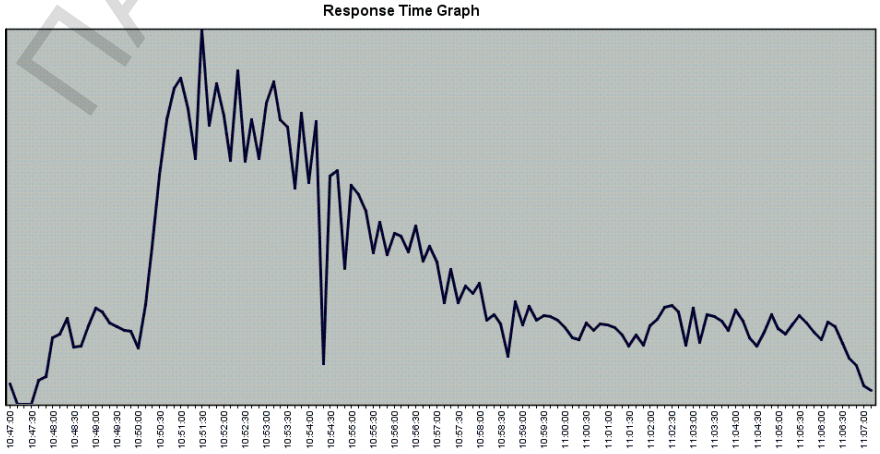
- Ένα μηχάνημα τύπου lite instance (0.5 cpu και 256 mb memory), ως εξισορροπητής φορτίου.
- Και ένα αντίστοιχο μηχάνημα για το ρόλο του εξυπηρετητή.

Για να μετρήθεί η συμπεριφορά της εφαρμογής προσομοιώθηκαν εκατό χρήστες, οι οποίοι έστελναν συνεχώς αιτήματα στην εφαρμογή (περιπου 7 αιτήματα το δευτερόλεπτο από κάθε χρήστη). Όπως φαίνεται και στην εικόνα[Εικόνα 20], σποραδικά παρατηρείται μεγάλη καθυστέρηση (>20 sec) στην επεξεργασία των αιτημάτων. Για αυτό των λόγο ενεργοποιήθηκε η παραπάνωροή εργασίας και προστέθηκε ένας επιπλέον υπολογιστικός πόρος έτσι ώστε να εξομαλυνθεί ο χρόνος απόκρισης των αιτήσεων.




Εικόνα 18: Χρόνος επεξεργασίας αιτημάτων χρηστών

Στη δεύτερη προσομοίωση, αρχικά υπήρχε ένας εξισσοροπηστής φορτίου τύπου lite instance (0.5 cpu και 256 mb memory) και δύο εξυπηρετητές τύπου lite instance (0.5 cpu και 256 mb memory). Προσομοιώθηκαν 250 χρήστες με περίπου 7 αιτήματα ο καθένας το δευτερόλεπτο, αλλά πλέον τα αιτήματα δεν ήταν σταθερά αλλά ακολουθούσαν μία Gaussian κατανομή, έτσι ώστε επιτευχθεί όσο το δυνατόν μεγαλήτερη τυχαιότητα στον τρόπο που στελνόταν ένα αίτημα στον εξυπηρετητή σε κάθε επανάληψη αιτημάτων. Όταν χρειάστηκε να ενεγοποιηθεί η ροή εργασίας, υπήρχε ήδη ένας διαθέσιμος υπολογιστικός πόρος ο οποίος χρησημοποιήθηκε από τον εξισορροπητή φορτίου, οπότε πλέον υπήρχαν 3 εξυπηρετητές να απαντάνε στα αιτήματα. Οι χρόνοι απόκρισης και πάλι δεν ήταν ικανοποιητικοί, και συνεπώς δημιουργήθηκαν δύο επιπλέον υπολογιστικοί πόροι (σύνολο πέντε εξυπηρετητές) ώστε να μειωθούν οι χρόνοι απόκρισης στα επιθυμητά επίπεδα.




Εικόνα 19: Χρόνος επεξεργασίας αιτημάτων για το 2ο σενάριο αποστολής αιτημάτων

### 4.2.3. Ο μηχανισμός ελέγχου και εποπτείας κατά τη πειραματική διαδικασία

Κατά την λειτουργία της πειραματικής διαδικασίας, ο μηχανισμός ελέγχου και εποπτείας είναι υπεύθυνος για την ομαλή λειτουργία του πειράματος. Σύμφωνα με πρώτους πειραματισμούς, ένας εξυπηρετητής μπορεί να χρειαστεί να καταναλώνει σημαντικό μέρος των πόρων ενός μηχανήματος, εάν οι αιτήσεις προς αυτόν καταφθάνουν μαζικά και σε μεγάλο αριθμό. Για παράδειγμα, σε ένα περιβάλλον με 4 πύρηνες και 8GBμνήμη RAM, παρατηρήθηκε ότι σε μία προσομοίωση 200 χρηστών που στέλνουν συνεχείς αιτήσεις, ο επεξεργαστικός φόρτος μπορεί να φτάσει έως και 70%, ενός η κατανάλωση της μνήμης μπορεί να φτάσει μέχρι και 60%. Γίνεται προφανές λοιπόν, ότι ο μηχανισμός ελέγχου και εποπτείας πρέπει παρακολουθεί σε συνέχη βάση όλα τα εικονικά μηχανήματς της υποδομής, ώστε να εξασφαλίσει την ομαλή λειτουργία του. Πρέπει να είναι σε συνεχή επικοινωνία με το διαχειρηστή ροών εργασίας. Ο μηχανισμός ελέγχου πραγματοποιεί εποπτεία και έλεγχο σε δύο επίπεδα.

Αρχικά, πραγματοποιεί έλεγχο σχετικά με την υγεία του εικονικού μηχανήματος. Ανά τακτά χρονικά διαστήματα τα οποία περιγράφονται από ένα αρχείο, ο μηχανισμός ανακτά δεδομένα σχετικά με τις παραμέτρους συστήματος, όπως επεξεργαστικός φόρτος, διάθεσιμη μνήμη και αποθηκευτικός χώρος. Έπειτα ο μηχανισμός πρέπει να ελέγξει το αν κάθε μία από αυτές τις τιμές ξεπερνά κάποιο προκαθορισμένο όριο, το οποίο περγράφεται από το αρχείο που αναφέρθηκε παραπάνω. Υπάρχουν δύο όρια για κάθε τιμή. Αν μία τιμή ξεπεράσει το πρώτο όριο, τότε το μηχάνημα τίθεται σε κατάσταση προειδοποιητική, ενώ αν ξεπεράσει το δεύτερο τίθεται σε κατάσταση κρίσιμη. Ο ρόλος του μηχανισμού στις παραπάνω περιπτώσεις, είναι ανάλογα να με τη κατάσταση ¨συναγερμού¨ να ενημερώσει κατάλληλα το διαχειριστή ροών εργασίας. Ο διαχειριστής αναλαμβάνει στη προκειμένη περίπτωση να δημιουργήσει ένα καινούργιο εικονικό μηχάνημα, ώστε να κατανεμηθεί από τον εξισσοροπηστή το φορτίο πιο ομαλά στη συστάδα εικονικών μηχανημάτων.

Σε δεύτερο επίπεδο ο μηχανισμός πραγματοποιεί έλεγχο, σε ένα αρχείο του εξυπηρετητή Apache. Σύμφωνα με το αρχείο αυτό ο μηχανισμός μπορεί να συμπεράνει το πόσες αιτήσεις έφτασαν στον εξυπηρετητή κατά τη διάρκεια ενός προκαθορισμένου χρονικού παράθυρου. Αν οι αιτήσεις δεν ξεπερνούν ένα κατώτατο κατώφλι, τότε ο μηχανισμός συμπεραίνει ότι ο φόρτος του μηχανήματος είναι πολύ μικρός, και δεν υπάρχει ανάγκη να βρίσκεται σε λειτουργία. Συνεπώς επικοινωνεί με το διαχειρηστή ροών εργασίας, ο οποίος με τη σειρά του τερματίζει τη λειτουργία του εικονικού μηχανήματος.Ο σκοπός της εποπτείας σε επίπεδο αιτήσεων είναι η παρατήρηση της εξισσορόπησης που επιτυγχάνει ο LoadBalancer. Παρατηρούμε εδώ το βαθμό ευφυίας του μηχανισμού ελέγχου και εποπτείας, καθώς επιτυγχάνεται η ελαστικότητα, ως βασικό συστατικό ενός υπολογιστικού νέφους, σε μεγάλο βαθμό.

## 4.3. Τεχνολογίες και εργαλεία

Στο παρόν υποκεφάλαιο θα αναφέρουμε τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του λογισμικού κάθε επιμέρους αρχιτεκτονικού τμήματος για τη δημιουργία του μηχανισμού ελέγχου και εποπτείας. Δόθηκε έμφαση σε εργαλεία ανοικτού κώδικα.

Ένα από τα πιο βασικά στοιχεία του μηχανισμού είναι ότι είναι υπηρεσιοστρεφής. Τα τμήματα επικοινωνούν μεταξύ τους, για την ανταλλαγή δεδομένων και μηνυμάτων, υλοποιώντας RESTfulυπηρεσίες διαδικτύου. Παρακάτω θα αναφέρουμε όλες τις τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν.

***JavaEE 6***

Η Java Enterprise Edition[80]είναι η πλατφόρμα της Oracle για την ανάπτυξη εφαρμογών και υπηρεσιών διαδικτύου, και άλλες πολυεπίπεδες, επεκτάσιμες, αξιόπιστες και ασφαλείς εφαρμογές. Η Java EE είναι επέκταση της JavaSE. Η πλατφόρμα υιοθετεί μία σχεδίαση, κατά ένα μεγάλο βαθμό βασισμένη σε πολλαπλά τμήματα λογισμικού τα οποία τρέχουν σε ένα εξυπηρετητή εφαρμογών (applicationserver). Η εφαρμογές που υπηστηρίζονται από τη πλατφόρμα Java EE είναι υλοποιήμενες στη γλώσσα προγραμματισμού Java.

***Jboss Application Server***

Ο Jboss AS [33] αποτελεί ένα εξυπηρετητή εφαρμοργών για την υποστήριξη εφαρμογών JavaEE. Είναι γραμμένος στη γλώσσα προγραμματισμού Javaκαι γι'αυτό υποστηρίζεται από όλα τα περιβάλλοντα που έχουν εγκατεστημένη την πλατφόρμα Java.

***MySQL***

Μία ανοικτού κώδικα, πλατφόρμα για σχεσιακές βάσεις δεδομένων[39]. Στηρίζεται στη γλώσσα επερωτήσεων SQLκαι είναι πολύ δημοφιλής για εφαρμογές διαδικτύου.

***RESTeasy***

Η βιβλιοθήκη RESTeasy[41] αποτελεί την υλοποίηση για το πρότυπο RESTfulυπηρεσιών JAX-RSγια εφαρμογές διαδικτύου JavaEE. Το πακέτο RESTeasy προσφέρεται απευθείας από τον εξυπηρετητή εφαρμογών Jboss, και προσφέρει μία πληθώρα εργαλείων για την εύκολη υλοποίηση υπηρεσιοστρεφών εφαρμογών, καθώς και εφαρμογών πελατών (client) που καλούν διαδικτυακές RESTful υπηρεσίες.

***NetBeans***

Αποτελεί ένα IDE[40] για τη ανάπτυξη εφαρμογών σε Java, αλλά και σε άλλες γλώσσες προγραμματισμού. Είναι ένα πρόγραμμα ανοικτού κώδικα, με μία μεγάλη και ενεργή κοινότητα υποστήριξης. Το NetBeans είναι πολύ δημοφιλές στους προγραμματιστές, καθώς προσφέρει εργαλεία για την εύκολη ανάπτυξη διαδικτυακών υπηρεσιών SOAPκαι REST.

***Eclipse***

Το eclipse[31] είναι ένα ακόμα IDE για την ανάπτυξη εφαρμογών σε Java, αλλά και σε άλλες γλώσσες προγραμματισμού. Είναι και αυτό λογισμικού ανοικτού κώδικά και έχει μία μεγάλη κοινότητα υποστήριξης. Είναι πολύ δημοφιλές καθότι παρέχει μία πολύ μεγάλη πληθώρα εργαλειών (plugins), τα οποία ο χρήστης μπορεί να εγκατασταστήσει πολύ εύκολα, καθώς το Eclipseπαρέχει ένα πολύ φιλικό προς το χρήστη, οδηγό εγκατάστασης λογισμικού.

# 5. Το πείραμα Ομόσπονδης Υποδομής Νέφους

Το πείραμα Ομόσπονδης Υποδομής Νέφους είναι ένα πρόγραμμα χρηματοδοτούμενο από την Ευρωπαική Ένωση, το οποίο σχεδιάζει, κατασκευάζει και λειτουργεί μία υποδομή υπολογιστικού νέφους, αποτελούμενη από 6 διαφορετικές υποδομές ανά την Ευρώπη. Η υποδομή προσφέρει ετερογενείς πόρους υπολογιστικών νεφών, συμπεριλαμβανομένων υπολογιστικών (compute),αποθηκευτικών (storage) και δικτυακών (network).

Η αρχιτεκτονική του πειράματος Ομόσπονδης Υποδομής Νέφους έχει σχεδιαστεί για να υποστηρίζει ερευνητικά πειράματα σε εφαρμογές, υπηρεσίες και συστήματα. Παρέχει συγκεκριμένες λειτουργίες, όπως

- Διαχείρηση ελέγχου, σε επίπεδο υποδομής και virtual machine.
- Διαχείρηση πειραμάτων με τη χρήση αρχείων περιγραφής πειραμάτων (dedcriptors/XML).
- Ελαστικότητα.
- Διαχείρηση πόρων για την εγκατάσταση και τη λειτουργία εφαρμογών σε μία ομάδα διαφορετικών εικονοποιημένω μηχανημάτων.

## 5.2. Χαρακτηριστικά

Η υποδομή προσφέρει τις ακόλουθες λειτουργίεας και χαρακτηριστικά:

***Πολλαπλά εργαλεία διαχείρησης***

Για την δημιουργία πειραμάτων και πόρων, μπορούν να χρησιμοποιηθούν:

- Εντόλες πρωτοκόλλου HTTP, χρησιμοποιώντας έτοιμα εργαλία (cURL).
- Η δικτυακή διεπαφή του πειράματος. Πρόκειται για μία διαδικτυακή εφαρμογή, εύκολη στη χρήση. Η εφαρμογή δίνει την δυνατότητα στο χρήστη να δημιουργεί νέα πειράματα βήμα-βήμα, καθώς επίσης και να καθορίζει τις αρχικές παραμέτρους που αφορούν τους διαθέσιμους πόρους για το πείραμα.
- Ένα αρχείο περιγραφής πειράματος (Experimentdescriptor). Ο χρήστης μπορεί να δημιουργήσει ένα αρχείο, το οποίο περιγράφει το πείραμα, σε μορφή JSONή OVF. Η υποδομή διαθέτει έναν διαχειριστή πειραμάτων, ο οποίος μεταφράζει αυτό το αρχείο και δεσμεύει τους πόρους. Η λειτουργία αυτή επιτρέπει στο χρήστη να δημιουργεί πειράματα, χωρίς να έχει γνώση OCCI και HTTPμηνυμάτων.
- Ηυπηρεσία Restfully, η οποία προσφέρει μία διεπαφή υπηρεσιών REST, σε γλώσσα προγραμματισμού Ruby. Σκοπός της είναι να παρέχει ένα επίπεδο ανταλλαγής HTTPμηνυμάτων μεταξύ χρήστη και εξυπηρετητή, καλύπτοντας τις λεπτομέρειες του πρωτοκόλλου. Η υπηρεσία επιτρέπει στο χρήστη να δημιουργεί αρχεία (scripts), που μπορούν να αυτοματοποιήσουν την διαδικασία δημιουργίας και εγκατάστασης πειραμάτων.
- Εργαλία εντολών κονσόλας (CLItools). Πρόκειται για εργαλία τα οποία δίνουν την δυνατότητα στο χρήστη να επικοινωνεί με το διεπαφή της υποδομής, μέσω της παραδοσιακής γραμμής εντολών.

