ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνολογίες διαχείρισης υπολογιστικών υποδομών και εφαρμογών σε υπηρεσιοστρεφείς αρχιτεκτονικές και περιβάλλοντα Νεφών

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Γρηγόριος Χ. Κατσαρός

Επιβλέπουσα Καθηγήτρια: Θ. Α. Βαρβαρίγου

Αθήνα, Μάιος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνολογίες διαχείρισης υπολογιστικών υποδομών και εφαρμογών σε υπηρεσιοστρεφείς αρχιτεκτονικές και περιβάλλοντα Νεφών

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Γρηγόριος Χ. Κατσαρός

Συμβουλευτική Επιτροπή : Θεοδώρα Α. Βαρβαρίγου
Βασίλειος Λούμος
Συμεών Παπαβασιλείου

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την …….

.....	.....	.....
Θ.Α. Βαρβαρίγου	Βασίλειος Λούμος	Συμεών Παπαβασιλείου
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπλ. Καθηγητής Ε.Μ.Π.
.....	.....	.....
Δ. Ασκούνης	Ν. Κοζύρης	Α. Σταφυλοπάτης
Αναπλ. Καθηγητής Ε.Μ.Π	Αναπλ. Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

.....
Ε. Βαρβαρίγος
Καθηγητής Πανεπιστημίου Πατρών

Αθήνα, Μάιος 2012

.....

Γρηγόριος Χ. Κατσαρός

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

ΠΡΟΛΟΓΟΣ

Η διδακτορική διατριβή που παρουσιάζεται στις επόμενες σελίδες εκπονήθηκε από το Οκτώβριο του 2008 μέχρι τον Μάιο του 2012, στο εργαστήριο Τηλεπικοινωνιών του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής, στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Κατά την διάρκεια της εκπόνησης αυτής της διατριβής, είχα την ευκαιρία να ασχοληθώ με αρκετά ενδιαφέροντα επιστημονικά θέματα που αφορούν κυρίως στους τομείς της χρήσης, διαχείρισης, και επίβλεψης υπηρεσιοστρεφών υποδομών και υποδομών Νεφών, και να αποκτήσω πολύτιμη εμπειρία και γνώσεις.

Θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου, την καθηγήτριά μου κ. Θεοδώρα Βαρβαρίγου για το ενδιαφέρον που έδειξε, για τις πολύτιμες συμβουλές της και για την ιδιαίτερη στήριξη που μου παρείχε κατά την διάρκεια αυτής της πορείας μου, καθώς επίσης τους καθηγητές κ. Συμεών Παπαβασιλείου και Βασίλειο Λούμο για την υποστήριξη και καθοδήγησή τους. Επίσης, θα ήθελα να ευχαριστήσω όλους τους συναδέλφους μου στην ερευνητική ομάδα με τους οποίους συνεργάστηκα άψογα και επιτυχώς όλα αυτά τα χρόνια.. Ιδιαίτερες ευχαριστίες ωστόσο θα ήθελα να απευθύνω στους στενούς μου συνεργάτες και κυρίως στους Ανδρέα Μενύχτα, Δημοσθένη Κυριαζή, Κλεοπάτρα Κωνσταντέλη, Γιώργο Κουσιουρή και Σπυρίδωνα Γωγουβίτη με τους οποίους μοιραστήκαμε τις πάρα πολλές ώρες της ερευνητικής εργασίας .

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά και τους φίλους μου για την υποστήριξη και πίστη τους σε εμένα και τις επιλογές μου.

Γρηγόριος Χ. Κατσαρός

Μάιος 2012

Πίνακας περιεχομένων

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

Περίληψη

Η ανάπτυξη του Υπολογιστικού Νέφους και γενικότερα των υπηρεσιοστρεφών υποδομών παροχής λογισμικού και άλλων πόρων ως υπηρεσίες καθιστούν εφικτή την χρησιμοποίηση αυτών από εξωτερικούς χρήστες με τη μορφή πληρωμής με βάση τη χρήση. Η αποτελεσματική όμως χρήση και η αποδοτική διαχείριση των υποδομών αυτών εγείρουν πολλαπλές και ιδιαίτερες απαιτήσεις όσον αφορά την συλλογή και διαχείριση των πληροφοριών που παράγονται κατά την λειτουργία αυτών των υποδομών. Υπό αυτές τις συνθήκες, οι μηχανισμοί και οι τεχνικές επίβλεψης στις υπηρεσιοστρεφείς υποδομές και περιβάλλοντα Νεφών αποτελούν ένα πολύ σημαντικό συστατικό κατά μήκος της αλυσίδας αξίας των αρχιτεκτονικών αυτών.

Η διατριβή επικεντρώνεται σε αυτήν την μελέτη της διαχείρισης της πληροφορίας προσπαθώντας να ξεπεραστούν οι αγκυλώσεις και οι περιορισμού που προκύπτουν λόγω της ύπαρξης διαφορετικών οντοτήτων (Πάροχος Εφαρμογής, Πάροχος Πλατφόρμας και Πάροχος Υποδομών) αλλά και των τεχνικών περιορισμών που οι νέες αυτές αρχιτεκτονικές εισαγάγουν. Συγκεκριμένα γίνεται ανάλυση και σχεδιασμός καινοτόμων μηχανισμών επίβλεψης και παρακολούθησης υποδομών και εφαρμογών που εκτελούνται σε αυτά τα περιβάλλοντα. Παρουσιάζονται εξειδικευμένες δυνατότητες και λειτουργίες οι οποίες επαληθεύονται και αξιολογούνται τόσο με προσομοιώσεις όσο και σε πραγματικές συνθήκες λειτουργίας. Ιδιαίτερη προσοχή έχει δοθεί στην δυνατότητα αναδιαμόρφωσης του μηχανισμού επίβλεψης βασιζόμενοι στις συλλεχθείσες πληροφορίες. Το δυναμικό αυτό χαρακτηριστικό επιτρέπει τόσο την αποφυγή παραβιάσεων στο συμβόλαιο χρήσης (SLA) όσο και την βελτιστοποίηση χρήσης της υποδομής με την λήψη συγκεκριμένων αποφάσεων. Η συγκεκριμένη λύση επίβλεψης αποτελεί ένα υπηρεσιοστρεφές πλαίσιο συλλογής

πληροφοριών τόσο από την εικονική όσο και από την φυσική υποδομή. Η πολιτική αναδιαμόρφωσης εφαρμόζεται πάνω στο συγκεκριμένο υπηρεσιοστρφές πλαίσιο επίβλεψης και αξιολογείται κατόπιν μέσω μιας πραγματικής εφαρμογής λογισμικού διόρθωσης χρώματος σε καρέ βίντεο.

Επιπλέον, παρουσιάζεται μια καινοτόμα υποδομή επίβλεψης Υπολογιστικών Νεφών η οποία λαμβάνει δεδομένα από πολλαπλές πηγές με σκοπό την βελτιστοποίηση της διαχείρισης των πόρων. Ιδιαίτερη έμφαση δίνεται στην παρακολούθηση της ενεργειακής απόδοσης όπου και πραγματοποιείται μοντελοποίηση αυτής σε επίπεδο υπολογιστικού κόμβου, υποδομής (φυσικής και εικονικής) και εφαρμογής. Η μοντελοποίηση αυτή καταλήγει στην εφαρμογή πολιτικών που βελτιώνουν την χρήση και την τοποθέτηση (*placement*) εικονικών μηχανών (VMs) στην υποδομή του Νέφους. Τα αποτελέσματα που παρουσιάζονται είναι πολύ ενθαρρυντικά όσον αφορά τις δυνατότητες διαχείρισης των πόρων λαμβάνοντας υπόψη την ενεργειακή αποδοτικότητα της υποδομής.

Τέλος, παρουσιάζονται επιπρόσθετες συνεισφορές στην κοινότητα ανοιχτού λογισμικού σχετικά με τεχνολογίες και τεχνικές επίβλεψης πόρων. Συγκεκριμένα, παρουσιάζονται η αρχιτεκτονική και η υλοποίηση επεκτάσεων του συστήματος παρακολούθησης Nagios με στόχο την υπηρεσιοστρεφή λειτουργία του τελευταίου καθώς επίσης και μια πειραματική εφαρμογή διαχείρισης φόρτου εργασίας σε ιδιωτικά Νέφη.

Λέξεις κλειδιά: Υπηρεσιοστρεφείς Υποδομές, Υπολογιστικό Νέφος, Επίβλεψη πληροφορίας, Εικονικοποίηση, Ενεργειακή απόδοση

Abstract

The evolution of Cloud Computing and the offerings of the Service Oriented Infrastructures (SOIs) in general, allowed the on-demand software and resource providing as a service. The effective usage and the efficient management of infrastructures are raising multiple and complex demands as far as the collection and the management of information generated from those infrastructures concerns. To this end, the monitoring techniques and mechanisms in SOIs and Cloud environments form an important component of the value chain of those infrastructures.

This thesis is focused in this study of information management towards surpassing the limitations and restrictions that are introduced by the existence of different entities (Application Provider, Platform Provider and Infrastructure Provider) but also technical restrictions caused by this new architectural paradigm. More specifically, the analysis and design of innovative monitoring mechanisms is presented, managing information deriving from infrastructures as well as applications executed in those. Special focus has been given in the capability of re-configuration of the monitoring mechanism based on specific values of the collected information. This dynamic characteristic allows for a proactive violation detection of SLAs as well as the optimization of the infrastructure through better decision-making policies. The particular monitoring solution constitutes a service-oriented framework for data collection so much from the virtual and physical infrastructure. The policy of re-configuration is applied throughout this service framework which is evaluated with a real-life application scenario of a color correction video post-production.

Moreover, a second innovative monitoring infrastructure for Cloud environments is presented which aims at collecting information from various sources and assess it in order for an optimization of all Cloud entities. Through this work we focus on the collection of energy consumption of the resources and the calculation of the eco-efficiency of the infrastructure. Based on power models, the definition of eco-efficiency is provided for all Cloud entities and forecasted values are being calculated. This modeling leads to the application of policies that improve the placement of virtual machines (VMs) on the Cloud infrastructure. The presented results are very encouraging with regard to the possibilities of management of resources taking into consideration the eco-efficiency of infrastructure.

Finally, this thesis concludes with some additional contributions to the open source community, related with technologies and techniques of resource monitoring. In detail, an architecture and implementation of a monitoring plug-in of Nagios system is presented, which allows for a service-oriented operation of the pre-mentioned monitoring system. Furthermore, an experimental set-up of a Cloud-enabled elastic web hosting scenario is being described and evaluated, using only open-source software and applied in a private Cloud infrastructure.

Key words: Service Oriented Infrastructures, Cloud Computing, Information monitoring, Virtualization, Eco-efficiency

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:

- 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.

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 [1]. 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 επίβλεψη και παρακολούθηση 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 επίβλεψης. Από την άλλη, οι δυναμικές αρχιτεκτονικές των υπηρεσιοστρεφών συστημάτων και των περιβαλλόντων Νεφών, εγείρουν σημαντικές προκλήσεις στον σχεδιασμό και λειτουργία των διαδικασιών επίβλεψης.

Στο παρόν κεφάλαιο παρατίθεται αρχικά ο ορισμός του Υπολογιστικού Νέφους (*Cloud Computing*), τα βασικά χαρακτηριστικά του και οι διάφοροι τύποι Νεφών. Στη συνέχεια αναλύονται οι προκλήσεις όσον αφορά τον σχεδιασμό ενός αποδοτικού συστήματος επίβλεψης και παρακολούθησης και καταγράφεται η συνεισφορά μας σε αυτό το θέμα. Τέλος, παρουσιάζεται η οργάνωση της παρούσης διδακτορικής διατριβής και περιγράφεται συνοπτικά το περιεχόμενο των κεφαλαίων.

1.1 Ορισμός και χαρακτηριστικά του Υπολογιστικού Νέφους

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.

Σχήμα 1: 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.

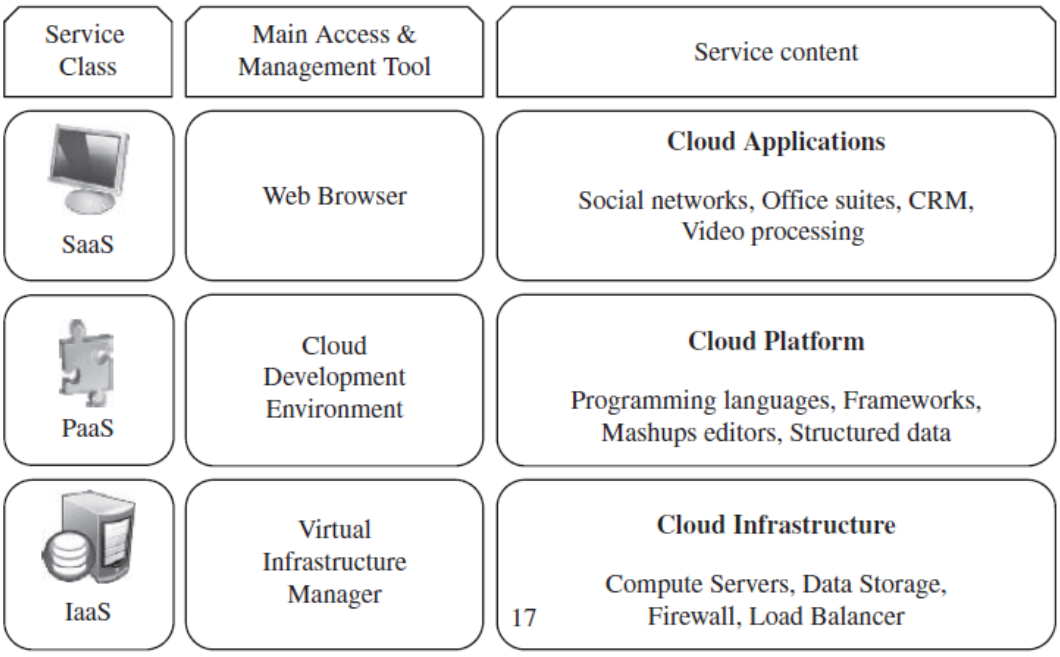
Ευρεία δικτυακή πρόσβαση. Οι δυνατότητες και οι υπηρεσίες αυτές είναι διαθέσιμες και προσβάσιμες μέσω του δικτύου χρησιμοποιώντας μηχανισμούς που προάγουν την χρήση ετερογενών προγραμμάτων/πλατφόρμων πελάτη (κινητά τηλέφωνα, ταμπλέτες, φορητοί υπολογιστές, σταθμούς εργασίας κ.α.).

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.



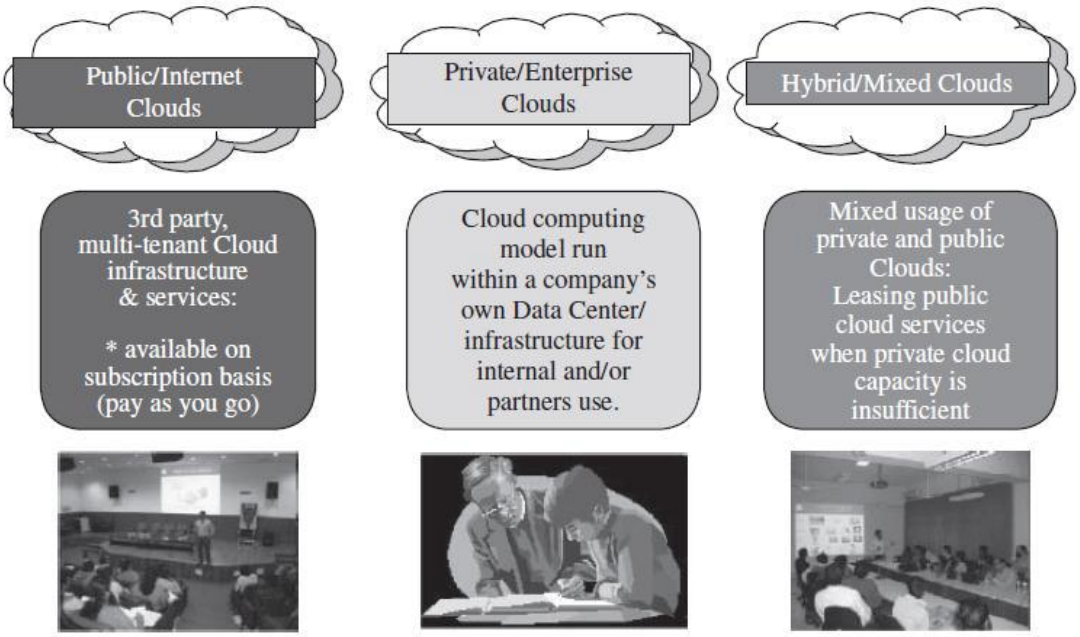
Σχήμα 2: Η στοίβα του υπολογιστικού Νέφους

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).



Σχήμα 3: Τύποι Νεφών βασισμένα σε μοντέλα ανάπτυξης

Μοντέλα ανάπτυξης:

Ιδιωτικό Νέφος (Private Cloud). 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 (Community cloud). 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).

1.2 Προκλήσεις επίβλεψης και παρακολούθησης σε Υπολογιστικά Νέφη

Ένα σύστημα επίβλεψης και παρακολούθησης πρέπει να σχεδιαστεί και υλοποιηθεί έτσι ώστε να είναι κατάλληλο για τον έλεγχο της υποδομής και των υπηρεσιών. Πρέπει να συμπεριλαμβάνει την όλη διαχείριση υπηρεσιών, και έτσι πρέπει να καλύπτει SLAs, ελαστικότητα, QoS, κ.λπ. Είναι σημαντικό να αναγνωριστεί ότι είναι ο μηχανισμός επίβλεψης που κλείνει το βρόχο από την αρχική εγκατάσταση, την εκτέλεση, και πίσω στην διαχείριση των υπηρεσιών. Το σύστημα επίβλεψης υπάρχει για να συγκεντρωθούν τα δεδομένα από όλα τα συστατικά μέσα σε μια αρχιτεκτονική Νεφών και έτσι είναι μια θεμελιώδης πτυχή ενός συστήματος υπηρεσιών που χρησιμοποιείται από την υποδομή και για τη διαχείριση υπηρεσιών. Ο μηχανισμός αυτός πρέπει να τροφοδοτήσει με αυτά τα δεδομένα το διαχειριστή υπηρεσιών έτσι ώστε να μπορεί ο τελευταίος να λάβει όλες τις απαραίτητες αποφάσεις για την αποδοτική λειτουργία μέσα στο περιβάλλον του Νέφους. Κατά συνέπεια χρειαζόμαστε το σύστημα επίβλεψης να είναι προσαρμόσιμο, εύκαμπτο, και επεκτάσιμο προκειμένου να υποστηριχθούν διάφορες καινοτόμες λειτουργίες. Για να εξετάσουμε όλες τις απαιτήσεις και τη λειτουργία ενός περιβάλλοντος υπηρεσιών Νεφών έχουμε καταγράψει τα κύρια χαρακτηριστικά γνωρίσματα και της προκλήσεις που ένα σύγχρονος μηχανισμός επίβλεψης πρέπει να καλύπτει:

- Εξελιξιμότητα (*scalability*) – έτσι ώστε να εξασφαλιστεί ότι ο μηχανισμός επίβλεψης μπορεί να αντιμετωπίσει μεγάλους αριθμούς αιτημάτων
- Ελαστικότητα (*elasticity*) - έτσι ώστε οι εικονικοί πόροι που δημιουργούνται και που καταστρέφονται με την επέκταση των δικτύων επιβλέπονται σωστά
- Μετανάστευση (*migration*) - έτσι ώστε οποιοσδήποτε εικονικός πόρος που κινείται από έναν φυσικό πόρο προς άλλο να επιβλέπεται σωστά
- Προσαρμοστικότητα (*adaptability*) - έτσι ώστε το πλαίσιο επίβλεψης και παρακολούθησης να μπορεί να προσαρμοστεί στον μεταβλητό υπολογιστικό και δικτυακό φόρτο
- Αυτονομία (*autonomic*) - έτσι ώστε το πλαίσιο επίβλεψης να μπορεί να συνεχίσει την αποτελεσματική λειτουργιά του χωρίς επέμβαση και επανασχηματισμό
- Ομοσπονδία (*federation*) - έτσι ώστε οποιοσδήποτε εικονικός πόρος που εγκαθίσταται σε μια άλλη υποδομή να επιβλέπεται σωστά
- Παρεμβατικότητα (*intrusiveness*) – ο μηχανισμός επίβλεψης να μην επηρεάζει την ομαλή και αποδοτική λειτουργία του συστήματος το οποίο παρακολουθεί

Για την καθιέρωση τέτοιων χαρακτηριστικών γνωρισμάτων σε ένα πλαίσιο επίβλεψης απαιτείται αρχικά μια αναλυτική καταγραφή και αξιολόγηση των υπαρχουσών μηχανισμών παρακολούθησης και στη συνέχεια μια προσεκτική σχεδίαση της αρχιτεκτονικής. Ένας μηχανισμός επίβλεψης αποτελείται από πολλαπλά συστατικά, κάθε ένα με τα δικά του χαρακτηριστικά και δυνατότητες. Ως εκ τούτου, ακόμα και η διαχείριση του ιδίου του μηχανισμού και των υπο-μονάδων αυτού αποτελεί μια πρόκληση.

1.3 Συνεισφορά

Η συνεισφορά της παρούσας διατριβής επικεντρώνεται στην προκλήσεις που παρουσιάστηκαν στην προηγούμενη παράγραφο αλλά επεκτείνεται επίσης και σε

επιπρόσθετα θέματα. Πιο αναλυτικά, γίνεται μια καταγραφή των λύσεων και μεθοδολογιών υλοποίησης IaaS υποδομών όπως και επίσης ανάλυση των υπαρχόντων μηχανισμών επίβλεψης περιβαλλόντων Νεφών και υποδομών SOI. Επίσης, προσδιορίζονται οι προδιαγραφές επίβλεψης και παρακολούθησης εφαρμογών και πόρων σε περιβάλλοντα Νεφών.

Η προηγούμενη ανάλυση καταλήγει στον σχεδιασμό και υλοποίηση ενός υπηρεσιοστρεφή μηχανισμού επίβλεψης και παρακολούθηση πόρων σε περιβάλλον Νέφους. Ο ιεραρχικός αυτός μηχανισμός αξιολογείται διεξοδικά μέσω συγκεκριμένων καταστάσεων χρήσης και παρατίθενται τα αποτελέσματα απόδοσης.

Επιπρόσθετα, σχεδιάζεται και υλοποιείται ένας δεύτερος μηχανισμός επίβλεψης και παρακολούθησης πολλαπλών πηγών δεδομένων. Σε αυτήν την λύση παρουσιάζεται και αναλύεται η μεθοδολογία συλλογής και συνάθροισης διαφορετικών τύπων δεδομένων. Επίσης, γίνεται ιδιαίτερη αναφορά στην συλλογή και επεξεργασία δεδομένων σχετικών με την ενεργειακή κατανάλωση της υποδομής και το κατά πόσον αυτά τα στοιχεία μπορούν να έχουν αντίκτυπο στην διαχείριση των πόρων της υποδομής.

Τέλος, η διατριβή αυτή συνεισφέρει συγκεκριμένες λύσεις και μηχανισμούς για την συλλογή δεδομένων από εξειδικευμένες υποδομές και την διαχείριση δεδομένων με στόχο την καλύτερη λειτουργία της υποδομής

Στα πλαίσια της εργασίας αυτής προέκυψαν οι παρακάτω δημοσιεύσεις:

Περιοδικά με κρίση

1. Katsaros, G.; Kousiouris, G.; Gogouvitis, S.; Kyriazis, D.; Menychtas, A.; Varvarigou, T.;,"A Self-adaptive hierarchical monitoring mechanism for Clouds", Journal of Software and Systems (JS&S), 2012

2. Kuebert, R.; Katsaros, G.;, "Using Free Software for Elastic Web Hosting on a Private Cloud", International Journal of Cloud Applications and Computing (IJCAC), 2011.

3. Gogouvitis, S.; Konstanteli, K.; Waldschmidt, S.; Kousiouris, G.; Katsaros, G.; Menychtas, A.; Kyriazis, D.; Varvarigou, T.;, "Workflow management for soft real-time interactive applications in virtualized environments", Future Generation Computer Systems, June 2011.

4. Kousiouris, G.; Menychtas, A.; Kyriazis, D.; Konstanteli, K.; Gogouvitis, S.; Katsaros, G.; Varvarigou, T.;, " Dynamic Design and Performance Analysis of a Multi-Layered, Decoupled Service-Oriented Framework for incorporating numerical software in SOI Performance Prediction ", under revisions for IEEE Transactions on Services Computing

5. Koutsoutos S.; Katsaros G.; Gogouvitis S.; Varvarigou T.;, "An Architecture for an Efficient and Reprogrammable Cloud Monitoring supporting high level Decision Making Algorithms", under review for Journal of Software and Systems (JS&S).

6. Katsaros, G.; Varvarigou, T.; Subirats J.; Fito, O.; Guitart, J.; Gilet, P.; Espling, D.;, "A Service Framework for Energy-aware Monitoring and VM Management in Clouds", under review for Future Generation Computer Systems, Special Issue in Cloud Monitoring Systems.

Κεφάλαια σε βιβλία

1. Hasselmeyer, P.; Katsaros, G.; Koller, B.; Wieder, P.;, "Cloud Monitoring", in Book "Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice",IGI Global, 2012.

2. Tordsson, J.; Djemame, K.; Henriksson, D.; Katsaros, G.; Ziegler, W.; Waldrich, O.; Konstanteli, K.; Sajjad, A.; Rajarajan, M.; Gallizo, G.; Nair, S.;, "Towards holistic

Cloud management", in Book "European Research Activities in Cloud Computing", Cambridge Scholars Publishing, Dec. 2011.

3. Katsaros, G.; Gallizo, G.; Kuebert, R.; Wang, T.; Fito, O.; Espling, D.;, "An integrated monitoring infrastructure for Cloud environments", in Book "Lecture Notes in Business Information Processing" (LNBIP), Springer-Verlag, Sep. 2011.

4. Katsaros, G.; Kuebert, R.; Gallizo, G.; Wang, T.;, "Monitoring: A Fundamental Process to Provide QoS Guarantees in Cloud-based Platforms", in Book "Cloud computing: methodology, systems, and applications", CRC, Taylor & Francis group, September 2011.

5. Katsaros, G.; Cucinotta, T.;, "Programming Interfaces for Realtime and Cloud-based Computing", in Book "Achieving Real-Time in Distributed Computing: From Grids to Clouds", IGI Global, July 2011.

6. Gogouvitis, S.; Konstanteli, K.; Kyriazis, D.; Katsaros, G.; Cucinotta, T.; Boniface, M.;, "Workflow Management Systems in Distributed Environments", in Book "Achieving Real-Time in Distributed Computing: From Grids to Clouds", IGI Global, July 2011.

Διεθνή επιστημονικά συνέδρια

1. Katsaros, G.; Kuebert, R.; Gallizo, G.;, "Building a service-oriented monitoring framework with REST and Nagios", IEEE International Conference on Services Computing (SCC), July 2011.

2. Katsaros, G.; Gallizo, G.; Kuebert, R.; Wang, T.; Fito, O.; Henriksson, D.;, "A multi-level architecture for collecting and managing monitoring information in Cloud

environments", Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER), May 2011.

3. Katsaros, G.; Kuebert, R.; Wang, T.;, "A RESTful implementation of the WS-Agreement specification", Proceedings of the Second International Workshop on RESTful Design at WWW 2011, March 2011.

4. Katsaros, G.; Kousiouris, G.; Gogouvitis, S.; Kyriazis, D.; Varvarigou, T.; , "A service oriented monitoring framework for soft real-time applications", IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Dec. 2010.

5. Gallizo, G.; Kuebert, R.; Oberle, K.; Satzke, K.; Gogouvitis, S.; Katsaros, G.; Oliveros, E.;," A service level agreement management framework for real-time applications in cloud computing environments", Proceedings of the 2nd International ICST Conference on Cloud Computing, October 2010.

6. Gogouvitis, S.V.; Konstanteli, K.; Kousiouris, G.; Katsaros, G.; Kyriazis, D.; Varvarigou, T.;, "A Service Oriented Architecture for achieving QoSaware Workflow Management in Virtualized Environments", International Conference on Network and Service Management (CNSM), Oct. 2010.

7. Kousiouris, G.; Kyriazis, D.; Konstanteli, K.; Gogouvitis, S.; Katsaros, G.; Varvarigou, T.;, "A Service-Oriented Framework for GNU Octave-Based Performance Prediction", IEEE International Conference on Services Computing (SCC), July 2010.

8. Boniface, M.; Nasser, B.; Papay, J.; Phillips, S.C.; Servin, A.; Xiaoyu Yang; Zlatev, Z.; Gogouvitis, S.V.; Katsaros, G.; Konstanteli, K.; Kousiouris, G.; Menychtas, A.; Kyriazis, D.;, "Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds", Fifth International Conference on Internet and Web Applications and Services (ICIW), May 2010.

9. Katsaros, G.; Antonopoulos, S.; Kyriazis, D.; Varvarigou, T.;, "Service oriented license providing", IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Jan. 2009.

10. Kyriazis, D.; Tserpes, K.; Kousiouris, G.; Menychtas, A.; Katsaros, G.; Varvarigou, T.;, "Data Aggregation and Analysis: A Grid-Based Approach for Medicine and Biology", International Symposium on Parallel and Distributed Processing with Applications (ISPA), Dec. 2008.

11. Katsaros, G.; Campos, F.; Kyriazis, D.; Varvarigou, T.;, "Computational Fluid Dynamics Automatic Optimization using OpenFOAM in Grid Environments"ù, OpenFOAM International Conference, November 2007.

1.4 Οργάνωση του εγγράφου

Το παρόν έγγραφο αποτελείται από οχτώ (8) κεφάλαια. Στις ενότητες των κεφαλαίων αυτών παρουσιάζεται ουσιαστικά και με αναλυτικό τρόπο το αντικείμενο της διδακτορικής διατριβής και η μέχρι τώρα πρόοδος αυτής. Το κεφάλαιο 2 παρέχει γενικές πληροφορίες για τις υπηρεσιοστρεφείς αρχιτεκτονικές, τα περιβάλλοντα Νεφών και σχετικές τεχνολογίες αυτών. Στο κεφάλαιο 3, παραθέτουμε μια εισαγωγή στο κύριο θέμα της διατριβής (παρακολούθηση και επίβλεψη λειτουργίας) καταγράφοντας τις λύσεις που υπάρχουν διαθέσιμες και αναλύοντας τις προδιαγραφές ενός σύγχρονου συστήματος επίβλεψης για

περιβάλλοντα Νεφών. Στο κεφάλαιο 4 παρουσιάζουμε την πρώτη προτεινόμενη αρχιτεκτονική, περιγράφουμε την υλοποίηση και αξιολογούμε την απόδοση βάσει συγκεκριμένων πειραμάτων που εκπονήθηκαν. Το κεφάλαιο 5 παρουσιάζει την δεύτερη αρχιτεκτονική προσέγγιση, στην οποία δίνεται έμφαση στον υπολογισμό και αξιολόγηση της ενεργειακής απόδοσης της υποδομής. Στο κεφάλαιο 6 παρουσιάζονται κάποιες επιπρόσθετες τεχνικές και τεχνολογικές συνεισφορές που αφορούν στην επίβλεψη και την διαχείριση υποδομών Νεφών. Τέλος, στο κεφάλαιο 7 δίνεται μια σύνοψη της διατριβής ενώ το κεφάλαιο 8 περιλαμβάνει Βιβλιογραφικές Αναφορές.

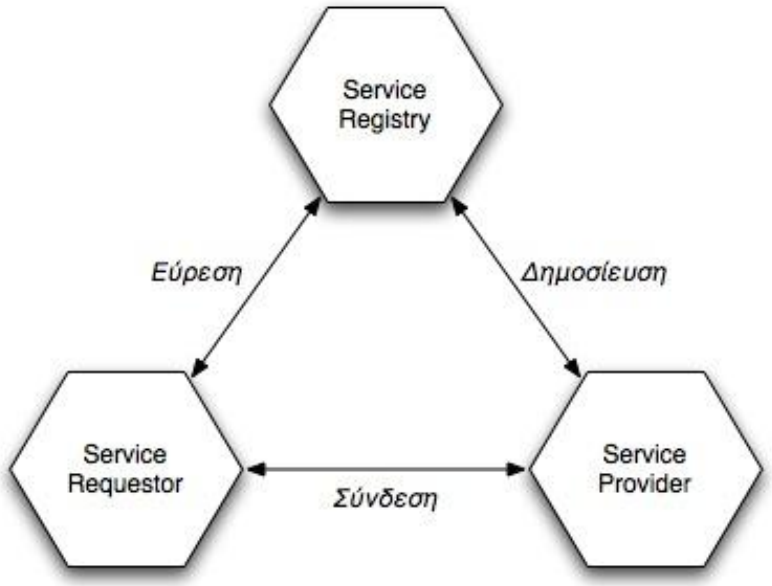
2 *Υπηρεσιοστρεφείς αρχιτεκτονικές και περιβάλλοντα Νεφών*

Στο κεφάλαιο αυτό παρουσιάζονται οι βασικές αρχές και η δομή της υπηρεσιοστρεφούς αρχιτεκτονικής όπως επίσης και οι τεχνολογίες πάνω στις οποίες βασίστηκε η υλοποίηση τέτοιων συστημάτων. Το μοντέλο αυτής της αρχιτεκτονικής συστημάτων και οι τεχνολογίες ανάπτυξης υπηρεσιών αποτέλεσαν τον τεχνολογικό πυρήνα για την δημιουργία περιβαλλόντων Νεφών. Στη συνέχεια του κεφαλαίου αυτού καταγράφουμε 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. Επίσης, παρουσιάζονται προυπάρχουσες τεχνολογίες που συνεισέφεραν τόσο στην ανάπτυξη περιβαλλόντων Νεφών αλλά και των μηχανισμών επίβλεψης και παρακολούθησης που αυτή η διατριβή εξετάζει.

2.1 Υπηρεσιοστρεφής αρχιτεκτονική

Η προσαρμογή κάποιων υπηρεσιών σε μια εφαρμογή δεν είναι μια δύσκολη διαδικασία και επιπρόσθετα αναμένεται να αποδώσει στην εφαρμογή πρόσθετα λειτουργικά χαρακτηριστικά. Τα χαρακτηριστικά αυτά όμως δεν δημιουργούν μια υπηρεσιοστρεφή αρχιτεκτονική (*Service Oriented Architecture* - SOA), αφού η διαφορά τους είναι πιο μεγάλη.

Το πρότυπο υπηρεσιοστρεφούς αρχιτεκτονικής είναι ένα σχεδιαστικό μοντέλο με κύριο χαρακτηριστικό την ενσωμάτωση λογικής εφαρμογών μέσα σε υπηρεσίες που θα αλληλεπιδρούν μέσω επικοινωνιακών πρωτοκόλλων. Βάσει αυτού, η υιοθέτηση σε μια εφαρμογή, SOA δομής σημαίνει αυτόματα την αποδοχή κάποιων σχεδιαστικών αρχών και πρόσθετων τεχνολογιών ως βασικού τμήματος του τεχνικού περιβάλλοντος της.



Σχήμα 4: Δομή υπηρεσιοστρεφής αρχιτεκτονικής

Σε επίπεδο σχεδιασμού συστημάτων, η χρήση της τεχνολογίας των *Web Services* οδηγεί στην υιοθέτηση της λεγόμενης *Service-Oriented* αρχιτεκτονικής. Οι βασικοί ρόλοι και λειτουργίες στην αρχιτεκτονική αυτή παρουσιάζονται στο Σχήμα 4. Η αρχιτεκτονική αυτή υποδεικνύει μια σχέση εξυπηρετητή-πελάτη (*server-client*) ανάμεσα στον προμηθευτή υπηρεσιών (*service provider*, που παίζει το ρόλο του *server*) και τον ζητών της υπηρεσία (*service-requestor*, που

παίζει το ρόλο του *client*). Ο *service-provider* είναι αυτός που παρέχει την υπηρεσία δεχόμενος μηνύματα κλήσεις από τους *requestors*. Είναι επίσης υπεύθυνος για τη δημιουργία της περιγραφής της υπηρεσίας (*service description*) και τη δημοσίευσή της σε κάποιο κατάλογο-οδηγό υπηρεσιών (*Universal Description, Discovery and Integration*- UDDI). Ο *service requestor* αναζητά την υπηρεσία και την περιγραφή της σε κάποιο κατάλογο υπηρεσιών (*service registry*) και στη συνέχεια καλεί κατάλληλα την επιθυμητή υπηρεσία. Ο κατάλογος υπηρεσιών φέρνει ουσιαστικά τις δύο πλευρές, *client* και *server* σε επαφή. Η συνέχεια αφορά μόνο τις δύο άλλες συμμετέχουσες μονάδες (*service requestor* και *service provider*).

2.1.1 Γενικά για τις υπηρεσίες

Τον τελευταίο καιρό έχει γίνει αρκετά μεγάλη συζήτηση γύρω από το θέμα των υπηρεσιών των εφαρμογών. Οι υπηρεσίες τείνουν να γίνουν τμήματα της εφαρμογής που αθροιστικά σχηματίζουν το περιβάλλον αυτής. Φυσικά δεν αποτελούν απλά ένα κομμάτι της εφαρμογής, αλλά έχουν χαρακτηριστικά που τις μετατρέπουν σε μέρος μιας αρχιτεκτονικής προσανατολισμένης στις υπηρεσίες (*Service Oriented Architecture*).

Ένα από τα χαρακτηριστικά αυτά είναι η αυτονομία από άλλες υπηρεσίες. Αυτό σημαίνει ότι κάθε υπηρεσία είναι υπεύθυνη για το δικό της εύρος λειτουργίας, περιορίζοντας έτσι και εξειδικεύοντάς την σε συγκεκριμένες επαγγελματικές χρήσεις. Αυτός ο σχεδιασμός έχει ως αποτέλεσμα τη δημιουργία ανεξάρτητων μονάδων, ελαστικά συνδεδεμένων μεταξύ τους με κάποιο πρότυπο πλαίσιο επικοινωνίας. Εξαιτίας αυτής της ανεξαρτησίας που απολαμβάνουν οι υπηρεσίες στο πλαίσιο αυτό, η προγραμματιστική λογική που κάθε υπηρεσία χρησιμοποιεί, δεν χρειάζεται να προσαρμόζεται σε συγκεκριμένη πλατφόρμα ή τεχνολογία. Είναι χαρακτηριστικό των υπηρεσιών ενός πλαισίου, ο πολυμορφισμός των μερών του με ταυτόχρονη ομαλή συνεργασία.

2.1.2 Διαδικτυακές Υπηρεσίες

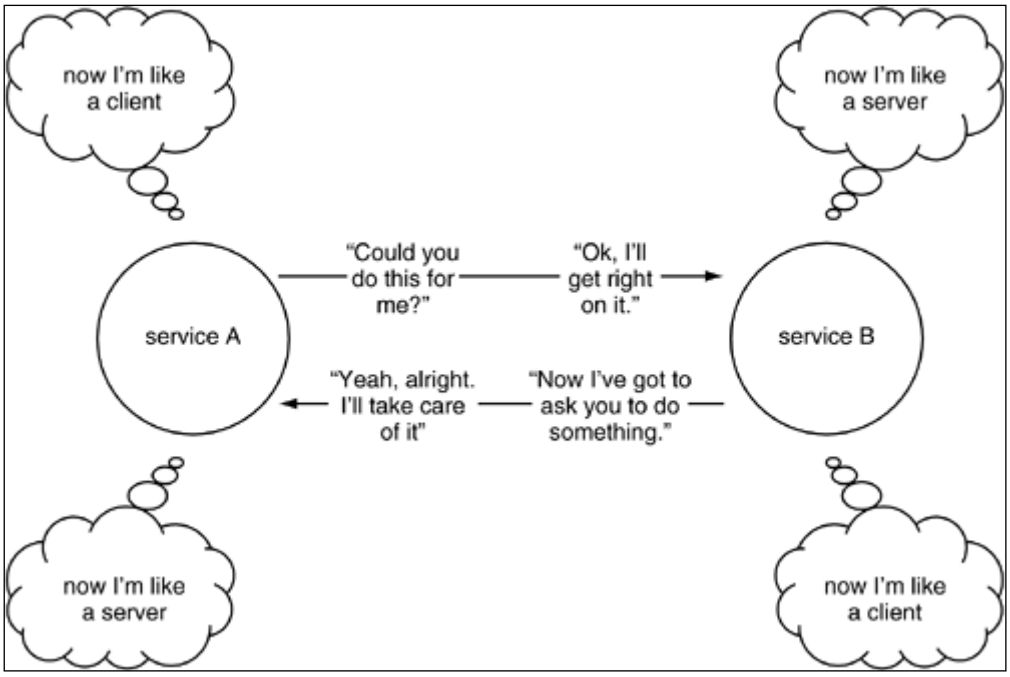
Ο πιο ευρέως διαδεδομένος και επιτυχημένος τύπος υπηρεσιών είναι οι διαδικτυακές υπηρεσίες XML *Services* γνωστές ως *Web Services*. Αυτός ο τύπος υπηρεσίας έχει δύο βασικές προαπαιτήσεις:

- επικοινωνεί μέσω πρωτοκόλλου Internet (κυρίως HTTP)
- στέλνει και δέχεται δεδομένα μέσα από XML αρχεία

Η ευρεία αποδοχή του μοντέλου των Web Services είχε ως αποτέλεσμα την ανάγκη πρόσθετων τεχνολογιών που βασίζονται σε αυτές και τη δημιουργία καινούργιων προτύπων. Έτσι η "παραγωγή" τέτοιων υπηρεσιών απαιτεί:

- τη περιγραφή της υπηρεσίας αναλυτικά, τουλάχιστον με ένα WSDL έγγραφο
- τη δυνατότητα μεταφοράς ενός XML εγγράφου χρησιμοποιώντας SOAP μέσω HTTP.

Επιπρόσθετα είναι συνηθισμένο μια υπηρεσία να λειτουργεί και ως πελάτης (*client/requestor*) και ως πάροχος (*provider*) υπηρεσίας. Αναλόγως λοιπόν με τη δραστηριότητα της υπηρεσίας κάθε στιγμή, μετατρέπεται από το ένα στο άλλο. Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα αυτής της συμπεριφοράς:



Σχήμα 5: Ανταλλαγή ρόλων μεταξύ Web services κατά τη διάρκεια μιας επικοινωνίας

2.1.3 Web Services Description Language (WSDL)

Οι *Web Services* χρειάζεται να ορίζονται με συγκεκριμένο τρόπο έτσι ώστε να μπορούν να εντοπιστούν και χρησιμοποιηθούν από άλλες υπηρεσίες και εφαρμογές. Για αυτό το σκοπό δημιουργήθηκε από τον οργανισμό W3C, μια γλώσσα περιγραφής γνωστή ως *Web Services Description Language* (WSDL).

Η γλώσσα αυτή είναι μια XML δομή εγγράφου, που περιέχει όλες τις πληροφορίες σχετικά με τα δεδομένα εισόδου και εξόδου, τις μεθόδους και ότι άλλο χρειάζεται να γνωρίζει κάποιος για την ακριβή χρήση μιας διαδικτυακής υπηρεσίας.

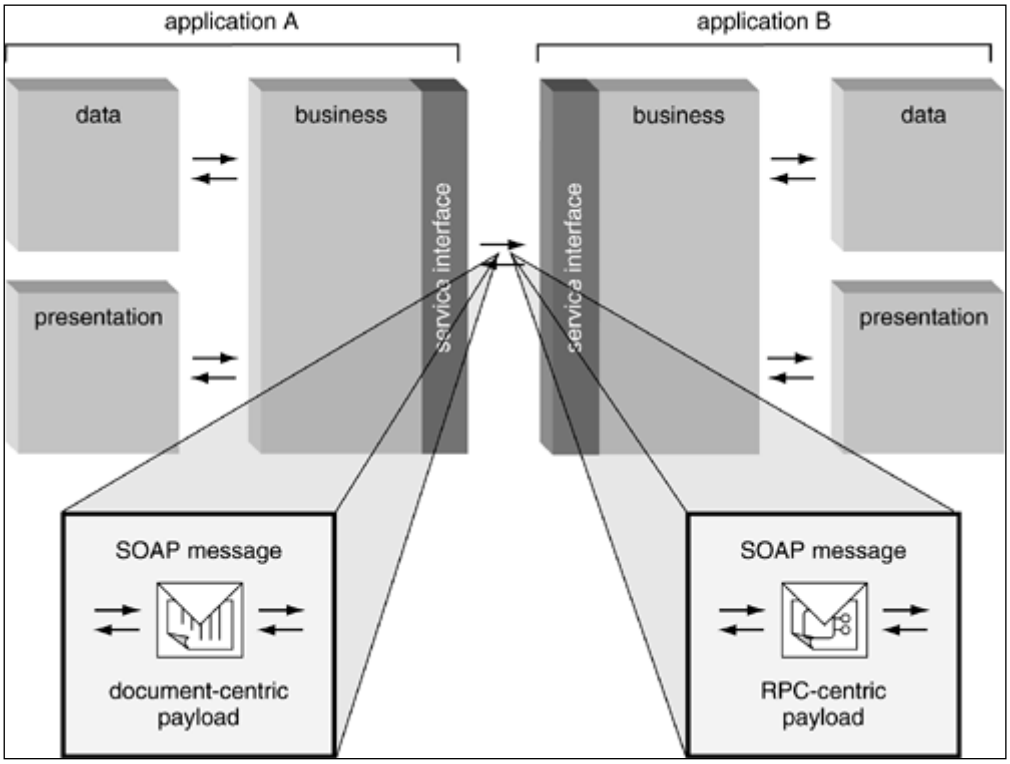
2.1.4 Simple Object Access Protocol (SOAP)

Αν και αρχικά είχε θεωρηθεί ως η τεχνολογία που θα γεφυρώσει το κενό μεταξύ ανόμοιων πλατφορμών βασισμένων σε RPC επικοινωνία, το SOAP έχει εξελιχθεί στο ευρέως χρησιμοποιούμενο πρότυπο επικοινωνίας για τη χρήση XML Υπηρεσιών Διαδικτύου (*Web Services*). Μετά από αυτή την κατάσταση γίνεται πολλές φορές η παράφραση του ακρωνύμου

από *Simple Object Access Protocol* σε *Service-Oriented Architecture (or Application) Protocol*.

Το πρωτόκολλο SOAP διαμορφώνει ένα πρότυπο μήνυμα που αποτελείται από ένα XML έγγραφο ικανό να περιγράψει δεδομένα όπως RPC κλήσεις κ.α. Το μήνυμα αυτό μεταφέρεται μεταξύ των υπηρεσιών και των εφαρμογών, χρησιμοποιώντας κυρίως το HTTP πρωτόκολλο δικτύου. Με τον τρόπο αυτό ολοκληρώνεται το πλαίσιο λειτουργίας και επικοινωνίας στην SOA δομή, αφού με την βοήθεια της περιγραφής WSDL είναι εφικτή η επικοινωνία και συνεργασία οποιωνδήποτε υπηρεσιών στο δίκτυο.

Στο παρακάτω σχήμα (Σχήμα 6) βλέπουμε τη χρήση του πρωτοκόλλου σε εφαρμογές είτε για προτυποποιημένη RPC επικοινωνία είτε για γενική χρήση μεταφοράς μηνύματος.