***Ετερογενείς υποδομές και πόροι***

Η πλατφόρμα του πειράματος Ομόσπονδης Υποδομής Νέφους περιλαμβάνει επιμέρους υποδομές υπολογιστικών νεφών, οι οποίες είναι γεωγραφικά κατανεμημένες ανά την Ευρώπη, προσφέροντας ετερογενείς πόρους. Οι υποδομές είναι η EPPCστο Ενδιμβούργο, η HPCellsστο Bristolτης Αγγλίας, η Inriaστη Rennesτης Γαλλίας, η HLRSστην Στουτγκάρδη της Γερμανίας, η IBBTVirtualWallστη Ghenttου Βελγίου, και η PSNCστο Poznantης Πολωνίας.

Κάθε περιοχή προσφέρει πόρους (compute,storage,network), οι οποίοι μπορούν να χρησιμοποιηθούν για την δημιουγία πειραμάτων, αλλά και εικονικών μηχανημάτων. Κάποιες υποδομές προσφέρουν πρόσθετες υπηρεσίες, όπως δυναμική ανάθεση πόρων κατ'εντόλη (on-requestresources, στην Inria) και προσομοίωση δικτύου (στο VirtualWall). Ο χρήστης, αφού πιστοποιήσει την ταυτότητα του, μπορεί να έχει πρόσβαση στους πόρους αυτούς μέσω SSH.

***Ανάθεση υπολογιστικών πόρων κατ'εντολή***

Η υποδομή Inria προσφέρει την δυνατότητα στο χρήστη να δεσμεύει πόρους την στιγμή που τους χρειάζεται. Πάρεχει μία μεγαλή ποσότητα πόρων (162 κόμβοι/1800 πυρήνες). Έτσι ο χρήστης έχει την ευελιξία να δημιουργεί μεγάλης κλίμακας πειράματα, όπως επίσης και να έχει περισσότερο έλεγχο στις παραμέτρους του πειράματος, καθώς μπορεί να επιλέξει σε ποιον φυσικό κόμβο να δεσμεύσει πόρους και να εκτελέσει το πείραμα του.

***Εικονικά μηχανήματα και τύποι στιγμιοτύπων***

Η πλατφόρμα του πειράματος Ομόσπονδης Υποδομής Νέφους προσφέρει πολλαπλά στιγμιότυπα για εικονικά μηχανήματα. Τα στιγμιότυπα βασίζονται στο linuxλειτουργικό σύστημα Debian Squeeze, τα οποία ποικίλλουν σε αποθηκευτικό χώρο. Ο αποθηκεύτικος χώρος μπορεί να επεκταθεί με την προσάρτηση επιπρόσθετου αποθηκευτικού χώρου (blockstorage), ο οποίος προσφέρεται ως ξεχωριστός τύπος πόρων.

Οι χρήστες μπορούν να δημιουργούν πολλαπλά εικονικά μηχανήματα, χρησιμοποιώντας τα στιγμιότυπα τύπου Debian, να εγκαταστήσουν και να τροποποιήσουν λογισμικό, και να τα αποθηκεύσουν. Η υποδομή, προς το παρόν, δεν προσφέρει τη δυνατότητα στο χρήστη να δημιουργήσει δικά του στιγμιότυπα και να ανεβάσει στην υποδομή του πειράματος, παρ'όλα αυτά έχει την δυνατότητα να αποθηκεύει μία κατάσταση (snapshot) ένος εικονικού μηχανήματος, αφού έχει εγκαταστήσει/τροποποιήσει λογισμικό, και να την χρησιμοποιήσει ως στιγμιότυπο για να δημιουργήσει καινούργια εικονικά μηχανήματα.

Όταν ο χρήστης δημιουργεί εικονικά μηχανήματα, ορίζει το τύπο στιγμιοτύπου, ο οποίος ποικίλλει σε αριθμό πυρήνων και μέγεθος μνήμης. Ο τύπος στιγμιοτύπου επίλεγεται είτε από μία λίστα (small,medium,large), είτε τον ορίζει ο χρήστης επιλέγοντας ο ίδιος τον αριθμό των πυρήνων και το μέγεθος της μνήμης.

***Αποθήκευση***

Όπως αναφέρθηκε και προηγουμένως, τα έτοιμα στιγμιότυπα παρέχονται με ένα συγκεκριμένο μέγεθος αποθηκευτικού χώρου. Αν ο χρήστης δεν ικανοποιείται από αυτό το μέγεθος, μπορεί να επεκτείνει τον αποθηκευτικό χώρο των εικονικών μηχανημάτων του πειραματός του, προσαρτώντας επιπλέον αποθηκευτικούς πόρους. Ο χρήστης μπορεί να καθορίσει το τύπο του αποθηκευτικού τμήματος που θέλει να δημιουργήσει (ext3,jfs,reiserfs), και φυσικά το μέγεθος του τμήματος. Είναι επίσης δυνατόν, να προσαρτηθεί το νέο τμήμα ως άμεση επέκταση του αποθηκευτικού χώρου του λειτουργικού συστήματος, του εικονικού μηχανήματος και όχι ως ξεχωριστή εγκατάσταση.

Εξ'ορισμού, τα τμήματα των αποθηκευτικών χώρων που δημιουργούνται αντιγράφονται όταν προσαρτούνται σε εικονικά μηχανήματα, και αντιστοίχως καταστρέφονται όταν τα εικονικά μηχανήματα τερματίζουν την λειτουργία τους. Ένω άλλοι πόροι (υπολογιστικοί, δικτύου) είναι δυνατό να δημιουργηθούν μόνο κατά την διάρκεια ενός πειράματος, οι αποθηκευτικοί πόροι μπορούν να δημιουργούνται ξεχωριστά και να είναι διαθέσιμοι και όταν ένα πείραμα λήξει. Επιπρόσθετα, είναι δυνατόν τα δεδομένα που αποθηκεύονται σε ένα τμήμα αποθηκευτικού χώρου που έχει δημιουργήσει ο

χρήστης, κατά τη διάρκεια λειτουργίας ενός εικονικού μηχανήματος, να εγγράφονται μόνιμα στο τμήμα αυτό. Έτσι ο χρήστης μπορεί να χρησιμοποιήσει το τμήμα αυτό, με τα δεδομένα, και να το προσαρτήσει σε νέα εικονικά μηχανήματα που θα δημιουργήσει.

Τέλος, ένας άλλος ειδικός τύπος αποθηκευτικού τμήματος είναι ο διαμοιρασμένος αποθηκευτικός χώρος. Προς το παρόν, μπορούν μόνο να δημιουργηθούν στην υποδομή IBBT. Αντίθετα με τους άλλους τύπους αποθηκευτικών πόρων, οι διαμοιρασμένοι αποθηκευτικοί πόροι, είναι προσπελάσιμοι από πολλαπλά εικονικά μηχανήματα την ίδια στιγμή. Εάν ένα τμήμα αποθηκευτικού χώρου, είναι διαμοιρασμένο, και έχει δημιουργήθει εκτός του εύρους ενός πειράματος, τότε είναι δυνατόν, αυτό το τμήμα να είναι προσπελάσιμο από εικονικά μηχανήματα, από διαφορετικά πειράματα.

***Δικτυακοί πόροι***

Οι επιμέρους υποδομές που αποτελούν το πείραμα Ομόσπονδης Υποδομής Νέφους, διασυνδέονται μέσω του δημόσιου διαδικτύου (publicInternet), όμως μόνο μέρικοι πάροχοι υποδομών έχουν την δυνατότητα να παρέχουν δημόσιες IPv4 διευθύνσεις. Ως εκ τούτου, η υποδομήλειτουργεί ένα ιδιωτικό εικονικό δίκτυο (VPN) για να δρομολογεί την κίνηση μεταξύ των υποδομών.

***Διαχείρηση ελέγχου και εποπτεία***

Η υποδομήπαρέχει εκτεταμένη πληροφορία, σχετικά με τη κατάσταση των εικονικών πόρων σε διάφορα πειράματα, καθώς επίσης και για την κατάσταση των φυσικών κόμβων, στους οποίους εγκαθίστανται τα εικονικά μηχανήματα. Αυτό είναι ένα μοναδικό χαρακτηριστικό του πειράματος Ομόσπονδης Υποδομής Νέφους, που επιτρέπει στο χρήστη να συσχετίζει παρατηρήσεις που αφόρουν εικονικά μηχανήματα, με παρατηρήσεις και συμβάντα που συμβαίνουν σε φυσικούς κόμβους.

Η πλατφόρμα παρακολούθησης που προσφέρεται από την υποδομή, βασίζεται στην υπηρεσία ελέγχου ανοικτού κώδικα Zabbix. Το λογισμικό αποτελείται από δύο κύρια προγράμματα. Το πρόγραμμα ZabbixServerκαι το πρόγραμμα Zabbix Agent. Το πρόγραμμα Zabbix Server, αναφέρεται και ως συγκεντρωτής (Aggregator), και εγκαθίσταται σε ξεχωριστό εικονικό μηχάνημα, ενώ οι ¨πελάτες¨(agents), εγκαθίστανται σε κάθε εικονικό μηχάνημα που δημιουργεί ο χρήστης κατά την διάρκεια ενός πειράματος. Ο συγκεντρωτής αναλαμβάνει να συλλέξει πληροφορίες που στέλνει κάθε εφαρμογή πελάτη, δινόντας μία ενοποιημένη εικόνα σχετικά με την κατάσταση των πόρων ενός πειράματος. Εκτενής αναφόρα στην πλατφόρμα και την αρχιτεκτονική ελέγχου και εποπτείας θα γίνει στο επόμενο υποκεφάλαιο.

***Ελαστικότητα***

Ως ελαστικότητα αναφέρεται η δυνατότητα να προσθέτονται και να αφαιρούνται δυναμικά πόροι, σύμφωνα με το φόρτο και τις ανάγκες ενός εικονικού μηχανήματος ή πειράματος. Η ενωποιημένη υποδομή παρέχει ένα ξεχωριστό εικονικό μηχάνημα με μία μηχανή Ελαστικότητας βασισμένη στο HAProxy και το Kamailio.

Η «ελαστικότητα ως υπηρεσία» (EaaS) στο πείραμα, χρησιμοποιεί την πληροφορία που συλλέγει από τα εικονικά μηχανήματα ο ZabbixAggregator, και δυναμικά προσθέτει ή αφαιρεί πόρους, σύμφωνα με μία ομάδα κανόνων ελαστικότητας, που ορίζονται από το χρήστη.

***Μηχανισμός ειδοποιήσεων***

Κατά την εκτέλεση του πειράματος, και ιδιαίτερα αν είναι ενεργοποιημένος ο μηχανισμός της ελαστικότητας, είναι πολύ σημαντικό για το χρήστη να μπορεί να επιβλέπει αλλαγές και συμβάντα που συμβαίνουν στο πείραμα. Η Ομόσπονδη Υποδομή Νέφους προσφέρει τη δυνατότητα στους πελάτες να λαμβάνουν ειδοποιήσεις όταν η κατάσταση του πειράματος ή των πόρων του πειράματος αλλάζει, όταν αυτοί δημιουργούνται, ανανεώνονται, ή καταστρέφονται. Οι αλλαγές κατάστασης είναι διαθέσιμες ως συμβάντα σε μία ουρά μηνυμάτων στην υποδομή, η οποία χρησιμοποιεί το RabbitMQ.

Οι καταστάσεις του πειράματος είναι σύμφωνες με το κύκλο ζωής του πειράματος, και οι καταστάσεις των πόρων είναι σύμφωνες με τις δηλωμένες καταστάσεις OCCI.

***Πλαισιοποίηση***

Τα στοιχεία πλαισιοποίησης του OCCI μπορούν να χρησιμοποιηθούν για να μεταφέρουν τιμές αρχικοποίησης για ένα πείραμα, σε μορφή από ζευγών τιμών-κλειδία. Αυτό χρησιμοποιείται από τις υποδομές του ολικού πειράματοςκαι το πλαίσιο λειτουργίας του, αλλά είναι επίσης διαθέσιμο και στους χρήστες για να ορίζουν, π.χ:

- Την IP διεύθυνση του εικονικού μηχανήματος όπου είναι εγκατεστημένος ο ZabbixAggregator.
- Οποιεσδήποτε παραμέτρους ελέγχου και εποπτείας.
- Κανόνες ελαστικότητας, εάν αυτή χρησιμοποιείται.
- Προγράμματα τα οποία εκτελούνται μετά την εγκατάσταση εικονικού μηχανήματος.
- Πρόσθετα κλειδιά SSH, για να επιτρέπεται η πρόσβαση σε ένα πείραμα από πολλαπλούς χρήστες.

Η πλαισιοποίηση είναι γενικής μορφής, έτσι ώστε να όριζονται οποιεσδήποτε τιμές κλειδιά. Οι παράμετροι πλαισιοποίησης είναι γραμμένες σε ένα ξεχωριστό φάκελο, ώστε να χρησιμοποιούνται τοπικά από κάθε εικονικό μηχάνημα.

## 5.3. Μηχανισμός ελέγχου και εποπτείας στο πείραμα Ομόσπονδης Υποδομής Νέφους

Η πλατφόρμα ελέγχου που προσφέρεται από το πείραμα, βασίζεται στην παρακολούθηση τη βοήθεια του λογισμικού Zabbix, με την εξαίρεση των timestamps. Το λογισμικό αποτελείται από δύο κύρια συστατικά στοιχεία: τον Zabbixserver και ZabbixAgent. Ο server αναφέρεται ως «συγκεντρωτής», ο οποίοςεγκαθίσταται σε ξεχψριστό εικονικό μηχάνημα, ενώ οι agentsείναι εγκατεστημένοι στα ήδη εγκατεστημένα εικονικά μηχανήματα που χρησιμοποιούνται για το πείραμα. Το συγκεντρωτής συλλέγει πληροφορίες παρακολούθησης που αποθηκεύουνοι agents σε μια βάση δεδομένων. Η βάση δεδομένων μπορεί να αποθηκευτεί είτε μέσα στο εικονικό μηχάνημα που λειτουργεί ο συγκεντρωτής, είτε σε κάποιο ξεχωριστό εικονικό μηχάνημα, με το οποίο επικοινωνεί ο συγκεντρωτής.

Το σύστημα παρακολούθησης του πειράματος Ομόσπονδης Υποδομής Νέφους επιτρέπει στους χρήστες την αποθήκευση των δεδομένων ελέγχου και εποπτείας με ευέλικτο τρόπο. Το μέγεθος του τμήματος αποθήκευσης δεδομένων μπορεί να προσφερθείon-demand, και ως εκ τούτου, ο χρήστης μπορεί να πάρει το επιθυμητό μέγεθος αποθήκευσης. Επιπλέον, ανάλογα με τη ρύθμιση του πειράματος, η παρακολούθηση των δεδομένων δεν είναι διαθέσιμη μόνο κατά τη διάρκει ενός πειράματος, αλλά μπορεί ακόμη και να διατηρηθεί μόνιμα μετά την διαγραφή ή τη λήξη του.

***Εποπτεία εικονικών μηχανημάτων***

Πολλαπλές παράμετροι είναι δυνατόν να μετρηθούν κατά την παρακολούθηση των εικονικών μηχανημάτων. Είναι πάνω από 100 σε αριθμό και μπορούν να ενεργοποιούνται και να απενεργοποιούνται, ανάλογα με τις απαιτήσεις του χρήστη. Μερικές από αυτές είναι:

- Ολική μνήμη συστήματος.
- Χρησιμοποιούμενη μνήμη.
- Διαθέσιμη μνήμη.
- Ολική εφεδρική μνήμη.
- Δεσμευμένη εφεδρική μνήμη.
- Διαθέσιμη εφεδρική μνήμη.
- Ολικός αποθηκευτικός χώρος.
- Δεσμευμένος αποθηκευτικός χώρος.
- Διαθέσιμος αποθηκευτικός χώρος.
- Φόρτος επεξεργαστή.
- Ποσοστό χρησιμοποίησης επεξεργαστή.
- Αριθμός πυρήνων.
- Παράμετροι δικτύου (εισερχόμενη και εξερχόμενη κίνηση).
- Παράμετροι σχετικές με το λειτουργικό σύστημα.
- Παράμετροι σχετικές με διεργασίες του λειτουργικού συστήματος.
- Παράμετροι σχετικές με υπηρεσίες (FTP/Email/SSH).

***Εποπτεία επιπέδου εφαρμογής***

Η παρακολούθηση εφαρμογών, ελέγχει παραμέτρους οι οποίες παρέχουν πληροφορίες σχετικά με τη κατάσταση μίας συγκεκριμένης εφαρμογής, των επιδόσεων της καθώς και άλλου τύπου πληροφορίες. Αυτές οι παράμετροι δεν παρέχονται από την υπηρεσία Zabbix, και οι χρήστες πρέπει να τις ορίσουν στους agentsαλλά και στον συγκεντρωτή.

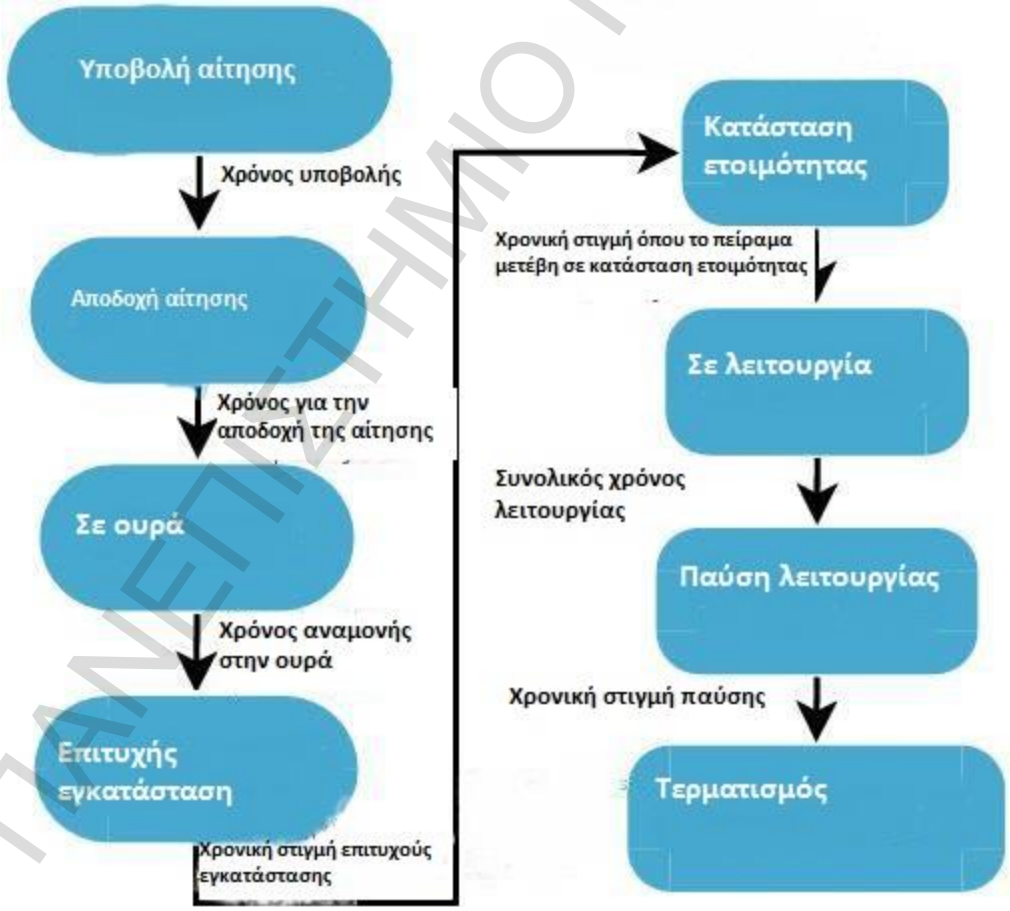
***Εποπτεία επιπέδου υποδομής***

Η πλατφόρμα Ομόσπονδης Υποδομής Νέφους παρέχει στους χρήστες τη δυνατότητα να συλλέγουν δεδομένα ελέγχου σχετικά με τους φυσικούς κόμβους των επιμέρους υποδομών, στους οποίους είναι εγκατεστημένα τα εικονικά μηχανήματα. Αυτή η υπηρεσία αναφέρεται ως εποπτεία υποδομής (Infrastracture monitoring). Οι κύριες παράμετροι ελέγχου υποδομής είναι:

- Φόρτος επεξεργαστή.
- Συνολική και διαθέσιμη μνήμη συστήματος.
- Διαθέσιμη εφεδρική μνήμη.
- Αριθμός εικονικών μηχανημάτων σε έναν φυσικό κόμβο.
- Διεργασίες εισόδου/εξόδου σκληρών δίσκων (diskI/O).
- Διεργασίες εγγραφής/ανάγνωσης σκληρών δίσκων (diskread/write).
- Εισερχόμενη και εξερχόμενη κίνηση στις διεπαφές δικτύου.

***Καταστάσεις συμβάντων***

Έχουν οριστεί οι ακόλουθες κατάστασεις πειράματος:




Εικόνα 20: Κατάστασεις πειράματος

- **Κατάσταση υποβολής αιτήματος (REUESTED),**όταν δηλαδή μία αίτηση για δημιουργία ενός πειράματος έχει φτάσει στο υπεύθυνο μεσάζοντα για τα πειράματα.
- **Κατάσταση αποδοχής αιτήματος (ACCEPTED).** Η μεσάζουσα υπηρεσία έχει αναλύσει την αίτηση, δεν έχει εντοπίσει σφάλματα και η αίτηση γίνεται αποδεκτή.
- **Κατάσταση αναμονής σε ουρά (QUEUED).**Το αίτημα περιμένει σε ουρά για να ανατεθεί σε ένα διαχειρηστή πειράματος που θα αναλάβει να το εγκαταστήσει.
- **Κατάσταση επιτυχούς εγκατάστασης (DEPLOYED),** όπου πια το πείραμα έχει δημιουργηθεί επιτυχώς από το διαχειρηστή πειράματος.
- **Κατάσταση ετοιμότητας (READY).** Το πείραμα περίμενει να μεταβεί σε κατάσταση λειτουργίας.
- **Κατάσταση λειτουργίας (RUNNING).**Το πείραμα έχει εκκινήσει τη λειτουργία του.
- **Κατάσταση παύσης (STOPPED).** Το πείραμα έχει σταμάτησει τη λειτουργία. Σε αυτή τη κατάσταση καμία ενέργεια που γίνεται στο πείραμα δεν πραγματοποιείται, εάν το πείραμα δεν επιστρέψει σε κατάσταση λειτουργίας.
- **Κατάσταση τερματισμού (TERMINATED).**Ο διαχειρηστής πειράματος έχει τερματίσει τη λειτουργία του πειράματος οριστικά και το διαγράφει από τη λίστα των ενεργών πειραμάτων.

Κατά το κύκλο ζωής ενός πειράματος, ο διαχειρηστής εποπτείας μετράει τους χρόνους που χρειάζεται για να μεταβεί το πείραμα από μία κατάσταση σε μία επόμενη, καθώς και χρόνους που έχουν να κάνουν με τη λειτουργία του πείραματος, όπως ο συνολικός χρόνος λειτουργίας του πειράματος ή ο συνολικός χρόνος ζωής του πειράματος, δηλαδή από τη κατάσταση READYμέχρι τη κατάσταση TERMINATED. Αξίζει επίσης να σημειωθούν τα ακόλουθα:

- Οι καταστάσεις REQUESTED, ACCEPTED, QUEUED, DEPLOYING και DEPLOYED υπάρχουν μόνο για διαχεριζόμενα πειράματα, που σημαίνει ότι προσκομίζονται από τον Διαχειρηστή πειράματος.
- Η κατάσταση READYμπορεί να υπολογιστεί καταγράφοντας το χρόνο όπου το τελευταίο εικονικό μηχάνημα ενός πειράματος μεταβαίνει σε κατάσταση RUNNING.

Προσοχή χρειάζεται στην περίπτωση εικονικών μηχανημάτων που χρησιμοποιούν την υπηρεσία της ελαστικότητας. Οι χρήστες θα πρέπει να αποφασίζουν το εάν και το πως θα επηρεάσουν τον υπολογισμό της κατάστασης READY.

Στην νέα έκδοση της πλατφόρμας Ομόσπονδης Υποδομής Νέφους, κάθε τμήμα ενός πειράματος αναμένεται να καταγράφει τις αλλαγές στην κατάσταση του. Αυτά τα αρχεία είναι διαθέσιμα από τα μηχανήματα στα οποία είναι εγκατεστημένο το κάθε τμήμα του πειράματος, είτε αυτό είναι εικονικό μηχάνημα, είτε αποθηκευτικό τμήμα, είτε δικτυακός πόρος. Μία εξαίρεση αποτελεί η περίπτωση του διαχειρηστή πειράματος, ο οποίος καταγράφει τις αλλαγές κατάστασης και τους χρόνους μεταξύ των πειραμάτων σε ένα αρχείο για κάθε πείραμα. Κατά την επιτυχή εγκατάσταση τμημάτων στη διάρκεια ενός πειράματος, ο διαχειρηστής αυτόματα ζητά την μετάβαση του πειράματος σε κατάσταση RUNNING.

## 5.4. Περιγραφή αρχιτεκτονικής της πλατφόρμας Ομόσπονδης Υποδομής Νέφους

Ο διαχειρηστής ροής εργασιών είναι σε θέση να λάμβάνει γεγονότα που παράγει ο αποτιμητής. Έχει αναπτυχθεί στην πλατφόρμα Drools, ώστε να υπάρχει η δυνατότητα διαχείρησης κανόνων. Οδιαχειρηστήςροήςεργασιώνρυθμίζεται και στη συνέχεια τα εικονικά μηχανήματα καλούν τις υπηρεσίες που βασίζονται σε ένα αρχείο περιγραφής ροής εργασιών, το οποίο εκτός από αυτή τη πληροφορία, περιέχει πληροφορία σχετικά με το ποια βήματα πρέπει να ακολουθηθούν σε κάθε πιθανό συμβάν. Τα βήματα αυτά περιλαμβάνουν κλήσεις άλλων υπηρεσιών ή αλλαγές στη ρύθμιση ήδη ενεργών υπηρεσιών. Οσυντάκτης ροής εργασιών είναι σε θέση να αντιδρά σε πιθανά λάθη, με τη βοήθεια συγκεκριμένων βημάτων χειρισμού λαθών, τα οποία ορίζονται στη περιγραφή ροής εργασιών. Αυτά μπορεί να είναι αρκετά απλά, όπως επανάληψη ή αποτυχία, ή πιο περίπλοκα, που μπορεί να ορίζονται από τον ίδιο τον προγραμματιστή και μπορούν να περιέχουν κλήσεις σε καινούργιες υπηρεσίες ή την επαναρύθμιση ήδη υπάρχοντων. Είναι πολύ σημαντικό να διαχωρίζονται οι διαφορές μεταξύ συμβάντων και σφαλμάτων. Το συμβάν είναι μία γνωστή και αναμενόμενη κατάσταση του συστήματος, στην οποία ο διαχειρηστής ροής εργασιών μπορεί να αντιδράσει ακολουθώντας κάποια προκαθορισμένα βήματα. Τα σφάλματα από την άλλη πλευρά, είναι άγνωστες και απρόβλεπτες καταστάσεις. Σε τέτοιες περιπτώσεις, πρέπει να πραγματοποιηθούν γενικού τύπου λειτουργίες, όπως η αποθήκευσης της κατάστασης ενός εικονικού μηχανήματος. Υπάρχει η δυνατότητα να οριστούν συγκεκριμένες δομές στη περιγραφή ροής εργασιών, καθώς επίσης και ειδικές περιπτώσεις, για περιστατικά επανάληψης ή αποτυχίας. Η επανάληψη π.χ μπορεί να προστεθεί ως μία εύκολη λύση για να αντιμετωπίζονται προβλήματα που εμφανίζονται σποραδικά.

Ο μηχανισμός διαχείρησης ελέγχου, είναι κάτι παραπάνω από ένα απλό εργαλείο μέτρησης, καθώς παρέχει συγκεντρωτικές λειτουργίες στη συλλογή δεδομένων της υποδομής. Ο μηχανισμός είναι σε θέση να συγκεντρώνει πληροφορία σε επίπεδο εφαρμογής, συστάδας εικονικών μηχανημάτων και υπολογιστικού νέφους. Ο λόγος που υπάρχει αυτή η πολυεπίπεδη προσέγγιση είναι ότι όσο περισσότερα επίπεδα εισάγονται, τόσο μεγαλύτερο είναι το κόστος επικοινωνίας. Συνεπώς το σύστημα διαχωρίζει διαφορετικά επίπεδα συγκέντρωσης πληροφορίας. Αναλυτικότερα, μπορεί να οριστούν ξεχωριστοί κανόνες για κάθε εφαρμογή, συστάδα εικονικών μηχανημάτων ή υποδομής υπολογιστικού νέφους συνδυασμένοι με ένα πλαίσιο χρόνου. Παρ'όλα αυτά, μπορεί να γίνει μερική συγκέντρωση πληροφορίας σε χαμηλότερα επίπεδα. Για κάθε εικονικό μηχάνημα υπάρχει ένα στιγμιότυπο του μηχανισμού ελέγχου, και είναι υπεύθυνο για τη συλλογή συμβάντων για το μηχάνημα. Κάθε στιγμιότυπο περιλαμβάνει ένα συλλέκτη πληροφορίας για όλες της εφαρμογές που είναι εγκατεστημένες στο μηχάνημα, ένα μηχανισμό κανόνων που αλλάζει τα χρονικά διαστήματα μεταξύ ελέγχων, ανάλογα με τη κρισιμότητα των συμβάντων, καθώς και έναν αποστολέα που στέλνει τη πληροφορία στον διαχειριστή ελέγχου. Τα τμήματα λογισμικού του στιγμιότυπου είναι ανεπτυγμένα στη γλώσσα προγραμματισμού Java. Ξεχωριστό τμήμα συλλέγει πληροφορίες σχετικά με τη κατάσταση του εικονικού μηχανήματος, και με τη σείρα του στέλνει τη πληροφορία αυτή στο τοπικό υποσύστημα παρακολούθησης. Η τμηματοποίηση του λογισμικού παρακολούθησης σε κάθε μηχάνημα επιτρέπει σε άλλα συστήματα μέτρησης να εγκατασταθούν και να χρησιμοποιηθούν ως πηγές πληροφορίας.