Σχήμα 6: Χρήση SOAP πρωτοκόλλου σε εφαρμογές

Το πρωτόκολλο SOAP καθορίζει ένα XML έγγραφο με μια συγκεκριμένη δομή που μέσα του θα εμπεριέχονται οι πληροφορίες. Η βασική δομή ενός τέτοιου εγγράφου είναι η παρακάτω:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    …
  </soap:Body>
</soap:Envelope>
```

Σχήμα 7: Δομή μηνύματος SOAP

Εσωτερικά του "σώματος" (*Body*) του εγγράφου, ενσωματώνεται η πληροφορία που θέλουμε να μεταφέρουμε από τη μία υπηρεσία στην άλλη. Η τεχνολογία που χρησιμοποιείται για απομακρυσμένη κλήση υπηρεσιών είναι η SOAP-RPC, (επέκταση της τεχνολογίας *Remote Procedure Call*) η οποία εισάγει στο SOAP έγγραφο τις απαιτούμενες πληροφορίες, σύμφωνα και με το WSDL της υπηρεσίας που θα κληθεί και στέλνει το μήνυμα. Επιπλέον, όπως φαίνεται και στο Σχήμα 7, εκτός από την χρήση που περιγράφηκε, το SOAP μπορεί να μεταφέρει οποιαδήποτε άλλη πληροφορία ενσωματωμένη σε XML δομή, μέσα στο "φάκελο" του SOAP μηνύματος, αρκεί βέβαια ο παραλήπτης να γνωρίζει πώς να αποκωδικοποιήσει το έγγραφο αυτό.

2.1.5 Αρχιτεκτονική προσανατολισμένη σε υπηρεσίες

Η υπηρεσία είναι μία οντότητα που παρέχει λειτουργίες στους πελάτες της μέσω ανταλλαγής μηνυμάτων. Καθώς η λειτουργία μίας υπηρεσίας ορίζεται από ακολουθίες συγκεκριμένων μηνυμάτων και μόνο, παρέχεται μεγάλη ευελιξία στην υλοποίηση και στον εντοπισμό της. Σε μία αρχιτεκτονική προσανατολισμένη σε υπηρεσίες, όλες οι οντότητες είναι υπηρεσίες, και άρα κάθε λειτουργία που παρέχεται από μία τέτοια αρχιτεκτονική είναι αποτέλεσμα ανταλλαγής μηνυμάτων.

Τα παρακάτω παραδείγματα αποδεικνύουν την ευρεία εφαρμογή των υπηρεσιών, οι οποίες περιλαμβάνουν από χαμηλού επιπέδου διαχείριση πόρων (*storage service*) μέχρι υψηλού επιπέδου συστήματα παρακολούθησης απόδοσης.

- Μία υπηρεσία αποθήκευσης μπορεί να υλοποιεί τις λειτουργίες που πραγματοποιούν την αποθήκευση και ανάκτηση δεδομένων, την παρακολούθηση της κατάστασης της

υπηρεσίας, και τον ορισμό και την πρόσβαση στις πολιτικές που ελέγχουν ποιος επιτρέπεται να χρησιμοποιήσει την υπηρεσία αυτή.

- Μια υπηρεσία μεταφοράς δεδομένων μπορεί να παρέχει λειτουργίες που πραγματοποιούν τη μεταφορά των δεδομένων από μία υπηρεσία αποθήκευσης σε μία άλλη, διαχειρίζοντας και παρακολουθώντας την κατάσταση της μεταφοράς, και ορίζοντας και παρέχοντας πρόσβαση στις πολιτικές που ελέγχουν την προτεραιότητα των μεταφορών.
- Μία υπηρεσία επίλυσης προβλημάτων μπορεί να παρακολουθεί την κατάσταση ποικίλων άλλων υπηρεσιών, όπως είναι οι προαναφερόμενες, και να παρέχει λειτουργίες που επιτρέπουν σε άλλες οντότητες (υπηρεσίες) να απαιτήσουν γνωστοποιήσεις (*notifications*) σχετικά με σφάλματα που μπορεί να προκύψουν και τέλος να ορίσουν και να παρέχουν πρόσβαση στις πολιτικές που καθορίζουν ποιος επιτρέπεται να παίρνει αυτές τις γνωστοποιήσεις.

Από τα παραπάνω παραδείγματα εξάγονται δύο σημαντικά συμπεράσματα που αφορούν την προσανατολισμένη στις υπηρεσίες αρχιτεκτονική. Κατά πρώτον, κοινές λειτουργίες, όπως η παρακολούθηση της απόδοσης και ο ορισμός και η πρόσβαση στις πολιτικές, εμφανίζονται σε διαφορετικές υπηρεσίες. Στόχος της σχεδίασης της συγκεκριμένης αρχιτεκτονικής είναι η διατύπωση τέτοιων λειτουργιών ανεξάρτητα από τις ειδικές περιπτώσεις που χρησιμοποιούνται, έτσι ώστε να απλοποιηθεί η σχεδίαση των εφαρμογών και να ενισχυθεί η επαναχρησιμοποίηση κώδικα.

Προς επίτευξη αυτού του σκοπού, πολλές λειτουργίες ομαδοποιούνται συνήθως μεταξύ τους, ώστε να σχηματίσουν μία διεπαφή υπηρεσίας (*service interface*). Οι διεπαφές μπορούν κατόπιν να συνδυαστούν για να ορίσουν μία άλλη υπηρεσία με τις επιθυμητές λειτουργίες. Κατά δεύτερον, βλέπουμε ένα παράδειγμα υψηλού επιπέδου λειτουργίας υπηρεσίας (μεταφορά δεδομένων), να υλοποιείται από τη σύνθεση απλούστερων υπηρεσιών (υπηρεσία αποθήκευσης). Η διευκόλυνση στη σύνθεση υπηρεσιών είναι ένας δεύτερος σημαντικός στόχος της σχεδίασης της αρχιτεκτονικής. Εισάγοντας λειτουργίες υπηρεσιών

μέσα σε ένα πλαίσιο ανταλλαγής μηνυμάτων, οι προσανατολισμένες σε υπηρεσίες αρχιτεκτονικές διευκολύνουν την παραγωγή εικονικών υπηρεσιών, απομονώνοντας τον χρήστη από λεπτομέρειες που αφορούν την εγκατάσταση υπηρεσιών και τον εντοπισμό τους. Παραδείγματος χάριν, η υπηρεσία αποθήκευσης που αναφέρθηκε προηγουμένως, η οποία παρέχει στον χρήστη μία διεπαφή που ορίζει, μεταξύ άλλων, μία λειτουργία αποθήκευσης αρχείου. Ένας χρήστης θα πρέπει να είναι σε θέση να καλέσει τη λειτουργία αυτή σε ένα συγκεκριμένο αντικείμενό της, χωρίς να χρειάζεται να ξέρει πώς σχετίζεται αυτό το αντικείμενο με την διεπαφή της υπηρεσίας αποθήκευσης. Διαφορετικές υλοποιήσεις μπορεί να αποθηκεύουν το αρχείο στον τοπικό υπολογιστή του χρήστη, σε ένα κατανεμημένο σύστημα, σε ένα απομακρυσμένο σύστημα αποθήκευσης αρχείων ή να επιλεγεί μεταξύ αυτών των εναλλακτικών ανάλογα με το εκάστοτε σενάριο, το φορτίο, το ποσό πληρωμής και άλλοι συντελεστές. Ανεξάρτητα από το πώς θα γίνουν αυτές οι υλοποιήσεις, ο χρήστης γνωρίζει μόνο ότι πραγματοποιούνται, με ποικίλες μάλιστα ποιότητες υπηρεσίας και συντελεστές που μπορεί να είναι αντικείμενο διαπραγμάτευσης μεταξύ πελάτη και υπηρεσίας.

Η αλληλεπίδραση με μία υπηρεσία επιτυγχάνεται με τη χρήση μίας γλώσσας περιγραφής διεπαφής (*interface definition language*), όπως είναι η WDSL, η οποία περιγράφει τη διεπαφή της υπηρεσίας. Ένα WDSL ορίζει τα μηνύματα που η υπηρεσία παράγει και δέχεται, χωρίς όμως να περιγράφει τι ακριβώς κάνει η υπηρεσία σε απάντηση της κλήσης της. Μία καλά ορισμένη WDSL διεπαφή και ο διαχωρισμός μεταξύ της διεπαφής της υπηρεσίας και της υλοποίησής της, απλοποιούν τη χρήση και διαχείριση των υπηρεσιών σε τέσσερις περιπτώσεις: ανακάλυψη υπηρεσιών, σύνθεση, εξειδίκευση, και επέκταση διεπαφής.

- Η ανακάλυψη υπηρεσιών είναι σημαντική στα κατανεμημένα συστήματα επειδή συχνά απαιτούνται υπολογισμοί σε ανεξερεύνητα περιβάλλοντα, στα οποία η ταυτότητα και τα χαρακτηριστικά των διαθέσιμων υπηρεσιών είναι άγνωστα. Σε μία αρχιτεκτονική προσανατολισμένη σε υπηρεσίες μπορούν να χρησιμοποιηθούν κατάλογοι μητρώων, που να περιέχουν πληροφορίες σχετικά με τη θέση ζητούμενων υπηρεσιών.

- Η σύνθεση υπηρεσιών από άλλες είναι επίσης σημαντική, καθώς επιτρέπει την επαναχρησιμοποίηση κώδικα και τη δυναμική υλοποίηση πολύπλοκων συστημάτων από απλούστερα στοιχεία. Μία καλά ορισμένη γλώσσα περιγραφής διεπαφής (WDSL) απλουστεύει τη σύνθεση επειδή ένας πελάτης χρειάζεται να γνωρίζει μόνο τον τρόπο που θα καλέσει την υπηρεσία.
- Η εξειδίκευση αναφέρεται στη χρήση διαφορετικών υλοποιήσεων μίας διεπαφής υπηρεσίας για διαφορετικές πλατφόρμες.
- Η δυνατότητα επέκτασης της διεπαφής είναι ένα σημαντικό χαρακτηριστικό μίας περιγραφικής γλώσσας διεπαφής, καθώς επιτρέπει σε εξειδικευμένες υλοποιήσεις να εισάγουν επιπρόσθετη λειτουργικότητα, ενώ παράλληλα θα υποστηρίζεται η υπάρχουσα διεπαφή. Οι υλοποιήσεις μπορούν να ανταγωνίζονται η μία την άλλη μέσω της προστιθέμενης αξίας επεκτάσεων που θα εισάγουν στις διεπαφές υπηρεσιών, και όχι μέσω της προστιθέμενης αξίας υλοποιήσεων νέων διεπαφών. Καταυτόν τον τρόπο είναι δυνατό να ικανοποιούνται διαφορετικές ποιότητες υπηρεσίας.

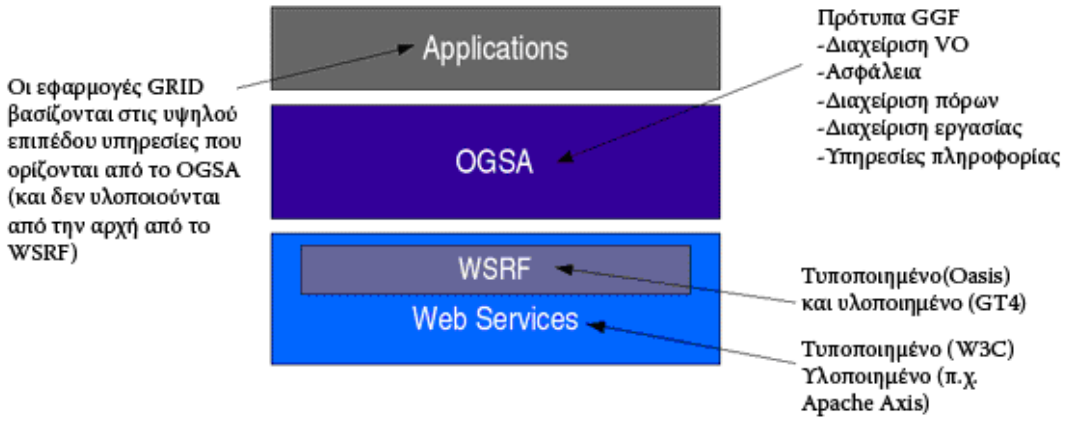
Τα δύο βασικά πρότυπα/τεχνολογίες αρχιτεκτονικών προσανατολισμένων στις υπηρεσίες δικτύου που υπάρχουν διαθέσιμα σήμερα είναι το WSRF [2] και το REST [3]. Το WSRF χρησιμοποιεί την αρχιτεκτονική (και τυποποίηση) που εισάγει το OGSA (είδη υπηρεσιών, σχέσεις μεταξύ τους κλπ.), αλλά ορίζει την υποδομή (τον τρόπο ορισμού και κατασκευής των υπηρεσιών) με διαφορετικό τρόπο από το OGSI . Το REST δεν είναι επίσημο πρότυπο αλλά ένα παράδειγμα προγραμματισμού υπηρεσιών διαδικτύου που βασίζεται στο HTTP πρωτόκολλο και είναι πολύ διαδεδομένο στις υπηρεσίες Νέφους. Τα δύο αυτά πρότυπα θα αναλυθούν στις επόμενες ενότητες του κεφαλαίου.

2.1.6 Το πλαίσιο WSRF (Web Service Resource Framework)

Βασικός άξονας της προσέγγισης των υπολογιστικών πλεγμάτων που εισάγει το WSRF είναι η δημιουργία πόρων με δυνατότητα αποθήκευσης της κατάστασής τους . Ένας πόρος (*WS-Resource*):

- Περιλαμβάνει ένα συγκεκριμένο σύνολο δεδομένων της κατάστασής του, αποθηκευμένα σε XML μορφή.
- έχει έναν καλά ορισμένο κύκλο ζωής.
- υπόκειται σε ανακάλυψη και χρήση από πολλές υπηρεσίες.

Ένας πόρος λοιπόν, με δυνατότητα αποθήκευσης κατάστασης, δεν είναι ο ίδιος μία υπηρεσία ιστού, αλλά ελέγχεται από υπηρεσίες. Η κατάσταση ενός πόρου ορίζεται από τις τιμές που περιέχονται σε ένα ξεχωριστό κείμενο ιδιοτήτων του πόρου (*WS-Resource Properties Document*). Σημαντική διαφορά μεταξύ του OGSI και του WSRF είναι ότι ενώ στο πρώτο, ένας πόρος αντιπροσωπεύεται από μία υπηρεσία πλέγματος (*Grid Service*), η οποία είναι μία επέκταση των ιδιοτήτων των υπηρεσιών δικτύου (*Web Services*), και ενθυλακώνει την κατάσταση του πόρου η ίδια (μέσω των SDEs), στο WSRF χρησιμοποιούνται κανονικές υπηρεσίες δικτύου, και η κατάσταση ενός πόρου αποθηκεύεται σε ξεχωριστό αρχείο το οποίο προσπελαύνουν διαφορετικά *interfaces*. Επίσης διαφορές υπάρχουν στους τρόπους που αναγνωρίζονται, διαχειρίζονται και ομαδοποιούνται τα στιγμιότυπα πόρων όπως θα δούμε στη συνέχεια.



Σχήμα 8: Σχέσεις των υπηρεσιών ιστού, του WSRF και του OGSA, και πώς συνδυάζονται μεταξύ τους για να χρησιμοποιηθούν από τις εφαρμογές

Η υποδομή WSRF δεν υλοποιεί νέες υπηρεσίες πλέγματος, αλλά χρησιμοποιεί την τυποποίηση (*standardization*) που είχε ήδη εισάγει το OGSA σε ότι αφορά τη λειτουργία τους. Αυτό που αλλάζει είναι ο τρόπος με τον οποίο ορίζονται, κατασκευάζονται και διευθυνσιοδοτούνται οι υπηρεσίες, και μεταχειρίζονται τα στιγμιότυπά τους. Με άλλα λόγια η υποδομή. Ειδικότερα, ενώ το OGSI μεταχειρίζεται έναν πόρο σαν να είναι ο ίδιος μία υπηρεσία πλέγματος, το WSRF διαχωρίζει τις υπηρεσίες δικτύου από τους πόρους που ελέγχουν.

2.1.6.1 Πληροφορίες κατάστασης

Κάθε πόρος περιέχει δεδομένα που σχετίζονται με αυτόν. Αυτά τα δεδομένα μπορεί να μεταβάλλονται κατά τη διάρκεια ζωής του πόρου. Η κατάσταση ενός πόρου περιγράφεται σε ένα κείμενο ιδιοτήτων υπηρεσίας, το οποίο προσπελαύνεται από απλές ανταλλαγές μηνυμάτων. Τις ανταλλαγές αυτές δεν είναι ανάγκη να πραγματοποιούν συγκεκριμένες λειτουργίες, αλλά το WSRF έχει υλοποιήσει μερικές λειτουργίες εντός του (WS-Resource Properties Document) για την ανάκτηση και τροποποίηση στοιχείων XML που περιέχουν τις ιδιότητες του πόρου. Αυτές είναι οι *GetResourceProperty*, η οποία επιστρέφει την τιμή ενός XML στοιχείου δεδομένου του ονόματός του, η *GetMultipleResourceProperties*, η οποία επιστρέφει τιμές για πολλά στοιχεία, η *SetResourceProperties* και η *QueryResourceProperties*, η οποία τελευταία επιστρέφει τις ιδιότητες που βρίσκονται στο *XPath* που δίνεται σαν όρισμα. Στο WSRF ορίζεται ξεχωριστή διεπιφάνεια που περιέχει τις πληροφορίες κατάστασης και τις μεθόδους για την προσπέλαση αυτών. Κάτι τέτοιο επιτρέπει επίσης τη χρήση της πρότυπης έκδοσης WSDL χωρίς επεκτάσεις.

2.1.6.2 Διευθυνσιοδότηση

Ένας πελάτης μπορεί να χρησιμοποιεί ταυτόχρονα πολλούς πόρους, ενώ ένας πόρος μπορεί επίσης να προσπελαύνεται ταυτόχρονα από πολλούς χρήστες. Έτσι, υπάρχουν πολλά στιγμιότυπα του ίδιου πόρου (*WS-Resources*), με την ίδια διεπαφή (*interface*) στον ίδιο χρόνο. Όταν ο πελάτης στέλνει ένα μήνυμα σε έναν εξυπηρετητή που μοιράζεται έναν πόρο

με άλλους, ο εξυπηρετητής πρέπει να γνωρίζει σε ποιο στιγμιότυπο του πόρου θα πρέπει να στείλει το μήνυμα.

Το WSRF χρησιμοποιεί ένα μοντέλο για τη διευθυνσιοδότηση των πόρων (*WS-Addressing model*) βάση του οποίου ορίζεται μία, ανεξάρτητη δικτύου, αναφορά δείκτη (endpoint) για κάθε υπηρεσία ιστού. Στην περίπτωση του WSRF, η αναφορά είναι δείκτης σε κάποιο πόρο (*WS-Resource*). Ο τρόπος αυτός διευθυνσιοδότησης επιτρέπει επίσης την παροχή επιπρόσθετων πληροφοριών μαζί με τη διεύθυνση του πόρου (π.χ. πληροφορίες για την υπηρεσία, ή τα πρωτόκολλα που χρησιμοποιούνται). Οι πληροφορίες και η διεύθυνση του πόρου παρέχονται σε μορφή XML από ένα αναγνωριστικό πόρου (*resource identifier*). Αυτό το αναγνωριστικό χρησιμοποιείται για να ξεχωρίζεται το ένα στιγμιότυπο από το άλλο. Μία αναφορά δείκτη (*WS-Addressing endpoint reference*) πρέπει να περιλαμβάνει ένα «στοιχείο-παιδί» που θα αναφέρει τις ιδιότητες του πόρου και θα αναγνωρίζει αυτόν που θα σχετίζεται με τις ανταλλαγές μηνυμάτων που θα πραγματοποιούνται με αυτή την αναφορά δείκτη. Οι πληροφορίες που περιέχει μία αναφορά δείκτη είναι ασήμαντες για τις υπηρεσίες δικτύου και χρησιμοποιούνται από το μοντέλο διευθυνσιοδότησης μόνο, για την αντιστοίχηση της αναφοράς με τον εκάστοτε πόρο. Το μοντέλο χρησιμοποιεί μία ανεξάρτητης πρωτοκόλλου μεταφοράς προσέγγιση για να συνδέσει τον αποστολέα και τον παραλήπτη. Αυτό σημαίνει ότι μπορεί να υλοποιηθεί πάνω από το SOAP ή οποιοδήποτε άλλο πρωτόκολλο. Οι πληροφορίες που αφορούν έναν πόρο ορίζονται στο πεδίο *ReferenceProperties* της αναφοράς δείκτη.