Όσον αφορά τον αποτιμητή, τα συμβάντα ακολουθούν ένα απλό σχήμα περιλαμβάνοντας μερικά στατικά πεδία, καθώς επίσης και πεδία τα οποία σχετίζονται με το συμβάν που δημιουργήθηκε. Οι κανόνες παρέχονται ως είσοδος στο μηχανισμό ελέγχου και παρακολούθησης, ώστε αυτός να καθορίσει το πως θα αντιμετωπίζεται κάθε συμβάν. Περιλαμβάνουν 3 κύριους τομείς:

- Καθορισμός θέματος κανόνα. Σε αυτό το σημείο ορίζεται ένα όνομα όπου θα λειτουργεί ως μοναδικό αναγνωριστικό για να καταγραφεί στην ουρά κανόνων. Επιπροσθέτως, στο ίδιο τμήμα ορίζονται το επίπεδο συγκέντρωσης και η διάρκεια του.
- Το τμήμα του κανόνα που παρέχει βασικές λογικές παραμέτρους στα πεδία των συμβάντων. Αν ένα ορισμένο πεδίο ενός κανόνα δεν υπάρχει σε ένα συμβάν, τότε αυτό απορρίπτεται.

- Το τμήμα που θα εκτελεί συγκεκριμένες λειτουργίες ανάλογα με τα συμβάντα και θα παραθέτει τα αποτελέσματα σε μία ουρά αποτελεσμάτων.

# 6. Αποτελέσματα πειραματικών διαδικασιών

Στο παρόν κεφάλαιο θα παρατεθούν τα αποτελέσματα δύο πειραματικών διαδικασιών, τα οποία διεξήχθησαν στη πλατφόρμα Ομόσπονδης Υποδομής Νέφους. Το χαρακτηριστικό του πρώτου πειράματος είναι ότι διενεργείται σε χρησιμοποιώντας πόρους μία φυσικής υποδομής (testbed, site), ενώ το δεύτερο χρησιμοποιεί πόρους από δύο υποδομές (cross-siteexpreriment).

Η φύση της πειραματικής διαδικασίας, όπως αυτή περιγράφηκε στο κεφ. 4.2, είναι τέτοια που δεν καταναλώνει αποθηκευτικό χώρο, οπότε η επόπτεια αποθηκευτικού χώρου, παρότι διενεργείται, δεν έχει καμία πρακτική αξία για τα συγκεκριμένα πειράματα.Επίσης πάλι λόγω της φύσης της πειραματικής διαδικασίας, δόθηκε μεγάλη έμφαση στη μνήμη των μηχανημάτων nodes, καθώς αυτή επηρεάζεται περισσότερο. Και στα δύο πειράματα, οι κανόνες εποπτείας είναι αυστηροί όσον αφορά τη μέτρηση μνήμης και επεξεργαστή. Για το $1^{ο}$ πείραμα το πρώτο όριο για τη μνήμη έχει οριστεί στο 42%, ενώ το όριο για το $2^{ο}$ στάδιο έχει οριστεί στο 48%. Για το $2^{ο}$ πείραμα και πάλι για τη μνήμη, έχουμε θέσει το όριο για το πρώτο στάδιο συναγερμού (yellowevent) στο 42%, ενώ για το δεύτερο στάδιο συναγερμού (redevent) το όριο είναι στο 46%. Αντίστοιχα και τα 2 πειράματα, για τον επεξεργαστή, το όριο για το πρώτο στάδιο συναγερμού είναι 40%, ενώ το όριο για το δεύτερο στάδιο είναι 60%. Επίσης γίνεται εποπτεία των αιτήσεων που λαμβάνει κάθε μηχάνημα. Ο αριθμός των αιτήσεων που εξυπηρετεί κάθε μηχάνημα μας δείχνει, το πόσο καλά λειτουργεί ο LoadBalancer, αλλά επίσης και το πόσο καλά λειτουργεί ο μηχανισμός εποπτείας, όχι μόνο σε επίπεδο εφαρμογής, αλλά και σε επίπεδο υποδομής και μηχανημάτων. Η αυστηρότητα των κανόνων έχει σκόπο να προσομοιώσει συνθήκες όπου απαιτείται υψηλό QoS, όταν δηλαδή απαιτείται από την υποδομή να δεσμεύει τους απαραίτητους πόρους για να εγγυηθεί την ομαλή λειτουργία της εφαρμογής.

Τέλος, γίνεται συνολική και συνοπτική αξιολόγηση του μηχανησμού εποπτείας και βγαίνουν συμπεράσματα σχετικά με τη λειτουργία του, και για το κατά πόσο επιτυγχάνεται μία σωστή εποπτείας και διαχείρηση των πόρων μίας υποδομής υπολογιστικού νέφους.

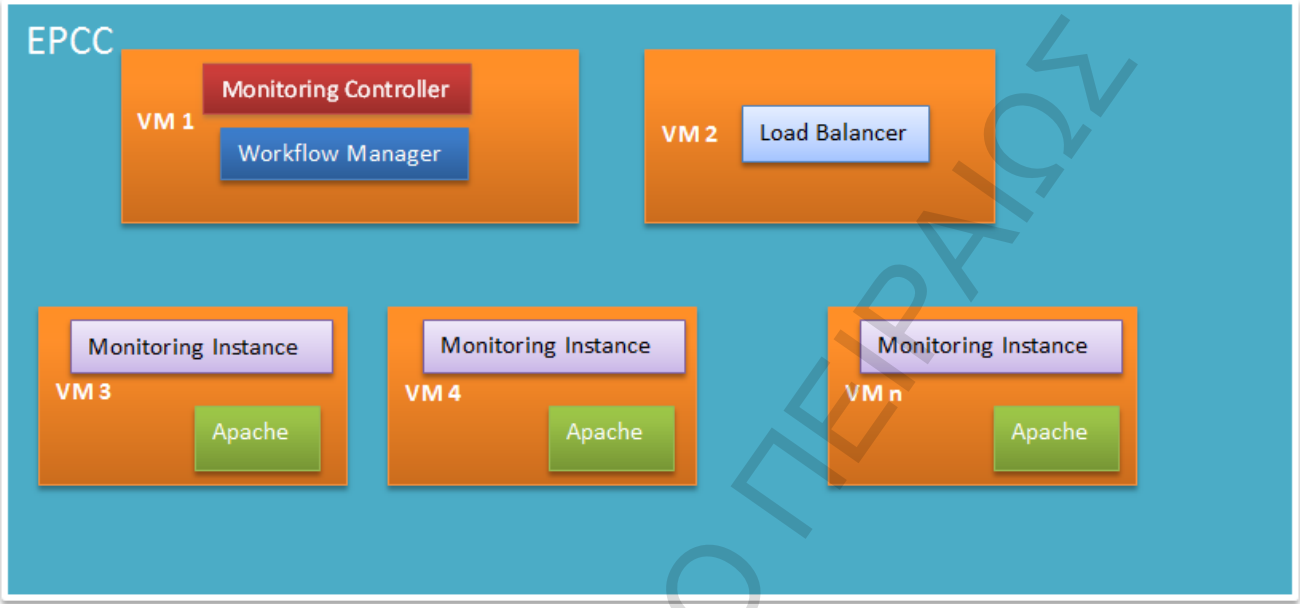
## 6.1. Πείραμα 1

Όπως αναφέρθηκε και παραπάνω για το συγκεκριμένο πείραμα δεσμεύτηκαν πόροι από μία κύρια φυσική υποδομή της πλατφόρμας Υπολογιστικού Νέφους. Το πείραμα εγκαταστάθηκε και παραμετροποιήθηκε στη υποδομή EPCC [81]. Όλες δηλαδή οι κύριες οντότητες των μηχανισμών εποπτείας και διαχείρησης ροών εργασίας, καθώς και τα στιγμιότυπα (nodes), που χρησιμοποιήθηκαν για την επεξεργασία των αιτήσεων, ήταν εγκατεστήμενα σε αυτή την υποδομή. Οι κύριες οντότητες του πειράματος είναι οι εξής:

- Ο **Monitoring Controller**, που είναι υπεύθυνος για την εποπτεία και συλλογή πληροφοριών σχετικά με τη κατάσταση όλων των μηχανημάτων (nodes), που είναι ενεργά και συμμετέχουν στο πείραμα.
- Ο **Workflow Manager**, που είναι υπεύθυνος για τις ροές εργασίας του πειράματος και αναλαμβάνει να δευσμεύσει και να εκκινήσει υπολογιστικούς πόρους, ανάλογα με τις ανάγκες του πειράματος.
- Ο **Load Balancer**, ο οποίος δέχεται όλες τις αιτήσεις και τις διαμοιράζει στα μηχανήματα nodes.
- Τα μηχανήματα **nodes**, τα οποία επεξεργάζονται τις αιτήσεις που τους αναθέτει ο Load Balancer. Τα εικονικά μηχανήματα nodes είναι εκείνα που βρίσκονται συνεχώς υπό την εποπτεία του Monitoring Controller.
- Τέλος, η διαδικτυακή εφαρμογή **Collector**, η οποία προσφέρει μία φιλική προς το χρήστη διεπαφή, για τη παρουσίαση των αποτελεσμάτων του μηχανισμού εποπτείας για όλα τα εικονικά μηχανήματα.

### 6.1.1. Τοπολογία πειράματος

Στη εικόνα 22 φαίνεται η τοπολογία του πειράματος. Όλες οι υπηρεσίες είναι εγκατεστημένες στην υπηδομή EPCC.




Εικόνα 21: Τοπολογία πειράματος 1

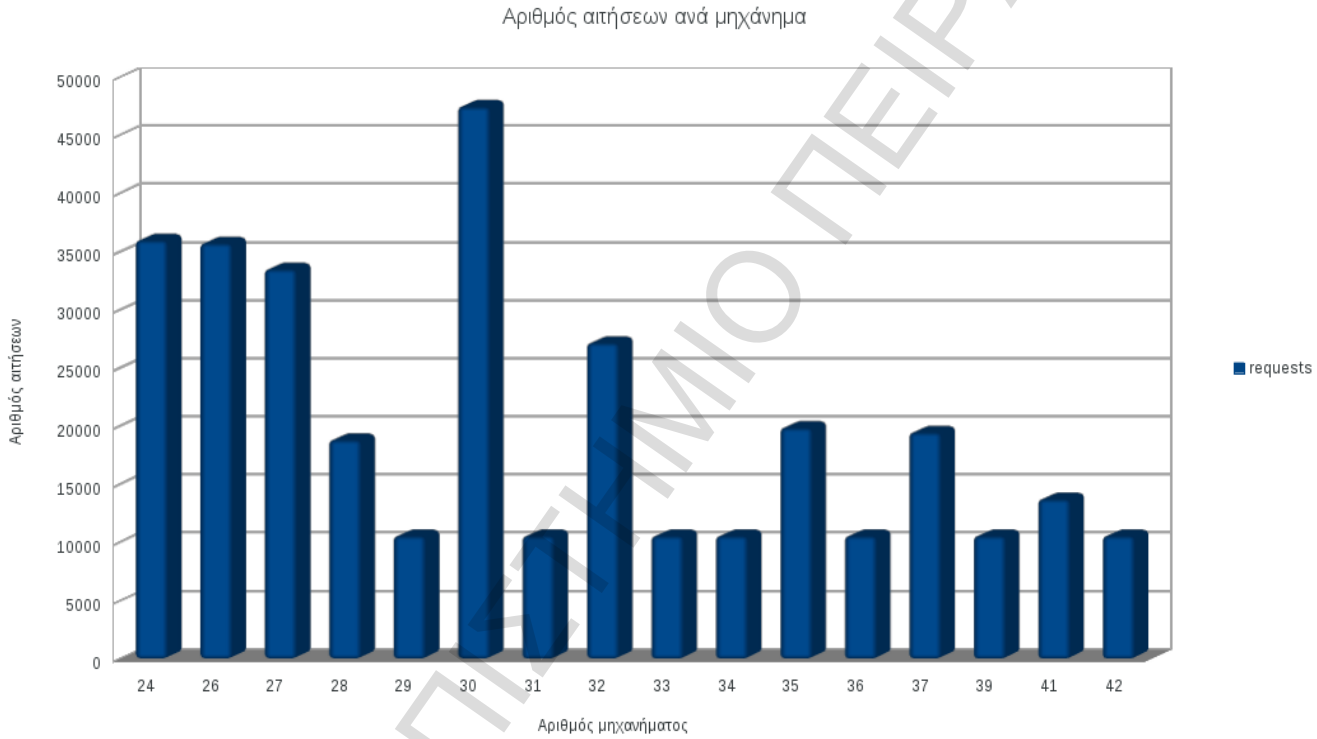
Το πρώτο VM χρησιμοποιήθηκε για να εγκατασταθούν οι βασικές υπηρεσίες των δύο μηχανισμών. Όπως είναι φυσικό, αυτό χρειάστηκε ένα περιβάλλον υψηλών προδιαγραφών. Για το λόγο αυτό χρησιμοποιήθηκε ένα εικονικό μηχάνημα με 2 επεξεργαστικούς πυρήνες και 4096MB μνήμης.

Για τα μηχάνηματα που φιλοξένουσαν την υπηρεσία του LoadBalancer, αλλά και για τα μηχανήματα nodes, χρησιμοποιήθηκαν στιγμιότυπα μικρότερων δυνατοτήτων, αφού η λειτουργία των μηχανημάτων δεν ήταν υψηλή σε απαιτήσεις πόρων. Για αυτό το λόγο χρησιμοποιήθηκαν μηχανήματος ενός (1) επεξεργαστικού πυρήνα και 1024MB μνήμης RAM.

### 6.1.2. Αποτελέσματα πειράματος

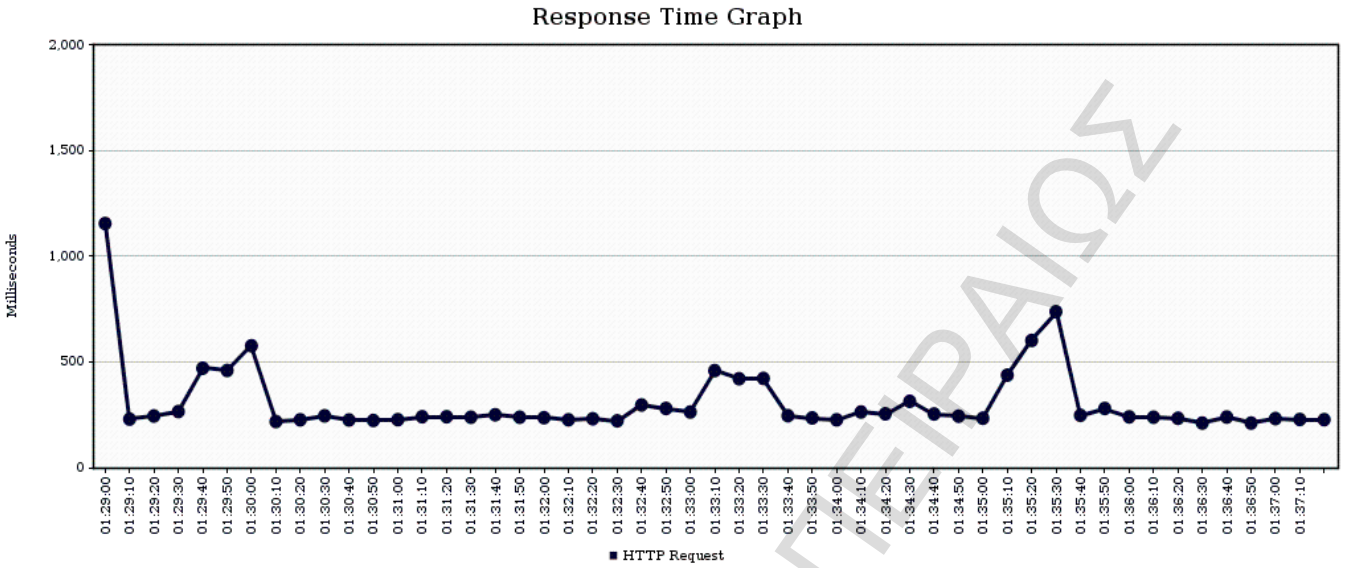
Για τις ανάγκες του πειράματος, η εφαρμογή προσομοίωσης HTTPαιτήσεων JMeter ρυθμίστηκε έτσι, ώστε να παράγει περίπου 140-700 HTTPαιτήσεις κάθε δευτερόλεπτο. Η εφαρμογή έστελνε τις αιτήσεις στη υπηρεσία του LoadBalancer, ο οποίος με τη σειρά του τις διαμοίραζε τα ενεργά εικονικά μηχανήματα για επεξεργασία.

Το πείραμα διήρκησε περίπου 40-45 λεπτά, και στο διάστημα αυτό δευσμέτηκαν πόροι για συνολικά 16 εικονικά μηχανήματα, το οποίο ισοδυναμεί σε 16 πυρήνες, 16384 MB μνήμης και 32 GBαποθηκευτικού χώρου. Στη παρακάτω εικόνα (εικόνα 23) βλέπουμε το συνολικό αριθμό αιτήσεων που επεξεργάστηκε κάθε κόμβος (node).



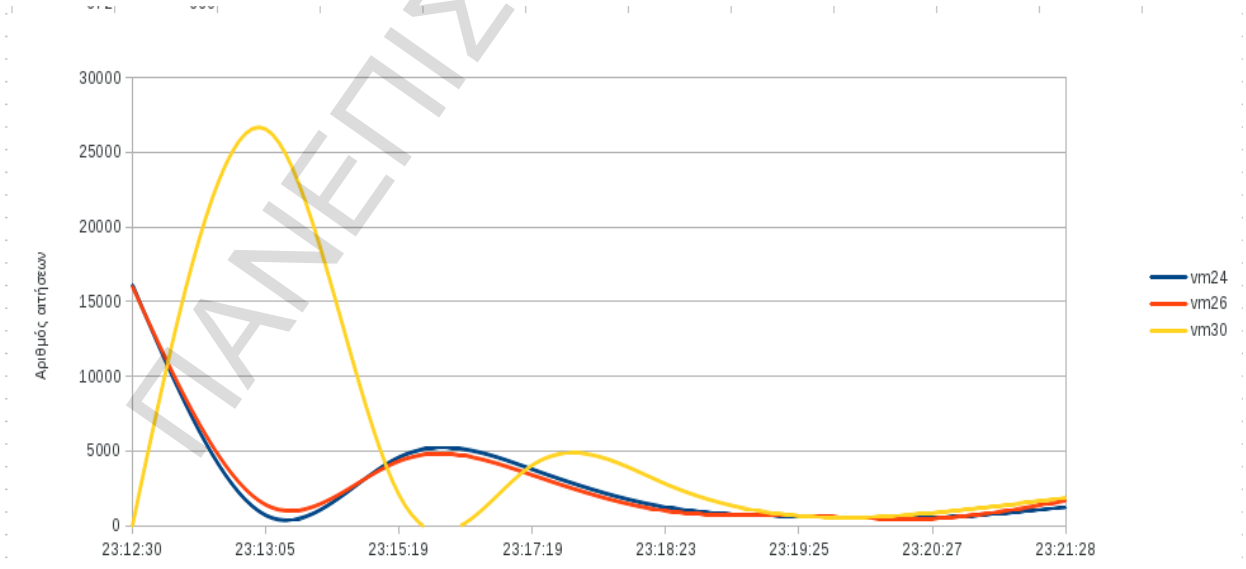

Εικόνα 22: Αριθμός αιτήσεων ανά μηχάνημα

Μία ένδειξη για τη ομαλότητα στη επεξεργασία των αιτήσεων μπορεί να μας δώσει το παρακάτων γράφημα (εικόνα 24):




Εικόνα 23: Χρόνος απόκρισης HTTP αιτημάτων

Η παραπάνω εικόνα δείχνει ότι κατά τη μεγαλύτερη διάρκεια του πειράματος ο χρόνος απόκρισης των αιτήσεων ήταν κάτω από 500 msec, ο οποίος είναι πολύ ικανοποιητικός, αν συνοπολογίσουμε τη μαζικότητα αποστολής αιτήσεων (140-700/sec), και τη μικρή δυναμικότητα των μηχανημάτων nodes. Αυτό δείχνει πως ο μηχανισμός εποπτείας κατάφερε και διατηρούσε την κατάσταση των μηχανημάτων σε ένα ικανοποιητικό επίπεδο, αφού παρατηρώντας τις μετρήσεις μνήμης κυριώς, έγκαιρα παρατηρούσε πότε ένα μηχάνημα ξεπέρναγε τα ανώτατα επιτρεπτά όρια και ειδοποιούσε το μηχανισμό διαχείρησης ροής εργασιών για να δεσμεύσει επιπρόσθετους πόρους για νέα εικονικά μηχανήματα. Τη παραπάνω υπόθεση μπορούμε να τη στηρίξουμε αναλύοντας και το παρακάτω γράφημα (εικόνα 25):

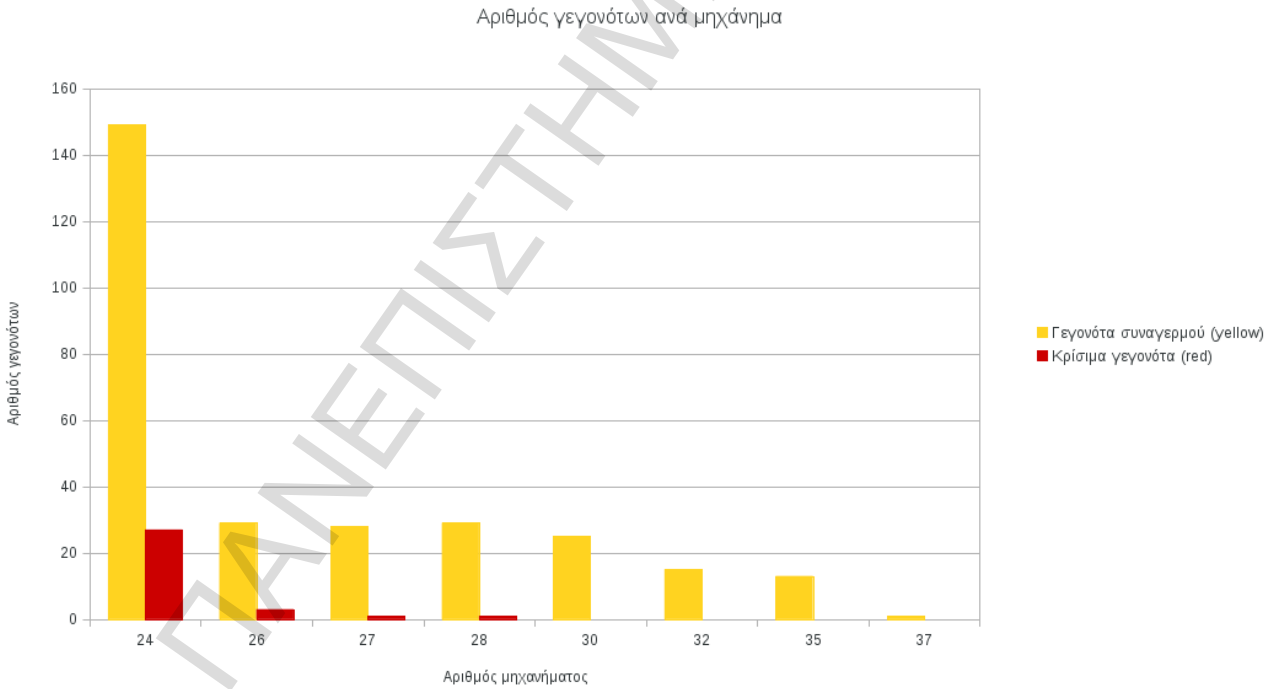



Εικόνα 24: Αριθμός αιτήσεων/χρονός

Το παραπάνω γράφημα παρουσιάζει το αριθμό των αιτήσεων που επεξεργάστηκε κάθε λεπτό το κάθε ένα από τα 5 πρώτα μηχανήματα, κατά τη διάρκεια του πειράματος. Παρατηρούμε ότι σε πολλά σημεία, και οι πέντε γραμμές συγκλίνουν, το οποίο σημαίνει ότι υπάρχει επαρκής αριθμός μηχανημάτων για να επεξεργαστεί τα αιτήματα, και η κατανομή του φόρτου γίνεται με απόλυτα επιτυχημένο τρόπο. Επίσης παρατηρούμες, ότι ο αριθμός των αιτήσεων που επεξεργάζεται κάθε μηχάνημα με τη πάροδο του χρόνου πέφτει, κάτι που σημαίνει ότι καταναλώνει πολύ λιγότερη μνήμη σε σχέση με την εκκίνηση του πειράματος. Αυτό μας δείχνει ότι ο μηχανισμός εποπτείας διέγνωσε έγκαιρα το αυξημένο φόρτο των πρώτων μηχανημάτων και ενημέρωσε κατάλληλα την υπηρεσία του διαχειρηστή ροής εργασίας. Καθώς περνά ο χρόνος και περισσότερα μηχανήματα προστίθενται, γίνεται ισοκατανομή του φόρτου, και τα μηχανήματα απαλλάσσονται από το υψηλό φόρτο.

Μία βασική λειτουργία του μηχανισμού εποπτείας είναι η παραγωγή γεγονότων κάθε φορά που μία παράμετρος ενός μηχανήματος ξεπερνά ένα από τα δύο όρια συναγερμού που έχουν οριστεί εκ των προτέρων. Αυτή η λειτουργία ενημερώνει τον διαχειρηστή ροής εργασιών για να πράξει ανάλογα με τη περίσταση. Σε περίπτωση που το γεγονός αντιπροσωπεύει παραβίαση του πρώτου στάδιου συναγερμού (yellowevent) και συμβαίνει για 1$^{η}$ φορά, τότε ο μηχανισμός εποπτείας ενημερώνει το μηχανισμό διαχείρησης για να εκκινήσει ένα νέο εικονικό μηχάνημα. Στη δεύτερη περίπτωση όπου έχουμε παραβίαση του δεύτερου ορίου συναγερμού, αυτό σημαίνει ότι κάποιο μηχάνημα καταναλώνει ένα σημαντικό μέρος των πόρων του και συνεπώς ο μηχανισμός εποπτείας ενημερώνει το διαχειρηστή για να εισάγει το νέο εικονικό μηχάνημα, που δημιούργησε προηγουμένως στη λίστα με τα μηχανήματα που διαμοιράζει ο LoadBalancertις αιτήσεις.

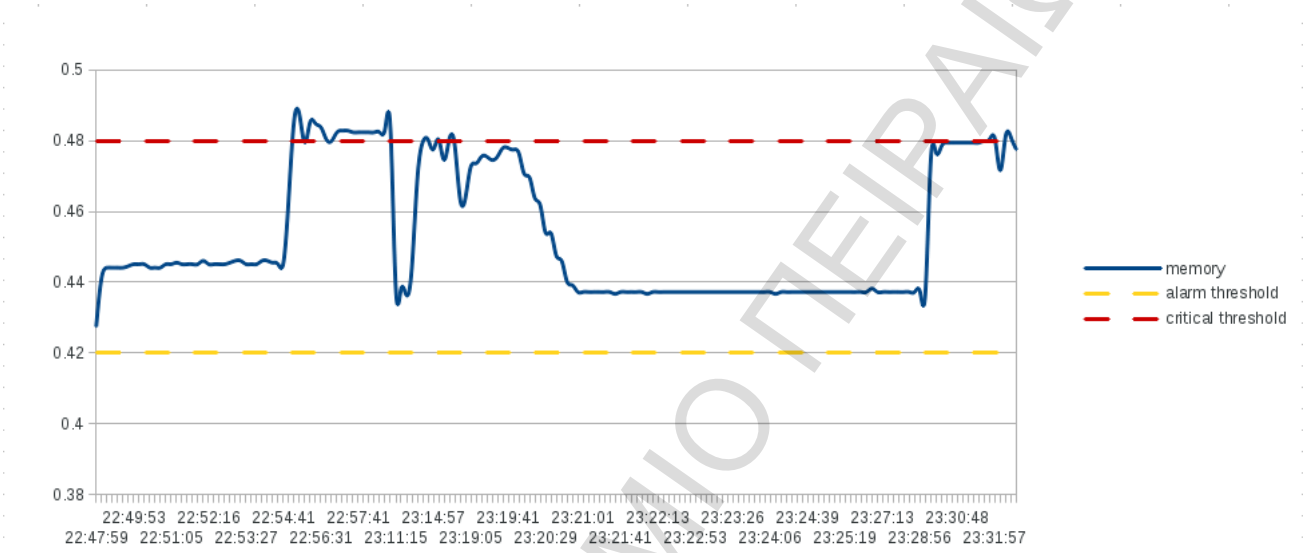
Το παρακάτω γράφημα (εικόνα 26), μας δείχνει τον αριθμό των γεγονότων που παράχθηκαν από συγκεκριμένα μηχανήματα:



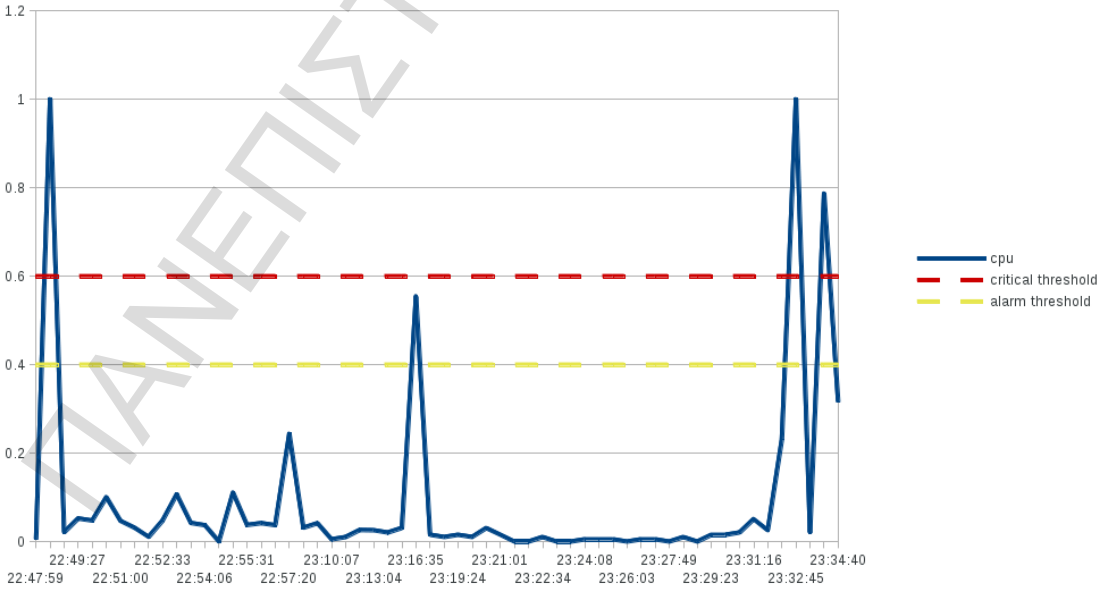

**Εικόνα 25: Αριθμός γεγονότων συναγερμού ανά μηχάνημα**

Παρατηρούμε ότι από τα 16 συνολικά μηχανήματα, τα 8 μόνο παρήγαγαν γεγονότα συναγερμού. Το μηχάνημα 24, που ουσιαστικά ήταν το 1° που εισήχθη στο πείραμα είναι και αυτό που έχει παράγει και τα περισσότερα γεγονότα κινδύνου. Παρ'όλα αυτά βλέπουμε ότι τα γεγονότα του δεύτερου σταδίου επικινδυνότητας είναι πολύ λιγότερα από τα γεγονότα του πρώτου σταδίου. Θεωρήσαμε ότι αυτό επιτεύχθηκε χάρη στη αυστηρή πολιτική του μηχανισμού εποπτείας που αναφέραμε στην αρχή του κεφαλαίου. Με αυτό το τρόπο η υποδομή είχε πάντα διαθέσιμα εικονικά μηχανήματα να παρέχει στο πείραμα.

Συγκεκριμένα για το μηχάνημα 24, μπορούμε να παρατηρήσουμε τις παρακάτω γραφικές παραστάσεις (εικονα 27,28):




Εικόνα 26: Κατανάλωση μνήμης κατά τη διάρκεια του πειράματος




Εικόνα 27: Φόρτος επεξεργαστή κατά τη διάρκεια του πειράματος

Η εικόνα 27 μας ότι για το μηχάνημα 24, που χρησιμοποιούταν καθ'όλη τη διάρκεια του πειράματος ελάχιστες φορές ξεπέρασε το ανώτατο όριο, και όταν έγινε αυτό ο μηχανισμός εποπτείας προέβει στις απαραίτητες ενέργειας για να εισαχθεί ένα καινούργιο μηχάνημα στο πείραμα. Αυτό φαίνεται στους μικρούς χρόνους που χρειάστηκε το μηχάνημα για να επανέλθει στη κατάσταση που ήταν πριν. Αντίστοιχα συνέβει και με τον επεξεργαστή του μηχανήματος (εικόνα 28), όπου σε πολύ συγκεκριμένες περιπτώσεις υπερέβει το πρώτο όριο σταδίου επικινδυνότητας.

Ένα άλλο κύριο χαρακτηριστικό του μηχανησμού εποπτείας είναι ότι εάν παρατηρήσει ότι μία παράμετρος ενός μηχανήματος υπερβεί ένα από τα δύο όριο, τότε αμέσως μειώνει το μεσοδιάστημα μεταξύ δύο διαδοχικών ελέγχων για τη συγκεκριμένη παράμετρο του συγκεκριμένου μηχανήματος. Σε κατάστασεις μη επικίνδυνες, ο μηχανισμός εποπτείας ελέγχει μία παράμετρο κάθε 30 sec. Εάν μία παράμετρος ενός μηχανήματος υπερβεί το $1^{ο}$ όριο επικινδυνότητας, τότε το μεσοδιάστημα αυτό μειώνεται στα 10 sec, ενώ εάν η παράμετρος υπερβεί και το $2^{ο}$ όριο, τότε το μεσοδίαστημα μειώνεται ακόμα περισσότερο στα 5 sec. Αυτό αυξάνει την εγκυρότητα του μηχανισμού εποπτείας και εγγυάται για την υγεία των μηχανημάτων. Τη διαφορά στους χρόνους μπορούμε να τη παρατηρήσουμε στις 2 προηγούμενες εικόνες (εικόνα 27,28). Παρατηρούμε ότι οι χρόνοι είναι πολύ πυκνότεροι στις μετρήσεις μνήμης, απ' ότι στις μετρήσεις επεξεργαστή. Αυτό συμβαίνει διότι η μνήμη του μηχανήματος είναι μόνιμα πάνω απ' το $1^{ο}$ όριο (42%) και συνεπώς το μεσοδιάστημα μεταξύ δύο διαδοχικών ελέγχων είναι 10 sec. Αντίθετα, επείδη το επίπεδο φόρτου του επεξεργαστή είναι κάτω από το $1^{ο}$ όριο (40%), ο μηχανισμός διακρίνει ότι δεν υπάρχει πρόβλημα με αυτή τη παράμετρο και το ελέγχει κάθε 30 sec. Οιπαρακάτω πίνακες μας δείχνουν πως μεταβάλλονται οι χρόνοι μεταξύ διαδοχικών μετρήσεων ανάλογα με τις διάφορες τιμές των παραμέτρων των εικονικών μηχανημάτων:

| # | memory_value | time(timestamp) |
|---|---|---|
| 36 | 0.44553483807654 6 | 22:54:41 |
| 37 | 0.464180569185476 | 22:54:51 |
| 38 | 0.485770363101079 | 22:55:03 |
| 39 | 0.487254901960784 | 22:55:05 |
| 40 | 0.479411764705882 | 22:55:31 |
| 41 | 0.485294117647059 | 22:56:21 |
| 42 | 0.484789008832188 | 22:56:27 |
| 43 | 0.483807654563297 | 22:56:31 |

Εικόνα 29: Ιστορικό μετρήσεων μνήμης για το μηχάνημα 24

| # | memory_value | time(timestamp) |
|---|---|---|
| 7 | 0.382352941176471 | 23:12:31 |
| 8 | 0.416094210009814 | 23:13:06 |
| 9 | 0.425490196078431 | 23:13:39 |
| 10 | 0.420019627085378 | 23:15:22 |
| 11 | 0.426470588235294 | 23:17:47 |
| 12 | 0.421000981354269 | 23:18:29 |
| 13 | 0.418627450980392 | 23:18:55 |
| 14 | 0.422963689892051 | 23:19:26 |

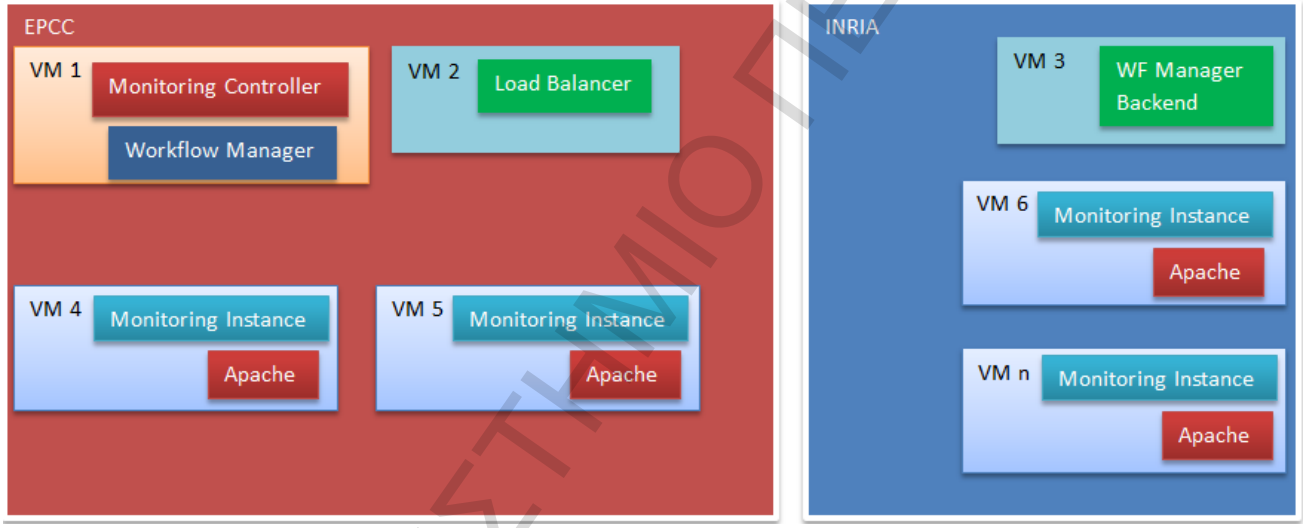
Εικόνα 28: Ιστορικό μετρήσεων μνήμης για το μηχάνημα 26

Οι παραπάνω πίνακες (εικόνες 29,30), μας δείχνουν ακριβώς το πως μεταβάλλονται τα διάστηματα. Στο συγκεκριμένο πείραμα, αν παρατηρήσουμε το πίνακα της εικόνας 29, θα δούμε αρχικά ότι ο μηχανισμός εποπτείας ελέγχει το μηχάνημα κάθε 10sec περίπου, αφού η τιμή της μνήμης έχει ξεπεράσει το $1^{ο}$ όριο (42%). Στη συνέχεια όμως όταν η τιμή ξεπερνά και το $2^{ο}$ όριο (48%), τότε το διάστημα πέφτει στα 5secή και λιγότερα. Αντίστοιχα, παρατηρώντας και το πίνακα ιστορικού μετρήσεων για το μηχάνημα 26 (εικόνα 30), βλέπουμε ότι αρχικά ο μηχανισμός εποπτείας λαμβάνει μετρήσεις κάθε 30sec, ενώ μόλις η μνήμη ξεπεράσει το 42%, το διάστημα αυτό πέφτει στα 10sec.

## 6.2. Πείραμα 2

Για το δεύτερο πείραμα, χρησιμοποιήθηκαν τα ίδια κύρια υποσυστήματα και υπηρεσίες που χρησιμοποιήθηκαν και για το 1$^{ο}$ πείραμα, με τη μόνη διαφορά ότι σε αυτό το πείραμα χρησιμοποιήθηκαν πόροι από 2 υποδομές (cross-site experiment). Η πρώτη υποδομή ήταν το EPCC και η δεύτερη υποδομή ήταν η INRIA[82]. Συνολικά χρησιμοποιήθηκαν 12 εικονικά μηχανήματα, κάτι το οποίο μεταφράζεται σε 12 επεξεργαστικούς πυρήνες, 12288MB μνήμης και 24GB συνολικά αποθηκευτικού χώρου. Το πείραμα διήρκησε γύρω στα 25-30 λεπτά. Ο Load Balancer δεχόταν αιτήσεις με ρυθμό 140-700 αιτήσεις/sec. Το πείραμα αυτό εκκινήθηκε χρησιμοποιώντας 2 μηχανήματα για αρχή, ένα από κάθε υποδομή.

### 6.2.1. Τοπολογία πειράματος



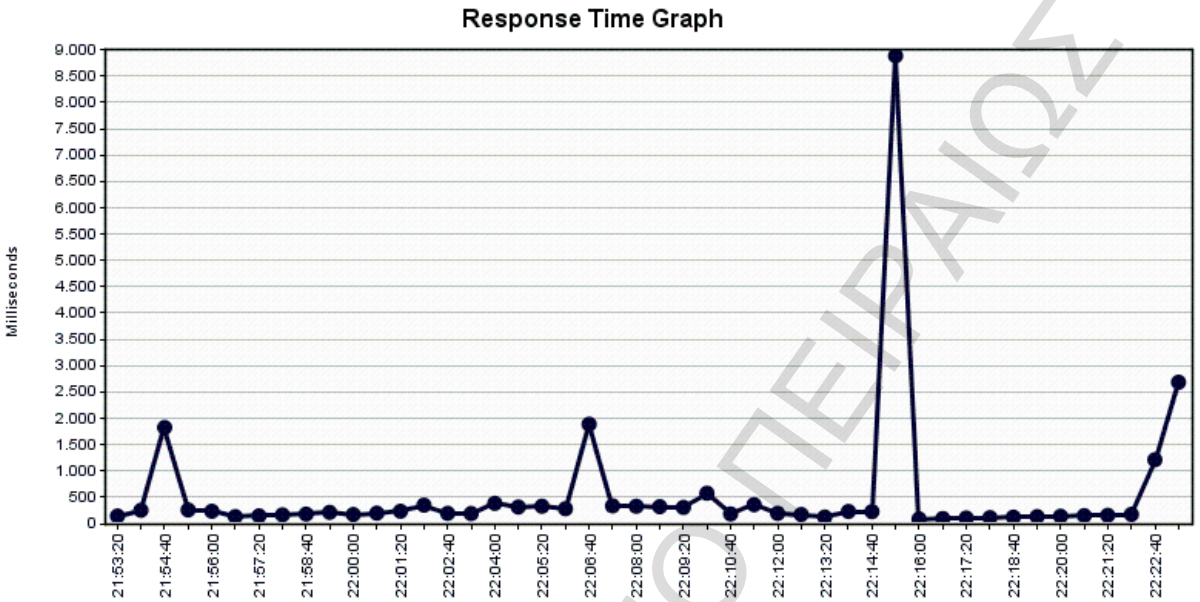

Εικόνα 30:Τοπολογία πειράματος 2

Στη εικόνα 31 φαίνεται η τοπολογία του πειράματος. ΟιυπηρεσίεςτουMonitoringController, Workflow Manager, Load Balancer και μερικά μηχανήματα nodes έχουν εγκατασταθεί στη υποδομή EPCC. Στην υποδομή INRIA, έχει εγκατασταθεί ένα υποσύστημα του Workflow Manager και τα υπόλοιπα μηχανήματα nodestου πειράματος.

Για το πείραμα χρησιμοποιήθηκαν μηχανήματα με δυνατότητες ίδιες με αυτές του πειράματος 1. Το υποσύστημα του διαχειριστή ροών εργασίας φιλοξενείται σε ένα μηχάνημα 2 επεξεργαστικών πυρήνων και 2048MB μνήμης.