```
<wsa:EndpointReference>
        <!-- Web Service address over a network endpoint -->
        <wsa:Address>
                http://helloworld.com/myWebService
        </wsa:Address>
        <!-- Meta Data -->
        <!-- Endpoint reference properties -->
        <wsa:ReferenceProperties>
                <tns:resourceID> ID-12345 </tns:resourceID>
        </wsa:ReferenceProperties>
</wsa:EndpointReference>
```

Πίνακας 1: WS-Addressing αναφορά δείκτη

2.1.6.3 Δημιουργία και καταστροφή στιγμιότυπων

Στο WSRF δεν ορίζεται με σαφήνεια ο τρόπος δημιουργίας ενός στιγμιότυπου. Ορίζεται μόνο ότι ένα νέο στιγμιότυπο μπορεί να δημιουργηθεί μέσω ενός «εξωτερικού μηχανισμού» ή μίας *factory* μεθόδου. Όταν δημιουργείται ένα στιγμιότυπο πόρου (*WS-Resource*), επιστρέφεται μία αναφορά δείκτη στο στιγμιότυπο πόρου με τον αρχικό χρόνο ζωής του. Ο χρόνος ζωής μπορεί να τροποποιηθεί από μεθόδους της *WS-ResourceLifetime* διεπαφής. Ένα στιγμιότυπο πόρου μπορεί επίσης να καταστραφεί αμέσως, μέσω της Destroy μεθόδου.

Σύμφωνα με τις προδιαγραφές της *WS-ResourceLifetime*, ο χρόνος λήξης είναι ένα στοιχείο του κειμένου ιδιοτήτων πόρου (*WS-ResourceProperties Document*). Αυτό το στοιχείο μπορεί να τροποποιηθεί μόνο μέσω της μεθόδου *SetTerminationTime* και όχι από τη *SetResourceProperties*.

2.1.6.4 Γνωστοποιήσεις (Notifications)

Το μοντέλο *WS-Resource* ορίζει μία περιγραφή XML των θεμάτων στα οποία ένας πελάτης μπορεί να δηλώσει ότι επιθυμεί να λαμβάνει γνωστοποιήσεις, σύμφωνα με τους όρους που ορίζει το πλαίσιο *WS-Topics*. Ο πελάτης (συνδρομητής) εγγράφεται σε έναν παραγωγό γνωστοποιήσεων για να λαμβάνει συγκεκριμένες γνωστοποιήσεις που τον ενδιαφέρουν. Ο συνδρομητής οφείλει να υλοποιεί μία διεπαφή που ονομάζεται *NotificationConsumer* για να μπορεί να δέχεται αυτά τα μηνύματα. Μαζί με την εγγραφή, ο πελάτης παραλαμβάνει μία αναφορά δείκτη σε μία *WS-Resource* συνδρομής. Αυτή η αναφορά δείκτη μπορεί να χρησιμοποιηθεί για έλεγχο και διαχείριση των γνωστοποιήσεων.

Μία άλλη προδιαγραφή ορίζει τη διεπαφή για μία διαμεσολαβητική υπηρεσία διαχείρισης συνδρομών άλλων υπηρεσιών που παράγουν γνωστοποιήσεις, που ονομάζεται *BrokeredNotification*. Το σχήμα γνωστοποιήσεων αποτελείται από τα ίδια τρία μέρη που συνθέτουν και το σχήμα που διατυπώνεται από το OGSA. Κατά πρώτον, είναι η διεπαφή της πηγής, η οποία στέλνει τα μηνύματα γνωστοποίησης και ονομάζεται *NotificationProducer*. Κατά δεύτερον, ο παραλήπτης των μηνυμάτων υλοποιεί τη διεπαφή *NotificationConsumer*. Τέλος, τη συνδρομή γνωστοποιήσεων διαχειρίζεται η διεπαφή *SubscriptionManager*.

Για να δεχθεί ο πελάτης γνωστοποιήσεις πρέπει να υλοποιήσει τα παρακάτω βήματα. Καταρχάς πρέπει να καλέσει τη μέθοδο *subscribe*. Ο παραλήπτης των γνωστοποιήσεων ορίζεται από μία αναφορά δείκτη (*WS-Addressing endpoint*). Μία γνωστοποίηση στέλνεται συνήθως καλώντας της μέθοδο notify της διεπαφής *NoificationConsumer*. Η λειτουργία μπορεί να μεταβληθεί αλλάζοντας την παράμετρο *UseNotify*. Όταν γίνεται κλήση της subscribe μεθόδου, το στοιχείο *SubscriptionPolicy* περιγράφει τις απαιτήσεις διαχείρισης (*policy requests*) του αιτούντα. Η μέθοδος επιστρέφει μία αναφορά δείκτη στο αντικείμενο διεπαφής *SubscriptionManager* με την οποία μπορεί ο πελάτης να διαχειρίζεται τη συνδρομή (π.χ. το χρόνο ζωής της).

Η διεπαφή WS-Topics ορίζει ένα μηχανισμό για την οργάνωση και κατηγοριοποίηση στοιχείων περιεχομένου που μπορεί να ενδιαφέρουν για την αποστολή γνωστοποιήσεων. Η διεπαφή *NotificationProducer* ορίζει επίσης ένα *WS-Resource property* στοιχείο, το οποίο περιέχει ένα σύνολο θεμάτων που υποστηρίζει.

Η αναφορά δείκτη στο αντικείμενο *SubscriptionManager* που επιστρέφεται μπορεί να χρησιμοποιηθεί επίσης για να παγώσει η διαδικασία αποστολής γνωστοποιήσεων και να συνεχιστεί στο μέλλον. Τη δυνατότητα αυτή υλοποιούν οι μέθοδοι *PauseSubscription* και *ResumeSubscription.*

2.1.7 RESTful υπηρεσίες

Το REST καθορίζει ένα σύνολο αρχιτεκτονικών αρχών με τις οποίες κάποιος μπορεί να σχεδιάσει υπηρεσίες Ιστού που εστιάζουν στους πόρους ενός συστήματος, συμπεριλαμβανομένου το πώς η κατάσταση των πόρων εξετάζεται και μεταφέρεται μέσω του HTTP πρωτοκόλλου από ένα ευρύ φάσμα των προγραμμάτων πελατών που γράφονται σε διαφορετικές γλώσσες προγραμματισμού. Εάν βασιστούμε στον αριθμό των υπηρεσιών Ιστού που το χρησιμοποιούν, το REST έχει προκύψει τα τελευταία έτη μόνο ως κυρίαρχο πρότυπο σχεδίου υπηρεσιών Ιστού. Στην πραγματικότητα, έχει ασκήσει μια μεγάλη επίδραση στο

διαδίκτυο που έχει εκτοπίζει το SOAP και WSDL πρωτόκολλο διεπαφών εξαιτίας της απλούστερης εφαρμογής του.

Το REST δεν προσέλκυσε πολύ προσοχή όταν εισήχθη αρχικά το 2000 από το Roy Fielding στο πανεπιστήμιο Καλιφόρνιας, Irvine, στην ακαδημαϊκή διατριβή, «Architectural Styles and the Design of Network-based Software Architectures» που αναλύει ένα σύνολο αρχών αρχιτεκτονικής λογισμικού που χρησιμοποιούν τον Ιστό ως πλατφόρμα για το κατανεμημένο υπολογισμό (*distributed computing*). Αρκετά χρόνια μετά από την εμφάνισή του, σημαντικά πλαίσια για το REST έχουν αρχίσει να εμφανίζονται και αναπτύσσονται ακόμα αποτελώντας αναπόσπαστο κομμάτι πολλών γνωστών διεπαφών προγραμματισμού (π.χ. Java™ 6, jsr-311).

Σύμφωνα λοιπόν με τις βασικές αρχές που προτείνει το REST, μια διαδυκτιακή υπηρεσία αυτού του τύπου πρέπει να ακολουθεί τις παρακάτω βασικές αρχές σχεδιασμού:

- Αποκλειστική χρήση HTTP μεθόδων.
- Υλοποίηση χωρίς αποθήκευση της κατάστασης (*stateless*).
- Διευθυνσιοδότηση πανομοιότυπη με την δομή φακέλων.
- Μεταφορά XML, JavaScript Object Notation (JSON), ή και των δύο.

2.1.7.1 HTTP μέθοδοι

Ένα από τα βασικά χαρακτηριστικά των *RESTful* υπηρεσιών είναι η αποκλειστική χρήση HTTP μεθόδων έτσι όπως ορίζεται στο πρωτόκολλο RFC 2616. Η HTTP GET για παράδειγμα ορίζεται ως μια μέθοδος παραγωγής δεδομένων που προορίζεται να χρησιμοποιείται από μια εφαρμογή πελάτη για να αποκτήσει έναν πόρο, να μεταφέρει δεδομένα από έναν εξυπηρετητή (*web server*), ή να εκτελέσει ένα ερώτημα (*query*) με την προσδοκία ότι ο εξυπηρετητής θα απαντήσει με ένα σύνολο πόρων που ταιριάζουν στο ερώτημα αυτό.

Αυτή η βασική αρχή σχεδιασμού διαμορφώνει μια ένα-προς-ένα αντιστοίχηση μεταξύ των λειτουργιών CRUD (*create*, *read*, *update*, *delete*) και των HTTP μεθόδων:

- Για την δημιουργία ενός πόρου χρησιμοποιείται η POST.
- Για την απόκτηση ενός πόρου χρησιμοποιείται η GET.
- Για την αλλαγή της κατάστασης ενός πόρου χρησιμοποιείται η PUT.
- Για την διαγραφή ενός πόρου χρησιμοποιείται η DELETE.

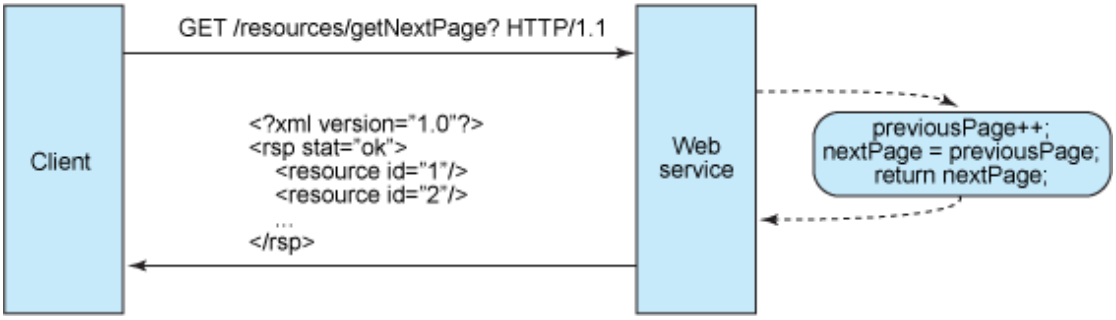
2.1.7.2 Σχεδιασμός χωρίς διατήρηση της κατάστασης (stateless)

Οι υπηρεσίες Ιστού REST πρέπει στην κλιμακώνονται ελαστικά για να ικανοποιήσουν τις όλο και περισσότερο απαιτήσεις υψηλής απόδοσης. Οι συστάδες των εξυπηρετητών με δυνατότητες εξισορρόπησης φορτίου και αντιμετώπισης σφαλμάτων, είναι συνήθως σχεδιασμένες με τέτοιο τρόπο ώστε που διαμορφώνουν μια τοπολογία υπηρεσιών, η οποία επιτρέπει στα αιτήματα να διαβιβαστεί από έναν κεντρικό υπολογιστή σε άλλο όπως απαιτείται για να μειώσει το γενικό χρόνο απόκρισης μιας κλήσης υπηρεσιών Ιστού. Η χρησιμοποίηση των ενδιάμεσων εξυπηρετητών για την βελτίωση της ελαστικότητας απαιτεί από τους πελάτες υπηρεσιών Ιστού REST να στέλνουν πλήρη και ανεξάρτητα αιτήματα δηλαδή να στέλνουν τα αιτήματα που περιλαμβάνουν όλα τα στοιχεία που πρέπει να εκπληρωθούν έτσι ώστε τα συστατικά στους ενδιάμεσους υπολογιστές να μπορούν να διαβιβάσουν, να καθοδηγήσουν, και να ισορροπήσουν το φορτίο χωρίς να αποθηκεύεται οποιαδήποτε κατάσταση τοπικά, μεταξύ των αιτημάτων.

Μια εφαρμογή πελάτης υπηρεσιών Ιστού REST περιλαμβάνει μέσα στις επικεφαλίδες και το σώμα ενός αιτήματος HTTP όλες τις παραμέτρους, περιεχόμενο, και στοιχεία που απαιτούνται από το εξυπηρετητή για να παράξει μια απάντηση. Η απουσία καταχώρησης της κατάστασης (*statelessness*) βελτιώνει από αυτή την άποψη την απόδοση υπηρεσιών Ιστού και απλοποιεί το σχέδιο και την υλοποίηση των συστατικών στην πλευρά του εξυπηρετητή επειδή η απουσία κατάστασης στον κεντρικό υπολογιστή αφαιρεί την ανάγκη να συγχρονιστούν τα στοιχεία συνόδου (*session*) με μια εξωτερική εφαρμογή.

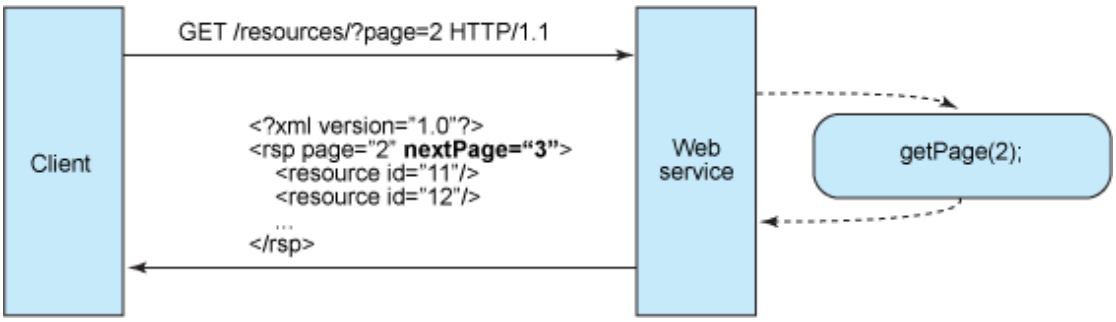
Το Σχήμα 9 επεξηγεί μια *stateful* υπηρεσία από την οποία μια εφαρμογή μπορεί να ζητήσει την επόμενη σελίδα από ένα σύνολο πολλαπλών σελίδων, υποθέτοντας ότι η

υπηρεσία παρακολουθεί σε ποια σελίδα η εφαρμογή σταματάει την πλοήγηση. Σε αυτό το *stateful* σχεδιασμό, η υπηρεσία προσαυξάνει και αποθηκεύει την μεταβλητή *previousPage* έτσι ώστε να μπορεί να ανταποκριθεί στα επόμενα αιτήματα.



Σχήμα 9: Αποθήκευση κατάστασης στην πλευρά του εξυπηρετητή

Από την άλλη, τα άνευ κατάστασης (*stateless*) συστατικά της πλευράς του εξυπηρετητή είναι λιγότερο περίπλοκα να σχεδιάσουν, να υλοποιηθούν, και να λειτουργήσουν σε κεντρικούς υπολογιστές. Μια τέτοια υπηρεσία όχι μόνο αποδίδει καλύτερα, αλλά μετατοπίζει το μεγαλύτερο μέρος της ευθύνης για την διατήρηση της κατάστασης στην εφαρμογή πελάτη. Σε μια *RESTful* υπηρεσία Ιστού, ο κεντρικός εξυπηρετητής είναι αρμόδιος για την παραγωγή των απαντήσεων και για την παροχή μιας διεπαφής που επιτρέπει στον πελάτη να διατηρήσει την κατάσταση της εφαρμογής από μόνο του. Παραδείγματος χάριν, στο αίτημα για ένα *multipage* σύνολο αποτελέσματος, ο πελάτης πρέπει να συμπεριλάβει τον πραγματικό αριθμό σελίδας που θέλει να ανακτήσει αντί απλά να ρωτήσει για την επόμενη (δείτε Σχήμα 10).



Σχήμα 10: Αίτηση υπηρεσίας με στοιχεία κατάστασης

Αυτή η πτυχή του *RESTful* σχεδίου υπηρεσιών Ιστού μπορεί να αναλυθεί σε δύο σύνολα ευθυνών που διευκρινίζει ακριβώς πώς μια *stateless* υπηρεσία μπορεί να διατηρηθεί:

Εξυπηρετητής (Server)

- Παράγει απαντήσεις που περιλαμβάνουν τις συνδέσεις με άλλους πόρους για να επιτρέψουν τις εφαρμογές να πλοηγηθούν μεταξύ των σχετικών πόρων.
- Παράγει απαντήσεις που δείχνουν εάν ένας πόρος είναι διαθέσιμος (*cacheable*) ή όχι για να βελτιώσουν την απόδοση με τη μείωση του αριθμού αιτημάτων για τους διπλούς πόρους. Ο κεντρικός εξυπηρετητής το υλοποιεί αυτό συμπεριλαμβάνοντας μια επικεφαλίδα HTTP για τα *Cache-Control* και *Last-Modified* (τιμή ημερομηνίας) πεδία.

Εφαρμογή πελάτη (Client application)

- Χρησιμοποιεί την επικεφαλίδα απάντησης *Cache-Control* για να καθορίσει εάν θα αποθηκεύσει ένα τοπικό αντίγραφο του πόρου (cache) ή όχι. Ο πελάτης διαβάζει επίσης την επικεφαλίδα απάντησης *Last-Modified* και στέλνει πίσω την τιμή της ημερομηνίας για να ρωτήσει τον εξυπηρετητή εάν ο πόρος έχει αλλάξει. Αυτή η διαδικασία καλείται *Conditional GET*, και σε αυτήν την περίπτωση ο τυποποιημένος κωδικός απάντησης ονομάζεται *304 HTTP response code*. Αυτός ο κωδικός απάντησης σημαίνει ότι ο πελάτης μπορεί ακίνδυνα να χρησιμοποιήσει ένα εναποθηκευμένο, τοπικό αντίγραφο του πόρου δεδομένου ότι είναι ο πιο ενημερωμένος.
- Στέλνει τα πλήρη αιτήματα που μπορούν να εξυπηρετηθούν ανεξάρτητα από άλλα αιτήματα. Αυτό απαιτεί τον πελάτη να αξιοποιήσει πλήρως τις επικεφαλίδες HTTP όπως διευκρινίζεται από τη διεπαφή υπηρεσιών Ιστού και να στείλει τις πλήρεις περιγραφές των πόρων στο σώμα αιτήματος. Ο πελάτης στέλνει τα αιτήματα που κάνουν πολύ λίγες υποθέσεις για τα προγενέστερα αιτήματα, την ύπαρξη μιας συνόδου στον κεντρικό υπολογιστή, τη δυνατότητα του κεντρικού υπολογιστή να προσθέτει περιεχόμενο σε ένα αίτημα, ή την κατάσταση της εφαρμογής που κρατείται μεταξύ των αιτημάτων.

Αυτή η αλληλεπίδραση μεταξύ της εφαρμογής πελάτη και του εξυπηρετητή είναι απαραίτητη για την αποδοτική υλοποίηση υπηρεσιών *stateless REST*. Αυτό βελτιώνει την απόδοση του

συστήματος, εξοικονομώντας εύρος ζώνης από το δίκτυο και ελαχιστοποιώντας την διατήρηση της κατάστασης από τα συστατικά της πλευράς του εξυπηρετητή.

2.2 Πάροχοι Νεφών και διεπαφές χρήσης

2.2.1 Amazon EC2 API

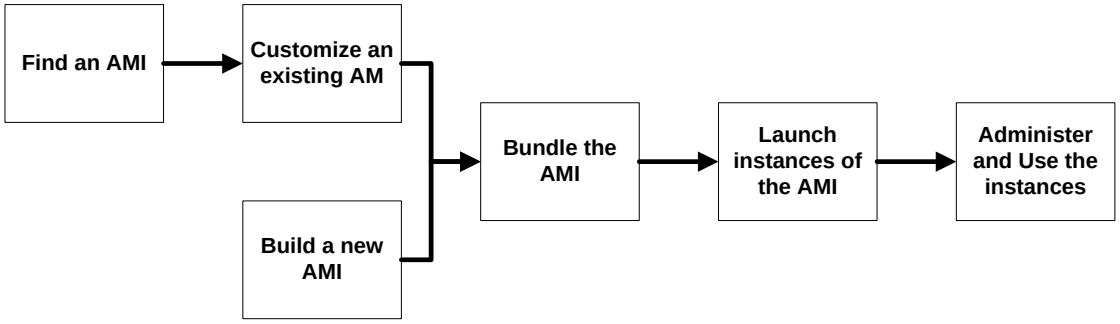
Η Amazon θεωρείται ως ο πρωτοπόρος πάροχος υπηρεσιών Υπολογιστικού Νέφους στην παρούσα αγορά. Η σχετική υπηρεσία που παρέχει ονομάζεται *Amazon Web Services* (AWS) [4] και παρουσιάστηκε αρχικά το 2002. Στην πράξη η AWS είναι ένα περιβάλλον διαδικτυακών υπηρεσιών οι οποίες στο σύνολό τους διαμορφώνουν την πλατφόρμα Νέφους της Amazon. Η εταιρία παρέχει επίσης μια πληθώρα σχετικών και εξειδικευμένων υπηρεσιών όπως: Elastic Compute Cloud (Amazon EC2) [5][6], Simple Storage Service (S3) [7], Simple Queue Service (SQS) [8], CloudFront [9], Simple DB [10] 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 (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.



Σχήμα 11: Amazon EC2 AMI ροή εγκατάστασης

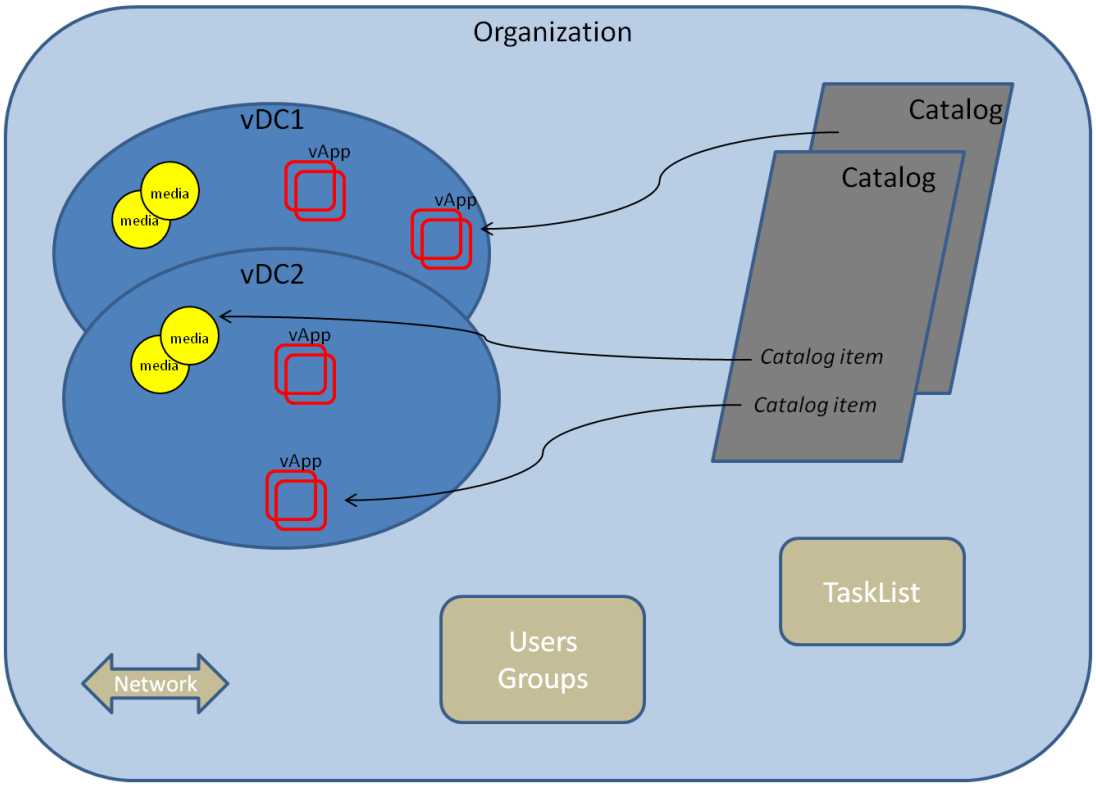
Το API της Amazon EC2 παρέχει πρόσβαση στην διαδικτυακή υπηρεσία είτε μέσω SOAP API είτε μέσω Query API. Στην πρώτη περίπτωση, οι διεπαφές ορίζονται μέσω ενός XML κειμένου περιγραφής υπηρεσίας (*Web Service Description Language* - WSDL). Αφού τα αιτήματα και οι απαντήσεις σε SOAP, στην Amazon EC2, ακολουθούν 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.

2.2.2 VMware vCloud

Η VMware ως πρωτοπόρος εταιρία στις τεχνολογίες εικονικοποίησης (*virtualization*), εξέδωσε το 2009 το προϊόν vCloud [11], 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 [12] 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.

Στο Σχήμα 12 παρουσιάζουμε γραφικά τις οντότητες και τους πόρους ενός vCloud. Τα αντικείμενα όλων των τύπων περιγράφονται μέσω XML κειμένων.



Σχήμα 12: Οντότητες πόρων στο vCloud

Οργανισμός (Organization): ένα vCloud μπορεί να συμπεριλαμβάνει περισσότερους από ένα οργανισμούς καθένας από τους οποίους είναι ένα υπερσύνολο χρηστών, ομάδων και πόρων.

vDC: ένα εικονικό κέντρο δεδομένων του vCloud (vDC) είναι ένας μηχανισμός για την ανάθεση υπολογιστικών πόρων όπως δίκτυα, αποθηκευτικό χώρο, CPU και μνήμη. Μέσα σε ένα vDC οι πόροι είναι εντελώς εικονικοποιημένοι και είτε δεσμεύονται κατ' απαίτηση (on-demand) είτε είναι προδιαγεγραμμένοι εξαρχής στη συμβόλαιο της υπηρεσίας (SLA). Υπάρχουν vDCs Παρόχου, τα οποία περιλαμβάνουν τους πόρους ενός παρόχου υπηρεσίας vCloud, και vDCs Οργανισμού, τα οποία παρέχουν ένα περιβάλλον στο οποίο εικονικά συστήματα μπορούν να αποθηκεύονται, εγκαθίστανται και εκτελούνται.

Κατάλογοι (Catalogs): Οι κατάλογοι περιέχουν αναφορές προς τα εικονικά κέντρα δεδομένων (vDCs). Ο κατάλογος αυτός μπορεί να είναι ορατός μόνο από τους δημιουργούς του ή μπορεί να δημοσιοποιηθεί και να είναι ορατός και από άλλα μέλη του οργανισμού.

Δίκτυο (Network): Το Δίκτυο αντιπροσωπεύει την δεσμευμένη χωρητικότητα δικτύου ενός εικονικού κέντρο δεδομένων (vDC) εντός ενός οργανισμού.

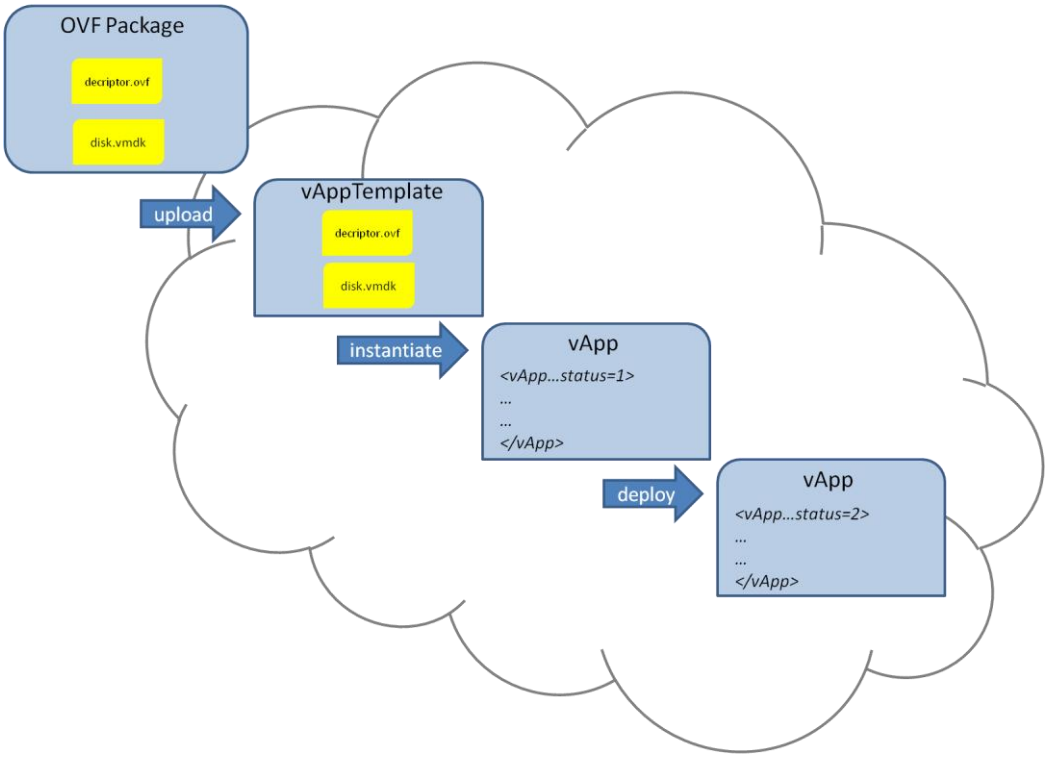
Χρήστες & Ομάδες (User & Groups): Ένας οργανισμός μπορεί να συμπεριλαμβάνει πολλούς Χρήστες και Ομάδες που δημιουργούνται από τον διαχειριστή του οργανισμού είτε να έχουν εισαχθεί από μια υπηρεσία αρχείου (LDAP). Οι ρόλοι και τα συγκεκριμένα δικαιώματα χρήσης ανατίθενται από τον διαχειριστή επίσης.

Λίστα Εργασιών (TasksList)**:** Οι εργασίες που δημιουργούνται κατά τη διάρκεια της εκτέλεσης μιας χρονοβόρας λειτουργίας αποθηκεύονται στην Λίστα Εργασιών.

Εικονικά Συστήματα και Εικόνες Μέσων (Virtual Systems and Media Images): Ένα vDC μπορεί να συμπεριλαμβάνει διάφορα Εικονικά Συστήματα και Εικόνες Μέσων (Media Images). Οι εικόνες αυτές είναι αποθηκευμένες στην αρχική τους αναπαράσταση (ISO, floppy etc) ενώ τα Εικονικά Συστήματα είναι αποθηκευμένα ως πρότυπα (*templates*) βασισμένα στο OVF 1.0 πρότυπο. Αυτά μπορούν μετά να αναζητηθούν από τον κατάλογο και να αρχικοποιηθούν έτσι ώστε να μετατραπούν σε εικονικές εφαρμογές (vApps). Μια εικονική εφαρμογή μπορεί να περιλαμβάνει μια ή και περισσότερες εικονικές μηχανές (*Virtual Machines* – VMs).

Ένα vCloud API υποστηρίζει πολλαπλές λειτουργίες για την ενεργοποίηση μιας διπλής επικοινωνίας για την μεταφορά των Εικόνων Μέσων και των πακέτων OVF μεταξύ του χρήστη και του Νέφους. Οι λειτουργίες μεταφόρτωσης δεδομένων υλοποιούνται μέσω αιτημάτων GET και POST. Ο κύκλος ζωής μιας vApp εφαρμογής περιλαμβάνει τα παρακάτω τρία βήματα:

- Μεταφόρτωση του πακέτου OVF (*Upload*)
- Αρχικοποίηση του προτύπου μιας vApp (*Instantiate*)
- Εγκατάσταση της vApp (*Deploy*)



Σχήμα 13: Κύκλος ζωής vApp

2.2.3 Google App Engine

Η είσοδος της Google στον χώρο των τεχνολογιών υπολογιστικού Νέφους έγινε μέσω της υπηρεσίας Google App Engine [13]. 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 [14] και Python [15] 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 2) παρουσιάζονται τα χαρακτηριστικά δωρεάν χρήσης.

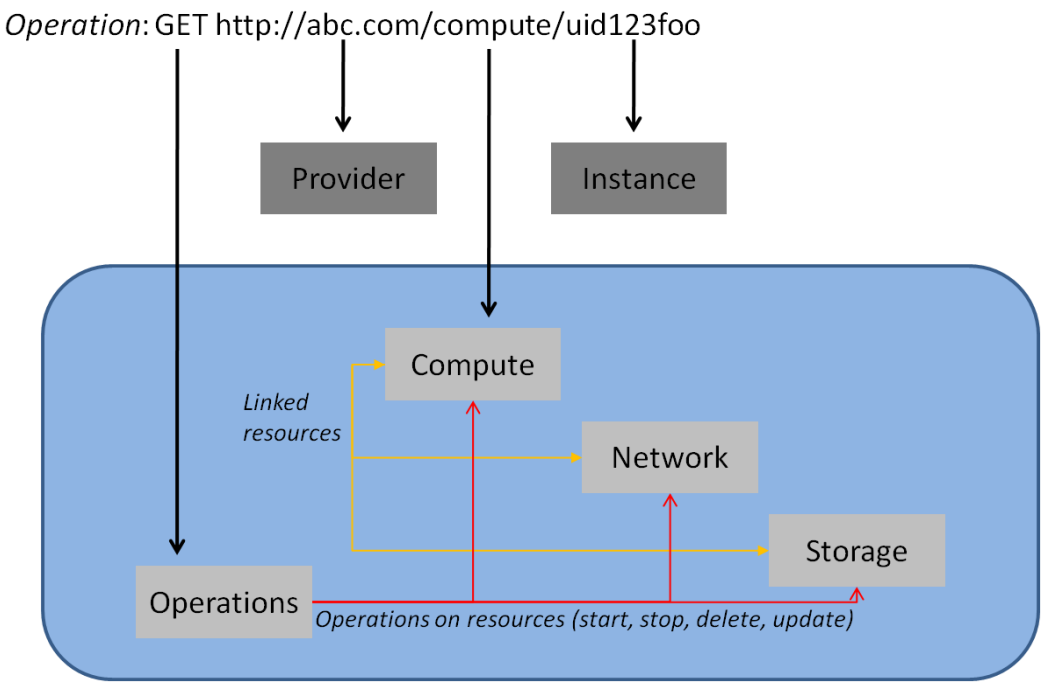
Quota	**Limit**
Apps per developer	10
Time per request	30 sec
Blobstore size (total file size per app)	1 GB
HTTP response size	10 MB
Datastore item size	1 MB
Application code size	150 MB

Πίνακας 2: Google App Engine free quota

2.2.4 Open Cloud Computing Interface (OCCI)

Κατά τη διάρκεια της OGF25 τον Μάρτιο του 2009, ο Ignacio M. Llorente του πανεπιστημίου της Μαδρίτης (UCM – OpenNebula [16]) και ο Thijs Metsch (Sun Microsystems - RESERVOIR project [17]) ίδρυσαν την ομάδα εργασίας Open Cloud Computing Interface Working Group (OCCI) [18] με τα 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 [19] 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.



Σχήμα 14: OCCI δομή πόρων στο URI

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).

2.2.5 Azure (.NET)

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 [20] 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++, VM

κλπ) 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.

2.2.6 Sun Cloud API

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 [21]. 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:

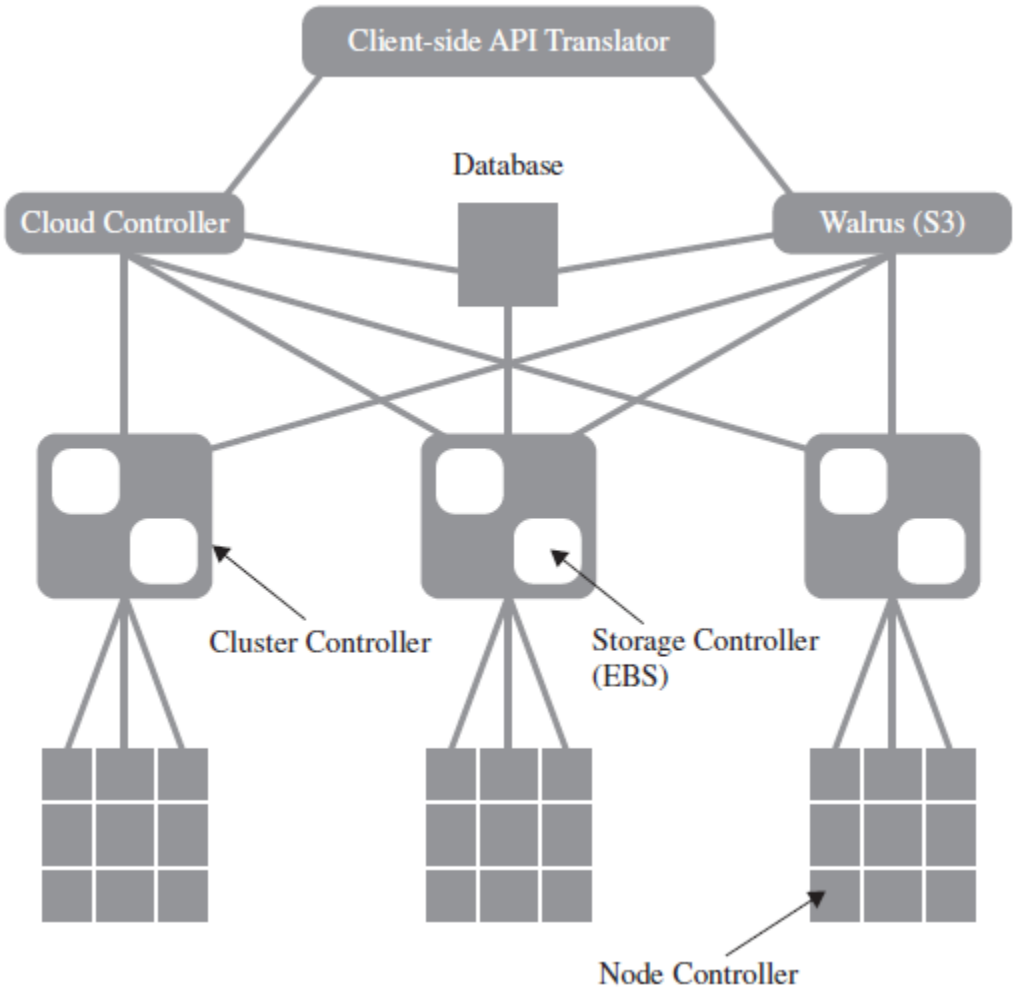
- *Νέφος (Cloud)*: Μια δομή υψηλού επιπέδου η οποία συγκεντρώνει όλα τα Εικονικά Κέντρα Δεδομένων στα οποία έχει πρόσβαση ένας χρήστης

- *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.

2.2.7 Eucalyptus

Το Eucalyptus 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)



Σχήμα 15: Αρχιτεκτονική υψηλού επιπέδου Eucalyptus

Η αρχιτεκτονική του Eucalyptus όπως παρουσιάζεται στο Σχήμα 15, 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.

2.2.8 Open Nebula

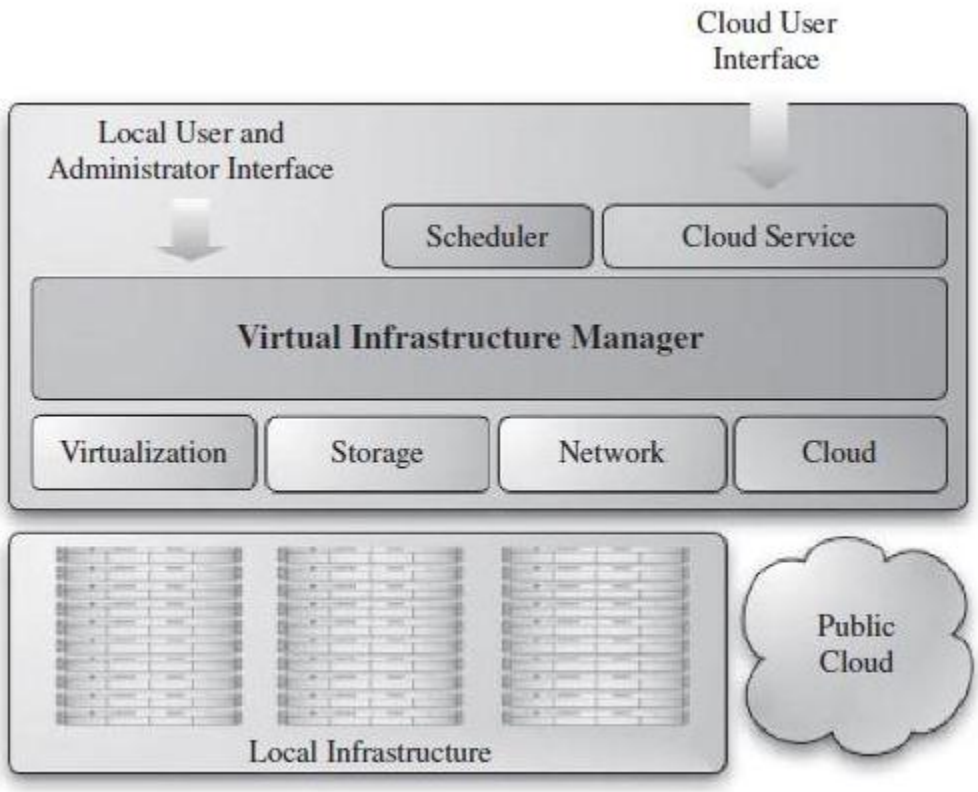
Το Open Nebula είναι μία από τις πιο πλούσιες εφαρμογές ανοιχτού κώδικα για υλοποίηση IaaS. Αρχικά είχε σχεδιαστεί για την διαχείριση εικονικών υποδομών και περιλάμβανε απομακρυσμένες διεπαφές που καθιστούσαν υλοποιήσιμη την κατασκευή δημόσιων Νεφών. Συνολικά τέσσερα APIs είναι διαθέσιμα:

- 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 (Amazon EC2, ElasticHosts)

- Εικονικά δίκτυα
- Δυναμική ανάθεση πόρων
- Προληπτική δέσμευση πόρων



Σχήμα 16: Αρχιτεκτονική υψηλού επιπέδου Open Nebula

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.

2.2.9 Cross Platform Cloud APIs

Εκτός από τους παρόχους Νεφών με συγκεκριμένα APIs υπάρχουν διεπαφές προγραμματισμού για Υπολογιστικά Νέφη ανεξάρτητα πλατφόρμας. Το Jclouds [22] 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 [23] 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 [24] 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.

3 *Παρακολούθηση & επίβλεψη λειτουργίας*

Στο κεφάλαιο αυτό παρουσιάζονται οι σχετικές τεχνολογίες και εργαλεία όσον αφορά την επίβλεψη και παρακολούθηση πόρων σε περιβάλλοντα Νεφών και άλλες αρχιτεκτονικές. Η μεθοδολογία έρευνας και ανάλυσης βασίστηκε στον διαχωρισμό των υπαρχουσών τεχνολογιών σε: (i) τεχνικές επίβλεψης εμπορικών Νεφών, (ii) συστήματα επίβλεψης ανοιχτού κώδικα (με δυνατότητα χρήσης και επέκταση τους), (iii) καταγραφή και ανάλυση λοιπών μηχανισμών επίβλεψης και σχετικών ερευνητικών δραστηριοτήτων. Επιπρόσθετα, η μελέτη των συστημάτων αυτών καταλήγει στον προσδιορισμό των προδιαγραφών και των απαιτήσεων σχεδιασμού ενός σύγχρονου μηχανισμού επίβλεψης και παρακολούθησης για υπολογιστικά Νέφη και υπηρεσιοστρεφείς υποδομές. Σε αυτό το πλαίσιο, συζητάμε και οριοθετούμε τις συνθήκες λειτουργίας ενός συστήματος επίβλεψης 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 Υπάρχουσες τεχνολογίες & εργαλεία

3.1.1 Επίβλεψη 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. Παραδείγματα τέτοιων υπηρεσιών είναι οι: Relational Database Service (RDS) [27], Auto Scaling Service [28] και Elastic Load Balancing Service [29]. 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 επίβλεψης και παρακολούθησης. Αυτοί συμπεριλαμβάνουν τον Azure Diagnostic Manager [30], 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 [31] για το Azure το οποίο επιτρέπει σε χρήστες και διαχειριστές Νεφών να αξιολογήσουν την χρησιμοποίηση των Azure πόρων τους.

Μια λύση επίβλεψης εμπορικού Νέφους ανεξάρτητη από τον πάροχο είναι το CloudStatus [32]. Βασισμένο στο Hyperic HQ, ένα εργαλείο παρακολούθησης και διαχείρισης για διαδικτυακές υποδομές, παρέχει τα μέσα για την επίβλεψη 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.

3.1.2 Μηχανισμοί επίβλεψης ανοιχτού κώδικα

Σε αυτό το υποκεφάλαιο παρουσιάζονται και αναλύονται επιλεγμένα εργαλεία και APIs που χρησιμοποιούνται ευρέως για την επίβλεψη 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 επίβλεψης υπολογιστικών Νεφών. Τέλος, όλοι οι μηχανισμοί που παρουσιάζονται είναι συστήματα ανοιχτού κώδικα (*open source*) και διατίθενται δωρεάν στο διαδίκτυο.

3.1.2.1 Ganglia

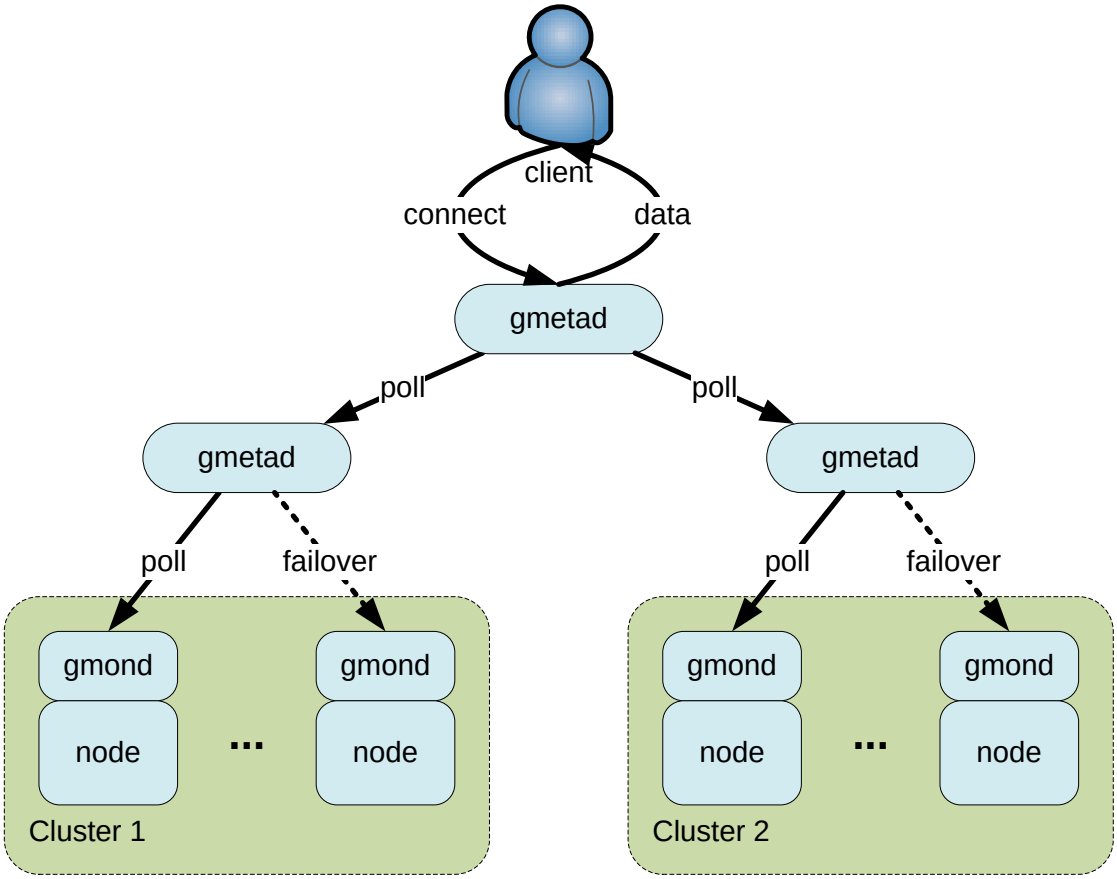
Το Ganglia [33] [34] είναι ένας μηχανισμός επίβλεψης πόρων που σχεδιάστηκε να εξυπηρετεί υπολογιστικά συστήματα υψηλής απόδοσης όπως συστάδες υπολογιστών και περιβάλλοντα πλέγματος. Είναι βασισμένο σε μία ιεραρχική αρχιτεκτονική η οποία παρέχει υψηλή επεκτασιμότητα και απόδοση, με δυνατότητα να υποστηρίζει μέχρι και 2000 υπολογιστικούς κόμβους. Η διαχείριση της πληροφορίας γίνεται μέσο XML, για την παροχή δεδομένων, και με XDR [35], για την μεταφορά δεδομένων, ενώ τα αποτελέσματα της επίβλεψης αποθηκεύονται με μορφή γράφου μέσο ενός RRD εργαλείου [36]. 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 επίβλεψη 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 επίβλεψης. 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.

3.1.2.1.1 Αρχιτεκτονική

Στο Σχήμα 17 παρουσιάζεται η ιεραρχική δενδροειδής αρχιτεκτονική του Ganglia, η οποία αποτελείται από τέσσερα δομικά συστατικά:

- Συλλέκτης δεδομένων *gmond*
- Συναθροιστής (*aggregator*) δεδομένων *gmetad*
- Εργαλείο RRD (***r**ound-**r**obin **d**atabase tool*)
- Διαδικτυακή διεπαφή σε PHP



Σχήμα 17: Αρχιτεκτονική Ganglia

Σε μια συστάδα υπολογιστών, το *gmond* συλλέγει τις παραμέτρους επίβλεψης 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.

3.1.2.2 Nagios

Το Nagios [37] είναι ένα ακόμα σύστημα επίβλεψης 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 επίβλεψης 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 επίβλεψης (*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 επίβλεψη 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 επίβλεψης. 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 [38].

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 την

αποστολή 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 [39], που παρέχουν επιπρόσθετες δυνατότητες συμπεριλαμβανομένων ισχυρού διαδικτυακού περιβάλλοντος διαχείρισης και διαχείριση παραμέτρων για εξειδικευμένους χρήστες.

3.1.2.2.1 Αρχιτεκτονική

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 επίβλεψης 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.*

3.1.2.3 Hyperic HQ Open Source

Το Hyperic HQ [40] είναι ένα λογισμικό επίβλεψης εφαρμογών, σχεδιασμένο να διαχειρίζεται και επιβλέπει διαδικτυακές εφαρμογές και υποδομές σε ετερογενή

περιβάλλοντα. 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 επίβλεψης, οι οποίες δεν μπορούν εύκολα να παρακολουθήσουν εφαρμογές, το Hyperic HQ υποστηρίζει λεπτομερή επίβλεψη 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.

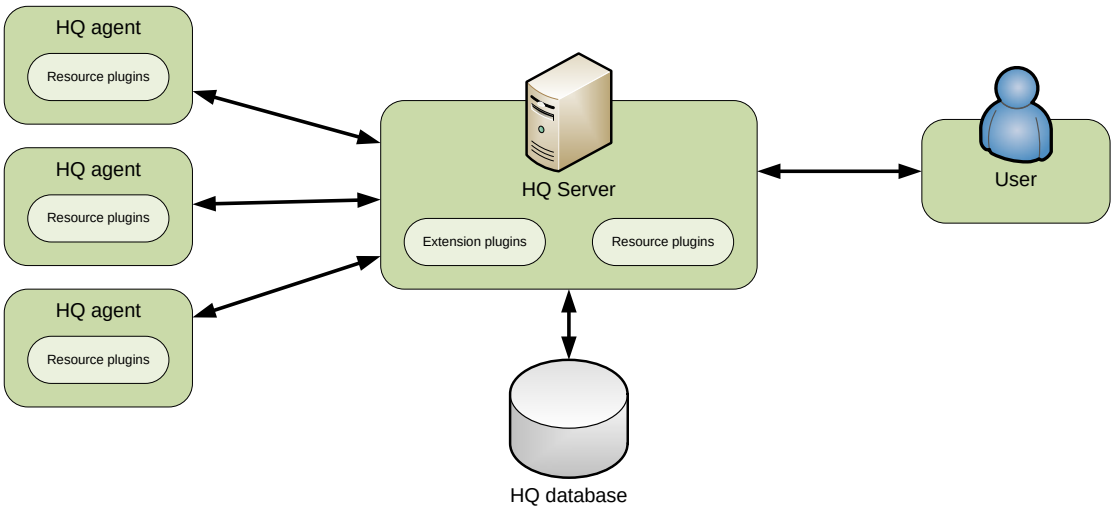
Οι δυνατότητες διαχείρισης του HQ βασίζονται στο μοντέλο απογραφής [41] το οποίο προσδιορίζει τις σχέσεις μεταξύ τον πόρων που διαχειρίζονται. Το HQ αυτόματα εντοπίζει το λογισμικό που εκτελείται σε ένα μηχάνημα και αποθηκεύει τις πληροφορίες στην βάση δεδομένων HQ σύμφωνα με το ιεραρχικό μοντέλο απογραφής. 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 επίβλεψης: 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 [42]. 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 επίβλεψης μέσω ενός γραφικού περιβάλλοντος το οποίο μπορεί να προσαρμοστεί έτσι ώστε να παρέχει συγκεκριμένες αναφορές για την επίβλεψη, 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.).

3.1.2.3.1 Αρχιτεκτονική

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 επίβλεψης (Σχήμα 18). 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*).



Σχήμα 18: Αρχιτεκτονική Hyperic HQ

3.1.2.4 Lattice monitoring Framework

Το πλαίσιο επίβλεψης Lattice [43] είναι ένα API ανοιχτού κώδικα για την ανάπτυξη συστημάτων επίβλεψης 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 επίβλεψη πόρων και υπηρεσιών όπως παρουσιάζεται στο [44]. 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 [45] για την επίβλεψη εικονικών δικτύων, όπως επίσης και στο RESERVOIR Project όπου χρησιμοποιήθηκε για την ανάπτυξη ενός ολοκληρωμένου συστήματος παρακολούθησης Νέφους υπηρεσιών.

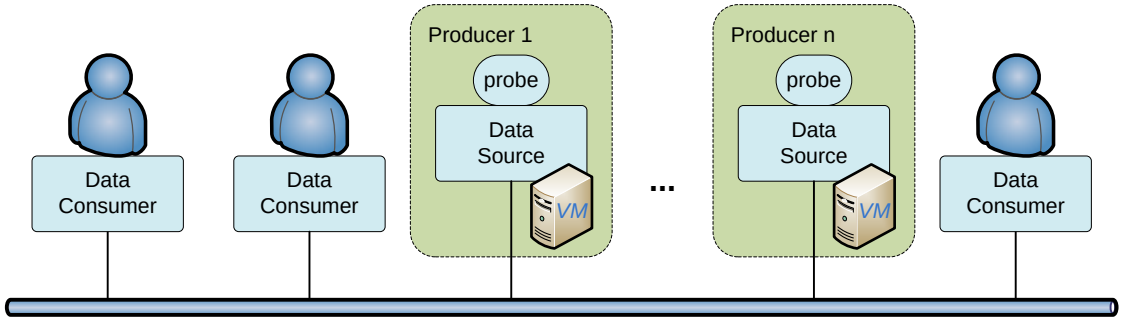
Παρόλο που το Lattice Framework παρέχει δυνατότητες για παρακολούθηση φυσικών πόρων (CPU, σκληρούς δίσκους κ.α.), χρησιμοποιείται κυρίως για την συλλογή μετρήσεων σε εικονικές μηχανές και παροχή των αποτελεσμάτων στο επίπεδο διαχείρισης. Οι λειτουργίες του επικεντρώνονται στην συλλογή και διανομή των δεδομένων επίβλεψης 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.

3.1.2.4.1 Αρχιτεκτονική

Όπως παρουσιάζεται στο Σχήμα 19, 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 επίβλεψης. 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.



Σχήμα 19: Λειτουργία Lattice

3.1.2.5 Zenoss

Το Zenoss Core [46] είναι ένα λογισμικό ελεύθερου κώδικα για τη διαχείριση και επίβλεψη, βασισμένο στον εξυπηρετητή εφαρμογών Zope [47]. 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 επίβλεψη και ειδοποίηση αφού πρώτα έχει εντοπίσει, επίσης αυτόματα, τις συσκευές. Για να διαχειριστεί πολύπλοκα δεδομένα επίβλεψης 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 επίβλεψης με γράφους ή με κάποιο προσωποποιημένο ταμπλό.

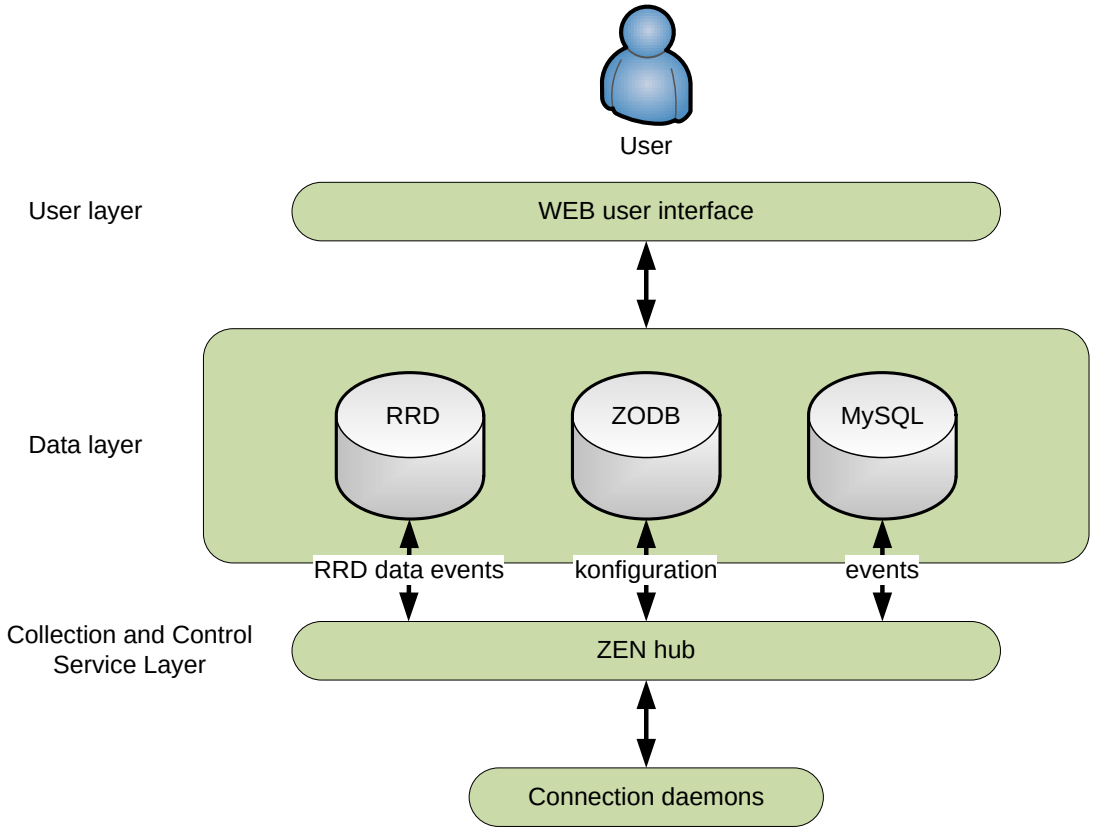
Υπάρχουν δύο εκδόσεις του συστήματος αυτού, το Zenoss Core και το Zenoss Enterprise. Το πρώτο διατίθεται δωρεάν υπό την άδεια GPL ενώ το δεύτερο είναι η εμπορική έκδοση, βασισμένη στην πρώτη, με επιπρόσθετες λειτουργίες.

3.1.2.5.1 Αρχιτεκτονική

Το Zenoss σύστημα μπορεί να χωριστεί σε τρία διαφορετικά στρώματα (Σχήμα 20):

- Στρώμα Χρηστών (*User layer*)
- Στρώμα Δεδομένων (*Data layer*)
- Στρώμα Υπηρεσιών Συλλογής και Ελέγχου (*Collection and Control Service layer*)

Το Στρώμα Χρηστών αποτελείται από ένα διαδικτυακό GUI το οποίο επιτρέπει στον χρήστη να έχει πρόσβαση στα δεδομένα επίβλεψης. 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.



Σχήμα 20: Αρχιτεκτονική Zenoss

Στον Πίνακας 3 έχουμε συγκεντρώσει τα χαρακτηριστικά και δυνατότητες κάθε συστήματος που παρουσιάσαμε σε αυτό το υποκεφάλαιο.

	Ganglia	Nagios Core	Hyperic HQ (OS)	Zenoss Core	Lattice
Generic info					
Homepage	http://ganglia.info/	http://www.nagios.org/	http://www.hyperic.com/products/open-source-systems-monitoring	http://www.zenoss.com/	http://clayfour.ee.ucl.ac.uk/lattice/
current version	verion 3.1.7	version 3.2.3	version 4.4.0	version 2.5.2	version 0.6.4
Supported Platforms	Linux, Solaris, FreeBSD, NetBSD, openBSD..	Linux and Unix variants	Windows, Linux, Solaris, Mac,	Red Hat Enterprise Linux, CentOS RPM, Novell Suse Enterprise,	Platform indepentent (Java based API)
License	BSD	GUN	GNU GPL v2	GNU GPL v2	LGPL licence
Architecture					
Frontend	Yes	Yes	Yes	Browser-GUI	No
Server / Agent	No	Yes	Yes	Yes	No
Plugin structure	No	Yes	Yes	Yes	No
Configuration					
Configuration via GUI	No	No	Part	Yes	No
Own User/Permission Management	No	Yes	Yes	Yes	No
Device Auto devices	in a cluster	No	Yes	Yes	No
Monitoring					
Monitoring Metrics	Yes	Yes	Yes	Yes	Yes
Monitoring availability	No	Yes	Yes	Yes	No
Track performance, configuration	Yes	Yes	Yes	Yes	
Physical Environment	Yes	Yes	Yes	Yes	Yes
Virtual Environment	No	Yes	Yes	Yes	Yes
Evalutaion, Alerting, Notification					
Availability Alerting	No	Yes	Yes	Yes	No
Event Alerting	No	Yes	Yes	Yes	No
Recovery Alerts/ corrective actions	No	Yes	Yes	Yes	No
Visualization					
performance diagram and charts	Yes	Yes	Yes	Yes	No
customizable Dashboard	No	Yes	Yes	Yes	No
Extensibility					
Web Service API	Yes		Yes	Yes	
Nagios plugin-in Integration	Yes	Yes	Yes	Yes	No
Plugin Development Kit	No	Yes	Yes	Yes	No
Support Services					
Document	Yes	Yes	Yes	Yes	Yes, but limited reources
Forum	Yes	Yes	Yes	Yes	No

Πίνακας 3: Σύγκριση χαρακτηριστικών συστημάτων επίβλεψης

3.1.3 Λοιπά συστήματα παρακολούθησης 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 επίβλεψη και παρακολούθηση πόρων έχει υπάρξει ενεργή περιοχή έρευνας για αρκετό καιρό [48]. 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 επίβλεψης για Νέφη. Όπως καταγράφεται στο [49], ένας μεγάλος αριθμός διαφορετικών λύσεων επίβλεψης 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 [50] είναι ένα αυτόνομο σύστημα επίβλεψης 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) [51] και το Globus MDS [52] 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 [53] 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 επίβλεψη και ανακάλυψη πόρων και υπηρεσιών. Παρέχεται μαζί με το Globus Toolkit σύστημα και διαθέτει τυποποιημένες διεπαφές που βασίζονται στα WS-Resource Framework (WSRF) [54] και WS-Notification (WS-N) [55]. 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 [56] 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 [57] [58]. 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) [60]. 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 επίβλεψης 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 επίβλεψη, υπάρχουν διάφορες ερευνητικές πρωτοβουλίες οι οποίες είναι άξιες αναφοράς. Στο [62] παρουσιάζεται ένα πλήρες πλαίσιο διαχείρισης πληροφορίας, επίβλεψης, διαχείρισης λογαριασμών και χρέωσης για πλατφόρμες Νεφών. Παρόλο που παρουσιάζουν μια καλή ανάλυση προδιαγραφών όσον αφορά την επίβλεψη σε

Υπολογιστικά Νέφη, στερούνται πληροφοριών υλοποίησης και στοιχείων για την απόδοση. Στο [63] 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 [64] αποδεικνύει ότι στις αντικειμενοστρεφείς αρχιτεκτονικές (SOA) η υπολογιστική επιβάρυνση που εισαγάγετε από το σύστημα επίβλεψης 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 επίβλεψης 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 επίβλεψης 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 επίβλεψης 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 [65]. 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 επίβλεψης συσχετίζονται με τους φυσικούς πόρους και έρχονται στο επίπεδο των φυσικών μηχανών, όχι των εικονικών μηχανών. Στα Νέφη, οι πληροφορίες επίβλεψης για τους φυσικούς πόρους είναι μόνο χρήσιμες στους προμηθευτές των πόρων, όχι στους πελάτες. Επιπλέον, ένας κύριος σκοπός πολλών συστημάτων επίβλεψης 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.

3.2 Επίβλεψη & διαχείριση

3.2.1 Προδιαγραφές

Η εγκατάσταση και η εκτέλεση εφαρμογών σε έντονα δυναμικές υποδομές, όπως τα υπολογιστικά Νέφη, εισαγάγουν ένα καινούριο σύνολο απαιτήσεων όσον αφορά την επίβλεψη το οποίο είναι απαραίτητο να ληφθεί υπόψη από τους προγραμματιστές και τους παρόχους των εφαρμογών και υπηρεσιών αυτών. Έτσι λοιπόν, ένα σύστημα επίβλεψης 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 (*multi-tenant*). 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 επίβλεψης, 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 [66] , 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 επίβλεψης 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. [67][68]

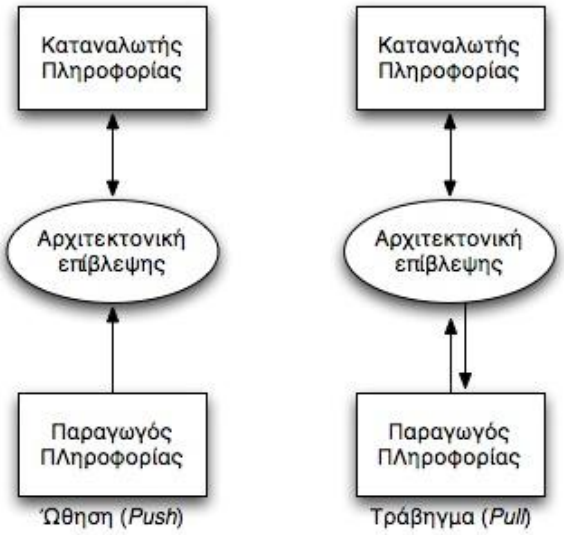
Μια ακόμη παράμετρος που θα πρέπει να ληφθεί υπόψη είναι η απόδοση του μηχανισμού επίβλεψης όταν αυτός λειτουργεί σε περιορισμένο χρονικά περιβάλλον (όλο και περισσότερες εφαρμογές πολυμέσων εγκαθίστανται σε εικονικές υποδομές εκμεταλλευόμενες τις δυνατότητες του υπολογιστικού Νέφους, προβάλλοντας χρονικούς περιορισμούς λειτουργίας [69]). 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 [64]. Φυσικά, είναι αδύνατο για οποιοδήποτε σύστημα επίβλεψης 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.

3.2.2 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 επίβλεψη 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.



Σχήμα 21: 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 σε προκαθορισμένες 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 επίβλεψης στην πλευρά του παραγωγού έχει ένα διπλό αντίκτυπο: δεν

επιβαρύνεται υπολογιστικά και λειτουργικά η μεριά του καταναλωτή, αλλά από την άλλη ο έλεγχος της λειτουργίας επίβλεψης είναι κατανεμημένος στους διαφόρους παραγωγούς. Ως εκ τούτου, μια εσφαλμένη λειτουργία του μηχανισμού συλλογής και αποστολής δεδομένων θα μπορούσε να μην αναγνωριστεί. Τελικά, μπορούμε να σχολιάσουμε, ότι το συγκεκριμένο πρότυπο είναι κατάλληλο σε συστήματα επίβλεψης στα οποία ο όγκος δεδομένων και η συχνότητα παρακολούθησης δεν θα επιβαρύνουν σημαντικά την δικτυακή υποδομή και επίσης μια εσφαλμένη λειτουργία του μηχανισμού επίβλεψης δεν θα επιφέρει κρίσιμες επιπτώσεις στην λειτουργία του συστήματος συνολικά.

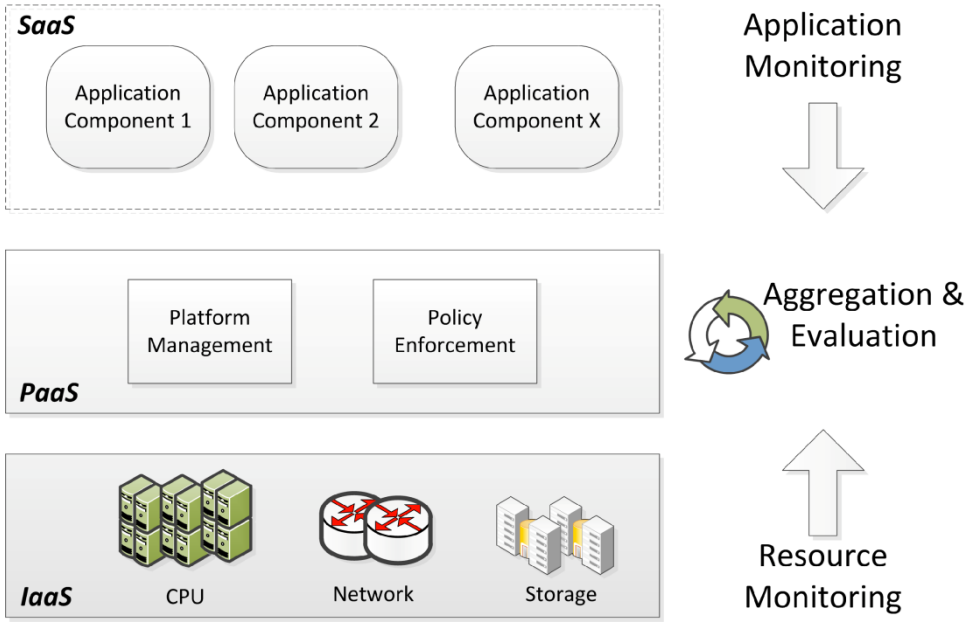
Στο πρότυπο τραβήγματος, ο καταναλωτής είναι αυτός που εκκινεί την διαδικασία μέσω μιας αίτησης. Υπό αυτήν τη μορφή, το χαμηλό ποσοστό τραβήγματος καταναλώνει λίγο εύρος ζώνης δικτύων, αλλά μπορεί να υπονοήσει απώλεια ενημέρωσης, το οποίο είναι ανεπιθύμητο για τους καταναλωτές. Εντούτοις, οι πληροφορίες 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 επίβλεψης και στις δύο μεριές (καταναλωτής, παραγωγός), κάτι που επίσης επιβαρύνει την όλη εγκατάσταση και λειτουργία. Από την άλλη, η αρχιτεκτονική αυτού του συστήματος επίβλεψης 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 επίβλεψης (π.χ. κατά τη διάρκεια της νύχτας).

Στην βιβλιογραφία έχουν προταθεί και συνδυαστικές μέθοδοι (P&P) [69], στις οποίες εφαρμόζονται και οι δύο τεχνικές επίβλεψης 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.

3.2.3 Επίβλεψη 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 στο Σχήμα 22.

Εξ ‘αιτίας της πληθώρας δεδομένων επίβλεψης, 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 επίβλεψης 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.



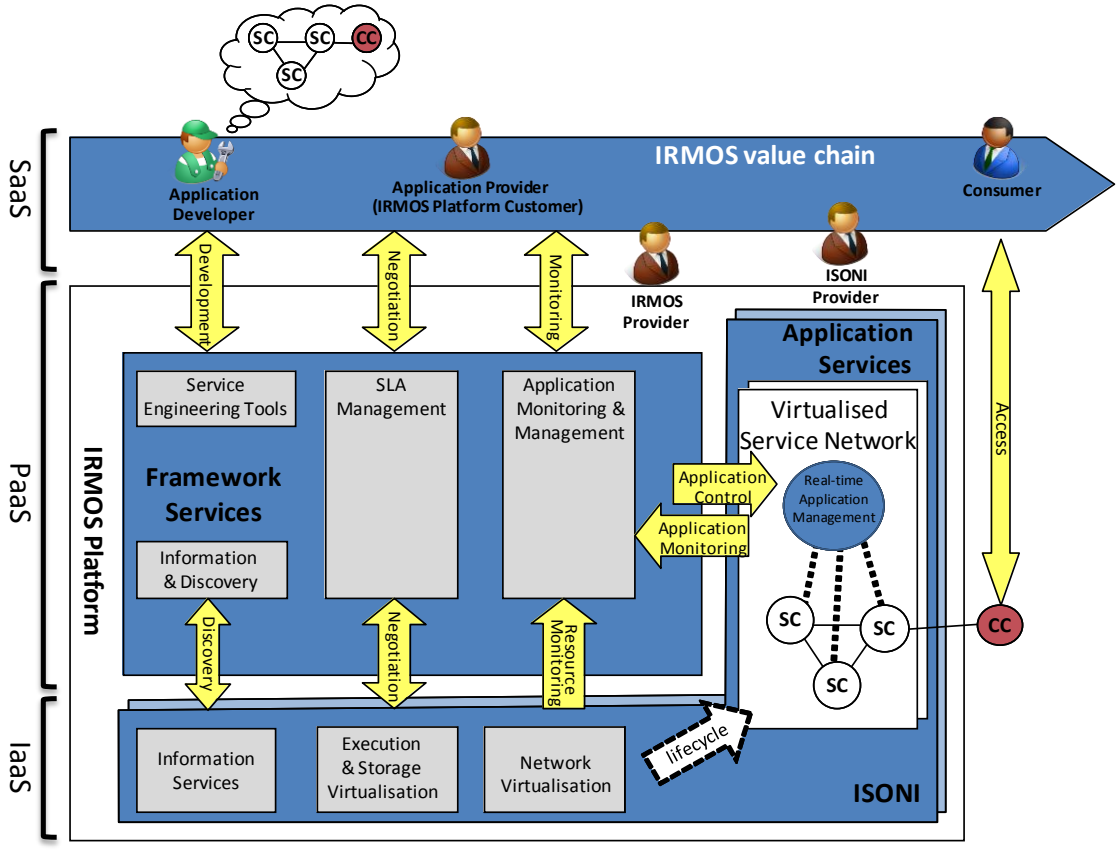
Σχήμα 22: Ροή πληροφορίας επίβλεψης

4 *Μηχανισμός παρακολούθησης και επίβλεψης πολλαπλών στρωμάτων*

Στο πλαίσιο αυτής της διδακτορικής διατριβής και κατόπιν της ανάλυσης των προδιαγραφών λειτουργίας, προχωρήσαμε στον σχεδιασμό, υλοποίηση και αξιολόγηση ενός μηχανισμού επίβλεψης και παρακολούθησης εφαρμογών και πόρων σε ένα υπηρεσιοστραφές περιβάλλων Νέφους. Ο προτεινόμενος μηχανισμός επίβλεψης εκτείνεται και στα τρία στρώματα του Νέφους, αποτελούμενος από πολλαπλές υπηρεσίες και δομικά συστατικά εγκατεστημένα στο SaaS, PaaS και IaaS στρώμα. Η εργασία αυτή υλοποιήθηκε στο πλαίσιο του ευρωπαϊκού ερευνητικού προγράμματος IRMOS. Στα υποκεφάλαια που ακολουθούν, τοποθετείται αρχικά η προτεινόμενη λύση στον συνολικό σχεδιασμό της πλατφόρμας του IRMOS και στη συνέχεια παρουσιάζεται λεπτομερώς η αρχιτεκτονική του μηχανισμού όπως επίσης και οι λειτουργίες κάθε μέρους ξεχωριστά. Τέλος, ο μηχανισμός αξιολογείται σε διάφορες συνθήκες λειτουργίας όπου και παρατίθενται τα αποτελέσματα.

4.1 Η πλατφόρμα IRMOS

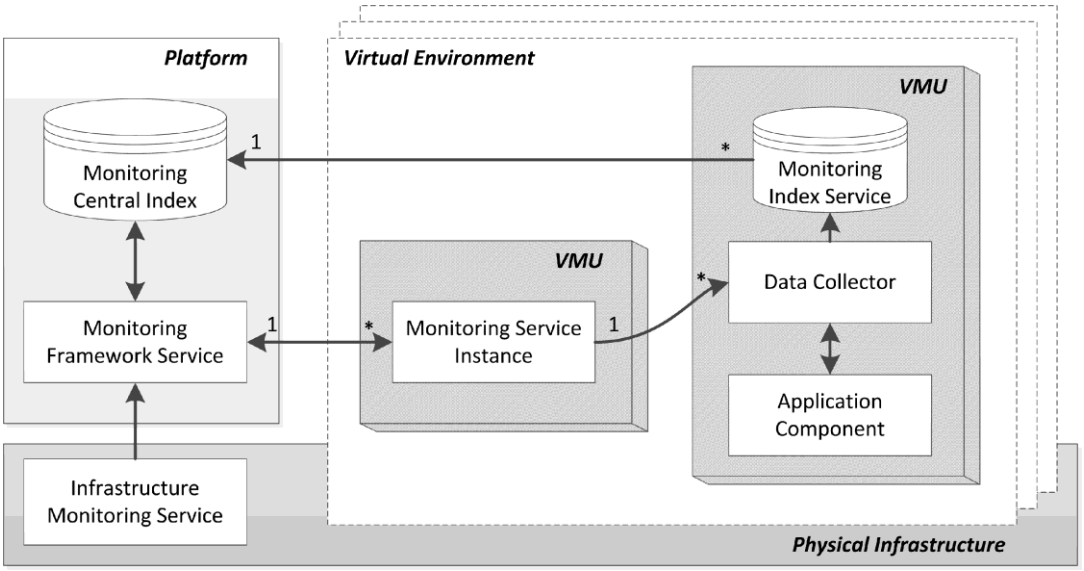
Η πλατφόρμα IRMOS είναι ένα σύνολο υπηρεσιών που επιτρέπουν την προσαρμογή, εγκατάσταση και εκτέλεση διαδραστικών εφαρμογών, με απαιτήσεις ταυτοχρονισμού (real-time), σε περιβάλλοντα Νεφών. Το σύνολο αυτών των υπηρεσιών καλύπτει και τα τρία στρώματα της αρχιτεκτονικής του Νέφους (SaaS, PaaS, IaaS) και παρέχει μια ολική λύση για την διαχείριση της πληροφορίας συμπεριλαμβάνοντας ένα σύστημα επίβλεψης και παρακολούθησης. Η πολύ-επίπεδη δομή της πλατφόρμας αυτής που απορρέει από το αρχιτεκτονικό παράδειγμα του Υπολογιστικού Νέφους οδήγησε στον σχεδιασμό και υλοποίηση ενός μηχανισμού επίβλεψης που συλλέγει δεδομένα από την υποδομή (IaaS) και τις εφαρμογές (SaaS) με την διαχείριση να πραγματοποιείται από τις υπηρεσίες της πλατφόρμας (PaaS), όπως φαίνεται και στο Σχήμα 23. Στα υποκεφάλαια που ακολουθούν αναλύεται με λεπτομέρεια ο σχεδιασμός, η υλοποίηση, η αξιολόγηση όπως και συγκεκριμένα χαρακτηριστικά και δυνατότητες του συστήματος αυτού.



Σχήμα 23: Η πλατφόρμα IRMOS

4.2 Αρχιτεκτονική σχεδίαση μηχανισμού

Ο μηχανισμός που προτείνεται, αποτελείται από έξι δομικά συστατικά, όπως παρουσιάζεται στο Σχήμα 24, τα οποία είναι εγκατεστημένα σε όλα τα στρώματα του Νέφους: α) στο PaaS για την συγκέντρωση και συνδυασμό δεδομένων από την εφαρμογή και την υποδομή, όπως και την αξιολόγηση των πληροφοριών αυτών, β) στο IaaS για την επίβλεψη τον πόρων της υποδομής (μηχανημάτων, υπολογιστών κλπ), γ) στο SaaS για την επίβλεψη της εφαρμογής/λογισμικού.



Σχήμα 24: Αρχιτεκτονική συστήματος παρακολούθησης πολλαπλών επιπέδων

4.2.1 Στρώμα PaaS

Monitoring Framework Service (MFS): Η υπηρεσία αυτή αποτελεί το βασικό δομικό συστατικό του μηχανισμού αφού διοργανώνει και διαχειρίζεται τη επίβλεψη όλων των εφαρμογών στο εικονικό περιβάλλον και έχει πρόσβαση στην συγκεντρωμένη πληροφορία μέσω του δομικού συστατικού Monitoring Central Index που περιγράφεται παρακάτω. Επιπρόσθετα, η MFS μπορεί να εξυπηρετεί παράλληλα πολλαπλές εφαρμογές που είναι εγκατεστημένες σε διαφορετικές εικονικές μηχανές. Το συστατικό αυτό παρέχει τις απαραίτητες διεπαφές έτσι ώστε οι υπόλοιπες υπηρεσίες της πλατφόρμας να αλληλεπιδρούν μαζί της, αλλά και οι υπηρεσίες που βρίσκονται στα επίπεδα της εικονικής και πραγματικής

υποδομής να μπορούν να έχουν πρόσβαση στην υπηρεσία αυτή. Ένα ακόμα σημαντικό χαρακτηριστικό της MFS είναι η αξιολόγηση των δεδομένων βάση συγκεκριμένων πολιτικών/κανόνων που επιτρέπουν τον επαναπροσδιορισμό του συνολικού μηχανισμού επίβλεψης ή την ενεργοποίηση διορθωτικών κινήσεων με σκοπό την επαναφορά του συστήματος από συνθήκες λανθασμένης λειτουργίας.

Monitoring Central Index: Αποτελεί την μονάδα αποθήκευσης δεδομένων της πλατφόρμας. Οι υψηλού επιπέδου παράμετροι παρακολούθησης όλων των δομικών συστατικών μια εφαρμογής, όπως επίσης και οι χαμηλού επιπέδου παράμετροι που προέρχονται από την υποδομή, αποθηκεύονται στο στοιχείο αυτό. Οι τιμές των παραμέτρων που καταχωρούνται εδώ ανανεώνονται με διαφορετική συχνότητα βάση της ορισμένης τιμής κάθε διαφορετικού δομικού συστατικού μιας εφαρμογής.

4.2.2 Στρώμα IaaS

Infrastructure Monitoring Service: Η υπηρεσία αυτή είναι υπεύθυνη για την συλλογή χαμηλού επιπέδου πληροφορίας, σχετικά με την λειτουργία των Μονάδων Εικονικών Μηχανών (VMUs) στου φυσικούς πόρους, και την δημοσίευση σχετικών αναφορών στο στοιχείο αποθήκευσης Monitoring Central Index. Μέσω συγκεκριμένων ελέγχων υπηρεσίας (service checks) στην φυσική υποδομή, αυτή η υπηρεσία παρακολούθησης συγκεντρώνει και συνδυάζει τα δεδομένα δημιουργώντας αναφορές γραμμένες σε XML.

4.2.3 Στρώμα SaaS

Monitoring Service Instance (MSI):

Τα στιγμιότυπα αυτής της υπηρεσίας επίβλεψης (MSI) βρίσκονται εγκατεστημένα μέσα σε Μονάδες Εικονικής Μηχανής (VMUs) και η κάθε υπηρεσία είναι μοναδική για κάθε δομικό συστατικό μιας ροής εργασίας. Το στοιχείο αυτό παρέχει διεπαφές προς την υπηρεσία MFS έτσι ώστε η τελευταία να εκκινήσει την λειτουργία παρακολούθησης και επίβλεψης. Κατά

την διάρκεια αυτής της διαδικασίας, παράμετροι ρύθμισης μεταφέρονται από την MFS στην MSI σχετικά με την παρακολούθηση κάθε ροής εργασίας: η ιδιωτική διεύθυνση της VMU, η χρονική συχνότητα επίβλεψης κάθε Συλλέκτη Δεδομένων και το όνομα κάθε εκτελέσιμου Συλλέκτη Δεδομένων.

Monitoring Index Service: Η υπηρεσία αυτή αποτελεί την τοπική αποθηκευτική μονάδα για τις παραμέτρους που παρακολουθούνται σε κάθε δομικό συστατικό μιας εφαρμογής. Εγκαθίσταται στην ίδια VMU με το συστατικό της εφαρμογής που επιβλέπει, έτσι ώστε ο αντίστοιχος Συλλέκτης Δεδομένων να παρέχει τις απαιτούμενες πληροφορίες (KPIs κ.α.). Εκτός από την αποθήκευση των δεδομένων, τα δημοσιεύει ταυτόχρονα στο Central Monitoring Index της πλατφόρμας, εξωτερικά του Εικονικού Περιβάλλοντος. Οι τιμές στην υπηρεσία Monitoring Index Service ανανεώνονται κάθε φορά που ένας Συλλέκτης παραδίδει ένα καινούριο σετ τιμών.

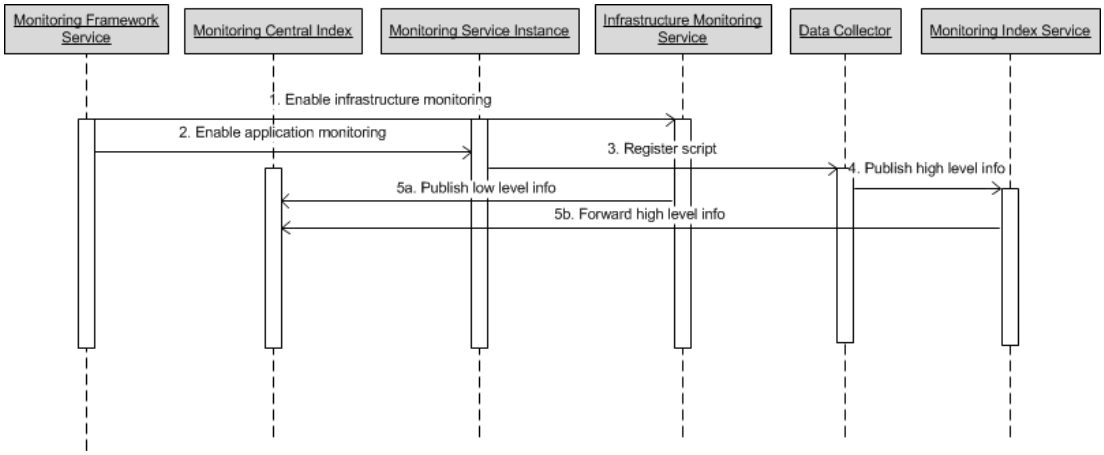
Data Collector: Το δομικό συστατικό αυτό είναι εγκατεστημένο στην ίδια μονάδα VMU στην οποία βρίσκεται ένα συστατικό μιας εφαρμογής. Με αυτόν τον τρόπο επιτυγχάνεται άμεση πρόσβαση στα στοιχεία της ροής εργασίας μια εφαρμογής και συλλέγονται τα δεδομένα έτσι ώστε να μεταφερθούν στο τοπικό Monitoring Index. Η υλοποίηση αυτού είναι μέρος της διαδικασίας προσαρμογής μιας εφαρμογής από τον προγραμματιστή έτσι ώστε να είναι συμβατή η λειτουργία της σε περιβάλλον Νέφους (ή υπηρεσιοστρεφούς γενικά αρχιτεκτονικής). Ο Συλλέκτης Δεδομένων εκτελείται αυτόματα με την συχνότητα λειτουργίας που προσδιορίζεται από την MSI υπηρεσία κατά την διάρκεια της διαδικασίας εγγραφής. Το αποτέλεσμα της εκτέλεσης αυτής είναι ένα σετ παραμέτρων (όνομα, τιμή, μονάδα κ.α.) που περιγράφεται σε γλώσσα XML και παραδίδεται στο τοπικό Monitoring Index.

4.3 Λειτουργία και αρχικοποίηση πλαισίου επίβλεψης

Η εγκατάσταση και η αρχικοποίηση του μηχανισμού ολοκληρώνεται σε δύο φάσεις: τα δομικά συστατικά που βρίσκονται στα στρώματα PaaS και IaaS διαμορφώνονται και

εγκαθίστανται στατικά/χειροκίνητα, ως μέρος του μόνιμου πλαισίου (*offline phase*) ενώ τα συστατικά στο στρώμα SaaS αρχικοποιούνται και εγκαθίστανται δυναμικά, κατά την διάρκεια εγκατάστασης της εφαρμογής (*online phase*). Λεπτομερέστερα η λειτουργία των συστημάτων παρουσιάζεται μέσω της ακολουθίας αλληλεπιδράσεων (Σχήμα 25):

1. Το MFS ενεργοποιεί την επίβλεψη της υποδομής (Infrastructure Monitoring) με την επίκληση της αντίστοιχης υπηρεσίας. Το τελευταίο συλλέγει τις χαμηλού επιπέδου πληροφορίες (π.χ. Ταχύτητα CPU, χρήση σκληρών δίσκων, εύρος ζώνης κ.λπ.) από τους φυσικούς πόρους όπου το VMUs της εφαρμογής είναι εγκατεστημένο.
2. Το MFS ενεργοποιεί την επίβλεψη της εφαρμογής με την επίκληση του MSI μέσα στο εικονικό περιβάλλον.
3. Το MSI εγγράφει το συλλέκτη δεδομένων (Data Collector) κάθε τμήματος εφαρμογής (*application component*) της ροής εργασίας στην κάθε υπηρεσία Monitoring Index Service.
4. Ο συλλέκτης δεδομένων δημοσιεύει τις υψηλού επιπέδου πληροφορίες παρακολούθησης του τμήματος εφαρμογής στην Monitoring Index Service. Οι τιμές των παραμέτρων ανανεώνονται μέσα στο Index με μια κοκκοποίηση/συχνότητα που καθορίζεται κατά τη διάρκεια της εγγραφής.
5. Η υπηρεσία Monitoring Index Service διαβιβάζει τις υψηλού επιπέδου ελεγχόμενες πληροφορίες στον Monitoring Central Index. Από αυτό, οι πληροφορίες όλων των συστατικών του εικονικού περιβάλλοντος δημοσιεύονται για την κεντρική αποθήκη στο επίπεδο πλατφορμών (5b). Αντίστοιχα, η Infrastructure Monitoring Service παραδίδει τις χαμηλού επιπέδου ελεγχόμενες παραμέτρους στο Central Index (5a).

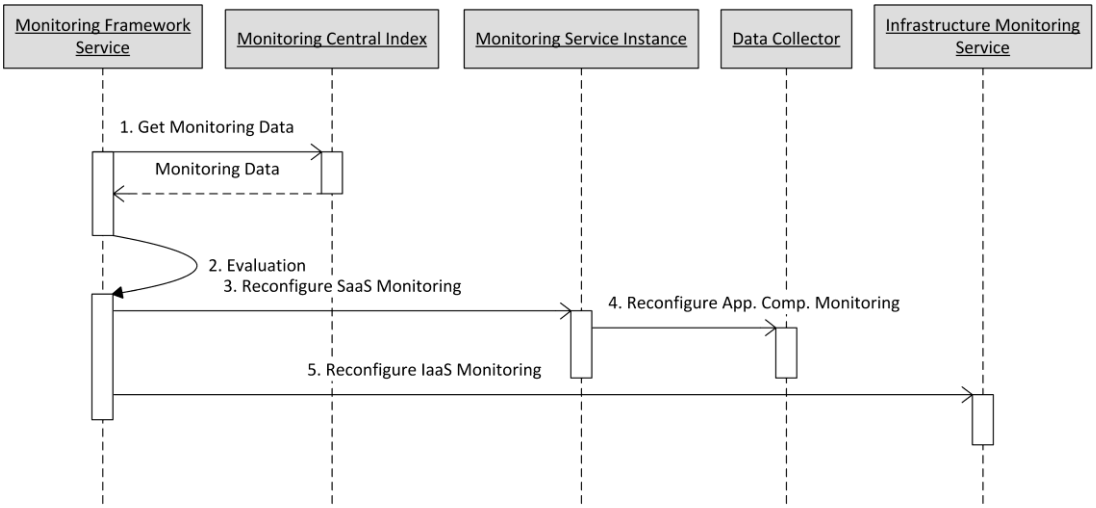


Σχήμα 25: Ακολουθία αλληλεπιδράσεων της λειτουργίας επίβλεψης

4.4 Αναδιαμόρφωση κατά τη διάρκεια εκτέλεσης

Ένα σημαντικό χαρακτηριστικό γνώρισμα του παρουσιαζόμενου μηχανισμού επίβλεψης είναι η δυναμική διαμόρφωση βασισμένη στις προτιμήσεις τελικών χρηστών ή τις προκαθορισμένες πολιτικές εφαρμογής ή πολιτικές της πλατφόρμας (*policies*). Αυτό το χαρακτηριστικό επιτρέπει σε όλα τα στοιχεία επίβλεψης (*Data Collectors* και *Infrastructure Monitoring Service*) να επαναπροσδιορίσουν τον κατάλογο του ελεγχόμενων KPIs ή των πόρων καθώς επίσης και του χρονικού διαστήματος ελέγχου. Αυτός ο βρόχος ελέγχου μεταξύ της πλατφόρμας και του εικονικού περιβάλλοντος δεν έχει επιπτώσεις ούτε διακόπτει την εκτέλεση του τμήματος εφαρμογής (*Application Component*) αλλά μόνο επαναπροσδιορίζει τη χρονική συχνότητα λειτουργίας και τις παραμέτρους έτσι ώστε να αναγκάσει το σύστημα για να πάρει περισσότερες μετρήσεις είτε να μειώσει το ποσοστό δειγματοληψίας στα τμήματα είτε τις περιόδους εφαρμογής που δεν είναι κρίσιμες.

Εκείνες οι προσωποποιημένες πολιτικές καθώς επίσης και το ίδιο το στοιχείο Data Collector αναπτύσσονται σε συνεργασία με τον υπεύθυνο για την ανάπτυξη εφαρμογής καθώς και οι δύο οντότητες πρέπει να προσαρμοστούν στις ανάγκες εφαρμογής. Στην τρέχουσα έκδοση του μηχανισμού που παρουσιάζεται η λειτουργία της πολιτικής είναι ενσωματωμένη με την υπηρεσία Monitoring Framework Service (Σχήμα 26) αλλά είναι μέσα στα μελλοντικά βήματά μας να υλοποιήσουμε ένα αυτόνομο συστατικό για να χρησιμεύσει ως αποθήκη πολιτικών (*policy registry*).

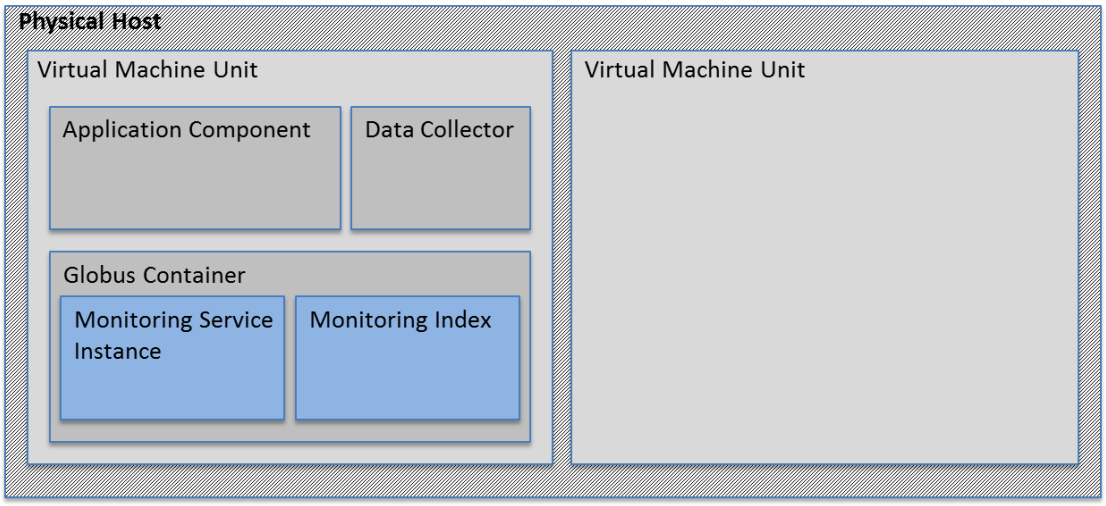


Σχήμα 26: Ακολουθία αλληλεπιδράσεων της λειτουργίας αναδιαμόρφωσης

Αυτός ο βρόχος ελέγχου ακολουθεί μια συναρμολογούμενη δομή έτσι ώστε το σύστημα να μπορεί να είναι εύκαμπτο και εξατομικεύσιμο για κάθε ροή της εργασίας εφαρμογής που εκτείνεται πάνω στην εικονική υποδομή. Καθώς η υπηρεσία Monitoring Framework Service βρίσκεται στο στρώμα PaaS (έτσι ώστε να έχει πρόσβαση στα δεδομένα επίβλεψης της εφαρμογής, τις πληροφορίες υποδομής καθώς επίσης και πρόσβαση σε ολόκληρο το πλαίσιο υπηρεσιών), οι υπεύθυνοι για την ανάπτυξη εφαρμογής μπορούν να επεκτείνουν τις σύνθετες όμως έξυπνες πολιτικές που θα τους επιτρέψουν να επιτύχουν καλύτερη απόδοση της εφαρμογής τους. Η λειτουργία αυτού του μηχανισμού ολοκληρώνεται σε τέσσερα βασικά βήματα: (1) απόκτηση των δεδομένων από το Central Index, (2) αξιολόγηση των δεδομένων ενάντια στις καθορισμένες πολιτικές, (3) ενεργοποίηση του SaaS αναδιαμόρφωσης (*reconfiguring*) στην υπηρεσία Monitoring Service Instance της αντίστοιχης εφαρμογής, (4) εφαρμογή του επαναπροσδιορισμού στους συλλέκτες δεδομένων (Data Collectors) όλων των τμημάτων εφαρμογής και (5) επαναπροσδιορισμός της υπηρεσίας IaaS Monitoring Service.

4.5 Υλοποίηση

Η υλοποίηση του υπηρεσιοστραφή μηχανισμού μας βρίσκεται σε διαφορετικά στρώματα σχεδίασης όπως απεικονίζονται στο Σχήμα 27. Κάθε φυσικός κόμβος (*Physical Host*) της υποδομής φιλοξενεί μία ή και πολλές μονάδες εικονικών μηχανών (*Virtual Machine Units* - VMUs), σε κάθε VMU έχουμε εγκαταστήσει ένα ή πολλά τμήματα εφαρμογής (*Application Components*) και συλλέκτες δεδομένων (*Data Collectors*) αντίστοιχα (υπάρχει ένα προς ένα σχέση μεταξύ των τμημάτων εφαρμογής και του κάθε συλλέκτη δεδομένων), τελικά κάθε VMU φιλοξενεί ένα εξυπηρετητή εφαρμογών (*application container*) Globus που έχει εγκατεστημένες τις υπηρεσίες Monitoring Service Instance και Monitoring Index για να ενεργοποιηθεί και λειτουργήσει ο μηχανισμός μέσα στο εικονικό περιβάλλον



Σχήμα 27: Διαστρωματική αρχιτεκτονική

Το μεσο-λογισμικό (*middleware*) που επιλέχτηκε για την εφαρμογή των βασικών υπηρεσιών ήταν το Globus Toolkit 4 (GT4) [61]. Το GT4 είναι ένα λογισμικό πλέγματος ανοικτού κώδικα που παρέχει την απαραίτητη λειτουργικότητα για να δημιουργήσει κάποιος και να εγκαταστήσει πλήρως λειτουργικές υπηρεσίες διαδικτύου (Web services). Παρέχει επίσης μια διεπαφή προγραμματισμού (API) που υποστηρίζει το πλαίσιο *Web Services Resource Framework* (WSRF) , *WS-Addressing*, *WS-Security* και *WS-BaseNotification*. Η επιλογή αυτού του API βασίστηκε στην ύπαρξη του συστήματος Monitor and Discovery System (MDS) του GT4: ένα εξειδικευμένο μηχανισμό που παρέχει πολλαπλές διεπαφές (*Aggregator*

Framework) για τη αλληλεπίδραση WSRF υπηρεσιών, μηχανισμό συνδρομής (*subscription*) για την συλλογή δεδομένων μέσω *WS- Notification* όσο και από την εκτέλεση εξωτερικών προγραμμάτων. Για αυτόν τον λόγο, αναπτύξαμε τις υπηρεσίες *Monitoring Framework Service, Monitoring Service Instance* και *Infrastructure Monitoring Service* ως *state-full* υπηρεσίες WSRF που εγκαθίστανται στο αυτόνομο GT4 container.

4.5.1 Υπηρεσία Επίβλεψης (Monitoring Service, Framework and Instance)

Η υπηρεσία Monitoring Framework Service είναι μια υπηρεσία WSRF Globus που υλοποιήθηκε με την γλώσσα προγραμματισμού Java και που εγκαθίστανται στο στρώμα πλατφόρμας. Υπάρχουν διαδικασίες για την ενεργοποίηση της επίβλεψης εφαρμογής (*application monitoring*) καθώς επίσης και για την απενεργοποίηση, χρησιμοποιώντας την κατάλληλη διεύθυνση αναφοράς (*Endpoint Reference* - EPR) κάθε στιγμιοτύπου της Monitoring Service Instance. Από την άλλη, η υπηρεσία Monitoring Service Instance είναι ένα *WS-Resource* (όπως καθορίζεται στην προδιαγραφή WSRF), που αρχικοποιείται από την *Monitoring Framework Service*, διαμορφώνεται με τις συγκεκριμένες παραμέτρους μιας ροής εργασίας μιας εφαρμογής και εγκαθίσταται μέσα στην εικονική υποδομή. Σε αυτό το πλαίσιο, ένα στιγμιότυπο a Monitoring Service περιλαμβάνει όλες τις λεπτομέρειες κάθε συστατικού (*private IP, name, ID, Data Collector time interval* κ.λπ.). Η εγγραφή κάθε συλλέκτη δεδομένων γίνεται χρησιμοποιώντας το *Aggregator Framework* που παρέχεται από τον MDS μηχανισμό. Χρησιμοποιώντας αυτό το API μπορούμε να καθορίσουμε ορισμένες παραμέτρους για το μεμονωμένο συλλέκτη στοιχείων μέσω ενός αρχείου διαμόρφωσης XML.

```
<ServiceGroupRegistrationParameters>
<!–Renew this registration every 600 seconds (10 minutes –>
<RefreshIntervalSecs>10</RefreshIntervalSecs>
<Content xsi:type=”agg:AggregatorContent”
xmlns:agg=”http://mds.globus.org/aggregator/types”>
<agg:AggregatorConfig xsi:type=”agg:AggregatorConfig”>
<agg:ExecutionPollType>
<!– Run our script every 1000 milliseconds –>
<agg:PollIntervalMillis>1000</agg:PollIntervalMillis>
<agg:ProbeName>datacollector</agg:ProbeName>
```

```
</agg:ExecutionPollType>
</agg:AggregatorConfig>
</Content>
</ServiceGroupRegistrationParameters>
```

Πίνακας 4: Registration / Configuration file

Όπως μπορεί να φανεί στον Πίνακας 4, η συχνότητα εκτέλεσης ενός συλλέκτη (*PollIntervalMills*), όπως και το όνομα του εκτελέσιμου αρχείου του συλλέκτη (*ProbeName*) είναι διαμορφώσιμο και μπορεί να αλλάζει βάσει των συγκεκριμένων αναγκών κάθε συστατικού εφαρμογής.

4.5.2 Συλλέκτης Δεδομένων (Data Collector)

Ο συλλέκτης δεδομένων (Data Collector) είναι ουσιαστικά ένα εκτελέσιμο αρχείο που γράφεται σε οποιαδήποτε γλώσσα (perl,shell script κ.λπ.) και είναι μια ευθύνη του υπεύθυνου για την ανάπτυξη εφαρμογής. Αυτό οφείλεται στο γεγονός ότι γνωρίζει τη λειτουργία των δομικών συστατικών της εφαρμογής και τον τρόπο συλλογής των παραμέτρων υψηλού επιπέδου (και ποιες από αυτές είναι σημαντικότερες). Αυτό το συστατικό εγκαθίσταται μέσα στην VMU προκειμένου να αλληλεπιδράσει με το Application Component και να συλλεχθούν οι απαραίτητες πληροφορίες. Η μόνη απαίτηση είναι ότι η εκτέλεση του αρχείου να παραγάγει ένα έγκυρο κείμενο XML συμπεριλαμβανομένης της ταυτότητας του συστατικού (*ID*), χρονική σφραγίδα (*timestamp*) της παραγμένης έκθεσης καθώς επίσης και των πληροφοριών για τις παραμέτρους (Πίνακας 5).

```
<MonitoringData>
<Application ID>xxx</Application ID>
<Component ID>yyy</Component ID>
<timestamp>yyyy-dd-mm
HH:mm:ss:msec</timestamp>
<MonitoringParameter>
<Name>name</Name>
<Value>value</Value>
<Type>type</Type>
<Unit>unit</Unit>
</MonitoringParameter>
</MonitoringData>
```

Πίνακας 5: Σχήμα δεδομένων της αναφοράς επίβλεψης

Κατόπιν της εγγραφής, από το Monitoring Service Instance, ο συλλέκτης δεδομένων αρχίζει να εκτελείται με την προσδιορισμένη συχνότητα λειτουργίας. Αυτός, αυτόματα δημοσιεύει το XML κείμενο στο τοπικό Monitoring Index.

4.5.3 Monitoring Index (Central and Local)

Το Central Monitoring Index όπως και το Monitoring Index (μέσα στην VMU) υλοποιούνται βασισμένα στην υπηρεσία Index Service που παρέχεται από GT4. Αυτή η υπηρεσία Index Service είναι ένα ευέλικτο στοιχείο αποθήκευσης το οποίο συλλέγει τις πληροφορίες και τις δημοσιεύει ως ιδιότητες των πόρων. Μπορούμε να ανακτήσουμε τις πληροφορίες από ένα Index με προγράμματα-πελάτες που χρησιμοποιούν τις τυποποιημένες διεπαφές ερώτησης WSRF όπως επίσης και από διεπαφές εγγραφής/ανακοίνωσης (*subscription/notification*). Επιπλέον, τα Indexes μπορούν να καταχωρήσουν ο ένας στον άλλο σε μια ιεραρχική μορφή (προς τα πάνω/προς τα κάτω εγγραφή) προκειμένου να αθροιστούν τα στοιχεία σε διάφορα επίπεδα. Η λύση μας εκμεταλλεύεται το χαρακτηριστικό αυτό προκειμένου να δημοσιευθούν οι υψηλού επιπέδου πληροφορίες επίβλεψης από κάθε Index μέσα στο VMUs στον κεντρικό Index επίβλεψης της πλατφόρμας (Monitoring Central Index).

Λεπτομερέστερα, ο μηχανισμός μας εκμεταλλεύεται μια διπλή ροή των πληροφοριών:

- Ο συλλέκτης δεδομένων (Data Collector) δημοσιεύει τα υψηλού επιπέδου στοιχεία επίβλεψης μέσω του πλαισίου Aggregator στο τοπικό Index. Σε αυτήν την περίπτωση κάνουμε χρήση της λειτουργίας πηγής εκτέλεσης που παρέχεται από το πλαίσιο. Ο τοπικός δείκτης μέσω μιας προς τα πάνω εγγραφής δημοσιεύει τα στοιχεία στον κεντρικό δείκτη ελέγχου.
- Η υπηρεσία υποδομής Infrastructure Monitoring Service, ως πηγή Aggregator (Subscription Aggregator Source), δημοσιεύει τις χαμηλού επιπέδου πληροφορίες επίβλεψης κατευθείαν στο Monitoring Central Index.

4.5.4 Υπηρεσία επίβλεψης της υποδομής (Infrastructure Monitoring Service)

Η υπηρεσία υποδομής Infrastructure Monitoring Service είναι μια υπηρεσία WSRF που συλλέγει τις χαμηλού επιπέδου παραμέτρους από το φυσικό πόρο σχετικά με την εκτέλεση και τις αλληλεπιδράσεις των VMUs. Η υπηρεσία παράγει τις εκθέσεις συμπεριλαμβανομένων των πληροφοριών για την απόδοση των εικονικών κόμβων (VMUs) καθώς επίσης και των συνδέσεων μεταξύ τους (Πίνακας 6). Η υπηρεσία έχει υλοποιηθεί βάσει των προδιαγραφών WS-Notification πραγματοποιώντας τον ακόλουθο μηχανισμό: ένας καταναλωτής των συλλεχθεισών πληροφοριών πρέπει να εγγραφεί στην υπηρεσία και να περιμένει έως ότου η Infrastructure Monitoring Service μεταδώσει την έκθεση υπό μορφή ανακοίνωσης (*notification*). Η υπηρεσία συνθέτει μια νέα έκθεση και επομένως προκαλεί την αποστολή ανακοίνωσης σε σταθερά χρονικά διαστήματα καθώς επίσης και όποτε υπάρχει μια σημαντική αποτυχία (διακοπή στο δίκτυο κ.λπ.).

<Avg used Bandwidth> <Value>xxx</Value> <Unit>kbps</Unit> </Avg used Bandwidth> <Avg Delay msec> <Value>yyy</Value> <Unit>msec</Unit> </Avg Delay msec> <Avg Jitter msec> <Value>zzz</Value> <Unit>msec</Unit> </Avg Jitter msec>	<CPU load> <Value>xxx</Value> <Unit>%</Unit> </CPU load> <Phys RAM> <Value>yyy</Value> <Unit>%</Unit> </Phys RAM> <Used volatile storage> <Value>zzz</Value> <Unit>%</Unit> </Used volatile storage>

Πίνακας 6: Παραδείγματα αναφορών επίβλεψης φυσικής υποδομής

Είναι σημαντικό ότι η έκθεση θα περιλαμβάνει συνεπή προσδιοριστικά (*identifiers*) με αυτά που χρησιμοποιούνται από το συλλέκτη δεδομένων κατά τη διάρκεια της επίβλεψης υψηλού επιπέδου (*high level monitoring*), προκειμένου να είναι σε θέση να αθροίσει τις πληροφορίες κατόπιν. Η συλλογή των στοιχείων εκτελείται μέσω του Nagios, ένα εργαλείο επίβλεψης και παρακολούθησης που είναι σε θέση να αναφέρει τη κατάσταση των διάφορων μετρικών της υποδομής.

4.5.5 Συνάθροιση (Aggregation)

Όπως παρουσιάστηκε στα προηγούμενα υποκεφάλαια, η ανάκτηση των συλλεχθεισών πληροφοριών σχετικά με τις συγκεκριμένες μετρικές της εφαρμογής και τις παραμέτρους κατάστασης υποδομής πραγματοποιείται με ασύγχρονο τρόπο: τα πρώτα μαζεύονται στο Central Index της πλατφόρμας ενώ τα δεύτερα καταναλώνονται μέσω μιας διεπαφής συνδρομής (*subscription interface*). Επομένως, η συνάθροιση όλων των πληροφοριών πρέπει να γίνει με έναν αποδοτικό τρόπο στο επίπεδο της πλατφόρμας (συγκεκριμένα από την υπηρεσία Monitoring Framework Service) βασισμένη στον ακόλουθο ψευδοκώδικα:

```
Infrastructure Report := InfrastructureMonitoringService. notification
for each Virtual Machine in the report
        DeploymentManager.getApplicationIDs list(VM ID)
        DeploymentManager.getComponentIDs list (VM ID)
        for each Application
                for each Component
                ComponentReport := Central Index.getMonitoring Report(ApplicationID, ComponentID)
                Aggregate(Infrastructure Report, ComponentReport ,ApplicationID, ComponentID)
```

Το αποτέλεσμα αυτής της διαδικασίας συνάθροισης (*aggregation*) είναι μια συνενωμένη έκθεση επίβλεψης που περιγράφει την κατάσταση της υποδομής, τις εφαρμογές που είναι εγκατεστημένες πάνω σε αυτήν καθώς επίσης και τις μετρικές QoS για κάθε συστατικό μιας εφαρμογής. Αυτή η έκθεση μπορεί να παρασχεθεί σε άλλα τμήματα πλατφορμών ή να αποθηκευτεί στην ιστορική βάση δεδομένων της πλατφόρμας για τη μελλοντική χρήση.

4.6 Αξιολόγηση

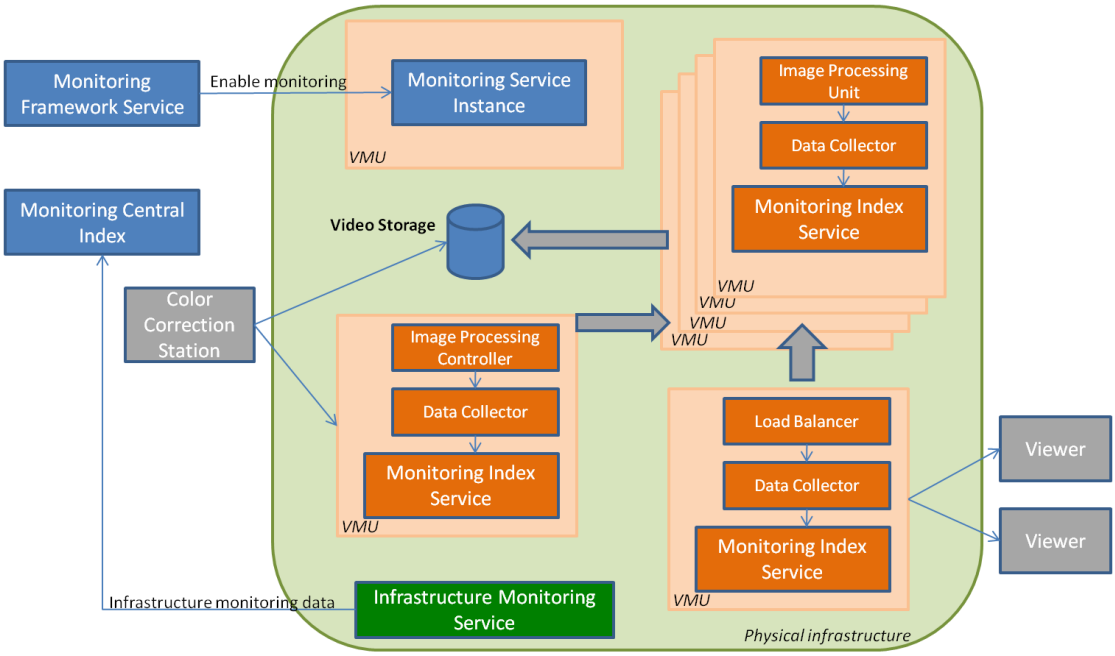
Η προτεινόμενη προσέγγιση βασίζεται στις αρχές σχεδιασμού του παραδείγματος υπηρεσιών υπολογιστικών Νεφών και επομένως ακολουθεί ένα αρχιτεκτονικό πρότυπο πολλαπλών επιπέδων: η εφαρμογή ελέγχεται μέσα από το εικονικό περιβάλλον, οι φυσικοί πόροι της υποδομής αναφέρουν την κατάσταση των ίδιων πόρων και το υπόλοιπο του πλαισίου επίβλεψης, το οποίο βρίσκεται στο επίπεδο της πλατφόρμας, λαμβάνει, αθροίζει και

αποθηκεύει αυτές τις πληροφορίες. Αυτή η ιεραρχική λειτουργία όχι μόνο επιτρέπει την επεκτασιμότητα και ελαστικότητα του συστήματος ανεξάρτητα στα διαφορετικά επίπεδα, αλλά και κρύβει την πολυπλοκότητα του χαμηλού επιπέδου μηχανισμού επίβλεψης από το χρήστη που καταναλώνει τις υπηρεσίες της πλατφόρμας. Το επιλεγμένο υλικολογισμικό παρέχει το απαραίτητο υπηρεσιοστρεφές API προκειμένου να αναπτυχθεί το ιεραρχικό πλαίσιό μας. Σύμφωνα με τα στοιχεία απόδοσης που παρουσιάζονται στην βιβλιογραφία [57][58] η τεχνολογίες του API που χρησιμοποιήσαμε μας επιτρέπουν να επιτύχουμε πολύ μικρό λειτουργικό χρονικό διάστημα και παρόλα αυτά να έχουμε μια υπηρεσία ανταγωνιστική από άποψη real-time εφαρμογής. Επιπλέον, το MDS4 παρέχει διαλειτουργικότητα, επιτρέποντας συλλογή πληροφοριών από τα Ganglia, Hawkeye, Reliable File Transfer Service (RFT) κ.α. Επιπλέον, μέσω της υπηρεσίας GRAM που παρέχεται μπορούμε να εκτελέσουμε, να ελέγξουμε και να επιτηρήσουμε ροές εργασίας όπως παρουσιάστηκε στο [59]. Για την περαιτέρω και πιο λεπτομερή αξιολόγηση έχουμε πραγματοποιήσει μια σειρά πειραμάτων και δοκιμαστικών εκτελέσεων με ένα σενάριο εφαρμογής όπως παρουσιάζεται στα επόμενα υποκεφάλαια.

4.6.1 Πειραματική εφαρμογή και αποτελέσματα

Προκειμένου να εξεταστεί και επικυρωθεί η λειτουργία και η απόδοση του προτεινόμενου μηχανισμού επίβλεψης επιλέχθηκε μία πειραματική εφαρμογή μιας συνεργατικής και κατανεμημένης εφαρμογής διορθώσεων χρώματος (*collaborative and distributed color correction*) που εκτελείται ως τμήμα της παραγωγής ταινιών. Η εφαρμογή αυτή εγκαθίσταται στην εικονική υποδομή ενώ ολόκληρη η λειτουργία ρυθμίζεται μέσω της πλατφόρμας υπηρεσιών [71]. Στο σενάριό μας μια εταιρία παραγωγής έχει ένα συμβόλαιο για να εκτελέσει τη διόρθωση χρώματος σε μερικά πλάνα ταινιών ενώ στον ίδιο χρόνο ο διευθυντής ταινιών και ο παραγωγός θα αναθεωρήσουν αυτό το τμήμα της ταινίας. Η ροή εργασίας της εφαρμογής αποτελείται από ένα τηλεοπτικό τμήμα αποθήκευσης, έναν μεταβλητό αριθμό μονάδων επεξεργασίας εικόνας (Image Processing Units - IPUs) ανάλογο με τον ρυθμό

μετάδοσης των δεδομένων, έναν ισορροπιστή φορτίων (Load Balancer - LB) υπεύθυνο για την παράδοση της ροής εικόνας (*stream*) στους δέκτες και έναν ελεγκτή επεξεργασίας εικόνας (Image Processing Controller – IPC) που κρατά τις παραμέτρους διορθώσεων χρώματος προερχόμενες από τον διορθωτή χρώματος (*colorist*). Οι συμμετέχοντες στο σενάριο είναι ο colorist που χρησιμοποιεί το σταθμό διορθώσεων χρώματος, ένα πρόγραμμα δέκτη/θεατή (*viewer*) για το διευθυντή ταινιών και ένα για τον παραγωγό.



Σχήμα 28: Σενάριο εφαρμογής διορθώσεων χρώματος

Η επιλογή αυτού του σεναρίου εφαρμογής έχει γίνει για τους ακόλουθους λόγους:

- Μέγεθος και πολυπλοκότητα: η ροή εργασίας της εφαρμογής αποτελείται από έναν ιδανικό αριθμό συστατικών, όχι αρκετά μεγάλο προκειμένου να αυξηθεί η δυσκολία της εγκατάστασης και του ελέγχου του μέσα σε ένα εικονικό περιβάλλον, όμως όχι πολύ μικρό έτσι ώστε να διακινδυνέψει της αξιολόγηση του πειράματος.
- Ελαστικότητα: η δομή της ροής εργασίας της εφαρμογής (ύπαρξη των ανεξάρτητων μονάδων επεξεργασίας) ευθυγραμμίζεται ιδιαίτερα με την ελαστικότητα που μια πλατφόρμα Νεφών προσφέρει.

- Ποσοτικές μετρικές: τα τμήματα λογισμικού προσφέρουν τις ποσοτικές μετρικές QoS (*frames per second* κ.λπ.) που έκαναν την ένταξη τους στο σύστημα επίβλεψης σχετικά άμεσα και εύκολα.
- Ρεαλιστική περίπτωση χρήσης: η συγκεκριμένη εφαρμογή είναι μια ρεαλιστική, βιομηχανική, ροή εργασίας παραγωγής που χρησιμοποιείται στα σύγχρονα στούντιο διορθώσεων χρώματος.

4.6.1.1 Εγκατάσταση και συλλογή αποτελεσμάτων

Αφού έχουν εγκατασταθεί και τα έξι τμήματα εφαρμογής της ροής εργασίας IPU1-4, IPC, LB) συνεχίζουμε με την ενεργοποίηση του μηχανισμού από το στρώμα πλατφόρμας μέσω της υπηρεσίας Monitoring Framework Service. Στον Πίνακας 7 παρουσιάζονται οι υψηλού επιπέδου πληροφορίες επίβλεψης που παραδίδονται στο Monitoring Central Index από όλα τα τμήματα εφαρμογής της ροής εργασίας μέσω του μηχανισμού μας. Οι παράμετροι επίβλεψης παρουσιάζονται για ένα ορισμένο σημείο της εκτέλεσης (timestamp: 2010-04-29 19:40: 15) όπως κάθε συλλέκτης δεδομένων παρέχει. Όπως μπορεί να δει κανείς από τα αποτελέσματα, κάθε τμήμα εφαρμογής (IPU1-4, IPC και LB1) με το προσδιοριστικό PP01, εκθέτει ένα σύνολο μετρικών με την παροχή του ονόματος παραμέτρου (*param name*), της τιμής (*param value*), του τύπου (*param type*) και της μονάδας (*param unit*). Μπορούμε επίσης να σημειώσουμε ότι ενώ οι μονάδες επεξεργασίας (IPUs) και ο ελεγκτής (IPC) παρακολουθούν το ίδιο σύνολο μετρικών (*state, ACname, cpu loadavg1* και *mem phys used*), ο Load Balancer (LB1), όντας ο «εγκέφαλος» της εφαρμογής, έχει ρυθμιστεί να επιβλέπει και να αναφέρει πρόσθετες μετρικές (*ImageQueue entries, request latency, b clients* και *p playing*).

timestamp	ComponentID	ApplicationID	param_name	param_value	param_type	param_unit
2010-04-29 19:40:15	IPU3	PP01	state	running	string	enum
2010-04-29 19:40:15	IPU3	PP01	ACname	IPU	string	identifier
2010-04-29 19:40:15	IPU3	PP01	cpu_loadavg1	0.000000	float	CPUs
2010-04-29 19:40:15	IPU3	PP01	mem_phys_used	1312344	int	kbyte
2010-04-29 19:40:15	IPU2	PP01	state	running	string	enum

2010-04-29 19:40:15	IPU2	PP01	ACname	IPU	string	identifier
2010-04-29 19:40:15	IPU2	PP01	cpu_loadavg1	0.700000	float	CPUs
2010-04-29 19:40:15	IPU2	PP01	mem_phys_used	1421224	int	kbyte
2010-04-29 19:40:15	IPU4	PP01	state	running	string	enum
2010-04-29 19:40:15	IPU4	PP01	ACname	IPU	string	identifier
2010-04-29 19:40:15	IPU4	PP01	cpu_loadavg1	0.000000	float	CPUs
2010-04-29 19:40:15	IPU4	PP01	mem_phys_used	1258976	int	kbyte
2010-04-29 19:40:15	IPU1	PP01	state	running	string	enum
2010-04-29 19:40:15	IPU1	PP01	ACname	IPU	string	identifier
2010-04-29 19:40:15	IPU1	PP01	cpu_loadavg1	0.000000	float	CPUs
2010-04-29 19:40:15	IPU1	PP01	mem_phys_used	1398864	int	kbyte
2010-04-29 19:40:15	IPC	PP01	state	running	string	enum
2010-04-29 19:40:15	IPC	PP01	ACname	IPC	string	identifier
2010-04-29 19:40:15	IPC	PP01	cpu_loadavg1	0.240000	float	CPUs
2010-04-29 19:40:15	IPC	PP01	mem_phys_used	339884	int	kbyte
2010-04-29 19:40:15	LB1	PP01	state	running	string	enum
2010-04-29 19:40:15	LB1	PP01	ACname	LBB	string	identifier
2010-04-29 19:40:15	LB1	PP01	cpu_loadavg1	0.560000	float	CPUs
2010-04-29 19:40:15	LB1	PP01	mem_phys_used	434272	int	kbyte
2010-04-29 19:40:15	LB1	PP01	ImageQueue_entries	18	int	count
2010-04-29 19:40:15	LB1	PP01	request_latency	272.498993	float	ms
2010-04-29 19:40:15	LB1	PP01	b_clients	2	int	count
2010-04-29 19:40:15	LB1	PP01	p_playing	1	int	binary flag
2010-04-29 19:40:15	LB1	PP01	p_playedframes	541	int	count
2010-04-29 19:40:15	LB1	PP01	p_droppedframes	40	int	count
2010-04-29 19:40:15	LB1	PP01	p_reliability	0.000000	float	0..1

Πίνακας 7: Αποτελέσματα επίβλεψης εφαρμογής

4.6.1.2 Αναδιαμόρφωση επίβλεψης

Όπως έχει αναφερθεί ήδη, ο μηχανισμός που υλοποιήσαμε είναι διαμορφώσιμος όσον αφορά το χρονικό διάστημα που κάθε συλλέκτης δεδομένων λειτουργεί αλλά και για τις παραμέτρους που συλλέγει. Η διαμόρφωση για κάθε τμήμα εφαρμογής ποικίλλει ανάλογα με τον τύπο του συστατικού της εφαρμογής, τον τύπο της παραμέτρου QoS που ελέγχουμε και τη σημασία ενός συστατικού μέσα σε ολόκληρη τη ροή εργασίας της εφαρμογής. Μέσα σε

αυτό το πλαίσιο, ο υπεύθυνος για την ανάπτυξη της εφαρμογής καθορίζει το αρχικό χρονικό διάστημα επίβλεψης, το εκτελέσιμο κώδικα του συλλέκτη καθώς επίσης και τις πολιτικές μέσα στον ελεγκτικό μηχανισμό επίβλεψης (*Monitoring Controller*) πριν από την εγκατάσταση και την εκτέλεση της εφαρμογής του μέσα στο εικονικό περιβάλλον. Σε αυτό το σενάριο εφαρμογής διόρθωσης χρώματος, το αρχικό χρονικό διάστημα επίβλεψης έχει τεθεί 20 δευτερόλεπτα και η QoS μετρική που έχει αξιολογηθεί από τον ελεγκτικό μηχανισμό ήταν ο αριθμός χαμένων καρέ (*dropped frames*) κατά τη διάρκεια της επεξεργασίας. Ο συλλέκτης δεδομένων έχει παράσχει τον αριθμό χαμένων καρέ από το τελευταίο σημείο αναφοράς του και όχι από την αρχή της εκτέλεσης (μη-αθροισμένη λειτουργία). Ο μηχανισμός επίβλεψης για αυτήν την εφαρμογή υποστηρίζει τη συλλογή των διάφορων μετρικών (Πίνακας 7) και της μεταπήδησης από έναν συλλέκτη δεδομένων σε άλλον αλλά χάριν της απλότητας θα εστιάσουμε στην παράμετρο χαμένων καρέ για την επικύρωση του μηχανισμού.

Επιπρόσθετα, το SLA που υπογράφεται για αυτή την συγκεκριμένη εφαρμογή καθορίζει έναν περιορισμό QoS 25 πλαισίων εικόνας ανά δευτερόλεπτο (fps) επιτυχούς επεξεργασίας για την ροή βίντεο της εξόδου. Υπό αυτές τις συνθήκες, η εφαρμογή διαμορφώνεται για να επεξεργαστεί τα καρέ με ένα ρυθμό λειτουργίας στα 25fps. Σε περίπτωση που υπάρχει μια αποτυχία στην επεξεργασία των καρέ περισσότερο από 10%, με άλλα λόγια εάν η τυπική απόκλιση στον ρυθμό μετάδοσης δεδομένων είναι μεγαλύτερη από 10% της συμφωνηθείσας στο SLA τιμής (περισσότερα από 2.5 χαμένα καρέ ανά δευτερόλεπτο βασισμένα στον περιορισμό 25fps), μια παραβίαση SLA προκαλείται. Επομένως, η πολιτική που έχουμε καθορίσει και έχουμε εφαρμόσει στο σενάριό μας βασίστηκε στην εξής δήλωση: η ανίχνευση της αποτυχίας επεξεργασίας καρέ (*dropped frames*) πάνω από 5% προκαλεί μια μείωση του διαστήματος δειγματοληψίας επίβλεψης. Όταν η εφαρμογή αποδίδει πάλι μέσα στα αποδεκτά όρια (ποσοστό αποτυχίας λιγότερο από 5%) για ένα ορισμένο χρονικό διάστημα, το διάστημα ελέγχου επανέρχεται στην αρχική τιμή.

Σύμφωνα με τα παραπάνω ορίζουμε:

R_{sla} : τον ρυθμό μετάδοσης πλαισίων εικόνας που έχει συμφωνηθεί στη SLA. Στην συγκεκριμένη περίπτωση χρήσης, R_{sla} = 25fps

F_d : ο αριθμός χαμένων, από την επεξεργασία, πλαισίων (dropped frames).

F_T : ο συνολικός αριθμός πλαισίων που επεξεργάστηκαν από το σύστημα

Ρυθμός μετάδοσης πλαισίων εικόνας για το διάστημα t_1 -> t_2:

$$R(t_2) = R_{t_1 -> t_2} = \frac{F_T(t_2) - F_d(t_2)}{t_2 - t_1}$$

Τυπική απόκλιση για το διάστημα δειγματοληψίας t_1 έως t_2 είναι:

$s = \sqrt{s^2}$ με διακύμανση: $s^2 = \int_{t_1}^{t_2} (R(t) - R_{SLA})^2$

Η κατάσταση κανονικής λειτουργίας της εφαρμογής (μη παραβίαση SLA) σε μια χρονική στιγμή t είναι η εξής:

$$\left|R(t) - R_{SLA}\right| \leq 10\% * R_{SLA}$$

Όπως περιγράψαμε στην προηγούμενη παράγραφο, η ενεργοποίηση της αναδιαμόρφωσης της συχνότητας λειτουργίας του μηχανισμού επίβλεψης γίνεται όταν πλησιάζει κίνδυνος παραβίασης SLA, και πιο συγκεκριμένα όταν διαπιστώνεται ρυθμός απώλειας πλαισίων εικόνα πάνω από 5% της συμφωνηθείσας στο SLA τιμής:

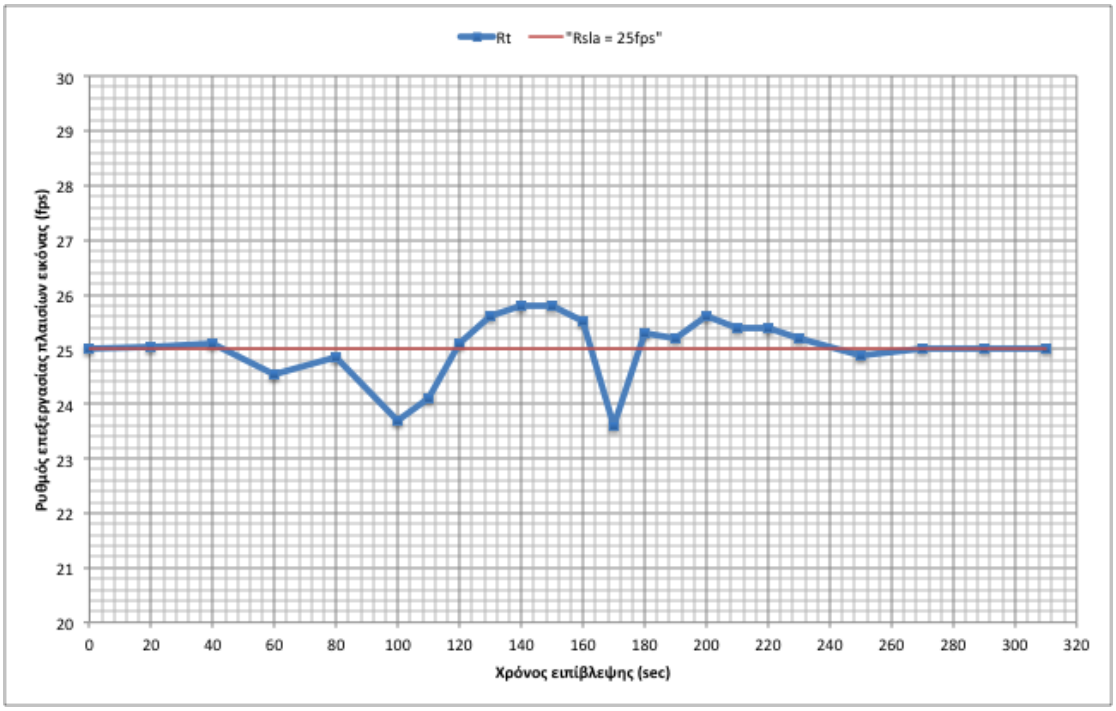
$$\left|R(t) - R_{SLA}\right| \geq 5\% * R_{SLA}$$

Στα ακόλουθα διαγράμματα (Σχήμα 29, Σχήμα 30) παρουσιάζουμε τα πειραματικά αποτελέσματα για το σενάριο εφαρμογής με το αρχικό χρονικό διάστημα 20 δευτερολέπτων και της αναδιαμόρφωση σε 10 και 5 δευτερόλεπτα. Παρατηρούμε και στις δύο περιπτώσεις ότι σε κάποιο χρονικό σημείο (συγκεκριμένα στο 100^o δευτερόλεπτο) ενεργοποιείται η πολιτική της αναδιαμόρφωσης και αλλάζει το διάστημα δειγματοληψίας. Αφού επανέλθει στο σύστημα σε συνθήκες κανονικής λειτουργίας, και δεν παρατηρηθεί κάποια παραβίαση σύμφωνα με την πολιτική μας, επαναφέρουμε τον χρόνο δειγματοληψίας στην αρχική τιμή. Επίσης, στο συγκεκριμένο πείραμα η επαναφορά της τιμής δειγματοληψίας στα προκαθορισμένο επίπεδο των 20 δευτερολέπτων γίνεται μετά το πέρας ενός λεπτού από την ανίχνευση κινδύνου παραβίασης (5% R_{SLA}).

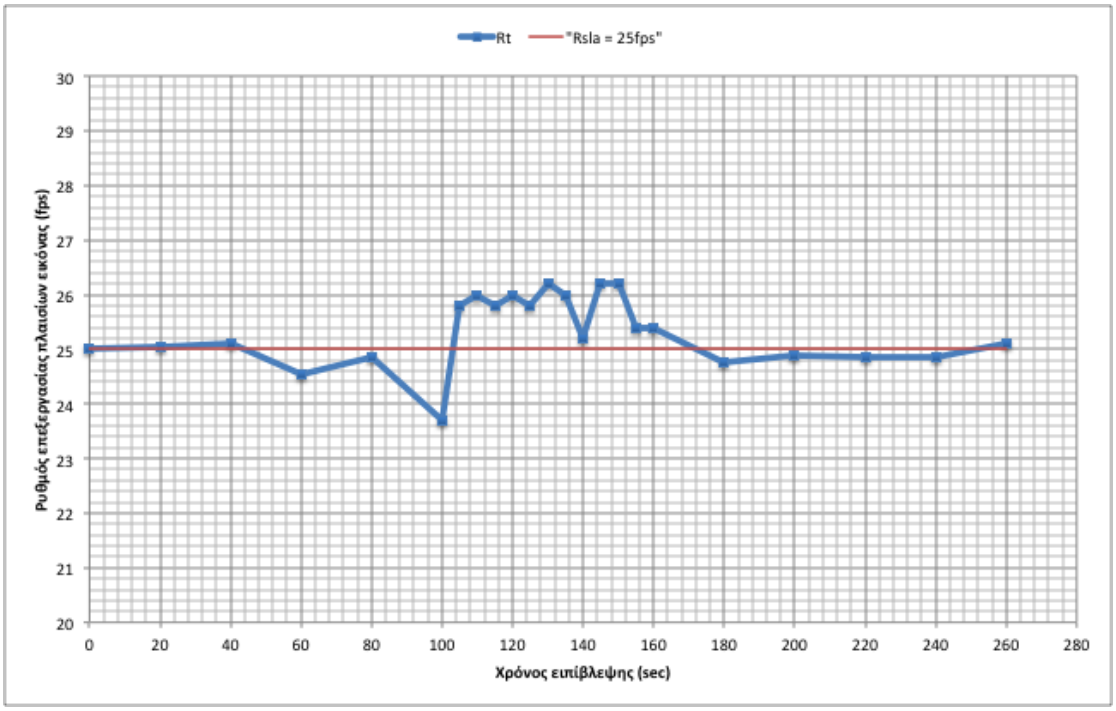
Με την εισαγωγή αυτής της πολιτικής ως τμήμα του μηχανισμού επίβλεψης κατά τη διάρκεια της εκτέλεσης του σεναρίου πετυχαίνουμε αποτελεσματικά να μειώσουμε το βήμα συλλογής δεδομένων παρακολούθησης και μέσω αυτής της δυνατότητας καταλήγουμε σε διάφορα σημαντικά αποτελέσματα:

- Καλύτερη ανάλυση συμπεριφοράς της εφαρμογής από τον υπεύθυνο για την ανάπτυξή της ή/και τον καταναλωτή: με τη μείωση του χρονικού διαστήματος επίβλεψης καταλήγουμε σε συλλογή περισσοτέρων πληροφοριών κατά τη διάρκεια της φάσης μιας ανώμαλης συμπεριφοράς της εφαρμογής.
- Γρηγορότερος χρόνος απόκρισης: έχοντας ένα μικρό βήμα δειγματοληψίας και με χρήση ευφυών πολιτικών, μπορούμε να λάβουμε ορισμένα μέτρα για την παρεμπόδιση μιας πιθανής παραβίασης SLA. Στη συγκεκριμένη περίπτωση χρήσης, εξετάσαμε επίσης μια δεύτερη πολιτική που επιτρέπει μια πρόσθετη μονάδα επεξεργασίας εικόνας (IPU) όταν υπάρχει μια ανίχνευση του ποσοστού χαμένων καρέ περισσότερο από 5%.
- Λιγότερα στοιχεία που μεταφέρονται, λιγότερα στοιχεία που αποθηκεύονται: μέσω του εύκαμπτου μηχανισμού επίβλεψης επιτυγχάνουμε την ελαχιστοποίηση του εύρους ζώνης που καταναλώνεται μεταξύ του εικονικού περιβάλλοντος και της πλατφόρμας καθώς επίσης και του χώρου στο δίσκο που χρησιμοποιείται για την αποθήκευση των ιστορικών πληροφοριών επίβλεψης.

Όπως αναφέρθηκε και προηγουμένως, ο μηχανισμός επίβλεψης που υλοποιήθηκε έχει την δυνατότητα εφαρμογής πολλαπλών πολιτικών και διορθωτικών κινήσεων. Ως συνέχεια του πειράματος σχετικά με την εφαρμογή διορθωτικών πολιτικών πάνω στα δεδομένα επίβλεψης που συλλέχθηκαν, θα παρουσιάσουμε παρακάτω τόσο τη μεταβολή στο διάστημα δειγματοληψίας (*PollIntervalMills* του αρχείου διαμόρφωσης) όσο και το όνομα του συλλέκτη δεδομένων (*ProbeName*) με σκοπό την συλλογή επιπλέον πληροφορίας.



Σχήμα 29: Αναδιαμόρφωση συχνότητας επίβλεψης από 20 sec σε 10 sec



Σχήμα 30: Αναδιαμόρφωση συχνότητας επίβλεψης από 20 sec σε 5 sec

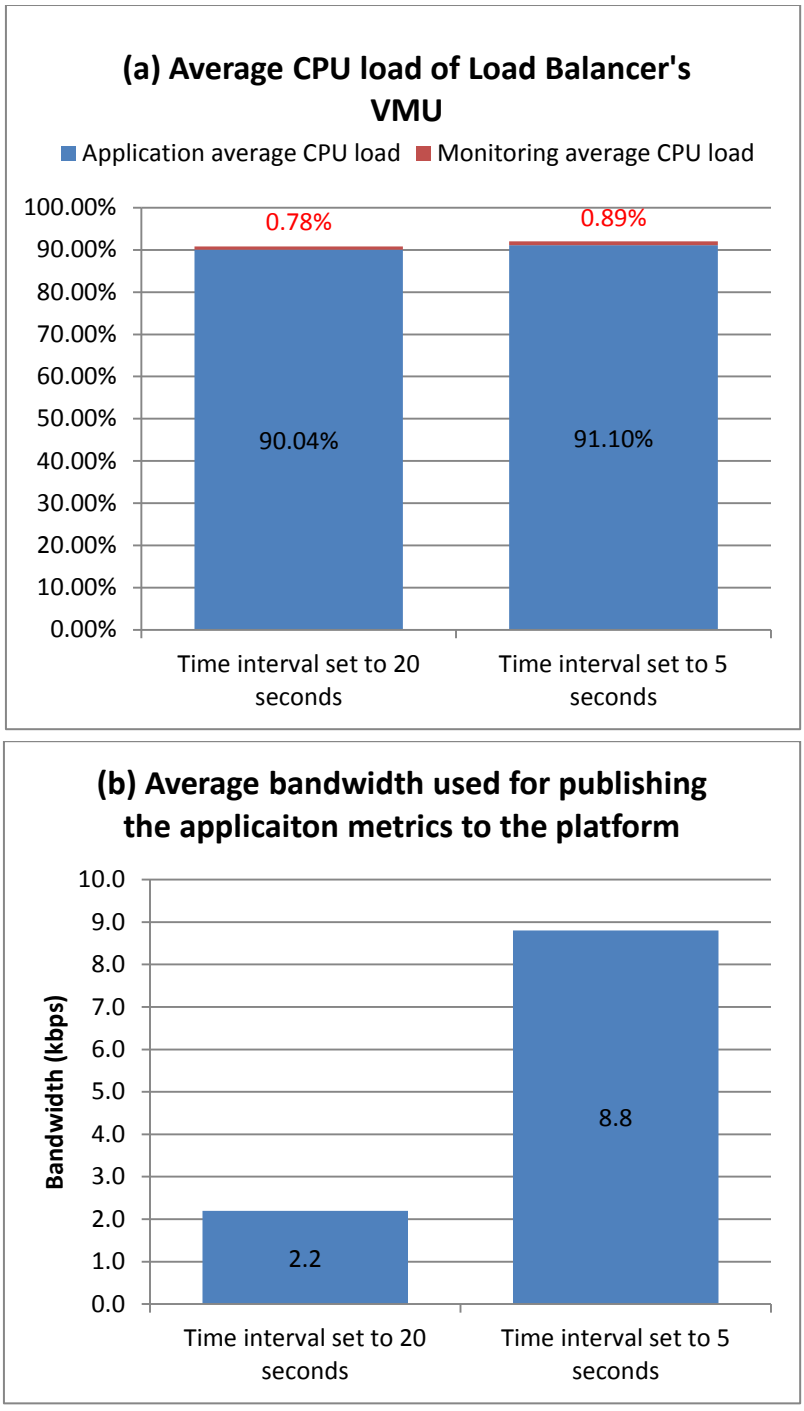
4.6.1.3 Αξιολόγηση απόδοσης

Η διαδικασία αξιολόγησης που εφαρμόσαμε περιλαμβάνει και μετρήσεις απόδοσης του μηχανισμού συλλογής δεδομένων από την εφαρμογή που είναι εγκατεστημένη στην εικονική

υποδομή (Data Collector και Monitoring Index Service). Υπό αυτό το πλαίσιο, καταγράψαμε τη χρησιμοποίηση φορτίων CPU του μηχανισμού επίβλεψης στην VMU όπου το συστατικό Load balancer είναι εγκατεστημένο. Όπως παρουσιάζεται στο Σχήμα 31 - (α), το μέσο φορτίο CPU που το στοιχείο επίβλεψης καταναλώνει είναι ασήμαντο σε σύγκριση με τη εφαρμογή Load balancer, η οποία χρησιμοποιεί περισσότερο από 90% της CPU. Το σημαντικό ποσό χρησιμοποίησης CPU που η LB καταναλώνει αποδεικνύει την υψηλή απαίτηση σε υπολογιστική ισχύ που το συστατικό έχει κατά τη διάρκεια του χρόνου εκτέλεσης. Από την άλλη, η σταθερή συμπεριφορά του και στις δύο περιπτώσεις αποδεικνύει ότι η λειτουργία της εφαρμογής δεν επηρεάζεται καθόλου από τα διαφορετικά ποσοστά δειγματοληψίας. Επιπλέον, μπορούμε με σιγουριά να πούμε ότι η επαναδιαμόρφωση του χρονικού διαστήματος επίβλεψης δεν έχει επιπτώσεις στο φορτίο του VMU σημαντικά ενώ η επίβλεψη της εφαρμογής συμπεριφέρεται ομοίως με το χρονικό διάστημα 20 και 5 δευτερολέπτων. Συγκεκριμένα, υπάρχει μια μικρή αλλά λογική αύξηση του φορτίου CPU που το στοιχείο επίβλεψης καταναλώνει, στην περίπτωση του πιο σύντομου χρονικού διαστήματος, αλλά συνολικά οι τιμές είναι πολύ μικρός (0.78-0.89%), δικαιολογημένος από την απλή όμως αποδοτική εφαρμογή του Data Collector και των συστατικών Monitoring Instance.

Στο Σχήμα 31 - (β), παρουσιάζεται η μέση αξία του εύρους ζώνης της δικτυακής κίνησης μεταξύ της VMU και της πλατφόρμας (Monitoring Index Service και Monitoring Central Index). Το πείραμα εκτελέσθηκε με την ίδια διαμόρφωση (λειτουργικό χρονικό διάστημα του συλλέκτη δεδομένων καθορισμένο σε 20 και 5 sec). Τα αποτελέσματα παρουσιάζουν την αναμενόμενη απόκλιση: η κίνηση στο δίκτυο αυξάνεται όταν ελαχιστοποιούμε το χρονικό διάστημα επίβλεψης αλλά αντίθετα όταν αποδίδει η εφαρμογή μέσα στα αποδεκτά όρια μπορούμε να επιτύχουμε την ελάχιστη χρησιμοποίηση δικτύου. Αυτή η αύξηση δικαιολογείται αφού η συνάθροιση των εκθέσεων επίβλεψης δεν εκτελείται σε εκείνο το επίπεδο και επομένως όταν έχουμε ένα μικρό διάστημα το σύστημα εκπέμπει τέσσερις φορές περισσότερες εκθέσεις. Και στις δύο περιπτώσεις εν τούτοις, οι τιμές που καταγράφονται (2.2 και 8.8 Kbps) για την μετάδοση μιας έκθεσης είναι σχετικά μικρές και

ακόμα κι αν «πιέζαμε» το σύστημα με πολλές συνδέσεις η κατανάλωση εύρους ζώνης θα ήταν ένα μικρό μέρος της χωρητικότητας του δικτύου.



Σχήμα 31: Αξιολόγηση απόδοσης

Πηγαίνοντας ένα βήμα παρακάτω με την αξιολόγηση του μηχανισμού, εκτελέσαμε ένα πείραμα με τον καθορισμό μιας συνδυασμένης πολιτικής. Χρησιμοποιώντας την ίδια συνθήκη ενεργοποίησης (ανίχνευση περισσότερο από 5% ποσοστού χαμένων καρέ) αντί

μόνο να αυξήσουμε το ποσοστό δειγματοληψίας της επίβλεψης ενεργοποιούμε επίσης έναν διαφορετικό συλλέκτη δεδομένων. Αυτός ο αναβαθμισμένος συλλέκτης θα περιλάβει στην έκθεση επίβλεψης τις πρόσθετες πληροφορίες που στη συγκεκριμένη περίπτωση χρήσης μας είναι μια παράμετρος «σημειώσεων» (*log*) με τις λεπτομέρειες της τελευταίας περιόδου παρακολούθησης. Σε αυτό το πλαίσιο, η επαναδιαμόρφωση του μηχανισμού επίβλεψης εφαρμόζεται επάνω σε δύο επίπεδα: συχνότητα ελέγχου και ποσό των πληροφοριών. Στον Πίνακας 8 παρουσιάζεται μια έκθεση από το LB1 συστατικό στο οποίο ο αναβαθμισμένος συλλέκτης δεδομένων εφαρμόστηκε.

Η αξία της παραμέτρου *log* είναι καθαρά εξαρτημένη από την εφαρμογή και προορίζεται να βοηθήσει τον υπεύθυνο για την ανάπτυξη εφαρμογής να αναπτύξει έναν αποδοτικό μηχανισμό πλατφόρμας για την λήψη αποφάσεων, την επίβλεψη και παρακολούθηση σε πραγματικό χρόνο. Επομένως, η ουσιαστική ιδέα είναι όποτε ανιχνεύουμε μια ανώμαλη συμπεριφορά πρέπει να αναδιαμορφώσουμε το σύστημα επίβλεψης προκειμένου να παρασχεθούν σε μας περισσότερα στοιχεία προς αξιολόγηση. Με το προτεινόμενο σύστημα μπορούμε να πετύχουμε σε αυτό είτε από την αλλαγή του ποσοστού δειγματοληψίας, είτε με την αύξηση των συλλεχθεισών πληροφοριών ή ακόμα και με τα δύο. Συγκρίναμε και τις τρεις περιπτώσεις με την αρχική κατάσταση στην βάση του μέσου όρου κατανάλωση εύρους ζώνης και τα αποτελέσματα παρουσιάζονται στο Σχήμα 32. Όπως αναμένεται, λόγω των πρόσθετων πληροφοριών που διαβιβάζονται έχουμε μια αύξηση της μέσης κατανάλωσης εύρους ζώνης κατά την χρησιμοποίηση του αναβαθμισμένου συλλέκτη στοιχείων. Επίσης παρατηρούμε ότι κατά εφαρμογή του νέου συλλέκτη αλλά και του μικρού διαστήματος, η απόκλιση του εύρους ζώνης είναι μικρότερη: από 2.2kbps στην περίπτωση 20sec φθάνουμε σε 4.3kbps (σχεδόν διπλό) ενώ στην περίπτωση 5sec από 8.8kpbs μετρήσαμε 9.8kbps στην περίπτωση του αναβαθμισμένου συλλέκτη δεδομένων. Ο λόγος πίσω από αυτό είναι ότι κατά το σύλληψη των πληροφοριών με το μικρό διάστημα, το μέγεθος των καταγραμμένων δεδομένων μειώνεται έναντι των στοιχείων log για την περίοδο 20sec. Στο σύνολο, αυτό το πείραμα κατέδειξε μια άλλη πτυχή της προσαρμοστικότητας του προτεινόμενου μηχανισμού. Με τον έλεγχο του ποσού πληροφοριών που συγκεντρώνεται και

διαβιβάζεται, κατά τρόπο δυναμικό βασισμένο στα διάφορα κριτήρια QoS, οδηγούμαστε σε ένα πολύ ευέλικτο σύστημα που επιτρέπει στον τελικό χρήστη να εκμεταλλευτεί περαιτέρω τις ικανότητες της εφαρμογής. Επιπρόσθετα, με τη ενεργοποίηση μόνο του αναβαθμισμένου συλλέκτη δεδομένων και όχι του αυξανόμενου ποσοστού δειγματοληψίας μπορούμε να ωφεληθούμε από τη μικρότερη μέση κατανάλωση εύρους ζώνης αλλά ακόμα λαμβάνουμε περισσότερες πληροφορίες σχετικά με τη συμπεριφορά εφαρμογής.

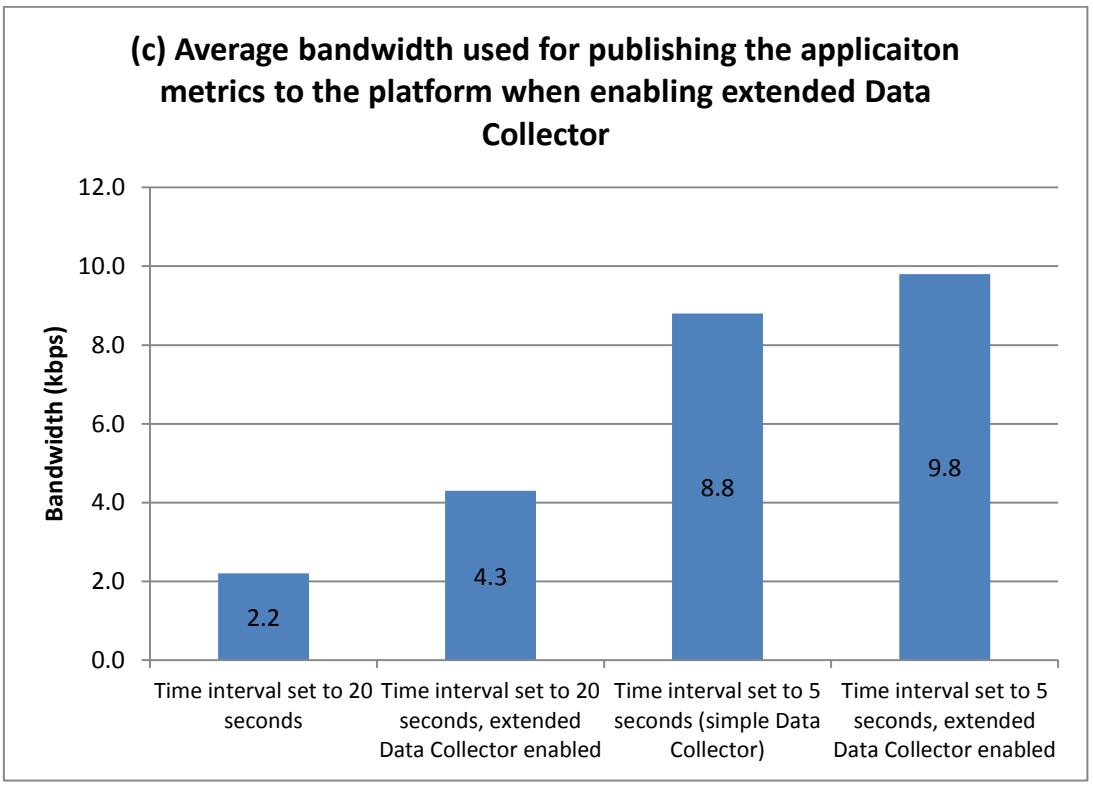
timestamp	ComponentID	ApplicationID	param_name	param_value	param_type	param_unit
2011-09-19 17:42:05	LB1	PP01	state	running	string	enum
2011-09-19 17:42:05	LB1	PP01	ACname	LBB	string	identifier
2011-09-19 17:42:05	LB1	PP01	cpu_loadavg1	0.45	float	CPUs
2011-09-19 17:42:05	LB1	PP01	mem_phys_used	681347	int	kbyte
2011-09-19 17:42:05	LB1	PP01	ImageQueue_entries	14	int	count
2011-09-19 17:42:05	LB1	PP01	request_latency	352.685473	float	ms
2011-09-19 17:42:05	LB1	PP01	b_clients	2	int	count
2011-09-19 17:42:05	LB1	PP01	p_playing	2	int	binary flag
2011-09-19 17:42:05	LB1	PP01	p_playedframes	456	int	count
2011-09-19 17:42:05	LB1	PP01	p_droppedframes	17	int	count
2011-09-19 17:42:05	LB1	PP01	p_reliability	0.000000	float	0..1
2011-09-19 17:42:05	LB1	PP01	log	[LB1] Viewer 001 conneted [LB1] Viewer 002 conneted [LB1] Viewer 001, frames 45-87, R:251, G:140, B:51, A:87, brightness:65%, contrast:53% [LB1] Applying parameters set to: IPU1, IPU2, IPU3, IPU4 [LB1] Viewer 002, frames 125-205,	string	log

				R:178, G:256, B:34, A:98, brightness:61%, contrast:49% [LB1] Applying parameters set to: IPU1, IPU2, IPU3, IPU4 [LB1] Viewer 001, frames 106-247, R:352, G:495, B:350, A:124, brightness:80%, contrast:74%		

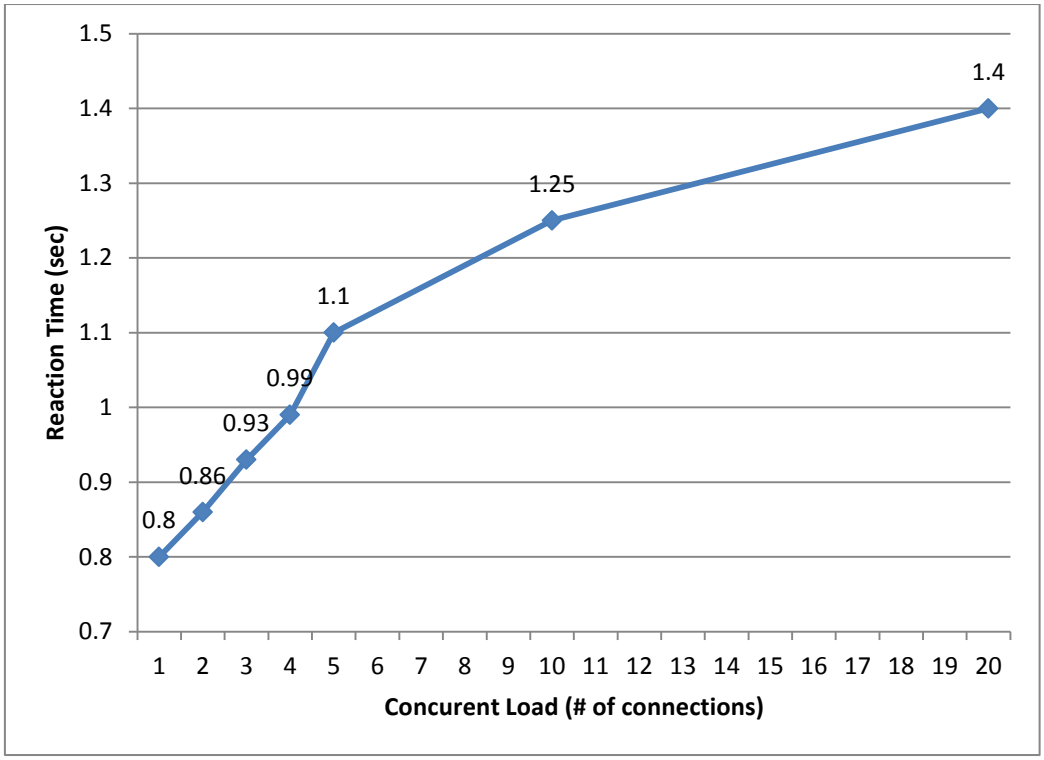
Πίνακας 8: Έκθεση επίβλεψης αναβαθμισμένου συλλέκτη δεδομένων

Οι πληροφορίες επίβλεψης από τα διάφορα συστατικά που εγκαθίστανται πάνω στο εικονικό περιβάλλον φθάνουν ασύγχρονα στο Monitoring Central Index. Προκειμένου να αξιολογηθεί η ελαστικότητα του μηχανισμού, επαναλάβαμε το αρχικό πείραμα του σεναρίου εφαρμογής, ρυθμίζοντας την κοκκοποίησης επίβλεψης των συστατικών (χρονικό διάστημα ελέγχου) σε 1 το δεύτερο (για τον LB) και τον χρόνο ανανέωσης δεδομένων κατά την εγγραφή των Monitoring Indexes (μέσα στις VMUs) και τον Monitoring Central Index καθορισμένο 1 sec επίσης. Με αυτήν την συγκεκριμένη εγκατάσταση, ο χρόνος διάδοσης (*propagation time*) του μηχανισμού επίβλεψης από τα τμήματα εφαρμογής μέσα στην VMU μέχρι το Central Index στην πλατφόρμα ήταν κοντά σε 1 sec (0.7-1.1 sec). Αυτή η μέτρηση πραγματοποιήθηκε με την αποσύνδεση των θεατών (*viewer*) από τον LB κατά τη διάρκεια της εκτέλεσης και τον υπολογισμό του χρόνου που η πρώτη έκθεση με το *b_clients=0* χρειάζεται για να φθάσει στο Monitoring Central Index. Με την επέκταση και την επανάληψη αυτού του πειράματος με περισσότερες από μια συνδέσεις (αριθμός Monitoring Indexes που μεταφέρουν δεδομένα στο ίδιο Central Index) μετρήσαμε το χρόνο αντίδρασης του συστήματος με τα αποτελέσματα που παρουσιάζονται στο Σχήμα 33. Ως «χρόνο αντίδρασης» (*Reaction time*) στον άξονα Y του διαγράμματος θεωρούμε το συνολικό χρόνο που απαιτείται για μια έκθεση να δημιουργηθούμε από το συλλέκτη δεδομένων, να δημοσιευθεί στην υπηρεσία Monitoring Index Service και να αποσταλεί τελικά στο Monitoring Central Index. Είναι επομένως μία

μονή διαδρομή, *end-to-end* χρονική μέτρηση από την παραγωγή των πληροφοριών έως ότου διατίθεται για την κατανάλωση. Παρατηρούμε ότι δεδομένου ότι ο αριθμός συνδέσεων αυξάνεται, ο χρόνος διάδοσης του μηχανισμού αυξάνει επίσης, παραμένοντας όμως κοντά σε 1 sec και αρχίζει να σταθεροποιείται ακόμα και όταν αυξάνεται το ταυτόχρονο φορτίο. Τα αποτελέσματα καταγράφθηκαν μέσω αυτού του πειράματος είναι επίσης σύμφωνα με τις τιμές αξιολόγησης που παρουσιάζονται στο [57] (σχήμα 7: *MDS4 Index service response time performance*) όπου ο χρόνος απόκρισης μιας υπηρεσίας MDS Index καταγράφεται όταν το σύστημα επιβαρύνεται με πολλαπλά φορτία. Η σημασία αυτής της συνεπής συμπεριφοράς βρίσκεται σε δύο γεγονότα: η επέκταση και η εισαγωγή των πρόσθετων υπηρεσιών πάνω από ένα υπάρχον API δεν έχουν επιπτώσεις στην απόδοση της υπηρεσίας MDS Index service, η ισχύς των αποτελεσμάτων μας ενισχύεται ενώ ευθυγραμμίζονται με αυτά που αναφέρονται από μια ανεξάρτητη ερευνητική ομάδα όταν εκτελώντας τα παρόμοια πειράματα.



Σχήμα 32: Αξιολόγηση αναβαθμισμένου συλλέκτη δεδομένων βάση χρησιμοποιούμενου εύρους ζώνης



Σχήμα 33: Χρόνος αντίδρασης υπηρεσίας δεδομένων (Index Service) σε διαφορετικές συνθήκες φόρτου συνδέσεων

Component	Reports expected	Reports received	Reports missed	Percentage
LB (1sec)	1800	1791	9	0.50%
IPU1 (2 secs)	900	900	0	0.00%
IPU2 (2 secs)	900	899	1	0.11%
IPU3 (2 secs)	900	900	0	0.00%
IPU4 (2 secs)	900	899	1	0.11%
IPC (5 secs)	360	360	0	0.00%

Πίνακας 9: Ποσοστά αποτυχίας παράδοσης αναφορών επίβλεψης

Συνεχίσαμε τη δοκιμή απόδοσης θέτοντας το χρονικό διάστημα των τμημάτων εφαρμογής ως εξής: 1 δεύτερος για τον LB, 2 δευτερόλεπτα για τα IPUs και 5 δευτερόλεπτα για τον IPC. Υπολογίζοντας τον αριθμό των εκθέσεων επίβλεψης που παραδόθηκαν στον Monitoring Central Index στην πλατφόρμα, για μια περίοδο 30 λεπτών της εκτέλεσης οδηγηθήκαμε στην απώλεια εννέα εκθέσεων ελέγχου για τον LB, μιας για την IPU2 και μιας για την IPU4 (Πίνακας 9). Αυτά τα αποτελέσματα δείχνουν ότι όταν «πιέζεται» ο μηχανισμός επίβλεψης με μια κοκκοποίηση $1^{ος}$ δευτερολέπτου και με πολλαπλά ενεργά συστατικά εφαρμογής, τότε έχουμε αποτυχία στην παράδοση εκθέσεων με ένα λάθος 0.5% των αναμενόμενων. Αφ' ενός, δεδομένου ότι η κοκκοποίηση της επίβλεψης τμημάτων εφαρμογής αυξάνεται, και ακόμη και

με τις τιμές 2 ή 5 δευτερολέπτων, το ποσοστό των αποτυχημένων παραδόσεων είναι 0%. Όσον αφορά την επίβλεψη της υποδομής, η αντίστοιχη υπηρεσία εξέθεσε τις χαμηλού επιπέδου παραμέτρους για τη χρήση των πόρων και δικτύων. Στον Πίνακας 10 παρουσιάζουμε μερικές επιλεγμένες μετρήσεις για τα συστατικά της εφαρμογής LB και IPU1.

timestamp	ComponentID	VMU_ID	param_name	param_value	param_type	param_unit
2010-04-29 19:40:16	IPU1ToLB1		Avg_used_Bandwidth	10	int	kbps
2010-04-29 19:40:16	IPU1ToLB1		Avg_Delay_msec	0	int	msec
2010-04-29 19:40:16	IPU1ToLB1		Avg_Jitter_msec	30	int	msec
2010-04-29 19:40:16	LB1	VMU01	CPU_load	10	int	%
2010-04-29 19:40:16	LB1	VMU01	Phys_RAM	20	int	%
2010-04-29 19:40:16	LB1	VMU01	Used_volatile_storage	0	int	%
2010-04-29 19:40:16	IPU1	VMU02	CPU_load	30	int	%
2010-04-29 19:40:16	IPU1	VMU02	Phys_RAM	80	int	%
2010-04-29 19:40:16	IPU1	VMU02	Used_volatile_storage	0	int	%

Πίνακας 10: Παράδειγμα αναφοράς επίβλεψης της υποδομής

Όπως παρουσιάσαμε σε προηγούμενο υποκεφάλαιο, οι εκθέσεις και από τις δύο πηγές (τμήματα υποδομής και εφαρμογής) πρέπει να περιλάβουν μερικά προσεκτικά επιλεγμένα προσδιοριστικά προκειμένου να επιτευχθεί μια αποδοτική συνάθροιση και να μπορεί κάποιος να επαναχρησιμοποιήσει τις πληροφορίες επίβλεψης κατόπιν. Στο σενάριο εφαρμογής μας, τα τμήματα της εφαρμογής υποβάλλουν μια έκθεση με το χαρακτηριστικό *Component ID* και *Application ID* ενώ η υπηρεσία υποδομής Infrastructure Monitoring Service υποβάλλει έκθεση με το χαρακτηριστικό *PhysicalNode ID* και *VMU ID*. Οι σχέσεις μεταξύ όλων αυτών των χαρακτηριστικών (IDs) και των αντίστοιχων οντοτήτων είναι οι ακόλουθες:

- κάθε εφαρμογή έχει ένα ή πολλά συστατικά (*Application Components*)
- κάθε φυσικός πόρος φιλοξενεί ένα ή πολλά VMUs
- κάθε VMU φιλοξενεί ένα ή πολλά συστατικά μιας εφαρμογής

Η συσχέτιση αυτών των προσδιοριστικών μιας εφαρμογής αποθηκεύεται μέσα σε έναν κατάλογο (*registry*) της πλατφόρμας αποκαλούμενο Deployment Manager και η συνάθροιση

εκτελείται όπως παρουσιάζεται στην παράγραφο 4.5.5. Η διαπραγμάτευση SLA, και η διαχείριση γενικά, στην πλατφόρμα μας πραγματοποιούνται ανά εφαρμογή. Επομένως, η παρακολούθηση της ιεραρχίας των παρεληφθεισών οντοτήτων (εφαρμογή, συστατικά, εικονικές μηχανές και φυσικοί κόμβοι) καθώς επίσης και των πληροφοριών που παράγουν επιτρέπει σε μας να εκμεταλλευτούμε τα συλλεχθέντα δεδομένα με οποιοδήποτε πιθανό τρόπο.

4.6.2 Αρχικά συμπεράσματα

Ο πρωταρχικός στόχος του μηχανισμού αυτού είναι να συλλεχθούν και να αθροιστούν οι πληροφορίες επίβλεψης όσον αφορά τους συγκεκριμένους περιορισμούς απόδοσης όπως τίθεται από τις μελλοντικές εφαρμογές διαδικτύου και επίσης να παρασχεθεί δυνατότητα αναδιαμόρφωσης που θα επιτρέψει στο σύστημα να προσαρμοστεί στις καινούριες συνθήκες λειτουργίας κατά το χρόνο εκτέλεσης. Επιπλέον, επικυρώσαμε και αξιολογήσαμε το πλαίσιο ενάντια σε ένα σενάριο περίπτωσης χρήσης χρησιμοποιώντας μια εφαρμογή διορθώσεων χρώματος (στην διαδικασία παραγωγής ταινιών). Εκτός από την παρουσίαση των αρχιτεκτονικών χαρακτηριστικών και των λειτουργιών που σχεδιάστηκαν, παρουσιάστηκε και η αξιολόγηση της απόδοσης του μηχανισμού σχετικά με την κατανάλωση CPU καθώς επίσης και της απόκρισης δικτύων χρόνος (διάδοση των πληροφοριών στα στρώματα). Η λειτουργία του προτεινόμενου μηχανισμού όσον αφορά τους περιορισμούς απόδοσης επιτυγχάνει τα όρια που τίθενται από τις εφαρμογές *soft-realtime*, δεδομένου ότι η συλλογή και η συνάθροιση δεδομένων επίβλεψης εκτελούνται μέσα στα μικρά χρονικά όρια. Τα πειράματα παρουσίασαν ελπιδοφόρα αποτελέσματα και επομένως η απόδοση του μηχανισμού θεωρείται ασφαλώς καθορισμένη επιτρέποντας έτσι την υιοθέτηση του από οποιοδήποτε ετερογενές σύστημα και ειδικά κάποιο σύστημα Νέφους που επιδιώκει να παρέχει τις εγγυήσεις QoS και να διευκολύνει την αλληλεπίδραση των στοιχείων του.

5 *Σύστημα επίβλεψης και διαχείρισης Νέφους με έμφαση στην ενεργειακή απόδοση*

Όπως αναφέραμε και σε προηγούμενα κεφάλαια, το Υπολογιστικό Νέφος έχει λάβει πρόσφατα την ιδιαίτερη προσοχή ως μια ελπιδοφόρα προσέγγιση για τις μελλοντικές τεχνολογίες πληροφοριών και επικοινωνιών (*Information and Communication Technologies - ICT*). Τα κέντρα δεδομένων (*Data Centers*) που υποστηρίζουν Υπολογιστικά Νέφη καταναλώνουν ένα τεράστιο ποσό ενέργειας [72], που αντιπροσωπεύει ένα οικονομικό βάρος των λειτουργικών εξόδων αυτών των οργανισμών, ένα βάρος στην υποδομή όσον αφορά την διαχείριση της ενέργειας, και ένα περιβαλλοντικό αντίκτυπο στην κοινωνία. Τα Υπολογιστικά Νέφη είναι ένα αναδυόμενο επιχειρηματικό παράδειγμα και καθώς η δημοτικότητά τους αυξάνεται, αυξάνεται και η σημασία του ενεργειακού αντίκτυπού του. Αυτή η αυξανόμενη χρήση του Νέφους, μαζί με τις αυξανόμενες ενεργειακές δαπάνες και την ανάγκη να μειωθούν οι εκπομπές άνθρακα απαιτούν τεχνολογίες ενεργειακής απόδοσης για να στηρίξουν και να διασφαλίσουν την αποδοτική λειτουργία των κέντρων δεδομένων των Νεφών.

Παρότι οι μεγάλες επιχειρήσεις Διαδικτύου (π.χ., Google, Microsoft) έχουν βελτιώσει σημαντικά την ενεργειακή αποδοτικότητα των κέντρων δεδομένων τους, έχουν εστιάσει μέχρι τώρα κυρίως στην βελτιστοποίηση της υλικοτεχνικής υποδομής. Υπάρχουν όμως ακόμα μεγάλες δυνατότητες μείωσης της ενεργειακής κατανάλωσης που μπορούν να πραγματοποιηθούν όσον αφορά τη λειτουργία συστημάτων. Εκτός από τα μεγάλα κέντρα δεδομένων, η αποδοτική λειτουργία θα ήταν εξαιρετικά χρήσιμη για τα μικρού και μεσαίου μεγέθους κέντρα, τα οποία διαμορφώνουν την πλειοψηφία της ενέργειας που καταναλώνεται [72] και δεν μπορεί γενικά να αντέξουν οικονομικά τις (ακριβές) βελτιώσεις του υλικού για να βελτιώσουν την ενεργειακή απόδοση.

Στο κεφάλαιο αυτό παρουσιάζεται ένα υπηρεσιοστρεφές πλαίσιο συλλογής και διαχείρισης πληροφοριών σε περιβάλλον Νέφους. Ιδιαίτερη έμφαση δίνεται στην συλλογή διαφορετικού τύπου πληροφοριών από πολλές πηγές και στην αξιολόγησή τους με σκοπό την ενεργειακή και οικολογική διαχείριση και βελτιστοποίηση.

Το σύστημα αυτό προσφέρει μεθόδους για την μέτρηση, ανάλυση και αξιολόγηση της ενεργειακής χρήση κατά τη διάρκεια της εγκατάστασης και λειτουργίας των υπηρεσιών. Εμπεριέχει μια υποδομή επίβλεψης για να παρέχει σε πραγματικό χρόνο μετρήσεις και προβλέψεις για πληροφορίες κατάστασης για τις υπηρεσίες, τους φυσικούς πόρους, τους εικονικούς πόρους, την κατανάλωση ενέργειας, την παραγωγή άνθρακα, κ.λπ. Χρησιμοποιώντας αυτές τις μετρήσεις κατανάλωσης ενέργειας, μαζί με τους μηχανισμούς για την πρόβλεψη του ενεργειακού αντίκτυπου μπορεί να αξιολογηθεί η οικολογική αποδοτικότητα ενός προμηθευτή Νεφών. Με αυτή την αξιολόγηση μπορεί να επιτευχθεί η εφαρμογή αυτό-διαχειριστικών πολιτικών προκειμένου να ικανοποιηθούν οι απαιτήσεις ενεργειακής αποδοτικότητας του προμηθευτή. Συγκεκριμένα, προτείνεται η χρήση ενεργειακών αξιολογήσεων για να βελτιστοποιηθεί η τοποθέτηση των εικονικών μηχανών (VM) σε έναν προμηθευτή Νεφών προκειμένου να βελτιστοποιηθεί η ενεργειακή αποδοτικότητά του.

5.1 Σχετικές εργασίες

Εκτός από το APIs και τα συστήματα επίβλεψης που είναι διαθέσιμα (και παρουσιάστηκαν σε προηγούμενα κεφάλαια) και κάποιος θα μπορούσε να χρησιμοποιήσει για να «χτίσει» μια λύση ελέγχου, υπάρχουν διάφορες ερευνητικές πρωτοβουλίες που αξίζουν αναφοράς. Στο [73], βρίσκουμε ένα σύστημα επίβλεψης για Νέφη υπηρεσιών που επιτυγχάνει στην ελαστικότητα και επεκτασιμότητα της επίβλεψης ενώ το σύστημα κατανέμεται στα διαφορετικά στρώματα (υπηρεσία, εικονικό περιβάλλον, φυσικοί πόροι, κ.λπ.). Το πλαίσιο προσφέρει τις βιβλιοθήκες μέσω των οποίων κάποιος θα μπορούσε να υλοποιήσει το σύστημα επίβλεψης και παρακολούθησης. Στο [74] παρουσιάζεται ένα πλήρες σύστημα διαχείριση πληροφοριών για υπολογιστικά Νέφη, συμπεριλαμβανομένου, επίβλεψης, διαχείριση λογαριασμών και τιμολόγησης. Η ανάλυση των απαιτήσεων που εκτελείται είναι ενδιαφέρουσα αλλά υστερεί σε λεπτομέρειες της υλοποίησης και σε μετρήσεις απόδοσης και αξιολόγησης. Επιπλέον, η ανάλυση στο [75] αποδεικνύει ότι σε SOA περιβάλλοντα το επιπρόσθετο υπολογιστικό κόστος που εισάγεται από το ίδιο το σύστημα επίβλεψης μπορεί να είναι σημαντικό και να έχει επιπτώσεις στη λειτουργία της υπηρεσίας ή εφαρμογής.

Ταυτόχρονα με το ξέσπασμα των κατανεμημένων συστημάτων και Υπολογιστικών Νεφών ήρθε το κύμα της «πράσινης» τεχνολογίας (π.χ. Green Computing). Ο κλάδος αυτός είναι ένας τομέας τεχνολογίας όπου οι μηχανικοί και οι υπεύθυνοι για την ανάπτυξη προσπαθούν να βελτιστοποιήσουν την ενεργειακή αποδοτικότητα των διάφορων διαδικασιών υπολογισμού είτε με το να επεμβαίνουν στο υλικό είτε την αρχιτεκτονική λογισμικού. Για αυτόν τον λόγο, τα τελευταία χρόνια έχουν υπάρξει διάφορες προσπάθειες που εισήγαγαν την έννοια της ενεργειακής αποδοτικότητας στον τομέα των Υπολογιστικών Νεφών και πρότειναν σχετικές λύσεις και έννοιες. Στο [76] οι συγγραφείς εκτέλεσαν μια εισαγωγικά, αλλά ενδιαφέρουσα έρευνα σχετικά με τον αντίκτυπο των στρατηγικών εξοικονόμησης ενέργειας στη διαχείριση των ενσωματωμένων συστημάτων και των Νεφών συγκεκριμένα, και ολοκληρώνουν με ορισμένες προτάσεις, όπως η μείωση του ενεργειακού κόστους του λογισμικού και του υλικού, βελτιώνοντας την εξισορρόπηση φορτίων, μειώνοντας την

κατανάλωση ενέργειας λόγω της επικοινωνίας και λαμβάνοντας υπόψη τις εκπομπές του διοξειδίου του άνθρακα (CO^2) των κέντρων δεδομένων. Στο ίδιο πλαίσιο, στο [77] σε μια εκτενή έρευνα πραγματοποιείται που τονίζει τη σημασία της κατανάλωσης ισχύος στα υπολογιστικά συστήματα, ταξινομεί τις τεχνικές διαχείρισης ενέργειας στα διαφορετικά επίπεδα, και ολοκληρώνει τελικά με τις απαιτήσεις υπολογιστικών Νεφών στο σχετικό θέμα. Οι ερευνητές στο [78] παρουσιάζουν μια τεχνική διαχείρισης πόρων που εφαρμόζεται σε κέντρα δεδομένων Νεφών. Η λύση που προτείνεται εστιάζει κυρίως στην τοποθέτηση VM πάνω στην υποδομή προσπαθώντας να βελτιστοποιήσει την κατανάλωση ενέργειας με την διασφάλιση του επιπέδου QoS όσο το δυνατόν πιο υψηλού. Επιπλέον, στο [79] παρουσιάζονται δύο αλγόριθμοι διαχείρισης εργασιών που χρησιμοποιούνται για να βελτιστοποιήσουν την τοποθέτηση μιας διαδικασίας σε υποδομή Νεφών. Τα συμπεράσματα του πειράματος έδειξαν ότι υπάρχει μεγάλη δυνατότητα στη μείωση της κατανάλωσης ενέργειας με τη βελτιστοποίηση της διαχείρισης των πόρων σε υποδομές Νεφών. Στο [80] αναλύεται ένας πολύ ενδιαφέρον μηχανισμός μέτρησης ενέργειας εικονικών μηχανών. Ο ερευνητής χρησιμοποιεί ένα ενεργειακό μοντέλο που βασίζει τη χρησιμοποίηση των πόρων κάθε VM για να υπολογίσει μια εκτίμηση της κατανάλωσης ισχύος. Στο [81] αξιολογούνται διαφορετικές διαμορφώσεις υλικού που έχουν αντίκτυπο στην κατανάλωση ενέργειας και την απόδοση κάποιων χαρακτηριστικών εφαρμογών. Το συμπέρασμα που συνάγεται είναι ότι οι διαφορετικοί φόρτοι εργασίας χρειάζονται διαφορετικές διαμορφώσεις υλικού για να επιτύχουν την αποδοτικότητα λειτουργίας και ενεργειακής κατανάλωσης. Αυτό που είναι ενδιαφέρον σε αυτή τη δημοσίευση είναι ότι προκειμένου να ληφθούν οι αξιολογήσεις της ενεργειακής αποδοτικότητας που μπορούν να συγκριθούν με τους φόρτους εργασίας, ορίζεται μια μετρική **iops/J** (input/output operations per Joule). Οι παράμετροι απόδοσης του συστήματος αρχείων του τύπου iops/J δεν μπορούν να χρησιμοποιηθούν για να συγκρίνουν διαφορετικό φορτίο εργασίας, καθώς εξαρτάται από τα κάθε πεδίο της εφαρμογής.

5.2 Μεθοδολογία

Ο κύριος στόχος της δουλειάς που παρουσιάζεται σε αυτό το κομμάτι τις διατριβής είναι ο σχεδιασμός, υλοποίηση και αξιολόγηση ενός διπλού μηχανισμού: (1) μιας υποδομής επίβλεψης και παρακολούθησης για περιβάλλοντα Νεφών η οποία συμπεριλαμβάνει την συλλογή πληροφοριών από διαφορετικές πηγές και αρχιτεκτονικά στρώματα, (2) έναν μηχανισμό αξιολόγησης και διαχείρισης πληροφορίας, που θα αλληλεπιδρά με τις υπηρεσίες επίβλεψης και θα εφαρμόζει πολιτικές διαχείρισης με σκοπό την βελτιστοποίηση της τοποθέτησης VM στο Νέφος λαμβάνοντας υπόψη την ενεργειακή απόδοση.

Όσον αφορά την επίβλεψη και συλλογή των δεδομένων, ακολουθείται παρόμοια μεθοδολογία με τον μηχανισμό που περιγράφθηκε στο προηγούμενο κεφάλαιο: μια κεντρική υπηρεσία επίβλεψης (*Monitoring Service*) που είναι εγκατεστημένη στο στρώμα πλατφόρμας (PaaS) συλλέγει τα δεδομένα από διαφορετικές πηγές και επίπεδα (SaaS, IaaS) τα συναθροίζει και τα αποθηκεύει σε μια βάση δεδομένων. Συγκεκριμένες λεπτομέρειες σχετικά με την τεχνική συλλογής και συνάθροισης των δεδομένων παρουσιάζονται στις παρακάτω ενότητες. Ιδιαίτερη έμφαση δίνεται στην μεθοδολογία συλλογής πληροφοριών από την υποδομή σχετικά με την ενεργειακή κατανάλωση.

Ερευνώντας την διαθέσιμη βιβλιογραφία στον τομέα της κατανάλωσης ενέργειας των συστημάτων υπολογιστών καταλήξαμε σε δύο σημαντικές μεθοδολογίες: (1) κάποιος μπορεί να υπολογίσει την κατανάλωση ενέργειας ενός κόμβου (εικονικού ή φυσικού) με την εφαρμογή ενεργειακών μοντέλων βάση της χρησιμοποίησης του πόρου και να καταλήξει σε μια εκτίμηση, (2) εξωτερικοί αισθητήρες και ειδικές συσκευές που να παρέχουν άμεσα την κατανάλωση ισχύος. Στη δεύτερη περίπτωση, όταν μια εικονική μηχανή φιλοξενείται σε ένα φυσικό πόρο μπορούμε να υπολογίσουμε την κατανάλωση ανά VM βασιζόμενοι στη χρησιμοποίηση των πόρων κάθε VM. Ο σχεδιασμός και η υλοποίηση του συστήματος μας βασίστηκε στην δεύτερη λύση για του παρακάτω λόγους:

• *Λύση ανεξαρτήτου συστήματος*: με τη χρησιμοποίηση μιας εξωτερικής συσκευής παρά ένα πρόγραμμα λογισμικού καταλήγουμε σε έναν ανεξάρτητο του τύπου υλικού

μηχανισμό, του λειτουργικού συστήματος, ή της έκδοσης αυτού. Κατά συνέπεια, κάποιος μπορεί να χρησιμοποιήσει τη λύση μας όχι μόνο σε υποδομές Νεφών αλλά και για τη συλλογή και διαχείριση της ενεργειακής αποδοτικότητας συστημάτων αποθήκευσης ή δικτύων.

• *Μη παρεμβατικός μηχανισμός (non-intrusive)*: αποφεύγοντας την εγκατάσταση συστατικών λογισμικού μέσα στο σύστημα που ελέγχουμε ελαχιστοποιούμε (ακόμη και μηδενίζουμε) την παρεμβατικότητα και τον αντίκτυπο που ο μηχανισμός επίβλεψης έχει στην υποδομή.

• *Ακριβείς μετρήσεις*: Ανεξάρτητα από το πόσο ακριβής είναι η εκτίμηση που ένα ενεργειακό μοντέλο υπολογίζει (βασισμένο στη χρησιμοποίηση CPU, τις διαδικασίες ανάγνωσης/εγγραφής στο δίσκο, την κατανάλωση μνήμης, κ.λπ.) δεν είναι παρά μια εκτίμηση. Με τη χρησιμοποίηση ενός εξωτερικού αισθητήρα για τη μέτρηση της κατανάλωσης ενέργειας μπορούμε να λάβουμε ακριβέστερες τιμές που λαμβάνουν υπόψη επίσης τις καταναλώσεις που παράγονται από τον ανεμιστήρα ή άλλα υποστηρικτικά συστήματα που ένα ενεργειακό μοντέλο θα μπορούσε να μην υπολογίσει.

• *Ευκολία εγκατάστασης*: τέλος, η λύση μας είναι εύκολο να εγκατασταθεί και να λειτουργήσει αφού, όπως αναφέραμε αρχικά, δεν παρεμβαίνει άμεσα το ελεγχόμενο σύστημα.

Επιπλέον, για τις διαχειριστικές ικανότητες του συστήματός μας, ο σχεδιασμός (καθώς επίσης και η υλοποίηση) είναι μέρος μιας ευρύτερης μεθοδολογίας στην οποία έχουμε εργαστεί, με το αρκτικόλεξο *TREC* (Trust/Risk/Eco/Cost). Αυτό το σύστημα στοχεύει κυρίως στο να βοηθήσει την ποσοτική αξιολόγηση και κατάταξη των συμμετεχόντων σε οικοσυστήματα Νεφών. Παραδοσιακά, οι σχέσεις μεταξύ αυτών επικεντρώνονται στους οικονομικούς παράγοντες. Εντούτοις, μια ευρύτερη οπτική που ενσωματώνει τους ποιοτικούς παράγοντες, όπως η εμπιστοσύνη μεταξύ των προμηθευτών, είναι τώρα μια ανάγκη λόγω της εμφάνισης των ανοικτών και δυναμικών περιβαλλόντων Νεφών. Στο τέλος, οι αξιολογήσεις TREC μπορούν να οδηγήσουν σε μια αλλαγή στη διαχείριση διάφορων οντοτήτων στο

Νέφος. Η ενότητα αυτής της διατριβής επικεντρώνεται στην διαχείριση βασισμένη στην ενεργειακή αποδοτικότητα, από τον έλεγχο της κατανάλωσης ενέργειας έως την λήψη αποφάσεων βασισμένοι στις συλληφθείσες τιμές.

Όσον αφορά την αξιολόγηση της ενεργειακής απόδοσης και τον υπολογισμό αυτής, ακολουθούμε μια παρόμοια μεθοδολογία με αυτήν που παρουσιάζεται στο [81]. Δεδομένου ότι η χρησιμοποίηση της CPU που υφίσταται από ένα VM εξαρτάται από την εφαρμογή που εκτελείται στο υλικό της υποδομής, αυτό δεν μπορεί να χρησιμοποιηθεί ως καθολική παράμετρος σύγκρισης απόδοσης στις διαφορετικές πλατφόρμες. Προκειμένου να είμαστε σε θέση να συγκρίνουμε την απόδοση του υπολογιστικού συστήματος του κάθε κόμβου που έχει διαφορετική αρχιτεκτονική και κατανάλωση ενέργειας, θα χρησιμοποιήσουμε μια μετρική του τύπου **CU/W** (Computing Unit per Watt). Μία υπολογιστική μονάδα (CU) ορίζεται ως το ποσό **MWIPS** (*Millions of Whetstone Operations per Second*) που εκτελείται όσον αφορά εκείνους που εκτελούνται από ένα SPARCstation 20-61 με 128 MB RAM, μια σειρά αποθήκευσης SPARC, και Solaris 2.3. Ενώ στο [81] η αξιολογημένη απόδοση βασιζόταν στις διαδικασίες ανάγνωσης/γραφής ανά ενεργειακή μονάδα, η προσέγγισή μας αξιολογεί την δυνατότητα εκτέλεσης υπολογισμών που παραδίδεται από μια μονάδα της πραγματικής ενέργειας που καταναλώνεται από τον κόμβο.

5.3 Αρχιτεκτονική σχεδίαση και υλοποίηση του συστήματος

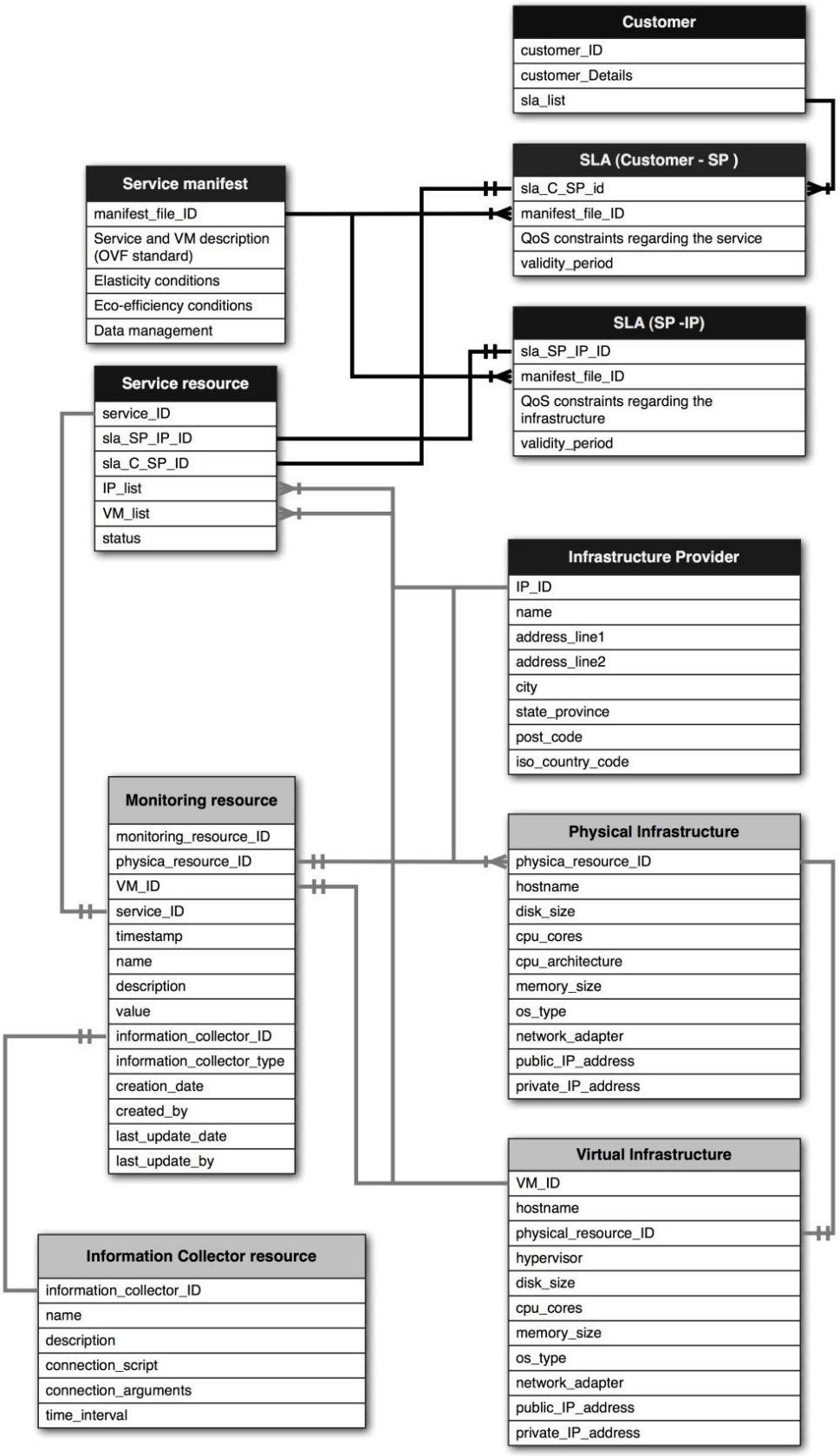
5.3.1 Μοντέλο δεδομένων

Η συλλογή, η διαχείριση και η αξιολόγηση των πληροφοριών ενός συστήματος γίνονται πάντα κάτω από τη «ομπρέλα» ενός προκαθορισμένου μοντέλου δεδομένων. Στις προσανατολισμένες στις υπηρεσίες αρχιτεκτονικές καθώς επίσης και στο υπολογιστικό πλέγμα (*Grid Computing*) έχει αναγνωριστεί αυτή η ανάγκη και έχει προκύψει ακόμη και το ζήτημα της διαλειτουργικότητας μεταξύ των διαφορετικών συστημάτων. [82][83][84] Για αυτόν τον λόγο, το ανοικτό φόρουμ OGF έχει προτείνει διάφορες προδιαγραφές μοντέλων

δεδομένων όπως το GLUE [85] και το *Activity Instance* (AID) [86]. Το πρώτο είναι ένα εννοιολογικό πρότυπο πληροφοριών για τις οντότητες πλέγματος, ανεξάρτητο από τις διάφορες υλοποιήσεις πλέγματος προκειμένου να επιβληθεί η διαλειτουργικότητα. Είναι αρκετά επιτυχές και υιοθετείται από διάφορα προγράμματα και πρωτοβουλίες [87][88]. Το δεύτερο προσπαθεί να ενσωματώσει πληροφορίες σχετικές με τις δραστηριότητες όπως η χρήση των πόρων, τα στοιχεία ασφάλειας, την κατάσταση ή τα δεδομένα επίβλεψης.

Αφ' ενός, όπως παρουσιάζεται σε προηγούμενο κεφάλαιο, το οικοσύστημα των Νεφών περιλαμβάνει διάφορους ρόλους και οντότητες. Οι διαφορετικοί τύποι υπηρεσιών προσφέρονται ως πραγματοποίηση αυτού του παραδείγματος Νεφών (SaaS, IaaS, Paas, NaaS κ.λπ.) και, δεδομένου ότι οι σχετικές τεχνολογίες ωριμάζουν, όλο και περισσότερο τύποι θα πλημμυρίσουν την αγορά. Καθώς η έρευνα αλλά και η επιχειρησιακή δραστηριότητα σχετική με τα Υπολογιστικά Νέφη αυξάνονται, το ίδιο συμβαίνει και στην παραγωγή των πληροφοριών και την απαίτηση για την καλύτερη διαχείρισή τους. Η υιοθέτηση μοντέλων και τεχνικών που έχουν σχεδιαστεί για SOA και Grid μπορεί να είναι μια λύση μέχρι ένα σημείο, αλλά οι καινοτομίες και οι διαφορές που το Υπολογιστικό Νέφος φέρνει προκαλούν συνήθως ασυνέπειες. Η εισαγωγή του στρώματος εικονικοποίησης, η ευρεία πρόσβαση που παρέχεται στο κοινό, τα διαφορετικά επιχειρησιακά μοντέλα που εφαρμόζονται και οι νέοι προσδιορισμένοι ρόλοι (προμηθευτής υποδομής, φορέας παροχής υπηρεσιών, προμηθευτής πλατφορμών, προμηθευτής αποθήκευσης κ.λπ.) είναι μερικές από τις συγκεκριμένες απαιτήσεις Νεφών που οδηγούν την ανάγκη για ένα καινούριο μοντέλο δεδομένων.

Υπο αυτές τις συνθήκες, έχουμε σχεδιάσει ένα μοντέλο δεδομένων που μπορεί να εφαρμοστεί σε διαφορετικά σενάρια και περιπτώσεις χρήσης Νεφών. Σε αυτό, ο φορέας παροχής υπηρεσιών (*Service Provider* - SP) είναι το κεντρικός ρόλος, και θεωρούμε την υπηρεσία (ή την εφαρμογή) ως κύρια οντότητα στο οικοσύστημα του Νέφους αλλά ενσωματώνουμε επίσης τους ρόλους του πελάτη, ως ο τελικός χρήστης της υπηρεσίας, και τον προμηθευτή υποδομής (*Infrastructure Provider* - IP). Φυσικά, το μοντέλο δεδομένων μπορεί να εμπλουτιστεί με περισσότερους τύπους προμηθευτών, που διαμορφώνονται με τον ίδιο τρόπο με την οντότητα IP.



Σχήμα 34: Μοντέλο δεδομένων Υπολογιστικών Νεφών

Παρακάτω αναλύονται οι οντότητες που ορίστηκαν όπως αυτές παρουσιάζονται στο Σχήμα 34. Οι πίνακες που παρουσίαζονται με μαύρο χρώμα σχετίζονται με συστατικά και διαδικασίες του παρόχου υπηρεσιών (SP), ενώ με γκρι του παρόχου υποδομών (IP).

Service manifest: είναι η οντότητα που αντιπροσωπεύει το πρότυπο μιας υπηρεσίας που πρόκειται να εγκατασταθεί και εκτελεστεί σε μια υποδομή Νέφους. Κάθε πρότυπο υπηρεσίας αποτελείται από διάφορα μέρη πληροφοριών: τη περιγραφή της υπηρεσίας χρησιμοποιώντας το πρότυπο OVF [89], τις απαιτήσεις ελαστικότητας (κανόνες και πολιτικές που καθορίζουν την ελαστικότητα της υπηρεσίας κατά τη διάρκεια του χρόνου εκτέλεσης), συνθήκες ενεργειακής αποδοτικότητας (πολιτικές και τιμές που συσχετίζονται με την κατανάλωση ενέργειας και την αποδοτικότητά της), διαχείριση δεδομένων (λεπτομέρειες για την αποθήκευση των δεδομένων και την πρόσβαση αυτών από την υπηρεσία).

SLA (Customer - SP): είναι το συμβόλαιο υπηρεσίας (*Service Level Agreement* - SLA) μεταξύ ενός πιθανού πελάτη/χρήστη και του φορέα παροχής υπηρεσιών (SP) που προσφέρει την πρόσβαση σε διάφορες υπηρεσίες/εφαρμογές. Ένα υπογεγραμμένο έγγραφο SLA περιλαμβάνει μια αναφορά στην περιγραφή της υπηρεσίας (*Service manifest*) που ο πελάτης έχει επιλέξει ως πρότυπο της υπηρεσίας που επέλεξε. Επιπλέον, περιλαμβάνει τους συγκεκριμένους όρους σχετικά με το QoS που καθορίζεται για τη συγκεκριμένη χρήση υπηρεσιών/εφαρμογής.

SLA (SP - IP): είναι το συμβόλαιο SLA μεταξύ του φορέα παροχής υπηρεσιών (SP) και του προμηθευτή υποδομής (IP). Ο SP επιλέγει έναν κατάλληλο IP βασισμένο στην περιγραφή της υπηρεσίας (*manifest file*) και τις ζητούμενες παραμέτρους QoS . Για αυτόν τον λόγο, το υπογεγραμμένο SLA, μεταξύ εκείνων των ρόλων, καθορίζει τους συγκεκριμένους περιορισμούς QoS σχετικά με την υποδομή, φυσική καθώς επίσης και εικονική, η οποία παρέχεται από τον αντίστοιχο IP.

Customer: είναι η οντότητα που αντιπροσωπεύει το τελικό χρήστη της υπηρεσίας που παρέχεται από τον SP. Η συγκεκριμένη δομή δεδομένων περιλαμβάνει τις λεπτομέρειες του

πελάτη (στοιχεία ονόματος, διεύθυνση, στοιχεία λογαριασμού, κ.λπ.) καθώς επίσης και των αναφορών στα SLAs που έχει υπογράψει με τον SP. Αυτή η οντότητα καταναλώνεται από τα τμήματα λογιστικής και τιμολόγησης του οικοσυστήματος Νέφους.

Infrastructure Provider (IP): είναι η αναπαράσταση ενός προμηθευτή υποδομής (IP) που προσφέρει υπηρεσίες Νεφών πάνω από τη φυσική υποδομή του. Οι λεπτομέρειες για την ιδιοκτησία και τη θέση του προμηθευτή (πόλη, χώρα, διεύθυνση κ.λπ.) ενσωματώνονται σε αυτή την δομή δεδομένων.

Physical Infrastructure: είναι η οντότητα που περιγράφει τους φυσικούς κόμβους μιας υποδομής. Κάθε αρχείο αυτής της δομής περιλαμβάνει τις τεχνικές λεπτομέρειες για έναν κόμβο, όπως ο χώρος στο δίσκο, οι πυρήνες CPU, η μνήμη, το λειτουργικό σύστημα κ.λπ. και σχετίζεται με έναν IP.

Virtual Infrastructure: αντιπροσωπεύει το εικονικό περιβάλλον που ένας προμηθευτής διατηρεί. Κάθε εγγραφή αυτής της οντότητας περιγράφει μια εικονική μηχανή (VM), που εγκαθίστανται πάνω σε έναν φυσικό πόρο, συμπεριλαμβανομένων όλων των τεχνικών προδιαγραφών αυτής της εγκατάστασης (hypervisor, χώρος στο δίσκο, μνήμη, πυρήνες CPU, διευθύνσεις IP κ.λπ.).

Information collector resource: η ουσιαστική συλλογή των δεδομένων επιτυγχάνεται μέσω διάφορων τμημάτων λογισμικού και συστημάτων ανάλογα με τον τύπο της ελεγχόμενης υποδομής. Παραδείγματος χάριν, ο φυσικός πόρος θα μπορούσε να επιβλεφθεί από ένα σύστημα Nagios, Ganglia ή οποιοδήποτε άλλο μηχανισμό ελέγχου. Για την εικονική υποδομή κάποιος θα μπορούσε να χρησιμοποιήσει το Libvirt API ή άλλη κάποια άλλη διεπαφή που μεσολογισμικό Νέφους παρέχει. Στο μοντέλο δεομένων μας, έχουμε καθορίσει την οντότητα *information_collector_resource* που συγκρατεί τις απαραίτητες πληροφορίες για την πρόσβαση σε οποιουδήποτε είδους συστήματος. Με τον καθορισμό σε αυτή τη δομή των λεπτομερειών σύνδεσης (αρχείο σύνδεσης, παράμετροι σύνδεσης), το σύστημα επίβλεψης μπορεί να εφαρμόσει μια λειτουργία «pull» και να συλλέξει τα δεδομένα. Η μόνη απαίτηση

είναι το αποτέλεσμα της εκτέλεσης να είναι μια δομή συμβατή με την οντότητα επίβλεψης (*Monitoring resource*).

Monitoring resource: αυτή η οντότητα αντιπροσωπεύει τις πληροφορίες επίβλεψης που συλλέγονται είτε από την υπηρεσία, τη φυσική είτε την εικονική υποδομή κατά τη διάρκεια της εκτέλεσης μιας υπηρεσίας. Κάθε εγγραφή περιλαμβάνει τα απαραίτητα προσδιοριστικά προς τις αντίστοιχες οντότητες (πόροι υπηρεσιών, φυσική και εικονική υποδομή) προκειμένου να πραγματοποιηθούν οι κατάλληλες σχέσεις. Η παράμετρος information_collector_type χρησιμοποιείται προκειμένου να επικυρωθεί η συνέπεια της έκθεσης επίβλεψης βασισμένης στον ακόλουθο κανόνα: εάν ο τύπος είναι «physical», το physical_resource_ID δεν πρέπει να είναι μηδενικό, προκειμένου να είναι σε θέση να αφορά εκείνη την έκθεση με τη φυσική υποδομή. Επιπλέον, όταν είναι «virtual», το VM_ID πρέπει να είναι συμπληρωμένο και όταν «service», το αντίστοιχο service_ID επίσης.

Service resource: αυτή η οντότητα είναι συσχετίζει την υπηρεσία, το SLAs, τους φυσικούς κόμβους και την εικονική υποδομή. Περιγράφει την πραγματική εγκατάσταση μιας υπηρεσίας, με υπογεγραμμένο SLAs, σε μια υποδομή ενός IP. Είναι μια πολύ σημαντική οντότητα που κρατά κεντρικά τα δεδομένα στον SP και επιτρέπει σε όλα τα σχετικά με τον SP συστατικά να παρακολουθήσουν την εγκατάσταση και την εκτέλεση μιας υπηρεσίας.

5.3.2 Μηχανισμός συλλογής πληροφοριών

Η υποδομή επίβλεψης στοχεύει στο να μετρήσει και να συλλέξει τις πληροφορίες από τα διαφορετικά επίπεδα του περιβάλλοντος του Νέφους, από το επίπεδο εφαρμογής/υπηρεσιών, στο εικονικό περιβάλλον, έως και από τη φυσική υποδομή (και αυτό περιλαμβάνει και τη συλλογή των σχετικών με την ενέργεια στοιχείων). Τα ακόλουθα διαφορετικά κομμάτια μπορούν να διακριθούν:

- Προμηθευτές πληροφοριών (*Monitoring information providers*), οι οποίοι περιλαμβάνουν τις διαφορετικές πηγές από όπου τα στοιχεία επίβλεψης συλλέγονται, καθώς επίσης και τα συστατικά υπεύθυνα για τη συλλογή τους (συλλέκτες δεδομένων). Η υποδομή επίβλεψης σχεδιάζεται κατά τέτοιο τρόπο ώστε αυτό το

κομμάτι να είναι επεκτάσιμο, επιτρέποντας την ενσωμάτωση πρόσθετων πηγών μέσω των αντίστοιχων συλλεκτών.

- Περιβάλλον βάσεων δεδομένων επίβλεψης (*Monitoring database environment*), που περιλαμβάνει τα συστατικά υπεύθυνα για την επεξεργασία των στοιχείων που συλλέχθηκαν και την αποθήκευσή τους σε μια αθροισμένη βάση δεδομένων. Περιλαμβάνει επίσης τα συστατικά που μπορεί κάποιος να χρησιμοποιήσει για να αλληλεπιδράσει με την βάση δεδομένων.
- Καταναλωτές πληροφοριών (*Monitoring information consumers*), που έχουν πρόσβαση στη βάση δεδομένων επίβλεψης έτσι ώστε να επεξεργαστούν τις συλλεχθείσες πληροφορίες.

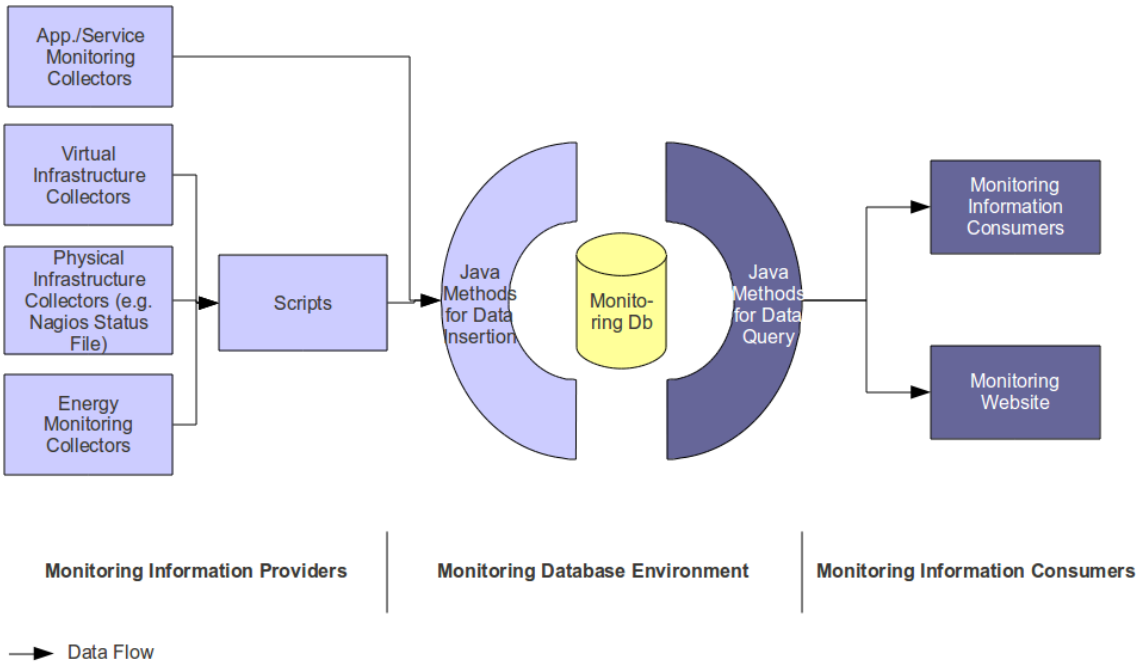
Η υποδομή ελέγχου προσφέρει επίσης πρόσβαση σε ένα γραφικό περιβάλλον προκειμένου να παρασχεθεί μια επισκόπηση των ιστορικών στοιχείων επίβλεψης. Ο ιστοχώρος αυτός θεωρείται επίσης ως ένας καταναλωτής πληροφοριών που ρωτά για στοιχεία από τη βάση δεδομένων.

Οι συλλέκτες πληροφοριών που χρησιμοποιούνται από την υποδομή επίβλεψης είναι οι ακόλουθοι:

- Συλλέκτες εφαρμογής/υπηρεσιών, που στοχεύουν να συλλέξουν τα στοιχεία σχετικά με την εφαρμογή ή την υπηρεσία που είναι εγκατεστημένη και εκτελείται μέσα στο περιβάλλον του Νέφους.
- Συλλέκτες εικονικής υποδομής, που στοχεύουν να συλλέξουν τα στοιχεία που αφορούν το εικονικό περιβάλλον στο οποίο οι υπηρεσίες εγκαθίστανται και εκτελούνται.
- Συλλέκτες φυσικής υποδομής, που στοχεύουν να συλλέξουν τα στοιχεία που αφορούν τη φυσική υποδομή μέσω της οποίας οι προμηθευτές της υποδομής προσφέρουν εικονικοποιημένους πόρους (αυτό περιλαμβάνει τους πόρους υπολογισμού, αποθήκευσης και δικτύωσης).

- Συλλέκτες ενεργειακής κατανάλωσης, που στοχεύουν να συλλέξουν στοιχεία σχετικά με την κατανάλωση ενέργειας των συσκευών που ανήκουν στον προμηθευτή της υποδομής.

Οι συλλέκτες πληροφοριών μπορούν να λειτουργήσουν είτε με έναν τρόπο ώθησης («push») είτε με έναν τρόπο τραβήγματος («pull»). Στον τρόπο ώθησης, οι συλλέκτες παίρνουν την πρωτοβουλία της σύνδεσης στη βάση δεδομένων, και υποβάλλουν τα στοιχεία άμεσα για την εισαγωγή μέσω μιας κλήσης στην υπηρεσία επίβλεψης σε ένα προκαθορισμένο χρονικό διάστημα. Αυτός ο τρόπος χρησιμοποιείται, π.χ., από τους συλλέκτες εφαρμογής/υπηρεσιών για να ελαχιστοποιηθεί το ίχνος που αφήνει το λογισμικό ελέγχου χρησιμοποιώντας μια μονόδρομη μετάδοση δεδομένων μέσα από την VM στο εξωτερικό σύστημα επίβλεψης. Αντιθέτως, στον τρόπο τραβήγματος η σύνδεση μεταξύ του κύριου συστήματος επίβλεψης και των συλλεκτών αρχίζει από το ίδιο το περιβάλλον βάσεων δεδομένων και φτάνει στους συλλέκτες, τους εκτελεί με προκαθορισμένη συχνότητα και καταχωρεί τελικά τα συλλεχθέντα στοιχεία στη βάση δεδομένων. Στο Σχήμα 35 παρουσιάζεται η γενική δομή της υποδομής επίβλεψης που προτείνεται.



Σχήμα 35: Υποδομή επίβλεψης και παρακολούθησης πολλαπλών πηγών δεδομένων

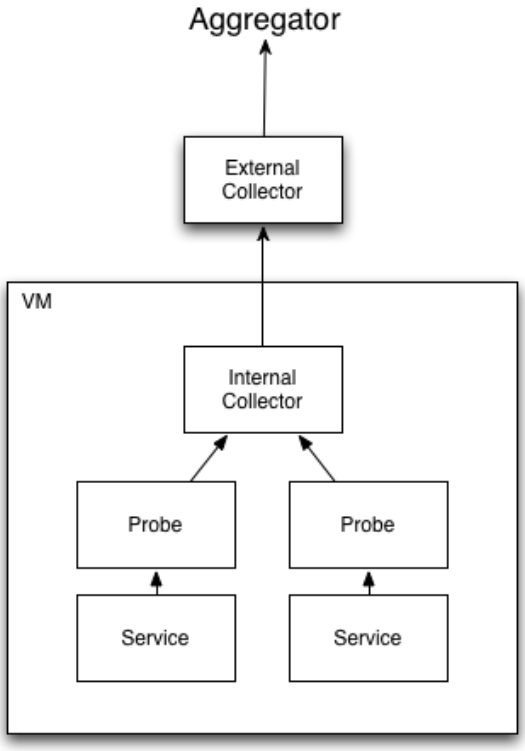
5.3.2.1 Συλλέκτης δεδομένων επίβλεψης εφαρμογής

Αυτό το συστατικό συλλέγει τις υψηλού επιπέδου πληροφορίες επίβλεψης από τις εφαρμογές ή τις υπηρεσίες που εκτελούνται μέσα στο εικονικό περιβάλλον. Η πρόκληση είναι να εξαχθούν τα στοιχεία μέσα από τα VMs και να κατασταθούν αυτά διαθέσιμα στη πλατφόρμα, κατά προτίμηση χωρίς εισαγωγή σφιχτών εξαρτήσεων μεταξύ του λογισμικού στο VM και του περιβάλλοντος της πλατφόρμας. Τα υπο-συστατικά αρμόδια για την ανάγνωση και την έκδοση των στοιχείων επίβλεψης αναφέρονται συχνά και ως «probes». Παραδείγματος χάριν, στο σύστημα επίβλεψης Lattice, υπάρχει ένα χωριστό probe για κάθε παράμετρο που πρόκειται να μετρηθεί. Οι έλεγχοι που μετρούν κάποια τιμή (παραδείγματος χάριν τρέχων αριθμός χρηστών που χρησιμοποιούν έναν κεντρικό υπολογιστή διαδικτύου Apache) μπορούν να επαναχρησιμοποιηθούν σε διάφορες διαφορετικές υπηρεσίες χωρίς τροποποίηση. Η εναλλακτική λύση είναι να υπάρξει ένας προσωποποιημένος συλλέκτης δεδομένων για κάθε τύπο υπηρεσίας, που διαχειρίζεται όλες τις ενδιαφέρουσες τιμές από μια ενιαία εικονική μηχανή.

Ένα κοινό μοντέλο δεδομένων για τα στοιχεία εφαρμογής ή υπηρεσιών πρέπει να υφίσταται μεταξύ των *probes* (που παράγουν τα στοιχεία) και των υψηλότερου επιπέδου συστατικών που καταναλώνουν τα στοιχεία. Ανεξάρτητα από το πώς τα δεδομένα περνούν μέσω των εμποδίων της εικονικής μηχανής, ένα εξωτερικό τμήμα συλλεκτών δεδομένων είναι αρμόδιο για να καταστήσει τα στοιχεία διαθέσιμα στο ανώτερο επίπεδο της υποδομής επίβλεψης. Αυτό μπορεί να γίνει είτε με έναν τρόπο ώθησης, δηλώνοντας το σύστημα επίβλεψης όταν το νέο στοιχείο είναι διαθέσιμο, είτε με έναν τρόπο τραβήγματος, που εκθέτει μια διεπαφή που επιτρέπει τα στοιχεία μετά από την απαίτηση. Στο επίπεδο της πλατφόρμας, η υπηρεσία επίβλεψης έχει πρόσβαση γενικά στα στοιχεία μέσω μιας διεπαφής, «pull», προκειμένου να τα αποφύγει για να υπερφορτωθεί από τα στοιχεία που ωθούνται.

Το διάγραμμα στο Σχήμα 36 παρουσιάζει την γενική δομή του μηχανισμού συλλογής δεδομένων επίβλεψης από τις εφαρμογές ή της υπηρεσίες. Τα δεδομένα συλλέγονται μέσα στην εικονική μηχανή από κάποιο probe, και έπειτα διαβιβάζονται σε ένα εξωτερικό

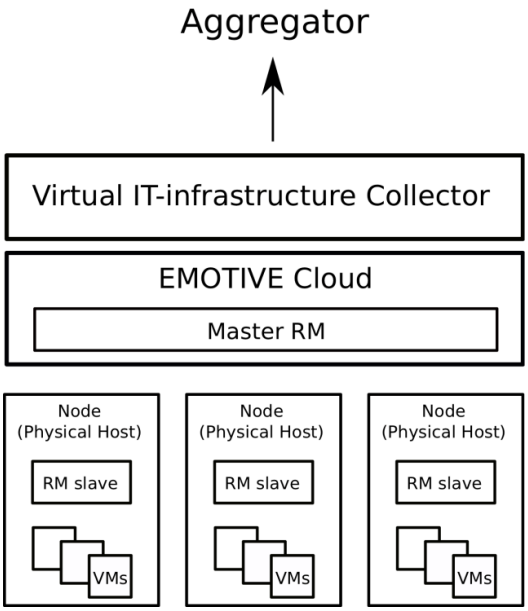
συστατικό που είναι αρμόδιο να καταστήσει τα στοιχεία διαθέσιμα στο κοινό σύστημα επίβλεψης (*Aggregator*).



Σχήμα 36: Αρχιτεκτονικό διάγραμμα συλλογή δεδομένων από την εφαρμογή

5.3.2.2 Συλλέκτης δεδομένων επίβλεψης εικονικού περιβάλλοντος

Το συστατικό αυτό είναι υπεύθυνο για την συλλογή δεδομένων επίβλεψης από την εικονική υποδομή και να τα παρέχει αυτές τις μετρήσεις στις υπηρεσίες διαχείρισης της πλατφόρμας και συγκεκριμένα στον συναθροιστή (Aggregator).



Σχήμα 37: Αρχιτεκτονικό διάγραμμα συλλογή δεδομένων από την εικονική υποδομή

Όπως παρουσιάζεται στο Σχήμα 37, ο μηχανισμός συλλογής πληροφοριών (σημειώνεται ως *Resource Monitor* - RM) βασίζεται στο μεσσολογισμικό EMOTIVE Cloud [90]. Είναι μια αποκεντρωμένη αρχιτεκτονική όπου υπάρχει ένα *Master RM* ανά εικονική υποδομή και ένα *Slave RM* ανά φυσικό κόμβο. Ο στόχος των *Slave RMs* είναι να συλλέγουν δεδομένα σχετικά με τις εικονικές μηχανές που εκτελούνται σε κάποιον κόμβο, ενώ των *Master RM* να διαχειρίζονται τα προηγούμενα και τις πληροφορίες που παρέχονται από αυτά. Τέλος, οι πληροφορίες επίβλεψης που συλλέγονται από την εικονική υποδομή, επιστρέφεται από το EMOTIVE σε ένα αρχείο XML όπως παρουσιάζεται παρακάτω:

```
<CLUSTER LOCALTIME="1311606671430" NAME="Cluster">
 <HOST NAME="host">
  <VM ID="06516cca-62a1-44bc-96d7-ebc1173fae25" IP="0.0.0.0" NAME="vm">
   <METRIC NAME="machine_type" UNITS="" VAL="x86_64" MEAN=""/>
   <METRIC NAME="os_name" UNITS="" VAL="CentOS" MEAN=""/>
   <METRIC NAME="virtual_domain" UNITS="" VAL="Dom-0" MEAN=""/>
   <METRIC NAME="state" UNITS="" VAL="Running" MEAN=""/>
   <METRIC NAME="cpu_vnum" UNITS="" VAL="2" MEAN="2"/>
   <METRIC NAME="cpu_speed" UNITS="MHz" VAL="2266" MEAN="2266.00"/>
   <METRIC NAME="cpu_user" UNITS="\%" VAL="37.12" MEAN="28.31"/>
   <METRIC NAME="mem_total" UNITS="KB" VAL="2048" MEAN="2048"/>
   <METRIC NAME="mem_used" UNITS="\%" VAL="72.91" MEAN="58.20"/>
   <METRIC NAME="swap_total" UNITS="KB" VAL="1024" MEAN="1024"/>
   <METRIC NAME="disk_total" UNITS="GB" VAL="300" MEAN="300" />
   <METRIC NAME="bytes_in" UNITS="bytes" VAL="110" MEAN="135"/>
```

```
    <METRIC NAME="bytes_out" UNITS="bytes" VAL="226" MEAN="306"/>
   </VM>
  </HOST>
</CLUSTER>
```

Η ετικέτα *CLUSTER* αντιπροσωπεύει μια δεδομένη εικονική υποδομή, η οποία συντίθεται από διάφορους φυσικούς κόμβους (ετικέτα *HOST*). Με τον ίδιο τρόπο, διάφορα VM (ετικέτα *VM*) μπορούν να εκτελούνται σε κάθε ένα από αυτούς τους κόμβους. Επίσης, ένα σύνολο μετρικών (*METRIC*) παρέχεται ανά VM. Για κάθε έναν από αυτούς ορίζεται, το όνομα, οι μονάδες, η αξία, και η μέση αξία. Όπως παρουσιάζεται, υπάρχει ένα ευρύ φάσμα μετρικών της εικονικής υποδομής που μπορούν να παρασχεθούν από αυτόν τον συλλέκτη: εικονική αρχιτεκτονική, λειτουργικό σύστημα, εικονική περιοχή όπου το VM τρέχει, κατάσταση του VM, αριθμός των εικονικών CPU, συχνότητα των CPUs, ποσοστό της κατανάλωσης CPU κ.α..

Τέλος, αξίζει να σημειωθεί ότι η προτεινόμενη λύση είναι ένα εργαλείο ανεξάρτητο από την πλατφόρμα εικονικοποιήσης και της τεχνολογίας αυτής. Έτσι, υποστηρίζουμε τη συλλογή των πληροφοριών επίβλεψης από διαφορετικές υλοποιήσεις Νεφών με τη βοήθεια του λογισμικού *Libvirt* [91]. Είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα που παρέχει, μεταξύ των άλλων, ένα API για να διαχειριστεί τις διαφορετικές υλοποιήσεις εικονικοποίησης (*hypervisors*).

5.3.2.3 Επίβλεψη φυσικής υποδομής

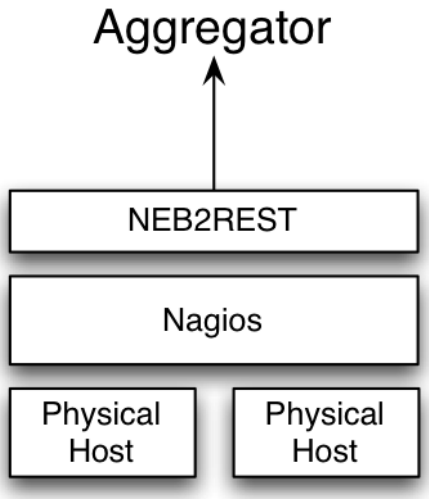
Το δομικό συστατικό αυτό συλλέγει πληροφορίες σχετικά με την φυσική υποδομή. Συγκεκριμένα με την υλοποίησή μας είμαστε σε θέση να συγκεντρώσουμε μετρήσεις για τις παραμέτρους που παρουσιάζονται στο Πίνακας 11.

Parameter	***Description***	***Data type***	***Example***
IP address	IP address of a VM	String	x.y.z.w
Architecture	The system's architecture of a node	String	Amd64
Nr. CPUs	The number of CPUs available on the node	Integer	8
CPU speed	The speed of the CPU (in MHz)	Integer	3000

Memory	The amount of memory (in megabytes)	Integer	8192
Free Memory	The amount of free memory (in megabytes)	Integer	2048
Bandwidth	The bandwidth of the node (in Kbytes/sec)	Integer	10
Disk space	The amount of disk space (in megabytes)	Integer	1

Πίνακας 11: Παράμετροι επίβλεψης φυσικής υποδομής

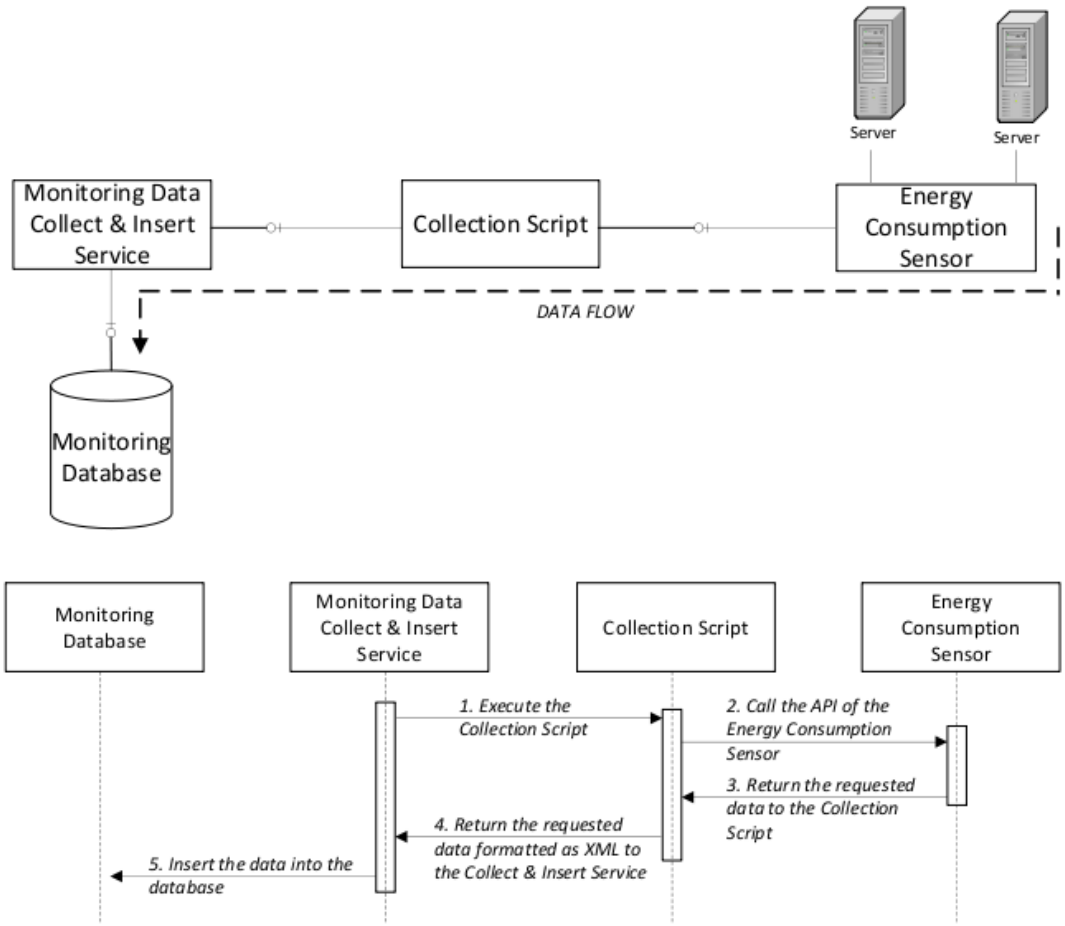
Η υλοποίηση αυτού του συλλέκτη δεδομένων βασίζεται στον μηχανισμό του Nagios. Η επιλογή αυτού του συστήματος έγινε καθώς το Nagios παρέχει μια πλατφόρμα συλλογής παραμέτρων πολύ αποδοτική όσον αφορά την κατανάλωση πόρων, με δυνατότητες επέκτασης και εγκατάστασης διαφορετικών μονάδων συλλογής δεδομένων και επίσης διατίθεται δωρεάν ως ένα λογισμικό ανοιχτού κώδικα. Συγκεκριμένα, χρησιμοποιούμε το σύστημα αυτό, διαμορφώνοντας μεμονωμένους ελέγχους για κάθε μετρική και προωθώντας τις μετρήσεις στο στρώμα της πλατφόρμας (Aggregator), μέσω του συστατικού NEB2REST. Το τελευταίο, όπως παρουσιάστηκε στο [92], αποτελεί ένα στοιχείο λογισμικού που μετατρέπει τα αποτελέσματα των μετρήσεων σε ένα έγγραφο XML (ως μια αναφορά επίβλεψης) και το ωθεί σε μια *RESTful* υπηρεσία, στην προκειμένη περίπτωση, την υπηρεσία Aggregator.



Σχήμα 38: Συλλογή δεδομένων από την υποδομή

5.3.2.4 Επίβλεψης ενεργειακής κατανάλωσης

Αυτή η ενότητα παρέχει περισσότερες λεπτομέρειες όσον αφορά τον τεχνικό σχεδιασμό και υλοποίηση της λύσης συλλογής και συγκέντρωσης δεδομένων σχετικών με την ενεργειακή κατανάλωση της υποδομής. Προκειμένου να συλλεχθούν τα σχετικά με την ενέργεια στοιχεία, οι φυσικοί πόροι είναι συνδεμένοι με τους αισθητήρες κατανάλωσης ενέργειας (για παράδειγμα, όπως εκείνοι της οικογένειας προϊόντων Racktivity EnergySwitch [93]). Οι αισθητήρες συνδέονται επίσης με το τοπικό δίκτυο της υποδομής και παρέχουν ένα API που μπορεί να χρησιμοποιηθεί από κάποιον για να συλλέξει τις πληροφορίες κατανάλωσης ενέργειας σε συνεχή βάση. Επιλέξαμε αυτή την οργάνωση επειδή αφ' ενός είναι μια οικονομική λύση για τη συλλογή δεδομένων (υπάρχουν πολυάριθμοι προσιτοί και αποδοτικοί αισθητήρες στην αγορά), και αφ' εταίρου είναι μια γενική λύση που μπορεί να υποστηρίξει οποιοδήποτε τύπο συσκευής υποδομής, συμπεριλαμβανομένων υπολογιστικών κεντρικών υπολογιστών, μονάδων αποθήκευσης, δρομολογητές, κ.λπ. Στο Σχήμα 39 παρουσιάζεται η ροή της πληροφορίας, από τους παραγωγούς των ενεργειακών στοιχείων (σε αυτήν την περίπτωση, κεντρικοί κόμβοι) στη βάση δεδομένων όπου οι συλλεχθείσες εκθέσεις αποθηκεύονται. Ένα διάγραμμα ακολουθίας απαριθμεί επίσης τα κύρια βήματα της διαδικασίας που εκτελείται για να μεταφέρει τις σχετικές με την ενέργεια πληροφορίες στη βάση δεδομένων.



Σχήμα 39: Επισκόπηση διαδικασίας επίβλεψης και αποθήκευσης δεδομένων

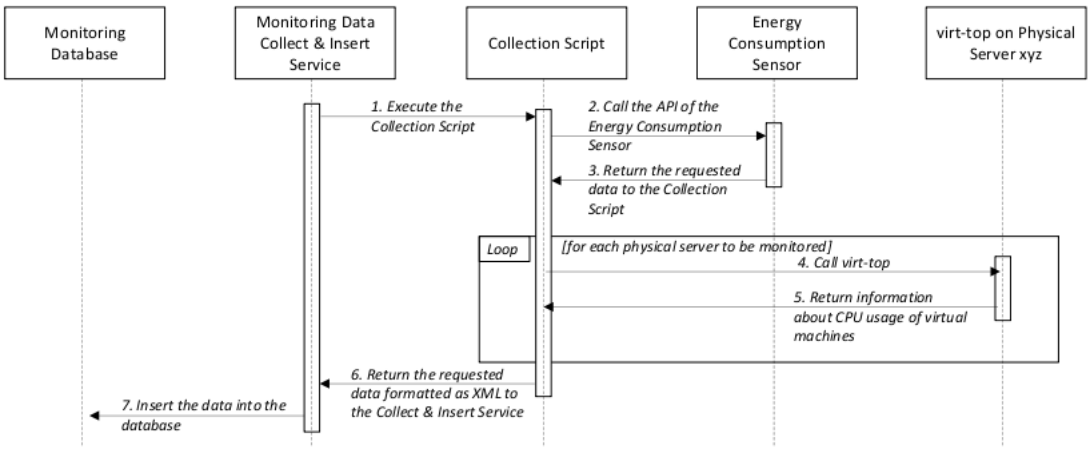
Η διαδικασία επίβλεψης για τα σχετικά με την ενέργεια δεδομένα, διαμορφώνεται με την τεχνική «τραβήγματος» (*pull*) και αρχικοποιείται από την υπηρεσία επίβλεψης συλλογής & εισαγωγής (*Monitoring data collect & insert service*) που εκτελεί τον κώδικα συλλογής (*Collection script*) περιοδικά, με ένα χρονικό διάστημα που διευκρινίζεται κατά τη διάρκεια της διαμόρφωσης της υποδομής επίβλεψης (κάθε συλλέκτης πληροφοριών που εκτελείται μπορεί να έχει διαφορετική χρονική περίοδο για τη συλλογή δεδομένων). Το εκτελέσιμο αρχείο κώδικα για την συλλογή δεδομένων είναι ένα αρχείο Python που καλεί το API του αισθητήρα κατανάλωσης ενέργειας προκειμένου να εξαχθούν οι απαραίτητες σχετικές με την ενέργεια πληροφορίες. Στην υποδομή μας, χρησιμοποιήσαμε τον αισθητήρα ES 1008-16 Racktivity EnergySwitch που εκθέτει διαφορετική διεπαφή για κάθε παράμετρο κατανάλωσης ενέργειας που συλλέγεται. Επομένως, το χειρόγραφο συλλογής που αναπτύξαμε καλεί τις διεπαφές του αισθητήρα σχετικού με τη διαδικασία μας, συλλέγει τις

μετρικές και τις αθροίζει σε μια έκθεση XML που επιστρέφεται στην υπηρεσία επίβλεψης. Οι σχετικές με την ενέργεια μετρικές που συλλέγονται μέσω των κλήσεων API παρατίθενται στον Πίνακας 12.

Metric Name	**Unit**
current	A
real power	W
real energy	kWh
apparent power	VA
apparent energy	kVAh
power factor in percent	-

Πίνακας 12: Παράμετροι ενέργειας

Η διαδικασία αξιολόγησης της ενεργειακής αποδοτικότητας όχι μόνο χρειάζεται αυτές τις μετρικές για να εκτελέσει το στόχο της αλλά απαιτεί επίσης πληροφορίες για τη χρήση CPU που συνδέεται με κάθε ένα από τα VM που τρέχουν στο περιβάλλον Νέφους. Τέτοιες πληροφορίες μπορούν να παρασχεθούν από το εργαλείο virt-top που παρέχεται στα λειτουργικά συστήματα Linux. Επομένως, επιπρόσθετα από την συλλογή των δεδομένων ενεργειακής κατανάλωσης, το εκτελέσιμο αρχείο συλλογής χρησιμοποιεί τον virt-top σε κάθε φυσικό κόμβο για να συγκεντρώσει πληροφορίες χρήσης CPU. Στο Σχήμα 40 παρουσιάζεται μια επέκταση του διαγράμματος ακολουθίας που δίνεται προηγουμένως συμπεριλαμβανομένων των κλήσεων που γίνονται στο virt-top.



Σχήμα 40: Ακολουθιακό διάγραμμα συλλογής ενεργειακών δεδομένων

Ένα παράδειγμα της πληροφορίας που επιστρέφεται από την κλήση στο virt-top είναι:

Domain-0.42;optimis-LMS 12.65;odfs-datanode-prototype 0;odfs-master-prototype 34.40;OptimisDB 6.00;ntua-datamanager 0.02

Οι αριθμοί δίπλα στο υποσύνολο VM (*VM domain*) είναι τα δεδομένα της κατανάλωσης CPU. Το τελευταίο βήμα στο ακολουθιακό διάγραμμα στο Σχήμα 40 αφορά την εισαγωγή των ενεργειακών δεδομένων, ως μια αναφορά σε XML, στην βάση δεδομένων. Η αναφορά αυτή παρουσιάζεται στο Σχήμα 41.

Η έκθεση XML παρουσιάζει μόνο σχετικά με την ενέργεια στοιχεία που συλλέγονται από έναν φυσικό κόμβο που προσδιορίζεται από το χαρακτηριστικό «*optimis1*». Σημειώνεται ότι οι συλλέκτες επίβλεψης πληροφοριών σχεδιάζονται κατά τέτοιο τρόπο ώστε να μπορούν να συλλέξουν με μία εκτέλεση δεδομένα από πολλαπλές πηγές, και συντάσσουν όλα τα συλλεχθέντα στοιχεία μέσα σε μια ενιαία έκθεση XML που υποβάλλεται για εισαγωγή στη βάση δεδομένων ελέγχου.

Μόλις ολοκληρωθεί το βήμα εισαγωγής στοιχείων, τα στοιχεία μπορούν να αποδοθούν στους πελάτες που εκτελούν τη διαδικασία αξιολόγησης της ενεργειακής αποδοτικότητας. Για αυτόν τον λόγο, το προαναφερθέν περιβάλλον βάσεων δεδομένων επίβλεψης παρέχει υπηρεσίες REST που οι πελάτες μπορούν να χρησιμοποιήσουν για να λάβουν όχι μόνο τα στοιχεία σχετικά με την κατανάλωση ενέργειας των φυσικών πόρων, αλλά και όλα τα άλλα στοιχεία που παρουσιάζονται στη βάση δεδομένων από άλλους συλλέκτες πληροφοριών (π.χ. συλλέκτες επίβλεψης υποδομής κ.λπ.). Στη βάση δεδομένων, κάθε εγγραφή έχει έναν τύπο πόρου *resource_type* που βοηθά να καθοριστεί ο τύπος και η πηγή της μετρικής αυτής. Για παράδειγμα, όλες οι εγγραφές που συλλέγονται παίρνουν καθορισμένο τύπο *resource_type* με τιμή «*energy*». Εκείνες που προέρχονται από τους φυσικούς συλλέκτες υποδομής παίρνουν την τιμή «*physical*». Οι συλλέκτες επίβλεψης της εικονικής υποδομής έχουν τον τύπο «*virtual*» στα δεδομένα που συγκεντρώνουν, και οι συλλέκτες εφαρμογής/υπηρεσιών τον τύπο «*service*». Σε γενικές γραμμές, ένας καταναλωτής

των πληροφοριών αυτών μπορεί να αποκτήσει τα δεδομένα από την βάση δεδομένων ρωτώντας με τα παρακάτω ορίσματα:

- Resource Type (allowed values: 'energy', 'physical', 'virtual', 'service'),
- Metric Name or Id Number identifying a physical server, a VM or a service,
- Time Window.