### 6.2.2. Αποτελέσματα πειράματος

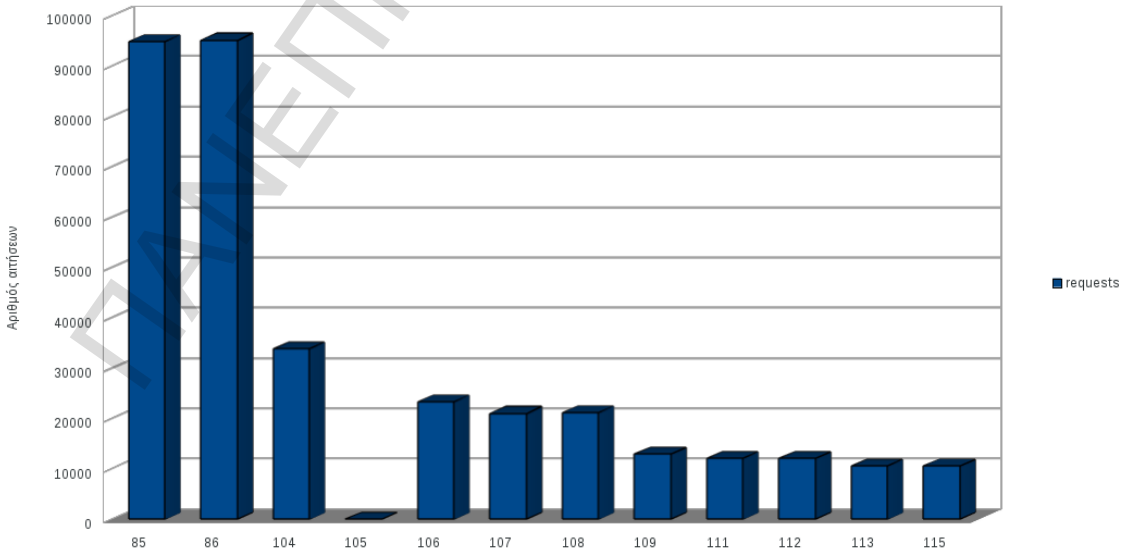
Σε αυτό το πείραμα αξίζει να παρατηρήσουμε το γράφημα της εικόνας 32, που περιγράφει το χρόνο απόκρισης αιτημάτων από τη μεριά του πελάτη:




Εικόνα 31: Χρόνος απόκρισης HTTP αιτημάτων

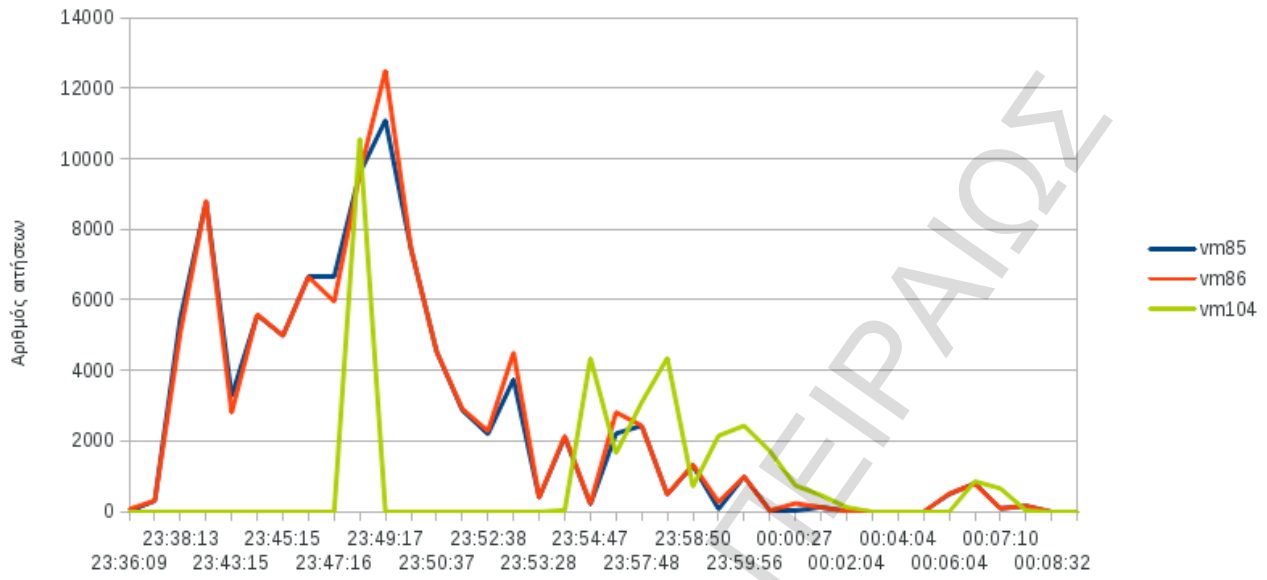
Παρατηρούμε ότι σε αυτό το πείραμα ο χρόνος απόκρισης είναι ελάχιστα μεγαλύτερος. Επίσης παρουσιάζονται εντονότερες διακυμάνσεις στο χρόνο. Αυτό συμβαίνει διότι μερικά μηχανήματα είναι εγκατεστημένα στη $2^{η}$ υποδομή, ενώ ο Load Balancer είναι εγκατεστημένος στη $1^{η}$ υποδομή όπως συνέβη και στο $1^{ο}$ πείραμα.

Στη παρακάτω εικόνα βλέπουμε το συνολικό αιτημάτων που δέχθηκε κάθε μηχάνημα του πειράματος.




Εικόνα 32: Συνολικός αριθμός αιτήσεων ανά μηχάνημα

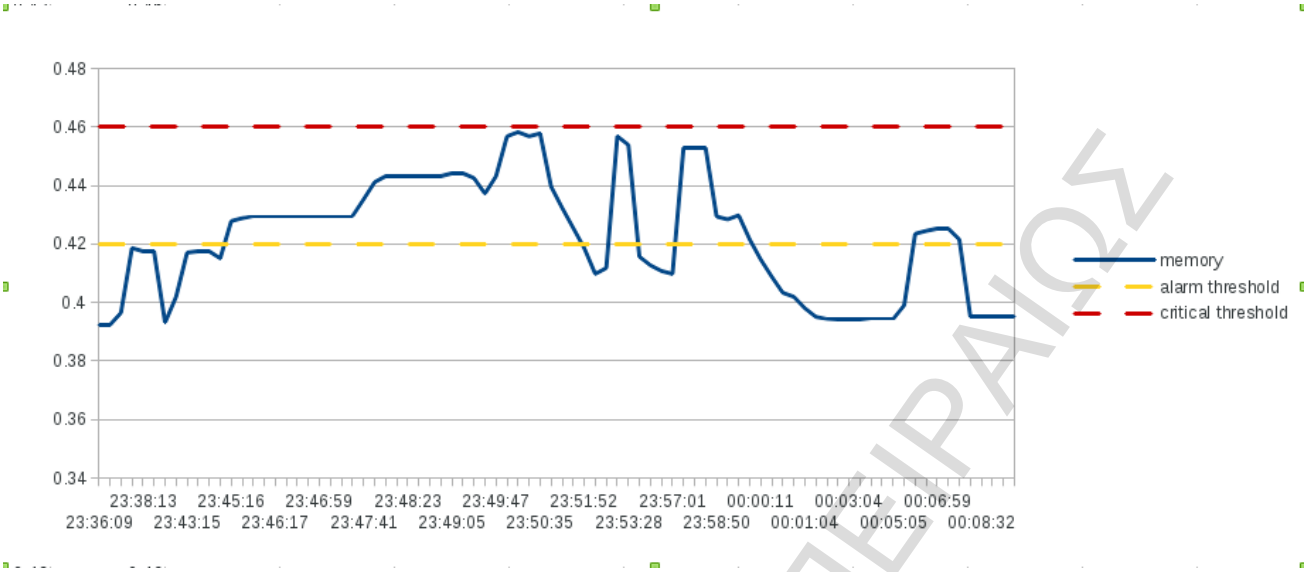
Μία ένδειξη για την ομαλότητα της κατανομής των αιτήσεων στα μηχανήματα αποτελεί η εικόνα 34:



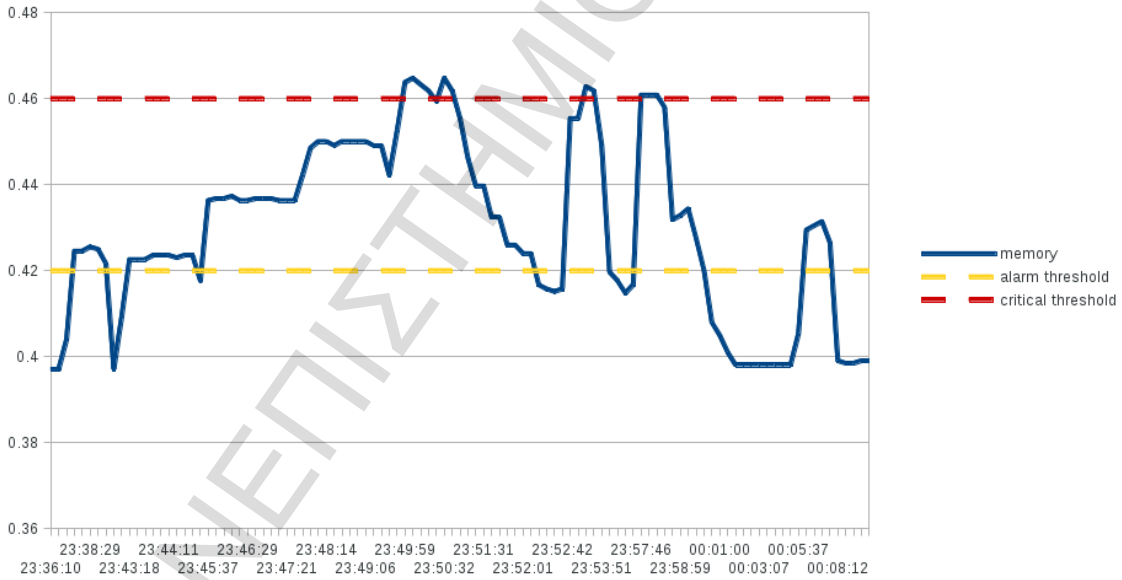

**Εικόνα 33: Αιτήσεις μηχανήματων προς το χρόνο**

Στο παραπάνω γράφημα χρησιμοποιήσαμε τα δύο πρώτα μηχανήματα του πειράματος (85,86) και ένα μηχάνημα το οποίο εισήχθει στο πείραμα ένα λέπτο αργότερα (104). Το γράφημα αυτό δείχνει το πως εξομαλύνεται η κατανομή των αιτήσεων με το πέρασμα του χρόνου. Για τα 2 πρώτα μηχανήματα ο αριθμός αιτήσεων είναι σχεδόν ο ίδιος. Οι μικρές διαφορές οφείλονται στο ότι τα μηχανήματα είναι εγκατεστημένα σε διαφορετικές υποδομές το καθένα. Το 3$^{ο}$ μηχάνημα παρότι εισάγεται το πείραμα ένα λεπτό αργότερα, παρατηρούμε ότι με το πέρας του χρόνου, εξυπηρετεί τον ίδιο αριθμό αιτήσεων με τα προηγούμενα δύο μηχανήματα.

Παρακάτω θα παραθέσουμε δύο γραφήματα (εικόνα 35,36) που δείχνουν το ιστορικό μετρήσεων μνήμης για τα δύο πρώτα μηχανήματα(85,86) που χρησιμοποιήθηκαν στο πείραμα:



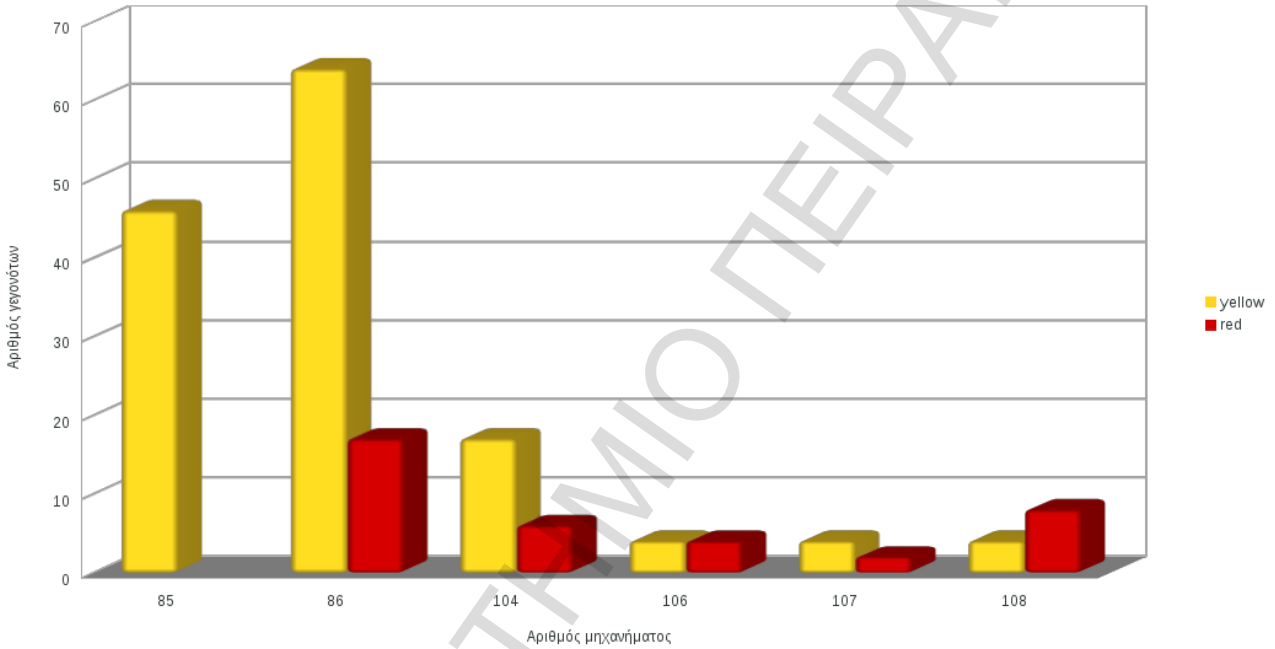

Εικόνα 35: Ιστορικό μετρήσεων μνήμης για το μηχάνημα 85




Εικόνα 34: Ιστορικό μετρήσεων μνήμης για το μηχάνημα 86

Παρατηρούμε ότι και τα δύο μηχανήματα, παρότι τα όρια 2ο του στάδιου επικινδυνότητας είναι μικρότερα, δεν ξεπερνάνε παρά σε ελάχιστες περιπτώσεις τα ανώτατα όρια. Αυτό συμβαίνει διότι το πείραμα ξεκίνησε με τα 2 μηχανήματα να είναι ενεργά, και έτσι είχαμε ισοκατανομή των αιτήσεων από την αρχή. Επίσης το παραπάνω φαινόμενο παρατηρήθηκε διότι, όπως προαναφέραμε, τα ανώτατα όρια ήταν μικρότερα σε σχέση με το 1ο πείραμα, το οποίο σήμαινε ότι πολύ πιο γρήγορα ένα μηχάνημα θα δηλώνόταν στον LoadBalancerκαι μετέπειτα στο μηχανισμό εποπτείας. Αυτό είχε ως αποτέλεσμα να πέφτει ο αριθμός των αίτησεων που είχε να επεξεργαστεί κάθε μηχανήμα και σε συνέχεια η κατανάλωση μνήμης του μηχανήματος αυτού.