```
<?xml version="1.0"?>
<MonitoringResources>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>current</metric_name>
<metric_value>0,00</metric_value>
<metric_unit>A</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>real_power</metric_name>
<metric_value>115,00</metric_value>
<metric_unit>W</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>real_energy</metric_name>
<metric_value>513,50</metric_value>
<metric_unit>kWh</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
```

<physical_resource_id>optimis1</physical_resource_id>
<metric_name>apparent_energy</metric_name>
<metric_value>825,00</metric_value>
<metric_unit>kVAh</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>apparent_power</metric_name>
<metric_value>151,50</metric_value>
<metric_unit>VA</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>power_factor</metric_name>
<metric_value>74,00</metric_value>
<metric_unit>%</metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
<monitoring_resource>
<physical_resource_id>optimis1</physical_resource_id>
<metric_name>xentop_cpu</metric_name>
<metric_value>Domain-0 .65;centos-optimis-SP .15;centos-optimis-IP .01;
TestECO 89.76;system-dummyComponent_instance-1 .09</metric_value>
<metric_unit></metric_unit>
<metric_timestamp>1333359108</metric_timestamp>
<service_resource_id></service_resource_id>
<virtual_resource_id></virtual_resource_id>
<resource_type>energy</resource_type>
</monitoring_resource>
</MonitoringResources>

Σχήμα 41: Αναφορά επίβλεψης σε XML που κατατίθεται στην βάση δεδομένων

5.3.3 Αξιολόγηση και διαχείριση πληροφορίας

Όσον αφορά την διαχείριση και αξιολόγηση της πληροφορίας προτείνουμε διάφορα εργαλεία και τεχνικές που στοχεύουν να βοηθήσουν στην αυτό-διαχείριση των περιβαλλόντων Νεφών. Η βελτιστοποίηση ολόκληρου του κύκλου ζωής των υπηρεσιών που αρχίζει από την κατασκευή υπηρεσιών, περιλαμβάνει την αξιολόγηση βασισμένη στην εμπιστοσύνη (*Trust*), τον κίνδυνο (*Risk*), την ενεργειακή αποδοτικότητα (*Eco-efficiency*) και το κόστος (*Cost*). Αυτό αποτυπώνει την ουσία ενός βελτιστοποιημένου οικοσυστήματος Νεφών που παράγεται από την εμπιστοσύνη μεταξύ των καταναλωτών και των προμηθευτών, τον κίνδυνο μη επίτευξης των στόχων, τους οικολογικούς και οικονομικούς στόχους. Ορίζουμε αυτές τις παραμέτρους ως παράγοντες TREC. Με βάση αυτές τις αξιολογήσεις το Νέφος μπορεί να προσαρμοστεί σε μία κατάσταση αυτό-συντήρησης και να προβλέπει προβλήματα στο μέλλον. Κάθε ένας από αυτούς τους παράγοντες αναλύονται ως εξής:

- *Trust*: Η εμπιστοσύνη του SP και του IP μετριέται με την εξέταση της προηγούμενης γνώσης απόδοσης και του νομικό πλαίσιο των συμμετεχόντων. Αυτό περιλαμβάνει το διεξαγωγή διάφορων ερωτήσεων για την εμπιστοσύνη μεταξύ των δύο συμβαλλόμενων μερών που συμμετέχουν στην υπηρεσία.
- *Risk*: Ο παράγοντας κινδύνου αξιολογεί πόσο επικίνδυνο θα ήταν για τον SP ή τον IP να εκτελέσει την υπηρεσία. Αυτό μετριέται σε σχέση με τους διάφορους παράγοντες όπως η απόδοση υπηρεσιών, το υλικό και η απόδοση των εικονικών μηχανών, η ασφάλεια, η συντήρηση και τα νομικά ζητήματα.
- *Eco-efficiency*: Η ενεργειακή αποδοτικότητα λαμβάνει υπόψη τους περιορισμούς για να ελαχιστοποιήσει την ενεργειακή κατανάλωση κατά τη διάρκεια της λειτουργίας μιας υπηρεσίας.
- *Cost*: Αυτό περιλαμβάνει το συνδυασμένο κόστος της ιδιοκτησίας μιας υπηρεσίας κατά τη διάρκεια του κύκλου της ζωής του συμπεριλαμβανομένης της απόκτησης των πόρων της χρήσης, ποινική ρήτρα στην αποτυχία υπηρεσιών και πληρωμή, κατά τη διάρκεια του κύκλου της ζωής του στο Νέφος.

Κάθε ένας από τους συμμετέχοντες στο οικοσύστημα Νεφών, έχει διαφορετικά ενδιαφέροντα που απεικονίζονται στους στόχους επιχειρησιακών επιπέδων τους και στις στρατηγικές που εφαρμόζουν. Αυτές οι πολιτικές καθορίζουν ποιες ενέργειες λαμβάνονται και ποιες πληροφορίες επίβλεψης πρέπει να συγκεντρωθούν. Από την άποψη του τελικού χρήστη, το κύριο όφελος αυτού του συμβαλλόμενου μέρους εναπόκειται στις εγγυήσεις ότι μια υπηρεσία θα εμμείνει σε μια συμφωνία επιπέδων υπηρεσιών με διευκρινισμένες λειτουργικές και μη λειτουργικές παραμέτρους. Αυτό το ενδιαφέρον διατηρείται μέσω της συνεχής επίβλεψης της κατάστασης μιας υπηρεσίας κατά τη διάρκεια της εγκατάστασης και εκτέλεσής της. Ένας φορέας παροχής υπηρεσιών είναι αρμόδιος για την επιλογή που προμηθευτής υποδομής για μια δεδομένη υπηρεσία και στην διαχείριση της υπηρεσίας κατά τη διάρκεια της εγκατάστασης και εκτέλεσης. Έχοντας αυτό το ρόλο, χρησιμοποιεί τις πληροφορίες επίβλεψης για να ελέγξει συνεχώς την φυσική και εικονική υποδομή. Ένας προμηθευτής υποδομής χρησιμοποιεί τις πληροφορίες επίβλεψης για να βελτιστοποιήσει τη χρήση της φυσικής υποδομής του για την εκτέλεση των υπηρεσιών και να μεγιστοποιήσει τους στόχους επιχειρησιακών επιπέδων όπως η αποδοτικότητα.

Τα εργαλεία αξιολόγησης TREC ενεργούν ως μηχανισμοί φίλτρων πριν από και κατά τη διάρκεια της εκτέλεσης των υπηρεσιών. Αυτά τα εργαλεία παρέχουν προτάσεις σε άλλα συστατικά, έτσι ώστε ένας συμμετέχων να μπορεί να λάβει βελτιωμένες ενημερωμένες αποφάσεις που βελτιστοποιούν τη χρήση ενός Νέφους. Οι πληροφορίες επίβλεψης χρησιμοποιούνται για να αποτρέψουν παραβίαση στα SLAs, με τη λήψη κατάλληλων μέτρων εάν οι απαιτήσεις πλησιάζουν ή έχουν φθάσει στην ποιότητα των κατώτατων ορίων των υπηρεσιών. Για να βελτιστοποιήσουν το Νέφος που χρησιμοποιεί προκαθορισμένους επιχειρησιακούς στόχους, οι πληροφορίες επίβλεψης μπορούν να χρησιμοποιηθούν για να ενισχύσουν τη συνολική απόφαση που είναι προς όφελος ενός συμμετέχοντα.

5.3.3.1 Αξιολόγηση Εμπιστοσύνης (Trust Assessment)

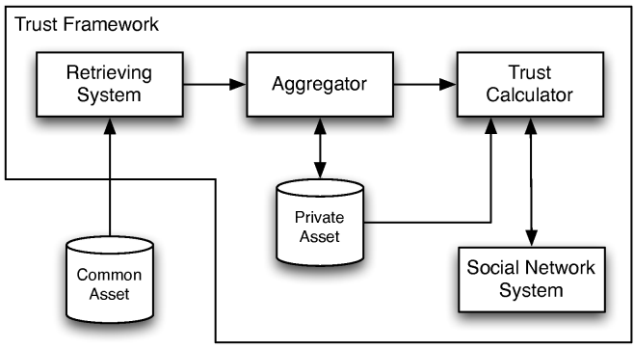
Στην περίπτωση του Υπολογιστικού Νέφους, η επιλογή μιας αξιόπιστης υποδομής ή ενός φορέα παροχής υπηρεσιών είναι αρκετά σημαντική διαδικασία αφού μέχρι κάποιο βαθμό θα τους παραδώσουμε τις πληροφορίες να τις διαχειριστούν έξω από την υποδομή μας.

Ως τμήμα των παραμέτρων TREC, ο μηχανισμός αξιολόγησης της εμπιστοσύνης (*Trust Framework*) χρειάζεται το σύστημα επίβλεψης έτσι ώστε να συγκεντρωθούν οι πληροφορίες οποιασδήποτε υπηρεσίας που τρέχει σε μια υποδομή IP, αλλά και από την ίδια την υποδομή, καθώς αυτές οι πληροφορίες θα διαχειριστούν και αποθηκευθούν κατά τέτοιο τρόπο ώστε ο μηχανισμός αυτός να αρχίσει να δημιουργεί ιστορικά στοιχεία (*assets*) από τα οποία η αξία της εμπιστοσύνης θα υπολογιστεί. Εντούτοις, αυτά τα ιστορικά στοιχεία δεν είναι η μόνη πηγή δεδομένων, αλλά επιπρόσθετα ένα κοινωνικό δίκτυο μεταξύ του SP και του IP αναπτύσσεται μαζί με αυτά τα ιστορικά στοιχεία.

Στο Σχήμα 42 μπορούμε να δούμε πώς τα αρχικά ακατέργαστα στοιχεία προσεγγίζονται καλώντας το *common asset*: αυτό είναι ένα ακατέργαστο σύνολο στοιχείων που συγκεντρώνει δεδομένα από όλους τους συμμετέχοντες στον κύκλο της ζωής μιας υπηρεσίας που εγκαθίστανται στο Νέφος. Εντούτοις οι αθροισμένες τιμές όπως η αξιοπιστία, η ηλικία, ή η ευρωστία, υπολογίζονται πάνω από τις προηγούμενες και αποθηκεύονται μέσα στο *Trust Framework*. Τα τελευταία χρησιμοποιούνται από τον υπολογιστή εμπιστοσύνης (*Trust Calculator*), ο οποίος είναι ο εγκέφαλος του τμήματος εμπιστοσύνης και αυτός που τροφοδοτεί το κοινωνικό δίκτυο για να παραγάγει και να διαχειριστεί τις σχέσεις.

Ο υπολογιστής εμπιστοσύνης, με τη χρησιμοποίηση ενός μοντέλου εμπιστοσύνης βασισμένου σε υποκειμενική [94], συγκεχυμένη [95] λογική και διαφορετικές άλλες τεχνικές σύγκρισης στοιχείων, είναι σε θέση να παραγάγει μια αξία εμπιστοσύνης για έναν συμμετέχοντα. Επιπλέον, τα στοιχεία που προέρχονται από την κοινωνική δικτύωση χρησιμοποιούνται ως ελεγκτής στον υπολογισμό εμπιστοσύνης, που σημαίνει ότι θα ελέγξουν τις μεγάλες αλλαγές της εμπιστοσύνης που προέρχεται από τις αλλαγές στο κοινό σύστημα ελέγχου (αποτυχίες συστημάτων ή δικτύων). Περισσότερες λεπτομέρειες για τις αρχές και τις

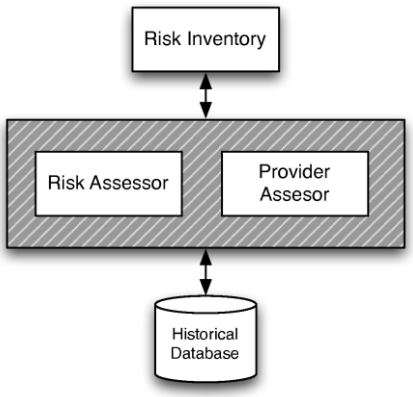
τεχνικές που χρησιμοποιούνται για την εφαρμογή του εργαλείου αξιολόγησης της εμπιστοσύνης παρουσιάζονται στα [96][97].



Σχήμα 42: Λειτουργία μηχανισμού αξιολόγησης της εμπιστοσύνης

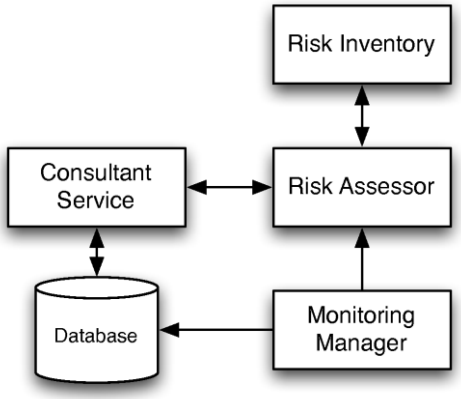
5.3.3.2 Αξιολόγηση Κινδύνου (Risk Assessment)

Ένα άλλο συστατικό που χρησιμοποιεί τις πληροφορίες επίβλεψης για να συμβάλει στην αυτό-διαχείριση του Νέφους είναι ο αξιολογητής κινδύνου (*Risk Assessment*). Ο κίνδυνος αξιολογείται στα διάφορα στάδια του κύκλου της ζωής μιας υπηρεσίας Νεφών από τους διαφορετικούς εμπλεκομένους. Για αυτόν τον λόγο, ο SP πρέπει να κάνει την αξιολόγηση του κινδύνου διαφόρων διαθέσιμων IPs και να επιλέξει έναν από αυτούς. Σε αυτό το επίπεδο δεν θα υπήρχε η ανάγκη τα συλλεχθέντα δεδομένα να τροφοδοτούνται συνεχώς στο εργαλείο ανάλυσης κινδύνου στο επίπεδο SP. Εντούτοις, στο επίπεδο IP, ο πάροχος πρέπει να κάνει συνεχή αξιολόγηση του κινδύνου όταν εκτελείται η υπηρεσία για τη μείωση των πιθανοτήτων αποτυχίας.



Σχήμα 43: Αξιολόγηση κινδύνου στον SP

Η ανάλυση κινδύνου απαιτεί τα στοιχεία να συλλέγονται σε δύο μορφές: στατική και δυναμική. Τα στατικά δεδομένα προσεγγίζονται χρησιμοποιώντας την ιστορική βάση δεδομένων σε συνδυασμό με τα εργαλεία επίβλεψης κινδύνου. Εντούτοις, τα δυναμικά στοιχεία τροφοδοτούνται συνεχώς από το τμήμα επίβλεψης και η συνεχής ανάλυση κινδύνου εκτελείται σε αυτά. Οι πληροφορίες επίβλεψης χρησιμοποιούνται από διάφορα συστατικά για να εκτελέσουν την αξιολόγηση στο δυναμικό περιβάλλον ενός Νέφους.



Σχήμα 44: Αξιολόγηση κινδύνου στον IP

Στο Σχήμα 44 παρουσιάζεται ότι τα εργαλεία επίβλεψης θα τροφοδοτούν τις πληροφορίες στα εργαλεία κινδύνου στο επίπεδο IP. Παραδείγματα τέτοιων πληροφοριών είναι ο τρέχων φόρτος εργασίας, οι διακοπές λειτουργίας συστημάτων, οι προσωρινές ελλείψεις απόδοσης, η κυκλοφορία δικτύων, διαθεσιμότητα εμπειρογνωμόνων ή γενικές πληροφορίες σχετικά με τον αριθμό υπηρεσιών που πρέπει να λειτουργήσουν. Αυτά τα επιβλεπόμενα δεδομένα βοηθούν στο να καθορίσει ο πάροχος τις δυσχέρειες της υποδομής έτσι ώστε να βελτιώσει τον προγραμματισμό, τη διοίκηση, και τη διαχείριση πόρων του.

Κατά τη διάρκεια της διαδικασίας ανάλυσης κινδύνου, τα εργαλεία αυτά θα αξιολογούν τον κίνδυνο βάσει συγκεκριμένων κατηγοριών που υλοποιούν το συμφωνθέν SLA. Οι διάφορες κατηγορίες κινδύνου που έχουν προσδιοριστεί είναι:

- Νομικές (*Legal*, π.χ SLA issues)
- Τεχνικές (*Technical*, π.χ.Hardware, VM failure)
- Πολιτικές (*Policy, π.χ.* Data jurisdiction policies)
- Γενικές (*Genera, π.χ.* Various issues such as security)

Μερικά από αυτά τα αντικείμενα κινδύνου έχουν στατική φύση αλλά κάποια άλλα, όπως τα τεχνικά ή τα νομικά θέματα, έχουν δυναμικό χαρακτήρα και πρέπει να επιβλέπονται και να αξιολογούνται συνεχώς. Στον Πίνακας 13 παρουσιάζονται μερικά από αυτά τα δυναμικά στοιχεία.

Asset identified: SLA
Vulnerability of asset: Lack of jurisdiction information
Threat to asset: Breach in data confidentiality
Resulting risk item: Changes in jurisdiction
Risk Category item belongs to: Policy
Risk Likelihood: Very high (5) [Range 1-5]
Risk Impact: High (4) [Range 1-5]
Resulting Risk level: Product of risk likelihood and risk impact [Range 1-25]
Risk event: Redeployment of data
Resulting risk mitigation: Seek legal advice

Πίνακας 13: Δυναμικά στοιχεία αξιολόγησης κινδύνου

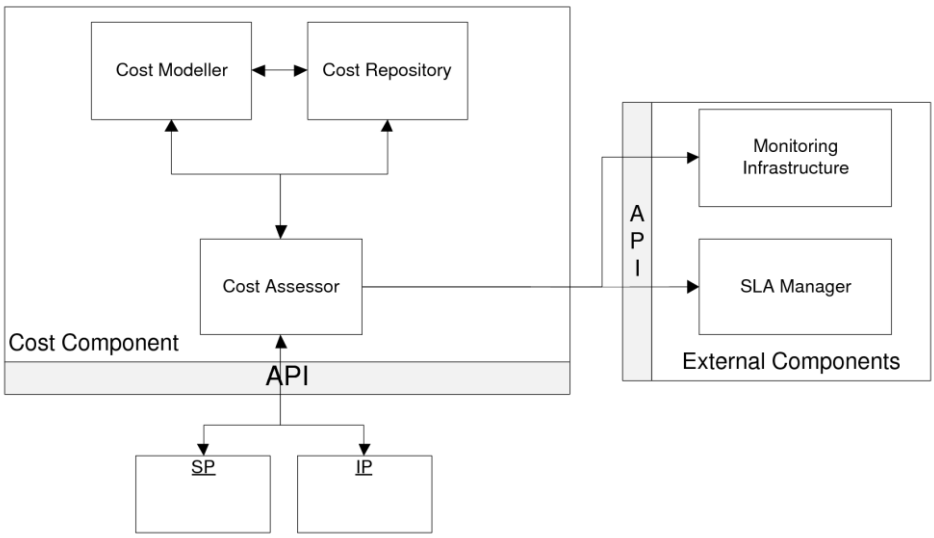
Μετά από τις αξιολογήσεις στους διάφορους παράγοντες κινδύνου και τον προσδιορισμό των σχετικών λύσεων μετριασμού, όπου είναι δυνατόν, οι κατάλληλες στρατηγικές θα αποφασιστούν για να εφαρμόσουν αυτές τις λύσεις. Αυτές οι στρατηγικές μετριασμού προτείνονται από το εργαλείο αξιολόγησης του κινδύνου στους σχετικούς συμμετέχοντες [98].

5.3.3.3 Αξιολόγηση κόστους (Cost Assessment)

Μαζί με την ελαστικότητα, το κόστος είναι ένας από τα κύρια χαρακτηριστικά που οδηγούν την μετάβαση στα Υπολογιστικά Νέφη. Υπάρχουν δύο κρίσιμα συστατικά της διαχείρισης δαπανών: αφενός να κατανοήσουμε τις τρέχουσες δαπάνες και αφετέρου να προβλέψουμε τις μελλοντικές δαπάνες. Και οι δύο ικανότητες είναι προσανατολισμένες προς τα δεδομένα, και απαιτούν τόσο ιστορικά όσο και δεδομένα πραγματικού χρόνου. Προκειμένου να συλλεχθούν

αυτά τα δεδομένα, ο μηχανισμός αξιολόγησης κόστους χρησιμοποιεί το σύστημα επίβλεψης που περιγράφεται προηγουμένως για να συγκεντρώσει όλες τις σχετικές πληροφορίες.

Ο μηχανισμός αυτός μπορεί να εγκατασταθεί ανεξάρτητα και στην υποδομή του SP και από την υποδομή του IP. Η δυνατότητα να αξιολογηθεί ακριβώς το κόστος μιας παρεχόμενης υπηρεσίας είτε στον SP είτε στο επίπεδο IP είναι μια απαίτηση για να επιτραπεί η διαφανής τιμολόγηση. Επιπλέον, η ικανότητα μιας οργάνωσης που προβλέπει ακριβώς και που διαχειρίζεται συνεπώς τις δαπάνες τους είναι κρίσιμη για την εξασφάλιση της τρέχουσας αποδοτικότητάς τους.



Σχήμα 45: Μηχανισμός αξιολόγησης κόστους

Ο μηχανισμός αποτελείται από τρία συστατικά: τον αξιολογητή δαπανών (*Cost Assessor*), ο μοντελοποιητής δαπανών (*Cost Modeller*) και την αποθήκη δαπανών (*Cost Repository*) όπως απεικονίζεται στο Σχήμα 45. Ο αξιολογητής δαπανών είναι το κεντρικό σημείο επικοινωνίας του μηχανισμού δαπανών και διαχειρίζεται όλα τα εισερχόμενα αιτήματα από τα εξωτερικά συστατικά. Ο μοντελοποιητής δαπανών είναι αρμόδιος για τον καθορισμό και την εκτέλεση των μαθηματικών μοντέλων για να καταγράψει ακριβώς και να προβλέψει το κόστος της εγκατάστασης/λειτουργίας των υπηρεσιών. Για να δημιουργήσει αυτά τα μοντέλα ο *Cost Modeller* χρησιμοποιεί στατιστικές τεχνικές στα στοιχεία που αποθηκεύονται μέσα στην αποθήκη δαπανών και που παρέχονται από το σύστημα επίβλεψης. Για τον SP αυτό μπορεί

να συμπεριλάβει τα στοιχεία σχετικά με τη φιλοξενία των δαπανών (από τον IP), της χρήσης υπηρεσιών, των επιπέδων SLA κ.λπ. Για τον IP αυτό μπορεί να συμπεριλάβει το εισόδημα φιλοξενίας (από τον SP), τη χρήση υπηρεσιών, τα επίπεδα SLA, τη χρησιμοποίηση υποδομής, την ενεργειακή χρήση κ.λπ.

Η αποθήκη δαπανών λειτουργεί ως μόνιμος αποθηκευτικός χώρος για τις τιμές κόστους που αναλύθηκαν και υπολογίστηκαν, αλλά και για τις σχετικές με το κόστος τιμές που συγκεντρώθηκαν από το σύστημα επίβλεψης. Αυτό χρησιμοποιείται στη συνέχεια για τον καθορισμό των μοντέλων κόστους. Οι περιγεγραμμένες λειτουργίες αξιολόγησης και πρόβλεψης που παρέχονται από αυτό το εργαλείο μπορούν να χρησιμοποιηθούν για να υποστηρίξουν τις διαδικασίες λήψης αποφάσεων απόφαση των SPs και IPs.

5.3.3.4 Αξιολόγηση ενεργειακής απόδοσης (Eco-Efficiency Assessment)

Μόλις συλλεχθούν οι σχετικές με την ενεργειακή κατανάλωση πληροφορίες επίβλεψης, κάποιος μπορεί να τις επεξεργαστεί προκειμένου να αξιολογήσει την ενεργειακή αποδοτικότητα των διαφορετικών οντοτήτων σε μια υποδομή Νεφών. Αυτή η διαδικασία, στην προτεινόμενη αρχιτεκτονική, είναι ευθύνη του μηχανισμού αξιολόγησης της ενεργειακής απόδοσης (*Eco-efficiency assessment tool*), ο οποίος χρησιμοποιεί τις πληροφορίες που παρέχονται από τους ενεργειακούς συλλέκτες επίβλεψης για να εκτελέσει δύο αξιολογήσεις αποδοτικότητας: την ενεργειακή αποδοτικότητα μιας δεδομένης οντότητας Νέφους (το σύνολο των χρήσιμων εργασιών ανά Watt που καταναλώνεται) και η οικολογική αποδοτικότητά του (το σύνολο των χρήσιμων εργασιών ανά κιλό του άνθρακα που εκπέμπεται στην ατμόσφαιρα). Αυτές οι αξιολογήσεις ρυθμίζονται πλήρως από σχετικούς με την ενέργεια (νομικούς) περιορισμούς όπως την κατοχή της πιστοποίησης LEED [99], η οποία μπορεί να καθοριστεί σύμφωνα με σχετικές με την ενέργεια παραμέτρους (π.χ., επίπεδα εκπομπής άνθρακα).

Ο αξιολογητής της ενεργειακής απόδοσης μπορεί να αξιολογήσει την αποδοτικότητα σύμφωνα με την παρούσα κατάσταση του προμηθευτή και να προβλέψει τέτοια

αποδοτικότητα για μια πιθανή μελλοντική θέση. Οι διενεργηθείσες αξιολογήσεις μπορούν να παρασχεθούν είτε με τεχνική τραβήγματος (*pull*), με την έκθεση των κατάλληλων διεπαφών για να επιτρέψουν την συλλογή δεδομένων κατ' απαίτηση, είτε με έναν τρόπο ώθησης (*push*), με τη βοήθεια των εγγραφών σε υπηρεσίες (*subscriptions*). Επιπλέον, οι αξιολογήσεις αυτές παρέχονται σε τέσσερα διαφορετικά επίπεδα: υπηρεσία, υποδομή, φυσικός κόμβος, και εικονικός κόμβος. Επομένως, τα αποτελέσματα ενεργειακής αποδοτικότητας που παρέχονται από αυτόν τον μηχανισμό έχουν τη δυνατότητα να χρησιμοποιηθούν από τους προμηθευτές των υποδομών σύννεφων (IaaS και PaaS) καθώς επίσης και του λογισμικού (SaaS). Στις παρακάτω ενότητες θα επικεντρωθούμε στην αξιολόγηση της ενεργειακής αποδοτικότητας και όχι της οικολογικής αποδοτικότητας. Θα παρουσιάσουμε τεχνικές τόσο για τον υπολογισμό όσο και την πρόβλεψη της αποδοτικότητας στα διάφορα επίπεδα του Υπολογιστικού Νέφους. Στον παρακάτω πίνακα παρουσιάζονται τα σύμβολα και οι μεταβλητές που ορίζονται και χρησιμοποιούνται στις υπόλοιπες ενότητες.

Symbol	***Meaning***
Eco_{Nodei}	Eco-efficiency assessment of node i
$\widehat{Eco}_{Node_i}$	Forecasted eco-efficiency value of node i
$Eco_{Infrastructure}$	Eco-efficiency assessment of the whole IaaS provider infrastructure
$\widehat{Eco}_{Infrastructure}$	Forecasted eco-efficiency value of the whole IaaS provider infrastructure
Eco_{VMj}	Eco-efficiency assessment of VM j
$\widehat{Eco}_{VM_j}$	Forecasted eco-efficiency of VM j
$Eco_{Servicek}$	Eco-efficiency assessment of service k
$\widehat{Eco}_{Service_k}$	Forecasted eco-efficiency of service k
P_{Nodei}	Performance of node i in Computing Units (CU)
P_{VMj}	Maximum performance in CU required by VM j
$U_{VMj,i}$	CPU consumption of VM j in node i
$\widehat{U}_{VM_{j,i}}$	CPU consumption estimation of VM j in node i
Domain - 0	Privileged domain of Xen Hypervisor
U_{0i}	CPU consumption of Domain-0 of node i
$\widehat{U}_{0_i}$	CPU consumption estimation of Domain-0 of node i
R_i	Real power consumption of node i
$\widehat{R}_i$	Forecasted real power consumption of node i
PUE	Power Usage Effectiveness
n	Number of nodes of the whole Iaas provider infrastructure
n_i	Number of VMs being executed in node i
m_k	Number of VMs of service k

Πίνακας 14: Ορισμός μεταβλητών ενεργειακής απόδοσης και αξιολόγησης

5.3.3.5 Ενεργειακή απόδοση σε επίπεδο κόμβου

Η ενεργειακή αποδοτικότητα αναφέρεται στην αναλογία μεταξύ της διενεργηθείσας χρήσιμης εργασίας και της ενέργειας που καταναλώνεται για να ολοκληρωθεί αυτή. Μετά από αυτόν τον καθορισμό, ο στόχος μας είναι να ορίσουμε έναν τύπο που αξιολογεί αυτήν την σχέση στο επίπεδο φυσικών κόμβων.

Από την άποψη IaaS, η χρήσιμη εργασία που εκτελείται από έναν κεντρικό υπολογιστή που φιλοξενεί N αριθμό VMs αντιστοιχεί στο σύνολο των χρήσιμων διαδικασιών που οι CPUs του έχουν εκτελέσει για αυτές. Υπό αυτές τις συνθήκες, οι πιθανές μετρικές θα μπορούσαν να είναι MFLOPS ή ECU. Στο υπόλοιπο της διατριβής, θα αναφερόμαστε σε αυτές τις μονάδες σχετικές με την απόδοση P_{Node} που επιτυγχάνεται από την CPU ενός κόμβου, ως υπολογιστικές μονάδες (*Computing Units - CU*).

Πολλαπλασιάζοντας τον αριθμό των CUs του επεξεργαστή (τη μέγιστη δύναμη υπολογισμού που μπορεί να παραδώσει), που φιλοξενεί τα προαναφερθέντα VMs, με το άθροισμα της χρησιμοποίησης της CPU από το σύνολο των εικονικών μηχανών, μπορούμε να λάβουμε το ποσό της υπολογιστικής δύναμης που παραδίδεται στα VM. Η χρησιμοποίηση της προνομιούχου περιοχής (Domain0 στην ονοματολογία του Xen Hypervisor) δεν έχει συμπεριληφθεί σε αυτό τον υπολογισμό, δεδομένου ότι δεν θεωρείται χρήσιμη εργασία. Στην πραγματικότητα, μπορεί να θεωρηθεί ως γενικά έξοδα που απαιτούνται για να φιλοξενήσουν αυτές τις εικονικές μηχανές.

Αφ' ενός, η ενέργεια που καταναλώνεται από τον κόμβο για να φιλοξενήσει αυτές τις εικονικές μηχανές μαζί με την προνομιούχο περιοχή αντιστοιχεί στην πραγματική ενέργεια και όχι στη φαινόμενη ενέργεια, δεδομένου ότι η πραγματική κατανάλωση απεικονίζει την ενέργεια που καταναλώνεται συνολικά και που χρεώνεται από τις συσκευές [100]. Λαμβάνουμε υπόψη επίσης τη ενέργεια που απαιτείται για να ψυχθεί ο κόμβος. Κατόπιν, η συνολική ενέργεια που καταναλώνεται θα είναι η πραγματική κατανάλωση του κεντρικού υπολογιστή που πολλαπλασιάζεται με το PUE του κέντρου δεδομένων. Βάση των προηγουμένων, η ενεργειακή αποδοτικότητα ενός κόμβου από την άποψη IaaS μπορεί να

υπολογιστεί χρησιμοποιώντας την εξίσωση (1). Σημειώνεται ότι οι τιμές χρησιμοποίησης της CPU κυμαίνονται από 0 έως 100 ανεξάρτητα από τον αριθμό επεξεργαστών στον κόμβο. Επομένως, η $U_{VMi,j}$ αναφέρεται στο συνολικό ποσό χρησιμοποίησης της CPU που καταναλώνεται από ολόκληρο τον κεντρικό υπολογιστή για ένα δεδομένο χρονικό διάστημα, και όχι από έναν μεμονωμένο πυρήνα του.

$$Eco_{Node_i} = \frac{\left(\sum_j U_{VM_{j,i}}\right) \cdot P_{Node_i}}{R_i \cdot PUE} \qquad (1)$$

Ένας τρόπος για να προβλέψουμε την ενεργειακή αποδοτικότητα των κόμβων για έναν δεδομένο χρόνο στο μέλλον θα ήταν να παρακολουθήσουμε τα προηγούμενα αποτελέσματα αξιολόγησης και από αυτά να εξάγουμε την τιμή χρησιμοποιώντας την τεχνική της γραμμικής παλινδρόμησης. Εντούτοις, αυτή η προσέγγιση δεν λαμβάνει υπόψη ότι οι εικονικές μηχανές που φιλοξενούνται σε έναν συγκεκριμένο κόμβο μπορούν να έχουν απολύτως ανεξάρτητες συμπεριφορές. Προκειμένου να αντιμετωπιστεί αυτό το πρόβλημα, υπολογίζουμε την πρόβλεψη κάθε ένα από τους όρους της εξίσωσης 1 και προβλέπουμε την ενεργειακή αποδοτικότητα χρησιμοποιώντας την ίδια σχέση. Αφ' ενός, η χρησιμοποίηση της CPU κάθε VM και του Domain-0 μπορούν να προβλεφθούν χρησιμοποιώντας γραμμική παλινδρόμηση για έναν δεδομένο χρόνο στο μέλλον. Αφ' εταίρου, η κατανάλωση ισχύος των κόμβων μπορεί να υπολογιστεί χρησιμοποιώντας έναν ενεργειακό χαρακτηρισμό του κόμβου, όπως στο [101], και την αξιολόγησή του για το προβλεπόμενο φορτίο CPU. Η υπολογιστική απόδοση P_{Node} του κόμβου παραμένει σταθερή, δεδομένου ότι είναι σταθερό χαρακτηριστικό του κόμβου, καθώς επίσης και το PUE, το οποίο είναι σταθερό στο κέντρο δεδομένων (*datacenter*). Τέλος, η προβλεφθείσα ενεργειακή αποδοτικότητα των κόμβων μπορεί να υπολογιστεί χρησιμοποιώντας την εξίσωση (2). Αυτή η προσέγγιση συνυπολογίζει τη διαφορετική συμπεριφορά των VM που τρέχουν σε έναν συγκεκριμένο κόμβο.

$$\widehat{Eco}_{Node_i} = \frac{\left(\sum_j \hat{U}_{VM_{j,i}}\right) \cdot P_{Node_i}}{\hat{R}_i \cdot PUE} \qquad (2)$$

Να σημειωθεί ότι αυτός ο ορισμός ισχύει επίσης σε ένα σενάριο όπου τα νέα VM εγκαθίστανται ή μεταφέρονται σε έναν συγκεκριμένο κόμβο. Είναι απλά απαραίτητο να υπολογιστεί ποιά θα είναι η χρησιμοποίηση της CPU που υφίσταται από αυτά τα VM στον κόμβο. Σε περίπτωση που ένα νέο VM εγκαθίσταται, μια απαίτηση απόδοσης (σε CUs) πρέπει να διευκρινιστεί για εκείνο το VM. Κατόπιν, η μέγιστη χρησιμοποίηση CPU που υφίσταται από αυτό το VM_j στον κόμβο *i* όπου εγκαθίσταται θα είναι όπως στην εξίσωση (3).

$$\widehat{U}_{VM_{j,i(deployment)}} = \frac{P_{VM_j}}{P_{Node_i}} \qquad (3)$$

Εάν μια εικονική μηχανή (*VM*) μεταφέρεται σε ένα συγκεκριμένο κόμβο, η προβλεπόμενη χρησιμοποίηση CPU από τον αρχικό κόμβο (*source*) χρειάζεται να προσαρμοστεί όπως παρουσιάζεται στην εξίσωση (4), έτσι ώστε να υπολογιστεί η τιμή που αναλογεί στον κόμβο προορισμού (*destination*).

$$\widehat{U}_{VM_{j,destination}} = \left(\frac{P_{Node_{source}}}{P_{Node_{destination}}}\right) \cdot U_{VM_{j,source}} \qquad (4)$$

5.3.3.6 Ενεργειακή απόδοση σε επίπεδο υποδομής

Από την σκοπιά του IaaS, η ενεργειακή αποδοτικότητα της συνολικής υποδομής μπορεί υπολογιστεί ως η μέση ενεργειακή αποδοτικότητα όλων των κόμβων που την συνιστούν, όπως παρουσιάζεται στην εξίσωση (5).

$$Eco_{Infrastructure} = \frac{\sum_i Eco_{Node_j}}{n} \qquad (5)$$

Παρομοίως, η πρόβλεψη της ενεργειακής αποδοτικότητας της υποδομής θα είναι η μέση τιμή της πρόβλεψης της ενεργειακής αποδοτικότητας όλων των κόμβων, όπως φαίνεται στην εξίσωση (6).

$$\widehat{Eco}_{Infrastructure} = \frac{\sum_i \widehat{Eco}_{Node_j}}{n} \qquad (6)$$

5.3.3.7 Ενεργειακή απόδοση σε επίπεδο εικονικών μηχανών

Ακολουθώντας τον ορισμό της ενεργειακής απόδοσης που παρουσιάστηκε στις προηγούμενες ενότητες, η αποδοτικότητα μιας εικονικής μηχανής η οποία εκτελείται στον κόμβο i ($Node_i$) ορίζεται σύμφωνα με την εξίσωση (7).

$$Eco_{VM_j} = \frac{U_{VM_{j,i}} \cdot P_{Node_i}}{R_i \cdot \frac{U_{VM_{j,i}} + \frac{U_0}{n}}{U_0 + \sum_i U_{VM_i}} \cdot PUE} \qquad (7)$$

Σε αυτή την εξίσωση, το ποσό χρήσιμης εργασίας που εκτελείται από ένα VM στο επίπεδο του IaaS αντιστοιχεί στο μέρος ($U_{VMj,i}$) της μέγιστης απόδοσης του κόμβου που η εικονική μηχανή εκτελείται (P_{Nodei}). Προκειμένου να καθοριστεί η κατά προσέγγιση κατανάλωση ισχύος που υφίσταται από το VM, το ανάλογο μέρος του της συνολικής κατανάλωσης ισχύος R_i παράγεται βάση της χρησιμοποίησης CPU, συμπεριλαμβανομένων των γενικών εξόδων. Όπως παρουσιάζεται και στο [100], θεωρείται μια γραμμική σχέση μεταξύ της κατανάλωσης της πραγματικής ισχύος και της χρησιμοποίησης της CPU των κόμβων. Δεδομένου ότι η χρησιμοποίηση CPU του Domain-0 απαιτείται για να λειτουργήσουν όλα τα n_i VMs που φιλοξενούνται στον κόμβο i, θεωρείται ως γενική κατανάλωση που διανέμεται ομοιόμορφα μεταξύ τους.

Όπως παρουσιάστηκε σε προηγούμενη ενότητα, η πρόβλεψη της ενεργειακής αποδοτικότητας μιας εικονικής μηχανής μπορεί να υπολογιστεί προσεγγίζοντας τις παραμέτρους της εξίσωσης 7, και έτσι να καταλήξουμε στην εξίσωση 8. Η χρησιμοποίηση της CPU μπορεί να εξαχθεί με γραμμική παλινδρόμηση, καθώς η ενεργειακή κατανάλωση

μπορεί να ληφθεί μέσω μοντελοποίησης του κάθε κόμβου για μια δοθείσα κατανάλωση CPU. Οι τιμές P_{Nodei} και PUE κληρονομούνται από τις ίδιες τιμές του κόμβου και κέντρου δεδομένων αντίστοιχα.

$$\widehat{Eco}_{VM_j} = \frac{\hat{U}_{VM_{j,i}} \cdot P_{Node_i}}{\hat{R}_i \cdot \frac{\hat{U}_{VM_{j,i}} + \frac{\widehat{U_0}}{n}}{\widehat{U_0} + \sum_i \hat{U}_{VM_i}} \cdot PUE} \qquad (8)$$

5.3.3.8 Ενεργειακή απόδοση σε επίπεδο υπηρεσίας

Μια υπηρεσία είναι ένα σύνολο από δομικά συστατικά τα οποία αλληλεπιδρούν μεταξύ τους έτσι ώστε να αποτελέσουν μια συνολική εφαρμογή στον χρήστη. Συνήθως, αυτά τα συστατικά είναι εγκατεστημένα σε διαφορετικά VMs που μπορεί να εκτελούνται σε διαφορετικούς κόμβους ή ακόμη και σε διαφορετικά κέντρα δεδομένων, διαφόρων IaaS παρόχων.

Ως μια πρώτη προσέγγιση για την αξιολόγηση της ενεργειακής αποδοτικότητας μια δοθείσας υπηρεσίας, υπολογίζουμε την μέση (ενεργειακή) αποδοτικότητα όλων των εικονικών μηχανών που φιλοξενούν την υπηρεσία αυτή (Εξίσωση 9).

$$Eco_{Service_k} = \frac{\sum_{j \epsilon Service} Eco_{VM_j}}{m_k} \qquad (9)$$

Στη συνέχεια, η τιμή της πρόβλεψης υπολογίζεται με την εξίσωση 10.

$$\widehat{Eco}_{Service_k} = \frac{\sum_{j \epsilon Service} \widehat{Eco}_{VM_j}}{m_k} \qquad (10)$$

Εντούτοις, αν και αυτή η προσέγγιση είναι χρήσιμη για έναν προμηθευτή IaaS επειδή δίνει μια προσέγγιση της υπολογιστικής ισχύος που παραδίδεται στην υπηρεσία σε σχέση με τα Watt που καταναλώνονται, δεν έχει τόσο πολύ νόημα από την σκοπιά των παρόχων SaaS. Ένας πάροχος SaaS ενδιαφέρεται για την απόδοση που παραδίδεται σε σχέση με μετρικές όπως ο χρόνος απόκρισης ή τη δικτυακή απόδοση αντί των MFLOPS που μπορεί να επιτύχει.

Επομένως, για να αξιολογηθεί η ενεργειακή αποδοτικότητα μιας δεδομένης εφαρμογής στο επίπεδο SaaS, πρέπει να υπολογισθεί η ενέργεια που καταναλώνεται από τα VMs που χρησιμοποιούνται, και έπειτα να παραχθούν αποτελέσματα όπως ο ρυθμός απόδοσης του δικτύου ανά Watt που καταναλώνεται, ή το ποσοστό συνόδου ανά Watt κλπ.

Όπως σχολιάστηκε προηγουμένως, κάποιες υπηρεσίες μπορεί να απαιτούν την παρουσία ορισμένων πιστοποιήσεων ή την εκπλήρωση των νομικών περιορισμών από το IaaS. Σε περίπτωση που οποιοσδήποτε από αυτούς τους όρους δεν τηρείται από ένα IaaS, η αξία ενεργειακής αποδοτικότητας για αυτή την υπηρεσία στον συγκεκριμένο προμηθευτή IaaS πρέπει να είναι 0.

5.3.3.9 Διαχείριση ενεργειακής απόδοσης σε παρόχους Νέφους

Ο ίδιος ο παράγοντας της ενεργειακής αποδοτικότητας είναι ένα στοιχείο της διαδικασίας αξιολόγησης που συζητήσαμε προηγουμένως με τον όρο TREC. Ο κύριος στόχος είναι να βελτιστοποιηθούν οι υπηρεσίες και οι υποδομές Νεφών μέσω της επίβλεψης και αξιολόγησης όλων των παραμέτρων αυτών.

Σε γενικές γραμμές, τα αποτελέσματα της αξιολόγησης της ενεργειακής αποδοτικότητας, όπως περιγράφηκε στα ανωτέρω, είναι πολύ χρήσιμα κατά τη διάρκεια της λειτουργίας του προμηθευτή Νέφους και σε οποιαδήποτε φάση του κύκλου της ζωής υπηρεσιών. Αυτές οι αξιολογήσεις προορίζονται να καταναλωθούν από πολιτικές λήψης αποφάσεων που χρησιμοποιούνται για να διαχειριστούν τους προμηθευτές και τις υπηρεσίες, οι οποίοι λειτουργούν σύμφωνα με ενεργειακές και οικολογικές απαιτήσεις καθώς επίσης και από επιχειρησιακούς στόχους υψηλού επιπέδου.

Για να εξηγήσουμε την επείγουσα ανάγκη για ένα πλαίσιο υπηρεσιών αξιολόγησης όπως προτείνεται, απαριθμούμε τις πιθανές διαδικασίες λήψης αποφάσεων που χρειάζονται πληροφορίες ενεργειακής και οικολογικής αποδοτικότητας για να θεσπίσουν τις κατάλληλες διοικητικές ενέργειες:

1) Προμηθευτές λογισμικού Νεφών (SaaS) στα σενάρια πολλαπλών υποδομών Νεφών:

α) Κατά την εγκατάσταση υπηρεσιών σε υποδομές προμηθευτές Νεφών, επιλέξτε την οικολογικότερη επιλογή.

β) Αυτόματες μεταφορές ή/και ανακατανομή των υπηρεσιών σε περίπτωση ανεπαρκούς ενεργειακής αποδοτικότητας.

γ) Διαμόρφωση των διοικητικών τμημάτων υπηρεσιών για μεγιστοποίηση της ενεργειακής αποδοτικότητας των υπηρεσιών κατά τη διάρκεια της λειτουργίας τους.

2) Προμηθευτές υποδομής Νεφών (IaaS, PaaS) σε ιδιωτικά σενάρια Νεφών:

α) Αποδοχή ή όχι μιας νέας υπηρεσίας στην ιδιωτική υποδομή Νέφους.

β) Απόφαση μεταξύ της τοπικής ή εξωτερικής εγκατάστασης ενός νέου συνόλου VM που συνθέτουν μια υπηρεσία. Σε περίπτωση εσωτερικής εγκατάστασης, επιλογή της ενεργειακά αποδοτικότερης τοποθέτησης των VM στους διαθέσιμους κόμβους.

γ) Αποτελεσματική διαχείριση του κύκλου ζωής των VMs με τεχνικές έξυπνης διαμόρφωσης (μέγεθος, χαρακτηριστικά κλπ).

δ) Αναδιαμόρφωση της αρχικής τοποθέτηση των VM για να εξασφαλιστεί μέγιστη ενεργειακή αποδοτικότητα σε οποιοδήποτε σημείο εγκαίρως.

ε) Διαχείριση των φυσικών κόμβων, με την ενεργοποίηση/απενεργοποίησή τους ανάλογα με το φορτίο που αντιμετωπίζεται από τον προμηθευτή.

φ) Επιλογή της αποδοτικότερες τοποθέτηση δεδομένων και τη διαμόρφωση ενός κατανεμημένου συστήματος προκειμένου να αποφευχθούν οι περιττές μεταφορές πληροφορίας.

χ) Διαμόρφωση διοικητικών τμημάτων VM με το στόχο να μεγιστοποιηθεί η ενεργειακή αποδοτικότητα της λειτουργίας του προμηθευτή.

5.3.3.9.1 Οικολογική διαχείριση IaaS Providers σε σενάρια ιδιωτικών Νεφών

Σε αυτή την ενότητα περιγράφεται με λεπτομέρεια ο τρόπος με τον οποίο οι αξιολογήσεις της ενεργειακής αποδοτικότητας μπορούν να είναι χρήσιμες για έναν προμηθευτή Νεφών

IaaS κατά τη διαχείριση των ιδιωτικών τμημάτων της υποδομής και υπηρεσιών (VM) που εκτελούνται σε αυτήν. Αυτό συμπεριλαμβάνει βασικά το υποσύνολο των αποφάσεων και τις ενέργειες που απαριθμήσαμε προηγουμένως και είναι χρήσιμες για τους προμηθευτές υποδομής Νεφών (στοιχείο 2 στην προηγούμενη λίστα).

Όπως δηλώνεται στο προηγούμενο τμήμα, το εργαλείο για την ενεργειακή αποδοτικότητα μπορεί να παρέχει τις αξιολογήσεις της τρέχουσας και μελλοντικής αποδοτικότητας, και σε διαφορετικά επίπεδα ενός ιδιωτικού Νέφους (φυσικός κόμβος, εικονικός κόμβος, ολόκληρη την υποδομή κλπ). Οι τιμές αυτές που υπολογίζονται αποτελούν πολύ σημαντική πληροφορία που παρέχεται στις διοικητικές πολιτικές έτσι να βελτιώνουν συνεχώς την συνολική ενεργειακή αποδοτικότητα του προμηθευτή.

Οι προμηθευτές IaaS μπορούν να χρησιμοποιήσουν διάφορες τεχνικές για να μειώσουν την κατανάλωση ενέργειας μιας δεδομένης υποδομής, όπως οι ενεργοποίηση/απενεργοποίηση κόμβων δυναμικά ή «πάγωμα» των εικονικών μηχανών σε ένα κόμβο κ.α. Εντούτοις, όλες αυτές οι τεχνικές διαχείρισης έχουν χρησιμοποιηθεί μέχρι τώρα λαμβάνοντας υπόψη μετρικές χαμηλού επιπέδου όπως την κατανάλωση της ενέργειας [102][103]. Στην ενότητα αυτή της διατριβής εστιάζουμε σε έναν νέο τρόπο για να λυθεί το πρόβλημα της εγκατάστασης και λειτουργίας υπηρεσιών ενεργειακά αποτελεσματικά στην εικονική υποδομή (ιδιωτικό Νέφος) ενός προμηθευτή. Εν προκειμένω, το εργαλείο αξιολόγησης της ενεργειακής αποδοτικότητας παρέχει τις αξιολογήσεις που συμπεριλαμβάνουν τόσο τεχνικές όσο και επιχειρηματικές (π.χ. οικολογικές) πτυχές των δεδομένων και της λειτουργίας του συστήματος. Μεταξύ άλλων, οι αξιολογήσεις επιτρέπουν στις διοικητικές πολιτικές να εξετάσουν την εκπλήρωση ή όχι της αποδοτικότητας ενός επιθυμητού προμηθευτή, καθώς επίσης και να διαχειριστεί αποτελεσματικά τον κύκλο της ζωής των VM με τη μετατροπή της τοποθέτησης και της διάστασής τους.

5.3.3.9.2 Πρόβλημα βελτιστοποίησης

Το πρόβλημα της διαχείρισης παρόχων IaaS σε σενάρια ιδιωτικών Νεφών, λαμβάνοντας υπόψη την ενεργειακή αποδοτικότητα, μπορεί να εκφραστεί ως πρόβλημα βελτιστοποίησης.

Η εγκατάσταση, η απεγκατάσταση, η μεταφορά, η ταξινόμηση και η σταθεροποίηση των VM, καθώς επίσης και η εκκίνηση/τερματισμός των φυσικών κόμβων είναι πιθανές διοικητικές ενέργειες για να επιτευχθεί αυτή η βελτιστοποίηση. Αυτοί οι κινήσεις επιτρέπουν την αυτοματοποιημένη διαμόρφωση ενός συνόλου VM που εκτελούνται σε μια εικονικοποιημένη υποδομή. Να σημειωθεί ότι πραγματικά αντιπροσωπεύουν τη βάση ενός υποσυνόλου διαδικασιών λήψης αποφάσεων που παρουσιάστηκαν προηγουμένως.

Symbol	***Meaning***
H	Set of physical nodes
n	Total number of in-house physical nodes
h_i	i^{th} physical node
VM	Set of service components (VMs) running in the infrastructure
m	Total number of VMs
vm_j	j^{th} VM
S	Subsets of elements $j \in VM$
s_k	k^{th} subset of VMs
x_{ki}	Decision variable indicating whether or not subset s_k is operating in node i
a_{kj}	Variable indicating whether or not vm_j belongs to subset s_k
Eco_{nodeki}	Eco-efficiency of node i if subset s_k is operating there

Πίνακας 15: Παράμετροι και μεταβλητές που χρησιμοποιήθηκαν στο πρόβλημα βελτιστοποίησης

Ο Πίνακας 15 παρουσιάζει τις παραμέτρους που χρησιμοποιήθηκαν στο πρόβλημα βελτιστοποίησης. Λαμβάνονται υπόψη ένα πεπερασμένο πλήθος κόμβων (φυσικών και εικονικών), οι οποίοι αποτελούν τις οντότητες που πρέπει να διαχειριστούν από τις πολιτικές.

Το πρόβλημα ορίζεται ως εξής:

$$Max\ Eco_{infrastructure} = Max\ \frac{\sum_{i=1}^{n}\sum_{k=1}^{2^m} Eco_{node_{ki}} \cdot x_{ki}}{n} \qquad (11)$$

Ισχύουν τα:

$$\forall\, i \in \{1 \dots n\}, \forall\, j \in \{1 \dots m\}, \forall\, k \in \{1 \dots 2^m\}: x_{ki} \in \{0,1\}, a_{kj} \in \{0,1\} \qquad (12)$$

$$\forall\, j \in \{1 \dots m\}: \sum_{i=1}^{n}\sum_{k=1}^{2m} x_{ki} \cdot a_{kj} = 1 \qquad (13)$$

$$\forall\, i \in \{1 \dots n\}: \sum_{k=1}^{2^m} x_{ki} = 1 \qquad (14)$$

Να σημειωθεί ότι οι μεταβλητές x_{ki} και a_{kj} μπορούν μόνο να έχουν τιμή ίση με ένα ή μηδέν (εξίσωση 12), κάθε VM πρέπει να εγκαθίσταται σε έναν κόμβο (εξίσωση 13), και μπορεί να υπάρξει μόνο ένα υποσύνολο k που εγκαθίσταται ανά κόμβο (εξίσωση 14). Η χρονική πολυπλοκότητα αυτού του προβλήματος βελτιστοποίησης είναι $O(2^m)$. Ως εκ τούτου, η επίλυση τέτοιου προβλήματος απαιτεί ευρυστικούς ή «άπληστους» αλγορίθμους. Για παράδειγμα, κάποιος μπορεί να περιορίσει τον αριθμό μεταφορών VM σε κάθε επανάληψη της βελτιστοποίησης, η οποία μειώνει πολύ το συνολικό αριθμό πιθανών εγκαταστάσεων ($\sum_i |D(h_i)|$). Εντούτοις, αυτό το είδος ευρυστικής επίλυσης δεν εξετάζεται στην διατριβή αυτή.

5.3.3.9.3 Οικολογική πολιτική διαχείρισης

Σε αυτή την ενότητα σκιαγραφείται μια διοικητική πολιτική που χρησιμοποιεί τις αξιολογήσεις και προβλέψεις της ενεργειακής αποδοτικότητας για να επιτρέψει την αποτελεσματική και οικολογική διαχείριση μιας ιδιωτικής υποδομής Νέφους και των οντοτήτων αυτής.

Ο στόχος μας είναι να παρουσιάσουμε μια πραγματική περίπτωση μιας απλής διοικητικής πολιτικής που βασίζει τις αποφάσεις της σχετικά με τις αξιολογήσεις και τις προβλέψεις της αποδοτικότητας. Η πολιτική μας εξετάζει μια απλοποίηση του περιγεγραμμένου προβλήματος βελτιστοποίησης. Οι στόχοι αυτής της πολιτικής είναι (1) να προσαρμόσουν τον αριθμό ενεργών κόμβων σύμφωνα με το φορτίο που αντιμετωπίζεται από τον προμηθευτή και (2) για να βελτιστοποιήσει την αρχική τοποθέτηση των VM πάνω στους ενεργούς κόμβους. Πρέπει να σημειωθεί ότι αυτή η πολιτική αντιπροσωπεύει ένα παράδειγμα για το πώς δύο διαδικασίες λήψης αποφάσεων (από αυτές που παρουσιάστηκαν προηγουμένως), μπορούν να βοηθηθούν σημαντικά από τις αξιολογήσεις της ενεργειακής αποδοτικότητας. Συνοψίζοντας, η προτεινόμενη πολιτική (Πίνακας 16) αντιμετωπίζει την πρώτη φάση του κύκλου της ζωής των VM, δηλ. την εγκατάστασή τους. Σε αυτήν την περίπτωση, η χρονική πολυπλοκότητα είναι O(n), όπου το n είναι ο συνολικός αριθμός φυσικών κόμβων.

Require: Eco-efficiency assessment tool
Ensure: Eco-aware optimization of VMs placement and private Cloud dimension
  if new VM to be deployed **then**
    if $\widehat{Eco}_{infrastructure} >= Eco^{MIN}_{infrastructure}$ **then**
      if Not available capacity for the new VM **then**
        Turn on the node with better eco-efficiency forecast
      else if Spare infrastructure capacity **then**
        Turn off nodes with worse current eco-efficiency and redeploy VMs running on them
      end if
      for all powered-on nodes do
        Eco improvement = Forecast node eco-efficiency - assess current node eco-efficiency
      end for
      Deploy VM in the node with higher improvement in its eco-efficiency
  else
    Outsource VM to the more eco-efficient external IaaS provider
  end if
end if

Πίνακας 16: Ψευδοκώδικας οικολογικής πολιτικής

Όπως δηλώνεται πριν, οι αξιολογήσεις της ενεργειακής αποδοτικότητας, είτε οι παρούσες είτε οι μελλοντικές, συμπεριλαμβάνουν διάφορες οικολογικές πτυχές που τις καθιστούν «διαφανείς» για τις διοικητικές πολιτικές. Γενικά, οι κόμβοι που παρέχουν καλύτερη ενεργειακή αποδοτικότητα είναι αυτοί που επιλέγονται να είναι ενεργοί, ενώ εκείνοι που έχουν μια χειρότερη τιμή είναι οι καλύτεροι υποψήφιοι που απενεργοποιούνται. Με παρόμοιο τρόπο, όταν δέχεται ο προμηθευτής την τοπική εγκατάσταση ενός νέου VM, η πολιτική ζητά τις προβλέψεις της ενεργειακής αποδοτικότητας σχετικά με την πιθανή εγκατάσταση του VM σε κάθε κόμβο ενδεχομένως ικανό να προσφέρει τις απαιτήσεις των πόρων της. Επιπλέον, ζητά την τρέχουσα ενεργειακή αποδοτικότητα εκείνων των κόμβων προκειμένου να υπολογιστεί η βελτίωση της αποδοτικότητα. Αυτός με την υψηλότερη αύξηση στην αξία της ενεργειακής αποδοτικότητάς του επιλέγεται. Διαφορετικά, εάν ο πάροχος αποφασίζει να μην εγκαταστήσει το εισερχόμενο VM στην υποδομή του, η πολιτική επιλέγει τον ενεργειακά αποδοτικότερο πάροχο IaaS για να μεταφέρει το VM. Η απόφαση της αποδοχής ή όχι ενός νέου VM ρυθμίζεται σχετικά με το εάν η αποδοτικότητα του μελλοντικού παρόχου θα είναι μεγαλύτερη από την επιθυμητή ελάχιστη τιμή ($Eco^{MIN}_{infrastructure}$). Τέλος, αυτή η απλή

πολιτική διασφαλίζει την μεγαλύτερη δυνατή χρησιμοποίηση των ενεργών κόμβων, η οποία επομένως θα βελτιώσει την συνολική ενεργειακή αποδοτικότητα του παρόχου IaaS.

5.4 Αξιολόγηση

5.4.1 Πειραματική υποδομή

Για την αξιολόγηση και επαλήθευση της υλοποίησής μας δημιουργήσαμε ένα εικονικό περιβάλλον βασισμένο σε τέσσερις φυσικούς κόμβους και πολλαπλά VMs χρησιμοποιώντας Xen τεχνολογία εικονικοποίησης. Στον Πίνακας 17 παρουσιάζονται τα τεχνικά χαρακτηριστικά του περιβάλλοντος που διαμορφώσαμε για την πειραματική αξιολόγηση του μηχανισμού επίβλεψης.

Hostname	optimis1	optimis2	optimis3	optimis4
Processor Name	Intel Xeon E5630	Intel Xeon E5630	Intel Xeon E5520	Intel Xeon E5310
Count of Processors	4	4	2	1
Cores per Processor	4	4	4	4
Processor Frequency	2.53 GHz	2.53 GHz	2.26 GHz	1.60 GHz
Memory	8956 MB	8956 MB	512 MB	1468 MB
Disk	999.6 GB	999.6 GB	999.6 GB	72.7 GB
Operating System	CentOS 5.5	CentOS 5.5	CentOS 5.5	CentOS 5.6
Hypervisor	2.6.18-238.9.1.el5xen	2.6.18-238.9.1.el5xen	2.6.18-238.9.1.el5xen	2.6.18-238.9.1.el5xen

Πίνακας 17: Τεχνικά χαρακτηριστικά πειραματικής υποδομής

Στην πειραματική αυτή εφαρμογή, ο φυσικός κόμβος με το όνομα *Optimis1* φιλοξένησε δύο εικονικές μηχανές με τα παρακάτω χαρακτηριστικά:

Domain Name	centos-optimis-IP	TestECO
Count of VCPUs	1	16
Memory	1024 MB	256 MB
Operating System	CentOS 5.6	Debian 5.0.4

Πίνακας 18: Διαμόρφωση εικονικών μηχανών στον κόμβο Optimis1

5.4.2 Αποτελέσματα επίβλεψης

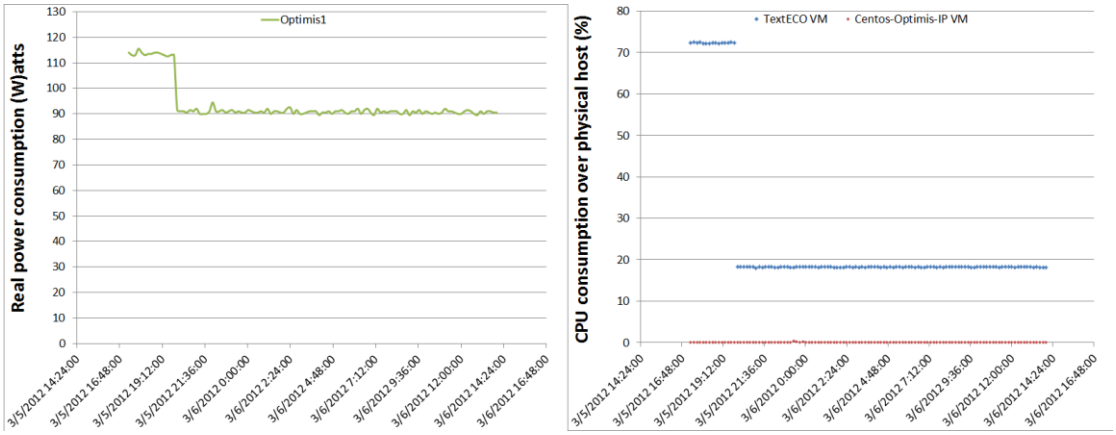
Το Σχήμα 46 παρέχει μια επισκόπηση των μετρικών που επιβλέπονται από τους διάφορους συλλέκτες που λειτουργούν στο φυσικό, το εικονικό, το στρώμα υπηρεσιών αλλά και την ενεργειακή κατανάλωση του υπολογιστικού περιβάλλοντος Νέφους (δεν παρατίθενται όλες οι μετρικές για λόγους συντόμευσης). Όπως έχει αναλυθεί προηγουμένως, διάφοροι συλλέκτες επίβλεψης πληροφοριών συγκεντρώνουν τα δεδομένα από τα προαναφερθέντα στρώματα. Όταν χρησιμοποιείται η τεχνική τραβήγματος (*pull*), κάθε συλλέκτης πληροφοριών εκτελείται σε ένα προκαθορισμένο χρονικό διάστημα που τίθεται ως παράμετρος σε έναν πίνακα διαμόρφωσης της βάσης δεδομένων επίβλεψης, και παρεμβάλλει τελικά τα συλλεχθέντα στοιχεία στη βάση δεδομένων. Οι συλλέκτες που λειτουργούν με την τεχνική ώθησης (*push*) εκτελούν επίσης έναν παρόμοιο στόχο, η μόνη σημαντική διαφορά είναι ότι το διάστημα εκτέλεσης για τη συλλογή δεδομένων δεν είναι μια παράμετρος που αποθηκεύεται σε κάποιο πίνακα της βάσης δεδομένων. Επομένως, το περιβάλλον βάσεων δεδομένων ελέγχου διαδραματίζει έναν μάλλον παθητικό ρόλο σε αυτήν την περίπτωση, και δεν μπορεί να ελέγξει τη συχνότητα της υποβολής στοιχείων. Με άλλα λόγια, η υποβολή των στοιχείων μπορεί να συμβεί σε κάθε στιγμή στον τρόπο ώθησης.

Τα δεδομένα που παρουσιάζονται στο Σχήμα 46 χρησιμοποιούνται από τους διάφορους πελάτες των διαδικασιών αξιολόγησης της υποδομής επίβλεψης που ασχολούνται για παράδειγμα με τον κίνδυνο που κάποιος διατρέχει κατά την απόφαση να εγκατασταθεί μια υπηρεσία μέσα σε ένα VM που τρέχει σε έναν δεδομένο φυσικό κόμβο. Για παράδειγμα, οι πληροφορίες σχετικά με την ιστορία των αποτυχιών υλικού ή η συχνότητα των εκ νέου επανεκκινήσεων ενός φυσικού κόμβου μέσα σε ένα δεδομένο χρονικό διάστημα μπορούν να είναι χρήσιμοι για ένα εργαλείο που τρέχει μια αξιολόγηση του κινδύνου. Τέτοιες διαδικασίες αξιολόγησης σχετικά με την εμπιστοσύνη, τον κίνδυνο, την ενεργειακή αποδοτικότητα και το κόστος που εκτελούνται κατά την επιλογή ενός παρόχου υποδομής Νέφους είναι ένα ιδιαίτερο χαρακτηριστικό γνώρισμα του προγράμματος OPTIMIS [104][105].

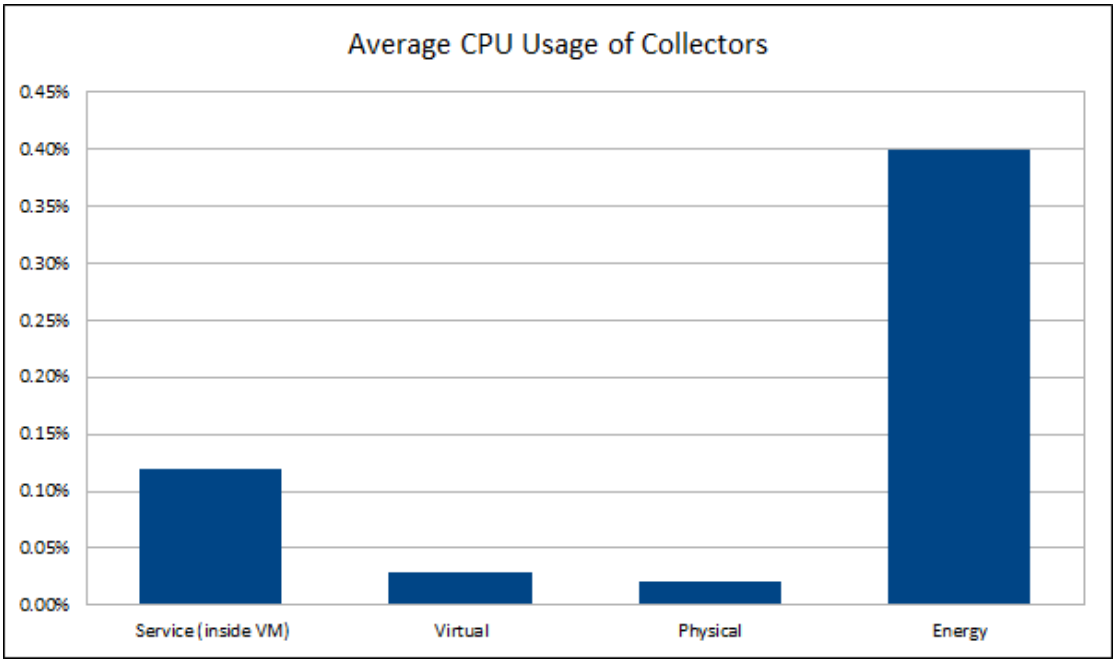
physical_resource_id	virtual_resource_id	monitoring_information_collector_id	metric_name	metric_value	metric_unit	metric_timestamp	resource_type	service_resource_id
optimis1		3	apparent_power	136	VA	2012-03-08 10:06:32	energy	
optimis1		3	real_energy	454.19	kWh	2012-03-08 10:06:32	energy	
optimis1		3	real_power	90.5	W	2012-03-08 10:06:32	energy	
optimis1		3	current	0.59	A	2012-03-08 10:06:32	energy	
optimis1		3	apparent_energy	740.53	kVAh	2012-03-08 10:06:32	energy	
optimis1		3	power_factor	66	%	2012-03-08 10:06:32	energy	
optimis1		3	virt-top_cpu	Domain-0 .38;centos-optimis-IP .04;TestECO 18.20		2012-03-08 10:06:32	energy	
optimis1		4	mac_address	BC:30:5B:CE:97:CB		2012-03-08 10:05:02	physical	
optimis1		1	count_of_users	3	users	2012-03-08 10:05:02	physical	
optimis1		4	No_of_cores	16	cores	2012-03-08 10:05:02	physical	
optimis1		1	cpu_average_load	0.000,0.010,0.000		2012-03-08 10:05:02	physical	
optimis1		4	total_memory	9170944	KB	2012-03-08 10:05:02	physical	
optimis1		1	free_memory	65448	KB	2012-03-08 10:05:02	physical	
optimis1		4	disk_total_space	903775	MB	2012-03-08 10:05:02	physical	
optimis1		4	fqdn	optimis1		2012-03-08 10:05:02	physical	
optimis1		1	disk_free_space	658399	MB	2012-03-08 10:05:02	physical	
optimis1		1	cpu_speed	2527.022	MHz	2012-03-08 10:05:02	physical	
optimis1		4	last_reboot	1317036662(14160840)		2012-03-08 10:05:02	physical	
optimis1		1	Downstream	129.024	Kbps	2012-03-08 10:05:04	physical	
optimis1		1	Upstream	113.672	Kbps	2012-03-08 10:05:04	physical	
optimis1		1	status	OK		2012-03-08 10:05:04	physical	
optimis1		5	peak_time	0	Kbps	2012-03-08 10:06:35	physical	
optimis1		6	trough_time	0	Kbps	2012-03-08 10:06:50	physical	
optimis1	5d1ce1c2-6206-4e49-bebf-710774459b8a	instance-jboss-1	ThreadCount	58		2012-03-08 09:44:53	service	DemoApp
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	disk_total	300	GB	2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	cpu_user	0.07	%	2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	machine_type	x86_64		2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	os_release	2.6.20-xen3.1		2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	mem_total	512	KB	2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	mem_used	100	%	2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	cpu_vnum	1		2012-03-08 09:37:52	virtual	
optimis1	34c46650-5b80-46e6-9bb6-767f49c694e6	2	cpu_speed	2527	MHz	2012-03-08 09:37:52	virtual	

Σχήμα 46: Αναφορά επίβλεψης πολλαπλών πηγών για τον φυσικό κόμβο Optimis1

Στη συγκεκριμένη περίπτωση σχετικά με την αξιολόγηση της ενεργειακής αποδοτικότητας, οι μετρικές όπως *real_power* (που μετρά σε Watt την κατανάλωση ενέργειας των φυσικών κόμβων) και *virt-top_cpu* (που μετρά το ποσοστό χρήσης CPU σε κάθε VM) διαδραματίζουν έναν σημαντικό ρόλο στον υπολογισμό της ενεργειακής αποδοτικότητας. Στο πείραμά μας, συλλάβαμε την ενέργεια και την κατανάλωση φορτίων VM CPU από τον κόμβο *optimis1* για μια περίοδο 24 ωρών. Ο συγκεκριμένος φυσικός κόμβος, κατά τη διάρκεια του χρονικού διαστήματος των μετρήσεων, φιλοξενεί δύο VM. Στο Σχήμα 47, επεξηγούμε την εξέλιξη της πραγματικής κατανάλωσης ενέργειας από κοινού με την διακύμανση του φορτίου CPU των φιλοξενημένων VM. Μπορούμε σαφώς να παρατηρήσουμε – όπως και αναμέναμε - τη σχέση και μεταξύ των δύο μετρικών.



Σχήμα 47: Σχέση πραγματικής ενεργειακής κατανάλωσης φυσικής υποδομής με την κατανάλωση CPU φιλοξενημένων εικονικών μηχανών



Σχήμα 48: Επίπεδο διεισδυτικότητας συλλεκτών δεδομένων επίβλεψης

Στο Σχήμα 48 παρουσιάζεται το επίπεδο διεισδυτικότητας (βάση της μέσης κατανάλωσης CPU κατά τη διάρκεια του χρόνου εκτέλεσης) για κάθε συλλέκτη της υποδομής επίβλεψης. Ο πρώτος συλλέκτης (*Service*) λειτουργεί μέσα στο εικονικό περιβάλλον και επομένως η κατανάλωση CPU που μετριέται αναφέρεται στο VM στο οποίο εγκαθίσταται. Παρατηρούμε ένα μικρό ποσοστό κατανάλωσης CPU που οδηγεί στο επιθυμητό μικρό επίπεδο διεισδυτικότητας. Οι υπόλοιποι συλλέκτες (*Virtual*, *Physical* και *Energy*) εγκαθίστανται στο φυσικό κόμβο (*Optimis1*) από όπου συλλαμβάνουν τις επιθυμητές μετρικές. Μετά από

τέσσερις ώρες λειτουργίας μετρήσαμε τη χρήση CPU κάθε συλλέκτη και υπολογίσαμε τις μέσες τιμές. Η μετρηθείσα χρησιμοποίηση CPU ήταν πολύ μικρή, αποδεικνύοντας μια υψηλή αποδοτικότητα της εφαρμογής και ένα ελάχιστος αντίκτυπο στην υποδομή. Ακόμα κι αν η αξία για τον ενεργειακό συλλέκτη ήταν υψηλότερη σύγκριση με τους υπόλοιπους, η κατανάλωση είναι πολύ μικρή κατ' απόλυτη τιμή (περίπου 0.4%) και επομένως πετυχαίνουμε στη μείωση της διεισδυτικότητας του μηχανισμού επίβλεψης τόσο στην εικονική και όσο και στη φυσική υποδομή.

Για την επικύρωση της λύσης μας μετρήσαμε επίσης το χρόνο απόκρισης του ενεργειακού συλλέκτη στα διαφορετικά σενάρια λειτουργίας. Η υπηρεσία επίβλεψης της πλατφόρμας επικαλείται τον ενεργειακό συλλέκτη προκειμένου να ανακτηθούν οι διάφορες ενεργειακές μετρικές από τους διαφορετικούς φυσικούς κόμβους. Κατά συνέπεια, είναι σημαντικό ότι ο μηχανισμός συλλογής να λειτουργεί αποτελεσματικά κάτω από ένα ταυτόχρονο φορτίο των αιτημάτων. Για αυτόν τον λόγο, εκτελέσαμε ένα πείραμα με 1, 2, 6 και 10 αιτήματα πελατών με διαφορετικές συνθήκες ταυτοχρονισμού (Πίνακας 19). Με βάση τις μετρήσεις, ένα μεμονωμένο αίτημα παίρνει περίπου 0.5 δευτερόλεπτα να επεξεργαστεί και να επιστρέψει τις σχετικές με την ενέργεια μετρικές. Κατά αύξηση του αριθμού αιτημάτων καθώς επίσης και του ταυτοχρονισμού, παρατηρούμε ότι ο μηχανισμός συλλογής συνεχίζει να αποδίδει αποτελεσματικά με την υποδοχή των παράλληλων αιτημάτων. Η λύση μας στην πραγματικότητα δεν θα εκτελέσει 10 αιτήματα δεδομένου ότι συλλέγουμε μόνο 6 σχετικές με την ενέργεια παραμέτρους (βλ. Πίνακας 19). Κατά συνέπεια, η ρεαλιστική απόδοση του εφαρμοσμένου συστήματος επίβλεψης, από την άποψη συλλογή ενεργειακών παραμέτρων, είναι περίπου 3 δευτερόλεπτα. Από μια πρώτη εντύπωση, κάποιος θα μπορούσε να πει ότι η απόδοση δεν είναι ικανοποιητική για ένα σύστημα ελέγχου, αλλά εάν θεωρούμε ότι οι συγκεκριμένες παράμετροι (*real_power, apparent_power*, κ.λπ.) δεν αλλάζουν τόσο γρήγορα, το χρονικό πλαίσιο λειτουργίας μερικών δευτερολέπτων επιτρέπει στα διοικητικά τμήματα για να προχωρήσει στις διορθωτικές ενέργειες σε πραγματικό χρόνο.

Number of requests	*Number of concurrent requests*	*Response time (msec)*
1	1	521.4
2	1	3027.398
2	2	944.3
6	1	14104
6	3	6395
6	6	3214.77
10	1	9083
10	10	7547.36

Πίνακας 19: Χρόνος απόκρισης συλλέκτη ενεργειακών δεδομένων

5.4.3 Αποτελέσματα αξιολόγησης και πρόβλεψης ενεργειακής αποδοτικότητας

Μετρήσαμε την ικανότητα υπολογισμού P_{Node} των κόμβων χρησιμοποιώντας το εργαλείο μέτρησης επιδόσεων *Double-Precision Whetstone* του UnixBench [106], η οποία μετρά την ταχύτητα και την αποδοτικότητα διαδικασιών *floating-point* μετρώντας την αριθμητική ακέραιων και κινητής υποδιαστολής αριθμών. Αυτή η δοκιμή πραγματοποιήθηκε τρεις φορές σε όλους τους πυρήνες κάθε κόμβου, εκτελώντας ένα αντίγραφο της δοκιμής *Double-Precision Whetstone* σε κάθε πυρήνα ταυτόχρονα. Τα αποτελέσματα καθώς επίσης και οι υπολογισμένες μέσες τιμές για κάθε κόμβο παρουσιάζονται στον Πίνακας 20. Μπορεί να παρατηρηθεί ότι τα επιτευχθέντα αποτελέσματα είναι σχεδόν αμετάβλητα και επομένως αξιόπιστα. Επιπλέον, είναι συνεπή με τα χαρακτηριστικά υλικού κάθε κόμβου.

Hostname	optimis1	optimis2	optimis3	optimis4
Processor Name	Intel Xeon E5630	Intel Xeon E5630	Intel Xeon E5520	Intel Xeon E5310
Count of Processors	4	4	2	1
Cores per Processor	4	4	4	4
Processor Frequency	2.53 GHz	2.53 GHz	2.26 GHz	1.60 GHz
Whetstone (Round 1)	7198.2	7180.8	3379.7	1287.2
Whetstone (Round 2)	7178.7	7133.6	3394.3	1287.6
Whetstone (Round 3)	7194.6	7113.7	3415.7	1283.5
Whetstone (mean)	**7190.5**	**7142.7**	**3396.6**	**1286.1**

Πίνακας 20: Αποτελέσματα αξιολόγησης *Double-Precision Whetstone*

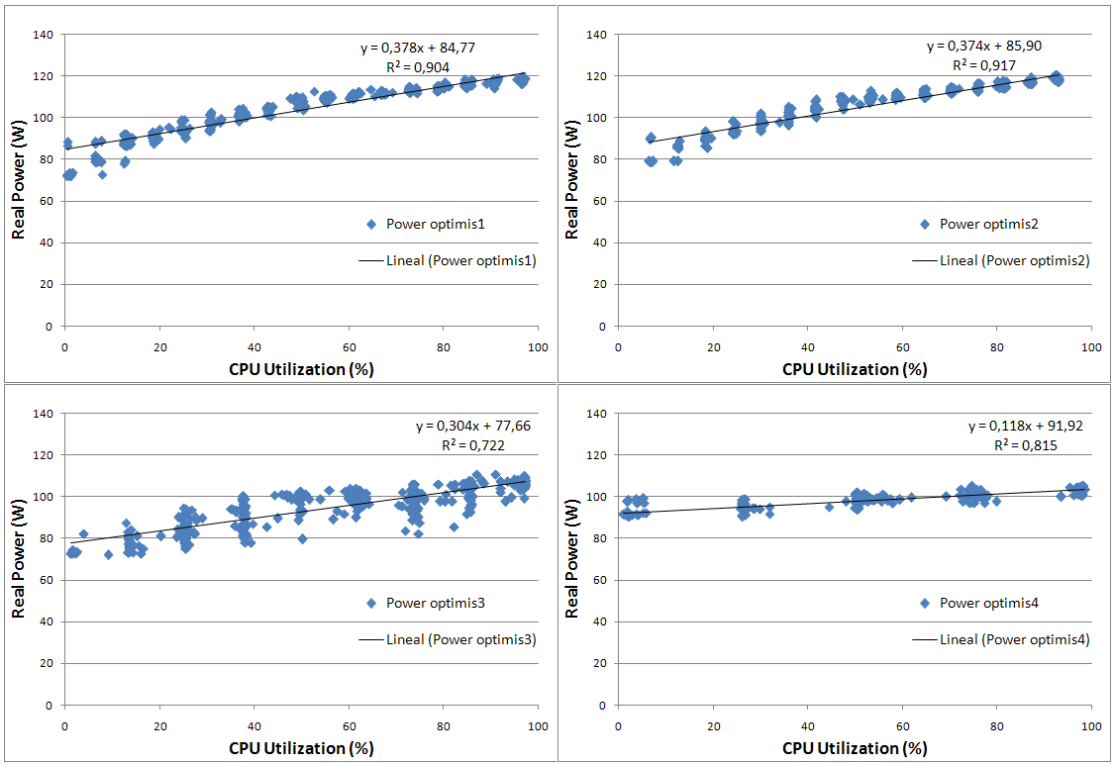
Προκειμένου να ληφθούν οι αξιολογήσεις της ενεργειακής αποδοτικότητας στους διαφορετικούς κόμβους, τέσσερα VMs δημιουργήθηκαν, ένα σε κάθε κόμβο, με τα

χαρακτηριστικά που περιγράφηκαν στον Πίνακας 21. Αυτές οι εικονικές μηχανές εκτέλεσαν ένα κώδικα που χρησιμοποίησε τη γεννήτρια φόρτου εργασίας [107] για να αυξήσει τη χρησιμοποίηση CPU του VM από 0% σε 100% με βήμα $\frac{100}{VCPU_{number}}$. Επομένως, 16 διαφορετικές χρησιμοποιήσεις CPU υπέστησαν στην TestECO, 16 στην TestECO2, 8 στην TestECO3 και 4 στην TestECO4. Συνεπώς, το ίδιο ποσό βημάτων χρησιμοποίησης CPU παρατηρήθηκε στους φυσικούς κόμβους.

Domain Name	TestECO	TestECO2	TestECO3	TestECO4
Hosting Node	optimis1	optimis2	optimis3	optimis4
Count of VCPUs	16	16	8	4
Memory	256 MB	256 MB	256 MB	256 MB
Operating System	Debian 5.0.4	Debian 5.0.4	Debian 5.0.4	Debian 5.0.4

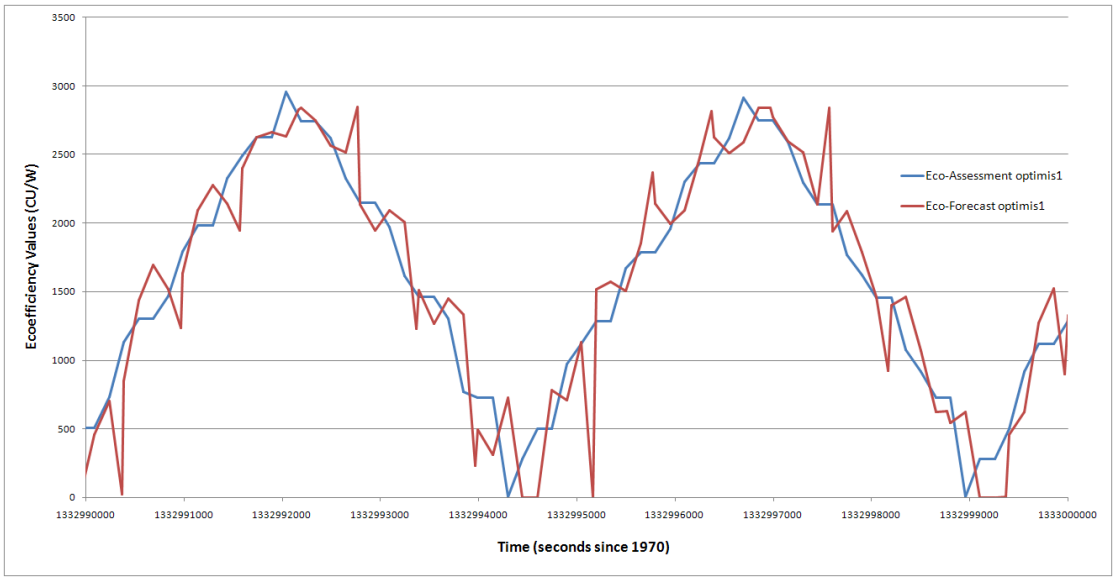
Πίνακας 21: Εικονικές μηχανές πειραματικής υποδομής

Αυτοί οι μεταβλητοί φόρτοι εργασίας μας επέτρεψαν να εκτελέσουν πολλαπλές μετρήσεις της πραγματικής κατανάλωσης ενέργειας (*real power*) για διαφορετικές χρησιμοποιήσεις CPU για όλους τους κόμβους. Όπως στο [101], τα αποτελέσματα, που παρουσιάζονται στο Σχήμα 49, επιβεβαιώνουν ότι η ενέργεια που καταναλώνεται από έναν κόμβο ακολουθεί μια γραμμική σχέση όσον αφορά τη χρησιμοποίηση CPU. Εντούτοις, μπορεί να παρατηρηθεί ότι σε όλες τις περιπτώσεις υπάρχει μια μικρή μεταβλητότητα (υψηλότερη στο optimis3) της ενέργειας που καταναλώνεται για ένα δεδομένο φορτίο CPU. Αυτό οφείλεται στην κατανάλωση ισχύος άλλων συστατικών, όπως η μνήμη, οι συσκευές δικτύων ή τους σκληρούς δίσκους, όπως περιγράφεται στο [80]. Παρόλα αυτά, το αποκτηθέν γραμμικό μοντέλο είναι αρκετά ακριβές για τους στόχους μας, δεδομένου ότι θα εξετάζουμε τις χρησιμοποιήσεις CPU όχι κοντά στο 0 % κατά την διαδικασία αξιολόγησης και τοποθέτησης των εικονικών μηχανών. Με αυτά τα αποτελέσματα είμαστε σε θέση να χτίσουμε ένα ενεργειακό μοντέλο για κάθε κόμβο της μορφής $\widehat{R} = a \cdot U_{CPU_i} + b$, με το οποίο θα υπολογίσουμε τις μελλοντικές καταναλώσεις ισχύος για ένα δεδομένο φορτίο CPU.



Σχήμα 49: Ενεργειακή κατανάλωση με μεταβλητή χρησιμοποίηση CPU

Αφότου ολοκληρώθηκαν αυτοί οι χαρακτηρισμοί των κόμβων, οι αξιολογήσεις και οι προβλέψεις της ενεργειακής αποδοτικότητας λήφθηκαν για όλους τους κόμβους. Συγκεκριμένα, το Σχήμα 50 παρουσιάζει τις αξιολογήσεις της ενεργειακής αποδοτικότητας και τις προβλεφθείσες τιμές τους στις 5:00 στις 7:46 της 29ης Μαρτίου 2012 για τον κόμβο optimis1, με μια αξία PUE 2. Οι αξιολογήσεις της ενεργειακής αποδοτικότητας εκτελέσθηκαν κάθε 2.5 λεπτά. Συγχρόνως, η προβλεπόμενη αξία για την επόμενη αξιολόγηση υπολογίζεται και παρουσιάζεται στον αντίστοιχο μελλοντικό χρόνο της. Η χρησιμοποίηση CPU ήταν μεταβλητή όπως εξηγείται προηγουμένως. Οι προβλέψεις χρησιμοποίησης CPU εκτελέσθηκαν για όλα τα VMs που εκτελούνται στον κόμβο, καθώς επίσης και για το Domain-0. Αυτές οι προβλέψεις εξήχθησαν χρησιμοποιώντας γραμμική παλινδρόμηση από τις τελευταίες 3 αξιολογήσεις της ενεργειακής αποδοτικότητας. Τα αποτελέσματα στο Σχήμα 50 δείχνουν ότι το σύστημα προβλέπει σωστά τις τιμές τις ενεργειακής αποδοτικότητας μέσα σε ένα ανεκτό περιθώριο λάθους.

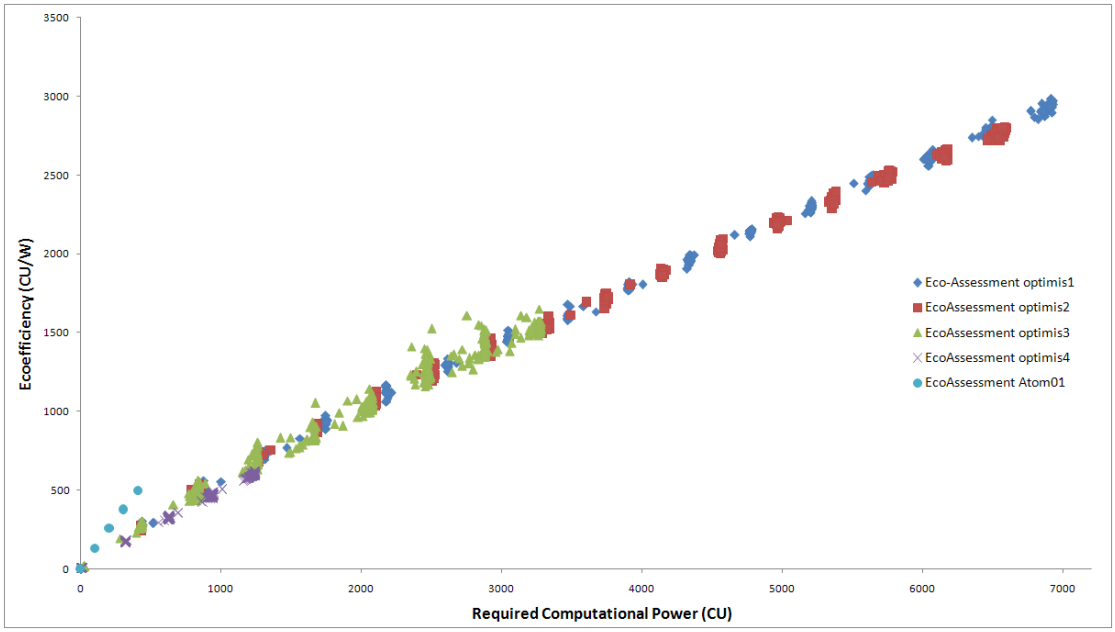


Σχήμα 50: Ενεργειακή αποδοτικότητα και προβλεφθήσες τιμές για τον κόμβο optimis1

Τελικά προχωρήσαμε στην αξιολόγηση του ποιος φυσικός κόμβος θα παρείχε τις καλύτερες τιμές ενεργειακής αποδοτικότητας για έναν δεδομένο φόρτο εργασίας, που εκφράζεται σε υπολογίστηκες μονάδες (CUs) που απαιτούνται. Εκτός από τους κόμβους που παρουσιάστηκαν στην αρχή αυτού του τμήματος, ένας πρόσθετος κόμβος χρησιμοποιήθηκε προκειμένου να είναι σε θέση να συγκρίνουμε διαφορετικές αρχιτεκτονικών εκτός από Intel Xeon. Επιλέχθηκε ένας χαμηλής κατανάλωσης ισχύος κόμβος, Atom του οποίου τα χαρακτηριστικά μπορούν να βρεθούν στον Πίνακας 22. Εφαρμόστηκαν διαφορετικά ποσοστά χρησιμοποίησης CPU (χρησιμοποιώντας το εργαλείο Stress), όπως περιγράφεται προηγουμένως και συλλέχθηκαν οι τιμές της ενεργειακής αποδοτικότητάς για αυτές τις χρησιμοποιήσεις. Τα επιτευχθέντα αποτελέσματα που συγκρίνουν την ενεργειακή αποδοτικότητα των διαφορετικών κόμβων για έναν δεδομένο φόρτο εργασίας παρουσιάζονται στο Σχήμα 51. Μπορεί να παρατηρηθεί ότι όλοι οι κόμβοι της διάταξής μας παρέχουν μια παρόμοια αποδοτικότητα για έναν δεδομένο φόρτο εργασίας. Εντούτοις, ο κόμβος Atom παρουσιάζει σαφώς καλύτερη ενεργειακή αποδοτικότητα για έναν δεδομένο φόρτο εργασίας. Αυτό το αποτέλεσμα σημαίνει ότι μερικές εργασίες χωρίς ιδιαίτερες απαιτήσεις υπολογισμού θα μπορούσαν να κατατεθούν σε αυτό το είδος κόμβων προκειμένου να ληφθεί μια καλύτερη τιμή ενεργειακής αποδοτικότητας και επομένως να μειωθεί το ποσό ενέργειας που καταναλώνεται, προσφέροντας όμως μια καλή υπηρεσία.

Hostname	atom01
Processor Name	Intel Atom 330
Count of Processors	4
Cores per Processor	2
Processor Frequency	1.60 GHz
Whetstone (Round 1)	409.1
Whetstone (Round 2)	409.7
Whetstone (Round 3)	409.3
Whetstone (mean)	**409.4**

Πίνακας 22: Αποτελέσματα αξιολόγησης *Double-Precision Whetstone για τον Atom01*



Σχήμα 51: Ενεργειακή αποδοτικότητα των κόμβων για μεταβλητό φόρτο εργασίας

5.4.4 Αποτελέσματα διαχείρισης ενεργειακής αποδοτικότητας

Έχοντας ως στόχο να εξηγήσουμε τη δυνατότητα εφαρμογής μιας οικολογικής διοικητικής πολιτικής (*eco-driven, ED*) που παρουσιάστηκε προηγουμένως, εξετάζουμε τώρα έναν πάροχο IaaS με ένα ιδιωτικό Νέφος που απαρτίζεται από 40 φυσικούς κόμβους, 10 κάθε τύπου που εκτίθεται λεπτομερώς προηγουμένως στον Πίνακας 17. Συγκεκριμένα, πρόκειται να αξιολογήσουμε τη μεγιστοποίηση της ενεργειακής αποδοτικότητας της υποδομής που επιτυγχάνεται από αυτήν την πολιτική (*ED*) σε σχέση με μία άλλη που βασίζεται στο κέρδος (*profit-driven, PD*).

Η διαδικασία λήψης αποφάσεων της πολιτικής *PD* καθοδηγείται κυρίως από τις λειτουργικές δαπάνες, όπως αυτές που προκαλούνται από την ενέργεια που καταναλώνεται. Με την παραλαβή μιας νέας υπηρεσίας προς εγκατάσταση, αυτή η πολιτική αξιολογεί το γενικό κόστος όλων των επιλογών εγκατάστασης, συμπεριλαμβανομένων εκείνων που οδηγούν στην μεταφορά σε άλλο πάροχο μιας τέτοιας υπηρεσίας. Το ίδιο είδος αξιολόγησης γίνεται εάν η πολιτική ανιχνεύει ότι μερικοί κόμβοι πρέπει να ενεργοποιηθούν ή απενεργοποιηθούν. Στο τέλος, οι κόμβοι που αντιστοιχούν στη μεγαλύτερη αποδοτικότητα είναι αυτοί που επιλέγονται να φιλοξενήσουν τις VMs.

Συνολικά 617 VMs που εκτελούν εφαρμογές συναλλαγών εγκαταστάθηκαν κατά τη διάρκεια του πειράματος (μια εβδομάδα): 285 με διάρκεια μιας εβδομάδα και 332 με διάρκεια μισή ημέρα που φθάνουν σε διαφορετικούς χρόνους λόγω έτσι ώστε να προσομοιωθούν περίοδοι μέγιστης ζήτησης. Ο χρησιμοποιούμενος φόρτος εργασίας λήφθηκε από ανώνυμο ευρωπαϊκό ISP (που συλλέχθηκε κατά τη διάρκεια του 2009), το οποίο αποτελείται από αιτήματα σε διάφορες διαδικτυακές υπηρεσίες.

Οι εικονικές μηχανές μπορούν να ταξινομηθούν σε τρεις διαφορετικούς τύπους, σύμφωνα με την ικανότητα υπολογισμού. Συγκεκριμένα, ορίζουμε χρυσό (*gold*), ασημένιο (*silver*), και χάλκινο (*bronze*), που απαιτούν 500, 400, και 300 μονάδες υπολογισμού (CUs), αντίστοιχα. Ο Πίνακας 23 παρουσιάζει την ενεργειακή αποδοτικότητα του παρόχου όπως αυτή υπολογίζεται από τις πολιτικές *ED* και *PD*.

Time slot (hours)	**0**	**12**	**24**	**36**	**48**	**60**	**72**
VMs running	285	305	285	305	285	305	285
Gold VMs	170	170	170	170	170	170	170
Silver VMs	115	115	115	115	115	115	115
Bronze VMs	0	20	0	20	0	20	0
CUs required	131000	137000	131000	137000	131000	137000	131000
Local VMs (PD)	285	305	285	305	285	305	285
Local VMs (ED)	285	305	285	305	285	305	285
Outsourced VMs (PD)	0	0	0	0	0	0	0
Outsourced VMs (ED)	0	0	0	0	0	0	0
Eco_optimis1	29812,7	29812,7	29812,7	29812,7	29812,7	29812,7	29812,7
Eco_optimis2	23192,9	25547,7	23192,9	25547,7	23192,9	25547,7	23192,9
Eco_optimis3 (PD)	-	-	-	-	-	-	-

Eco_optimis3 (ED)	-	-	-	-	-	-	-
Eco_optimis4	-	-	-	-	-	-	-
Eco_infrastructure(PD)	26502,7	27680,1	26502,7	27680,1	26502,7	27680,1	26502,7
Eco_infrastructure(ED)	26502,7	27680,1	26502,7	27680,1	26502,7	27680,1	26502,7

Time slot (hours)	**84**	**96**	**108**	**120**	**132**	**144**	**156**
VMs running	305	305	330	327	335	330	335
Gold VMs	170	170	170	170	170	170	170
Silver VMs	115	115	115	115	115	115	115
Bronze VMs	20	20	45	42	50	45	50
CUs required	137000	137000	144500	143600	146000	144500	146000
Local VMs (PD)	305	305	330	328	335	330	335
Local VMs (ED)	305	305	326	326	326	326	326
Outsourced VMs (PD)	0	0	0	0	0	0	0
Outsourced VMs (ED)	0	0	4	1	9	4	9
Eco_optimis1	29812,7	29812,7	29812,7	29812,7	29812,7	29812,7	29812,7
Eco_optimis2	25547,7	25547,7	28032,8	28032,8	28032,8	28032,8	28032,8
Eco_optimis3 (PD)	-	-	553,8	127,0	1265,0	553,8	1265,0
Eco_optimis3 (ED)	-	-	-	-	-	-	-
Eco_optimis4	-	-	-	-	-	-	-
Eco_infrastructure(PD)	27680,1	27680,1	19466,4	19324,1	19703,5	19466,4	19703,5
Eco_infrastructure(ED)	27680,1	27680,1	28922,7	28922,7	28922,7	28922,7	28922,7

Πίνακας 23: Ενεργειακή αποδοτικότητα του παρόχου για τις πολιτκές PD και ED

Όπως αναφέρθηκε προηγουμένως, η οικολογική πολιτική (*ED*) είναι ενήμερη για μία ελάχιστη τιμή ενεργειακής αποδοτικότητας που πρέπει να επιτευχθεί. Σε αυτό το πείραμα, η μεταβλητή $Eco_{infrastructure}^{MIN}$ ισούται με 2000 CU/W.

Όπως μπορεί να φανεί στα αποτελέσματα, αυτό το γεγονός κάνει τη οικολογική πολιτική να μεταφέρει μερικά VMs στους εξωτερικούς προμηθευτές (από τη 108η ώρα). Ο λόγος είναι ότι η ενεργειακή αποδοτικότητα της υποδομής θα ήταν κάτω από το ελάχιστο όριο που απαιτείται εάν οι πρόσθετοι φυσικοί κόμβοι με τη χαμηλή χρησιμοποίηση ενεργοποιούνταν. Αυτό συμβαίνει στην πραγματικότητα με την κερδοσκοπικού χαρακτήρα πολιτική, η οποία προτιμά να ενεργοποιήσει μερικούς κόμβους και να εγκαταστήσει τα VMs επειδή το κόστος είναι δευτερεύον από αυτό που πληρώνεται εάν μεταφερθούν στους εξωτερικούς προμηθευτές IaaS. Γενικά, η πολιτική των *ED* επιλέγει την κατανομή VMs που ικανοποιεί τις απαιτήσεις σε CUs και υπονοεί την υψηλότερη αξία ενεργειακής

αποδοτικότητας, ενώ η *PD* εξασφαλίζει επίσης τις απαιτήσεις σε CUs των VMs αλλά επιδιώκει την καλύτερη επιλογή όσον αφορά τις οικονομικές παραμέτρους. Όσον αφορά τους ενεργούς και ανενεργούς κόμβους, είναι ξεκάθαρο ότι η πολιτική *ED* χρησιμοποιεί μόνο τους μισούς κόμβους του ιδιωτικού Νέφους, ενώ η *PD* ενεργοποιεί σε μερικές περιπτώσεις το κόμβο *optimis3* έτσι ώστε να εξυπηρετήσει κάποια αιτήματα.

Εν γένει, ο πάροχος επιτυγχάνει μια μέση ενεργειακή αποδοτικότητα της υποδομής 24434.0 CU/W και 27787.5 CU/W κατά χρησιμοποίηση της κερδοσκοπικού χαρακτήρα (*PD*) και οικολογικής πολιτικής (*ED*), αντίστοιχα. Αυτό αντιπροσωπεύει μια αύξηση της ενεργειακής αποδοτικότητας της υποδομής του παρόχου κατά περίπου 14%, η οποία επιβεβαιώνει τα πλεονεκτήματα του οικολογικού πλαισίου που παρουσιάζεται σε αυτή τη διατριβή.

5.4.5 Αρχικά συμπεράσματα

Στο κεφάλαιο αυτό παρουσιάστηκε μια συνολική λύση επίβλεψης δεδομένων σε περιβάλλον Νέφους. Ιδιαίτερη έμφαση δόθηκε στην συλλογή στοιχείων από πολλαπλές πηγές αλλά και στον υπολογισμό και αξιολόγηση της ενεργειακής απόδοσης της υποδομής με στόχο την καλύτερη διαχείριση των πόρων. Μέσω της διαδικασίας επικύρωσης της λύσης μέσω μιας πειραματικής υποδομής εξαγάγαμε διάφορα πολύτιμα συμπεράσματα. Όσον αφορά την διαδικασία επίβλεψης, η αξιολόγηση του πλαισίου απέδειξε μια αποδοτική λειτουργία τόσο σε σχέση με την υπολογιστική αποδοτικότητα του συστήματος όσο και με τον χρόνο απόκρισης του μηχανισμού συλλογής. Και οι τέσσερις συλλέκτες δεδομένων εκτέλεσαν αποτελεσματικά τη συγκέντρωση των πληροφοριών και την αλληλεπίδραση με την υπηρεσία επίβλεψης της πλατφόρμας. Στα μελλοντικά βήματά μας, είναι η βελτιστοποίηση της αποθήκευσης πληροφοριών με την έρευνα των κατανεμημένων τεχνικών αποθήκευσης δεδομένων.

Η ενότητα της ενεργειακής αξιολόγησής υπέθεσε ένα γραμμικό μοντέλο για την πραγματική ενέργεια που καταναλώθηκε όσον αφορά τη χρησιμοποίηση CPU. Παρά την

ύπαρξη ικανοποιητικών αποτελεσμάτων για τα πειράματά μας, στοχεύουμε να εξετάσουμε άλλες πτυχές που έχουν επιπτώσεις στην κατανάλωση ισχύος προκειμένου να αυξηθεί η ακρίβεια, όπως η μνήμη, το δίκτυο, ή οι συσκευές σκληρών δίσκων. Επίσης παρατηρήσαμε ότι οι προβλέψεις γραμμικής παλινδρόμησης παρείχαν μια καλή προσέγγιση των μελλοντικών χρησιμοποιήσεων CPU που υπέστησαν από τα VM ενός κόμβου. Εντούτοις, υπήρξαν μερικές αποκλίσεις στις προβλέψεις της ενεργειακής αποδοτικότητας. Προκειμένου να λυθεί αυτό, καλύτερες μέθοδοι πρόβλεψης θα μπορούσαν να υιοθετηθούν, όπως η χρησιμοποίηση μεθόδων πρόβλεψης χρονικής σειράς.

Επιπλέον, παρουσιάστηκε, ως παράδειγμα μιας οικολογικής διαχείρισης στους προμηθευτές Νεφών, μια ενεργειακά οικολογική πολιτική που στόχευσε να μεγιστοποιήσει την ενεργειακή αποδοτικότητα ενός ιδιωτικού Νέφους. Χρησιμοποίησε τις αξιολογήσεις που παράχθηκαν από το πλαίσιο ως δεδομένα εισόδου στη διαδικασία λήψης αποφάσεών του. Αυτή η πολιτική πέτυχε μια μεγιστοποίηση της ενεργειακής αποδοτικότητας της υποδομής του προμηθευτή κατά 14 % όσον αφορά αυτό που επιτεύχθηκε από μια κερδοσκοπικού χαρακτήρα πολιτική.

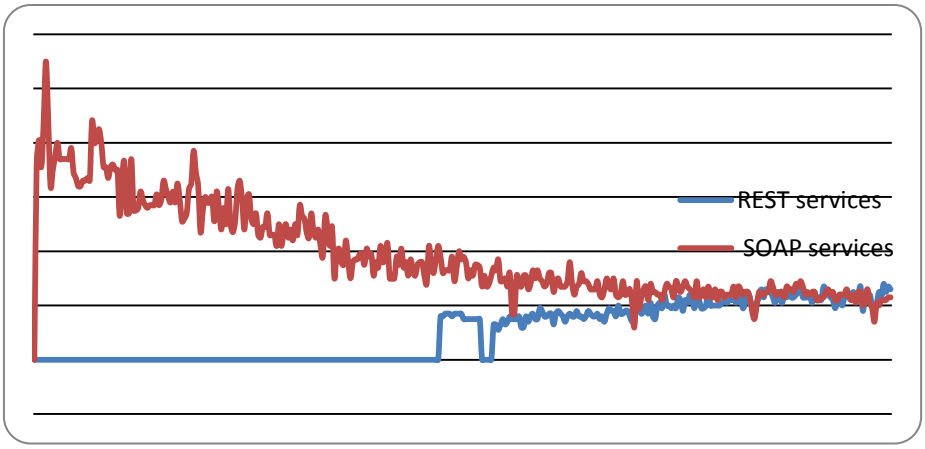
6 *Επιπρόσθετες συνεισφορές στην επίβλεψη και διαχείριση υποδομών*

Στο κεφάλαιο αυτό παρουσιάζονται δυο συνεισφορές της διατριβής αυτής στην κοινότητα ανοιχτού λογισμικού που σχετίζονται με την επίβλεψη και διαχείριση υποδομών. Συγκεκριμένα, αναλύεται ο σχεδιασμός και η υλοποίηση μιας επέκτασης του συστήματος επίβλεψης Nagios μέσω της οποίας γίνεται εφικτή η παρακολούθηση πόρων χρησιμοποιώντας ένα υπηρεσιοστρεφές πλαίσιο λειτουργίας μέσω της τεχνολογίας REST. Στη συνέχεια, παρουσιάζεται η εγκατάσταση και η λειτουργία ενός συστήματος ελαστικής φιλοξενίας διαδικτύου σε ιδιωτικά Νέφη με χρήση αποκλειστικά τεχνολογιών ανοιχτού λογισμικού. Στην προτεινόμενη αρχιτεκτονική δίνεται έμφαση στην διαχείριση των εικονικών πόρων μέσω της χρήσης ενός μηχανισμού επίβλεψης και εξισορρόπησης φορτίου εργασίας.

6.1 Υπηρεσιοστρεφές μηχανισμός επίβλεψης με REST και Nagios

Όπως αναλύθηκε σε προηγούμενα κεφάλαια αυτής της διατριβής, η επίβλεψη τόσο των πόρων μιας υπολογιστικής υποδομής όσο και των εφαρμογών που εκτελούνται σε αυτήν αποτελεί αναπόσπαστο μέρος των σύγχρονων συστημάτων με κύριο στόχο την διασφάλιση της ποιότητας των παρεχόμενων υπηρεσιών. Ένα αποδοτικό σύστημα παρακολούθησης και επίβλεψης θα πρέπει να ευέλικτο όσον αφορά τις παραμέτρους τις οποίες επιβλέπει αλλά και επίσης και με τις τεχνολογίες και τα συστήματα που χρησιμοποιεί για να πραγματοποιήσει την επίβλεψη.

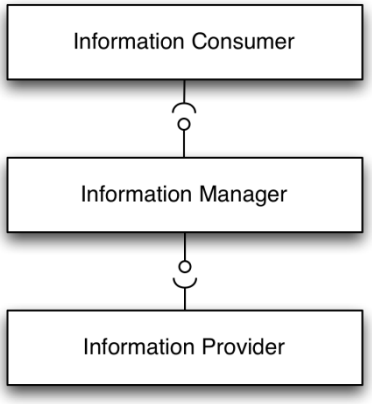
Η προτεινόμενη λύση αντιμετωπίζει τις προκλήσεις επίβλεψης Υπολογιστικών Νεφών μέσω μιας ευέλικτης και επεκτάσιμης σχεδίασης βασισμένη σε υπηρεσίες. Συνδυάζει δύο πολύ διαδεδομένες τεχνολογίες επίβλεψης πόρων (Nagios) και υλοποίησης υπηρεσιών (REST). Οι *RESTful* υπηρεσίες αποτελούν μια όλο και αυξανόμενη τάση προγραμματισμού, με αρκετά πλεονεκτήματα σε σχέση με το παραδοσιακό πρωτόκολλο SOAP. Η ευκολία υλοποίησης τόσο της πλευράς του πελάτη αλλά και της πλευράς της εφαρμογής του εξυπηρετητή, το μικρό επιπρόσθετο φορτίο εκτέλεσης (σε σχέση με τις υπηρεσίες SOAP) και τα αποδοτικά χαρακτηριστικά ελαστικότητας (εξαιτίας της απουσίας κατάστασης) είναι μερικά από τα δυνατά σημεία του τεχνολογικού προτύπου REST.



Σχήμα 52: REST vs SOAP: συχνότητα αναζήτησης από το 2004 μέχρι το 2010.
[*Google Trends: REST services, SOAP services. Google Inc., http://www.google.de/trends?q=rest+services,+soap+services&ctab=0&geo=all&date=all&sort=0 (accessed February 21, 2011)]*

6.1.1 Προτεινόμενη αρχιτεκτονική

Το σύστημα επίβλεψης που παρουσιάζεται σε αυτό το υποκεφάλαιο ακολουθεί τις αρχές σχεδιασμού της προσανατολισμένης προς τις υπηρεσίες αρχιτεκτονικής (SOA), η οποία περιλαμβάνει τους βασικούς ρόλους του καταναλωτή (*consumer*) και του παραγωγού (*provider*) της πληροφορίας. Κάθε σύστημα υπολογιστών, πόσο μάλλον στα κατανεμημένα συστήματα και στης προσανατολισμένες προς τις υπηρεσίες υποδομές (SOIs), υπάρχουν πληροφορίες που παράγονται σε μια τοποθεσία που πρέπει να χρησιμοποιηθούν σε μια άλλη θέση, ενδεχομένως σε διαφορετικά στρώματα. Σε αυτό το πλαίσιο, ένας παραγωγός πληροφοριών (*Information Provider*) θα μπορούσε να είναι μια υπηρεσία ή μια εφαρμογή που εγκαθίστανται σε ένα περιβάλλον εκτέλεσης, ένας διαχειριστής εικονικής υποδομής ή ακόμα και η ίδια φυσική υποδομή ενός περιβάλλοντος Υπολογιστικού Νέφους. Ο καταναλωτής της πληροφορίας (*Information Consumer*) αντιπροσωπεύει εκείνες τις οντότητες (άνθρωποι ή άλλα τμήματα υπηρεσιών) που χρησιμοποιούν τα στοιχεία για τη λήψη ορισμένων αποφάσεων. Επιπλέον, έχουμε εισαγάγει ένα ενδιάμεσο διοικητικό επίπεδο (*Information Manager*), μέσω του οποίου εκτελείται η συνάθροιση των πληροφοριών από τις διάφορες πηγές καθώς επίσης και η αποθήκευση εκείνων των στοιχείων σε μια βάση δεδομένων για μελλοντική χρήση (ανάσυρση δεδομένων, στατιστική κ.λπ.).



Σχήμα 53: Ρόλοι και διεπαφές

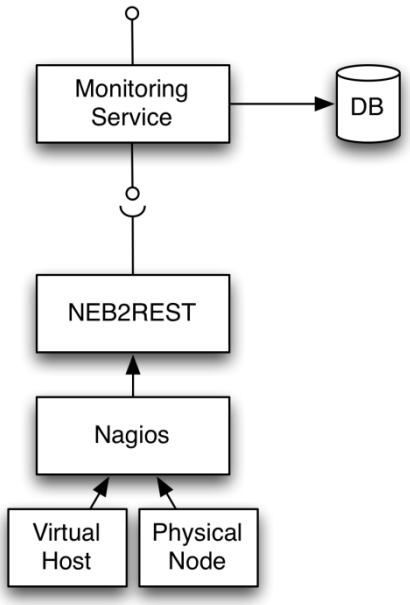
Η προτεινόμενη αρχιτεκτονική πραγματοποιεί ένα διπλό πρότυπο ώθησης & εξαγωγής (*push & pull*): οι πληροφορίες επίβλεψης ωθούνται προς το διοικητικό στρώμα ενώ κάθε καταναλωτής μπορεί να τραβήξει τα κομμάτια των στοιχείων από το διευθυντή

πληροφοριών κατ' απαίτηση. Αυτή η αρχιτεκτονική αρχή παρέχει την απαραίτητη ελαστικότητα στη συλλογή των πληροφοριών, αποφεύγοντας τις περιττές υπερφορτώσεις, ενώ συγχρόνως η κατανάλωση εκείνων των πληροφοριών δεν έχει επιπτώσεις καθόλου στην υποδομή που την παρέχει.

Λεπτομερέστερα, η λύση που σχεδιάσαμε και εφαρμόσαμε εξυπηρετεί και τα τρία προαναφερθέντα στρώματα. Εκμεταλλευόμενοι τα χαρακτηριστικά της *RESTful* ανάπτυξης υπηρεσιών, σχεδιάσαμε και υλοποιήσαμε την υπηρεσία παρακολούθησης (*Monitoring Service*) βάσει αυτού του προτύπου. Το συστατικό αυτό είναι εγκατεστημένο στο διοικητικό στρώμα μαζί με την βάση δεδομένων και οποιοδήποτε άλλο συστατικό του πλαισίου υπηρεσιών (πλατφόρμας). Ο ρόλος του παραγωγού πληροφοριών εκπληρώνεται στην περίπτωσή μας από την υποδομή ενός παρόχου Νέφους, τόσο την φυσική όσο και την εικονική. Για αυτόν τον λόγο, έχουμε χρησιμοποιήσει το Nagios, ένα ευρέως γνωστό σύστημα ελέγχου, για τη συλλογή των πληροφοριών των πόρων από τους κόμβους μιας υποδομής, είτε εικονικής είτε φυσικής. Μέσω του Nagios ο διαχειριστής της υποδομής μπορεί να καθορίσει τους ελέγχους [37] μέσω των οποίων μπορεί να αποκτήσει τη κατάσταση των φυσικών πόρων. Για την εξαγωγή των πληροφορών από τη φυσική υποδομή και την αλληλεπίδραση με την υπηρεσία παρακολούθησης, έχουμε αναπτύξει το συστατικό **NEB2REST**. Το δομικό συστατικό αυτό λειτουργεί σαν ενδιάμεσος συνδετικός βρόγχος μεταξύ του χαμηλού επιπέδου συστήματος συλλογής πληροφορίας του Nagios, και της υπηρεσίας επίβλεψης στην πλατφόρμα (Monitoring Service). Κάθε φορά που εκτελείται ένας έλεγχος από το Nagios, το NEB2REST καλεί την υπηρεσία (μέσω της διεπαφής ώθησής του) και στέλνει το σύνολο των δεδομένων.

Η προτεινόμενη προσέγγιση καλύπτει τις δύο σημαντικές απαιτήσεις επίβλεψης ενός παρόχου Υπολογιστικού Νέφους: παρατήρηση της κατάστασης των φυσικών πόρων, έλεγχο της λειτουργίας του εικονικού περιβάλλοντος. Ο υπηρεσιοστρεφής σχεδιασμός της προτεινόμενης λύσης επιτρέπει την επέκταση του συστήματος και τη συλλογή των οριζόμενων από εφαρμογή μετρικών. Με τη χρησιμοποίηση ελεγκτικών διεπαφών και πρακτόρων, που θα μπορούσαν να συνοδεύσουν την εφαρμογή στην εικονική μηχανή, θα

μπορούσαμε να χρησιμοποιήσουμε την διεπαφή της υπηρεσίας επίβλεψης και να ωθήσουμε τα υψηλού επιπέδου οριζόμενα από εφαρμογή στοιχεία στο διοικητικό στρώμα. Μέσω αυτού του μηχανισμού, είναι εφικτές οι διαδικασίες αξιολόγησης του SLA ή τη διαχείριση των πόρων συνολικά.



Σχήμα 54: Αρχιτεκτονική μηχανισμού επίβλεψης (RESTful Nagios)

6.1.1.1 Nagios, NEB2REST and libvirt.

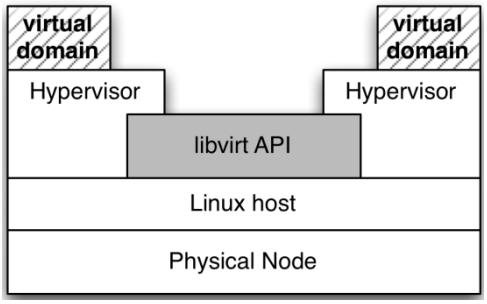
Όπως παρουσιάσαμε σε προηγούμενο κεφάλαιο, το Nagios είναι ένα σύστημα επίβλεψης ανοιχτού κώδικα, το οποίο επιτρέπει την παρακολούθηση και επίβλεψη των πόρων μιας υποδομής μέσω ελέγχων κατάστασης. Από λειτουργική άποψη, η αρχιτεκτονική Nagios είναι χωρισμένη σε δύο στρώματα: ένα στρώμα λογικής ελέγχου και ένα στρώμα λειτουργίας ελέγχου. Το βασικό API του συστήματος περιλαμβάνει ήδη κάποια plug-ins για τον έλεγχο των βασικών πόρων ενώ επιπρόσθετα plug-ins μπορούν να αναπτυχθούν σύμφωνα με τις οδηγίες για το API στο [38].

Ακόμα κι αν το Nagios, σε συνδυασμό με κάποια από τα δημοφιλή plug-ins [108][109][110], παρέχει ένα πλήρες πλαίσιο για τη συλλογή μετρικών, δεν προσφέρει έναν προσανατολισμένο προς τις υπηρεσίες μηχανισμό επίβλεψης. Για αυτόν τον λόγο, σχεδιάσαμε και υλοποιήσαμε το συστατικό **NEB2REST** το οποίο τοποθετείται μεταξύ του

συστήματος Nagios και της *RESTful* υπηρεσίας Ιστού που βρίσκεται στρώμα διαχείρισης. Η υλοποίηση αυτού του συστατικού είναι βασισμένη στο *Nagios Event Broker API* (NEB) [111] που προσφέρει τη δυνατότητα του χειρισμού των γεγονότων που παράγονται από Nagios και συγκεκριμένα σχετικών με: *Host and service checks, Notifications, Flapping states, Downtime starting/ending* κλπ.

Κατά συνέπεια, το NEB2REST είναι ένα συστατικό που γράφεται σε C γλώσσα προγραμματισμού, που «καταναλώνει» τα γεγονότα που παράγονται από το NEB API για κάθε έλεγχο υπηρεσιών που πραγματοποιείται από το Nagios. Για κάθε τέτοιο έλεγχο, λοιπόν, εξάγει όλες τις απαραίτητες πληροφορίες για έναν πόρο ελέγχου (Σχήμα 56) και καλεί την υπηρεσία επίβλεψης μέσω μιας POST μεθόδου HTTP. Η αλληλεπίδραση NEB2REST με την υπηρεσία πραγματοποιείται μέσω του cURL και του libcurl APIs[112]. Το Libcurl είναι μια βιβλιοθήκη προγραμμάτων πελάτη που υποστηρίζει, μεταξύ πολλών άλλων, τα πρωτόκολλα HTTP και HTTPS. Είναι ελεύθερο λογισμικό και παρέχει ένα C API μεταξύ άλλων γλωσσών προγραμματισμού. Χρησιμοποιώντας αυτήν την βιβλιοθήκη και με τη χρήση της POST μεθόδου κλήσης υλοποιήσαμε την πλευρά-πελάτη του συστατικού NEB2REST που αλληλεπιδρά με την υπηρεσία επίβλεψης.

Για την πραγματοποίηση της επίβλεψης της εικονικής υποδομής, χρησιμοποιήσαμε το plug-in Nagios libvirt. Το Libvirt [91] είναι ένα API ανοιχτού κώδικα που χρησιμοποιείται για τη διαχείριση πλατφορμών εικονικοποίησης. Υποστηρίζει έναν μεγάλο αριθμό μεσολογισμικών εικονικοποίησης όπως: Xen, QEMU, KVM, LXC, Virtual Box, VMware κ.α. Επιπλέον, υπάρχουν διάφορες εφαρμογές και εργαλεία που έχουν αναπτυχθεί πάνω από το libvirt API που υλοποιούν διαφορετικά χαρακτηριστικά γνωρίσματα διαχείρισης εικονικής υποδομής. Το Nagios- virt είναι μια επέκταση (plug-in) που επιτρέπει στο σύστημα Nagios να επιβλέψει τις εικονικές περιοχές μιας υποδομής. Εκμεταλλεύεται όλες τις ικανότητες του πυρήνα libvirt API με αποτέλεσμα να είναι σε θέση να παρακολουθεί μια μεγάλη ποικιλία εικονικών περιβαλλόντων. Όπως όλα τα plug-ins του Nagios, είναι εύκολα διαμορφώσιμο μέσω του Nagios Core API.



Σχήμα 55: Διεπαφή εικονικού περιβάλλοντος libvirt API

6.1.1.2 Η RESTful προσέγγιση της υπηρεσίας επίβλεψης(Monitoring Service)

Όπως αναφέρθηκε προηγουμένως, η υλοποίηση της υπηρεσίας επίβλεψης πραγματοποιήθηκε βάσει του REST αρχιτεκτονικού παραδείγματος. Οι πόροι (*resources*) επίβλεψης που ορίστηκαν για αυτόν τον λόγο περιγράφονται από το παρακάτω σχήμα XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="MonitoringResource">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="physical_host_id"/>
    <xs:element ref="virtual_host_id"/>
    <xs:element ref="timestamp"/>
    <xs:element ref="name"/>
    <xs:element ref="value"/>
    <xs:element ref="description"/>
    <xs:element ref="timeinterval"/>
    <xs:element ref="isPhysicalHost"/>
    <xs:element ref="isVirtualHost"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="physical_host_id" type="xs:ID" nillable="true"/>
 <xs:element name="virtual_host_id" type="xs:ID" nillable="true"/>
 <xs:element name="timestamp" type="xs:dateTime"/>
 <xs:element name="name" type="xs:NCName"/>
 <xs:element name="value" type="xs:NCName"/>
 <xs:element name="description" type="xs:string"/>
 <xs:element name="timeinterval" type="xs:long"/>
 <xs:element name="isPhysicalHost" type="xs:boolean"/>
 <xs:element name="isVirtualHost" type="xs:boolean"/>
</xs:schema>
```