Αυτό επιβεβαιώνεται και από τη παρακάτω εικόνα (37), η οποία δείχνει τον αριθμό των γεγονότων που δημιουργήθηκαν από κάθε μηχάνημα:




**Εικόνα 36: Αριθμός γεγονότων ανά μηχάνημα**

Παρατηρούμε ότι το 1ο μηχάνημα (υποδομή INRIA), δεν παρήγαγε κανένα γεγονός υψηλής κρισιμότητας (redevent). Αντίστοιχα το δεύτερο μηχάνημα παρήγαγε πολύ λιγότερα από το αντίστοιχο του πειράματος 1.

Τέλος και για αυτό το πείραμα θα παρατηρήσουμε το πως διαχειριζόταν ο μηχανισμός εποπτείας τις καταστάσεις χαμηλής και υψηλής επικινδυνότητας παρατηρώντας τους παρακάτω πίνακες μετρήσεων μνήμης για τα μηχάνηματα 85 και 86:

| # | memory_value | time(timestamp) |
|---|---|---|
| 10 | 0.417647058823529 | 23:43:45 |
| 11 | 0.417647058823529 | 23:44:16 |
| 12 | 0.415112855740922 | 23:44:46 |
| 13 | 0.427870461236506 | 23:45:16 |
| 14 | 0.428851815505397 | 23:45:46 |
| 15 | 0.429411764705882 | 23:45:56 |
| 16 | 0.429411764705882 | 23:46:07 |
| 17 | 0.429411764705882 | 23:46:17 |
| 18 | 0.429411764705882 | 23:46:28 |
| 19 | 0.429411764705882 | 23:46:38 |
| 20 | 0.429411764705882 | 23:46:49 |

| # | memory_value | time(timestamp) |
|---|---|---|
| 41 | 0.45 | 23:49:06 |
| 42 | 0.449019607843137 | 23:49:17 |
| 43 | 0.449019607843137 | 23:49:27 |
| 44 | 0.442156862745098 | 23:49:37 |
| 45 | 0.452404317958783 | 23:49:48 |
| 46 | 0.463725490196078 | 23:49:59 |
| 47 | 0.464705882352941 | 23:50:03 |
| 48 | 0.463199214916585 | 23:50:06 |
| 49 | 0.461764705882353 | 23:50:09 |
| 50 | 0.459273797841021 | 23:50:11 |

**Εικόνα 37: Ιστορικό μετρήσεων μνήμης μηχανήματών 85 και 86**

Παρατηρούμε και πάλι πως μειώνεται το διάστημα παρακολούθησης από 30sec σε ~10sec, όταν παρατηρείται παραβίαση του 1$^{ου}$ ορίου, και αντίστοιχα από 10secσε ~5sec, όταν παρατηρείται παραβίαση του 2$^{ου}$ ορίου.

## 6.3. Συμπεράσματα

Τα παραπάνω πειράματα έγιναν στο πλαίσιο αξιολόγησης μίας υπηρεσιοστρεφούς πλατφόρμας που παρέχει ένα εξελιγμένο μηχανισμό εποπτείας υποδομής υπολογιστικού νέφους και υπηρεσιών. Κατά τη διάρκεια των πειραμάτων, ήταν άξιο σημασίας το πόσο αξιόπιστα ανταποκρίθηκε ο μηχανισμός εποπτείας στις ανάγκες του πειράματος και σε συνεργασία με το διαχειριστή ροών εργασίας διατήρησαν τα επίπεδα της πειραματικής διαδικασίας και των μηχανημάτων της υποδομής σε πολύ ικανοποιητικά επίπεδα. Σκοπός του μηχανισμού εποπτείας ήταν η παρακολούθηση συγκεκριμένων παραμέτρων συστήματος των μηχανημάτων, καθώς και συγεκριμένων μετρικών σχετικά με την υπηρεσία που λειτουργούσε σε κάθε μηχάνημα.

Η υπηρεσία ισοκατανομής αιτημάτων HTTP, έδειξε να λειτουργεί σωστά, και ήταν σε θέση να διαμοιράσει σωστά τα αιτήματα, τα οποία κατεύθαναν μαζικά (140-700sec) στα ενεργά εικονικά μηχανήματα nodes, της υποδομής. Σε αυτό έπαιξε πολύ μεγάλο ρόλο η ακρίβεια και η εγκυρότητα του μηχανισμού εποπτείας, καθώς όπως παρατηρήσαμε και στις μετρήσεις είχε υλοποιηθεί η κατάλληλη λογική έτσι ώστε να εντοπίζεται έγκαιρα οποιαδήποτε απόκλιση από τα φυσιολογικά όρια των παραμέτρων συστήματος και να ειδοποιείται ο κατάλληλος μηχανισμός για την ανάθεση νέων πόρων στην υποδομή και το πείραμα.

Η παράμετρος που παρατηρήθηκε περισσότερο και στα 2 πειράματα που έγιναν, ήταν η κατανάλωση της μνήμης. Διαπιστώθηκε ότι η υπηρεσία ισοκατανομής των αιτημάτων ήταν πολύ πιο απαιτητική σε μνήμη, απ' ότι σε επεξεργαστική ισχύ. Για αυτό και ο μηχανισμός εποπτείας πραγματοποίησε το μεγαλύτερο αριθμό ελέγχων στη κατανάλωση μνήμης για κάθε μηχάνημα.

Μία ακόμα χρήσιμη παρατήρηση είναι ότι το υποσύστημα του MonitoringController, ανταποκρίθηκε πολύ σωστά σε μεγάλο αριθμός εικονικών μηχανημάτων που παρακολουθούσε. Και στα 2 πειράματα ήταν σε θέση να διεξάγει έλεγχο σε πάνω από 12 μηχανήματα ταυτόχρονα, με εξαιρετικό συγχρονισμό. Επίσης, χαρακτηριστικό πλεονέκτημα του μηχανισμό είναι η ευελιξία του μηχανισμού, καθώς παρακολουθούσε ξεχωριστά κάθε μία από τις 3 παραμέτρους συστήματος και τη μετρική της υπηρεσίας ισοκατανομής. Έτσι ήταν σε θέση να καθορίζει το μεσοδιάστημα μεταξύ ελέγχων πιο δυναμικά και πιο αποτελεσματικά.

Τα παραπάνω πειράματα, μπορούν να ερμηνευτούν και ως χρήσιμα παραδείγματα τα οποία σκιαγραφούν και αναλύουν τις απαιτήσεις, τις δυσκολίες και τις ανάγκες για την πραγματοποιήση εποπτείας σε απλές ή σύνθετες υποδομές υπολογιστικών νεφών.

# Βιβλιογραφία

1. Γ. Κατσαρός:*Τεχνολογίες διαχείρισης υπολογιστικών υποδομών και εφαρμογών σε υπηρεσιοστρεφείς αρχιτεκτονικές και περιβάλλοντα Νεφών,* Μάιος 2012
2. Konstantinos Kostantos, Andrew Kapsalis, Dimosthenis Kyriazis, Marinos Themistocleous: *Open-source IaaS fit for purpose: A comparison between OpenNebula and OpenStack*, July 2012
3. Peter Mell, Timothy Glance,*"The NIST Definition of Cloud Computing",* September 2011
4. Cloud Computing Expert Group Report, *"The Future of Cloud Computing"*, European Commission, available online at: http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf
5. Dimosthenis Kyriazis, Spyridon Gogouvitis, Sotiris Stamokostas: *SILICON: QoS-Oriented Service Management in Large Scale Federated Cloud,* October 2012
6. OpenNebula: http://opennebula.org/
7. OpenStack: http://www.openstack.org/
8. Amazon Elastic Compute Cloud: http://aws.amazon.com/ec2/
9. Amazon Web Services: http://aws.amazon.com/
10. Amazon EC2 API: http://docs.aws.amazon.com/AWSEC2/2009-11-30/APIReference/
11. Amazon Cloud Watch: http://aws.amazon.com/cloudwatch/
12. OpenStack Nova project: http://nova.openstack.org
13. OpenStack Swift project:http://swift.openstack.org/
14. Google App Engine: https://developers.google.com/appengine/
15. Windows Azure NET:http://www.windowsazure.com/en-us/develop/net/
16. Sun Cloud (Wikipedia):http://en.wikipedia.org/wiki/Sun_Cloud
17. Eucalyptus:http://www.eucalyptus.com/
18. Flexiscale Public Cloud:http://www.flexiscale.com/products/flexiscale/
19. Joyent SmartMachines:http://joyent.com/products/joyent-cloud/smartmachine-features
20. Joyent SmartDataCenter:http://joyent.com/products/smartdatacenter
21. JClouds:http://www.jclouds.org/
22. Delta Cloud:http://deltacloud.apache.org/
23. Apache libcloud API:http://libcloud.apache.org/
24. Open Cloud Computing Interface:http://occi-wg.org/
25. UDDI:http://uddi.xml.org/
26. Oasis:https://www.oasis-open.org/
27. W3C: http://www.w3.org/
28. Open Virtualization Format Specification:(URL)http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.0.0.pdf ,December 2012

29. He Huang and Liqiang Wang. 2010. "*P&P: A Combined Push-Pull Model for Resource Monitoring in Cloud Computing Environment*". In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD '10). IEEE Computer Society, Washington, DC, USA, 260-267.
30. G. Heward, I. Muller, J. Han, J.-G. Schneider, S. Versteeg, "*Assessing the performance impact of service monitoring"*, Software Engineering Conference, 2010.
31. Eclipse Project: http://eclipse.org/
32. Jersey Project: http://java.jersey.net
33. JBoss Application Server: http://www.jboss.org
34. Rackspace Cloud:http://www.rackspace.com/cloud/
35. Rackspace CloudFiles:http://www.rackspacecloud.com/cloud_hosting_products/files/
36. Rackspace CloudServers API reference:http://docs.rackspacecloud.com/servers/api/cs-devguide-latest.pdf
37. VMware vCloud:http://www.vmware.com/products/vcloud/
38. vCloudAPIProgrammingGuide:http://communities.vmware.com/servlet/JiveServlet/previewBody/12463-102-1-13007/vCloud_API_Guide.pdf
39. MySQL: http://www.mysql.com/
40. NetBeans: http://netbeans.org/
41. RESTEasy: http://www.jboss.org/resteasy
42. Foster, I., Yong Zhao, Raicu, I., Lu, S., "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008. GCE '08 , vol., no., pp.1-10, 12-16 Nov. 2008.
43. G. Kousiouris, D. Kyriazis, K. Konstanteli, S. Gogouvitis, G. Katsaros, T. Varvarigou, "A service-oriented framework for gnu octave-based performance prediction", 2010 IEEE International Conference on Services Computing (SCC), 2010, pp. 114–121.
44. I. A. Moschakis, H. D. Karatza, Performance and cost evaluation of gang scheduling in a cloud computing system with job migrations and starvation handling, in: ISCC, 2011, pp. 418–423.
45. Lattice Monitoring Framework, 2009, http://clayfour.ee.ucl.ac.uk/lattice.
46. Relational Database Service, http://aws.amazon.com/rds/
47. Azure Diagnostic Manager , http://www.cerebrata.com/Products/AzureDiagnosticsManager/Default.aspx
48. Azure Subscription Manager, http://communities.quest.com/docs/DOC-9911
49. CloudStatus, http://www.hyperic.com/products/cloud-status-monitoring
50. Ganglia Monitoring System, http://ganglia.info.
51. XDR, External Data Representation Standard, Sun Microsystems 1987, http://tools.ietf.org/html/rfc1014
52. RRD, Round Robin Database tool, http://oss.oetiker.ch/rrdtool/
53. Nagios 1996, http://www.nagios.org
54. Nagios plug-in development guidelines 2000, http://nagiosplug.sourceforge.net/developer-guidelines.html
55. Nagios Enterprises - Nagios XI. http://www.nagios.com/products/nagiosxi

56. Hyperic HQ, 2004, http://www.hyperic.com
57. HQ Inventory Model, 2009, http://support.hyperic.com/display/DOC/HQ+Inventory+Model
58. Hyperic Community, http://www.hyperic.com/community
59. Lattice Monitoring Framework, 2009, http://clayfour.ee.ucl.ac.uk/lattice
60. Stuart Clayman et. al., 2010, Monitoring Virtual Networks with Lattice, IEEE Network Operations and Management Symposium Workshops.
61. AutoI project, 2008, http://ist-autoi.eu/autoi/index.php
62. Zenoss, 2005, Open Source IT management. http://community.zenoss.org/docs/DOC-2614
63. Zope, http://www.zope.org
64. G. Carlson, "How to save money with computer monitoring", in: Proceedings of the ACM annual conference - Volume 2, ACM '72, ACM, New York, NY, USA, 1972, pp. 1018–1023.
65. Zanikolas, S., & Sakellariou, R. (2005). "A taxonomy of grid monitoring systems", Future Generation Computer Systems 21, 163-188
66. S. Andreozzi, N. D. Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, M. C. Vistoli, Gridice: a monitoring service for grid systems, Future Generation Computer Systems 21 (4) (2005) 559–571, high-Speed Networks and Services for Data-Intensive Grids: the DataTAG Project. doi:DOI: 10.1016/j.future.2004.10.005.
67. H. B. Newman, I. Legrand, P. Galvez, R. Voicu, C. Cirstoiu, "Monalisa : A distributed monitoring service architecture", CoRR cs.DC/0306096
68. J. M. Schopf, I. Raicu, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. DArcy, "Monitoring and discovery in a web services framework: Functionality and performance ofglobus toolkit mds4", MCS Preprint #ANL/MCS-P1315-0106, http://www.mcs.anl.gov/uploads/cels/papers/P1315.pdf (2006).
69. S. Microsystems, Jini architecture specification, version 1.2 (2001).
70. OASIS wsrf v1.2, http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf, (2009).
71. OASIS WS Notifation 1.2, http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf, 2004.
72. OGF Glue specification, http://www.ogf.org/documents/GFD.147.pdf, (2009).
73. J. M. Schopf, I. Raicu, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. DArcy, Monitoring and discovery in a web services framework: Functionality and performance of globus toolkit mds4 (2006).
74. H. Keung, J. Dyson, S. Jarvis, G. Nudd, "Predicting the performance of globus monitoring and discovery service (mds-2) queries", Fourth International Workshop on Grid Computing, 2003. Proceedings, 2003.
75. Tierney, B., Aydt, R., Gunter, D., Smith, W., Swany, M., Taylor, V., & Wolski, R. (2002). "GFD-I.7: A Grid Monitoring Architecture". The Open Grid Forum Informational Document. Retrieved August 8, 2011, from http://www.ogf.org/documents/GFD.7.pdf
76. M. Lindner, M. F. G., C. Chapman, S. Clayman, H. Daniel, E. Erik, "The cloud supply chain: A framework for information, monitoring, accounting and billing", 2nd International ICST Conference on Cloud Computing (CloudComp 2010), 2011.
77. D. J. Colling, J. Martyniak, A. S. McGough, A. Krenek, J. Sitera, M. Mulac, F. Dvorák,"Real time monitor of grid job executions", Journal of Physics: Conference Series 219, 2010.

78. G. Heward, I. Muller, J. Han, J.-G. Schneider, S. Versteeg, Assessing the performance impact of service monitoring, Software Engineering Conference, 2010.
79. Hasselmeyer, P. "Removing the Need for State Dissemination in Grid Resource Brokering". Proceedings of the 5th International Workshop on Middleware for Grid Computing (MGC 2007), ACM Press.
80. Java EE (Wikipedia): http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
81. EPCC, http://www.epcc.ed.ac.uk/
82. INRIA, http://www.inria.fr/en/