Σχήμα 56: Σχήμα πληροφορίας πόρου επίβλεψης

Οι διεπαφές χρήσης μια *RESTful* υπηρεσίας ορίζονται από τις διευθύνσεις (URIs) των πόρων που υποστηρίζει η υπηρεσία αυτή. Στην περίπτωση της υπηρεσίας επίβλεψης, που παρουσιάζουμε, έχουμε τις εξής διεπαφές:

/monitoring-service/information-providers: με την χρήση της μεθόδου GET, αυτή η διεπαφή επιστρέφει μια λίστα με πληροφορίες για όλους του παρόχους. Η POST μέθοδος επιτρέπει την δημιουργία ενός νέου πόρου για ένα πάροχο πληροφορίας (information provider).

/monitoring-service/information-providers/{id}: με την χρήση της μεθόδου GET, αυτή η διεπαφή επιστρέφει μια αναπαράσταση της πληροφορίας για τον πάροχο πληροφοριών με το αναγνωριστικό *id*. Η PUT μέθοδος ανανεώνει τα δεδομένα του πόρου, ενώ η DELETE διαγράφει τον πόρο.

/monitoring-service/physical-hosts: με την χρήση της μεθόδου GET, αυτή η διεπαφή επιστρέφει μια λίστα με όλους τους διαθέσιμους φυσικούς πόρους. Η POST μέθοδος επιτρέπει την δημιουργία ενός καινούριου πόρου.

/monitoring-service/physical-hosts/{id}: με την χρήση της μεθόδου GET, αυτή η διεπαφή επιστρέφει μια αναπαράσταση ενός πόρου της φυσικής υποδομής με αναγνωριστικό *id*. Η PUT μέθοδος ανανεώνει τα δεδομένα του πόρου, ενώ η DELETE διαγράφει τον πόρο.

/monitoring-service/virtual-hosts: Η λειτουργία αυτή είναι πανομοιότυπη με την */monitoring-service/physical-hosts* αλλά εφαρμόζεται σε εικονικούς πόρους.

/monitoring-service/virtual-hosts/{id}: Η λειτουργία αυτή είναι πανομοιότυπη με την */monitoring-service/physical-hosts/{id}* αλλά εφαρμόζεται σε εικονικούς πόρους.

/monitoring-service/monitoring-resources: με την χρήση της μεθόδου GET, αυτή η διεπαφή επιστρέφει μια λίστα με όλους τους διαθέσιμους πόρους επίβλεψης Using the GET method, this URI returns a list of all monitoring resources. Η POST μέθοδος επιτρέπει την δημιουργία ενός καινούριου πόρου.

/monitoring-service/monitoring-resources/{id}: Με τον καθορισμό του αναγνωριστικού id και μέσο της μεθόδου GET, η διεπαφή αυτή επιστρέφει μια αναπαράσταση του

συγκεκριμένου πόρου επίβλεψης. Με την DELETE μέθοδο μπορούμε να διαγράψουμε τον αντίστοιχο πόρο.

/monitoring-service/monitoring-resources/physical-hosts/{id}: Με την χρήση της GET μεθόδου σε αυτήν την διεπαφή μπορούμε να «φιλτράρουμε» τους αποθηκευμένους πόρους επίβλεψης και να λάβουμε μια λίστα με αυτούς που σχετίζονται με το φυσικό πόρο με τον δοσμένο αναγνωριστικό *id*. Η DELETE μέθοδος διαγράφει την ίδια λίστα πόρων.

/monitoring-service/monitoring-resources/virtual-hosts/{id}: η λειτουργία αυτής της διεπαφής είναι πανομοιότυπη με την προηγούμενη αλλά εφαρμόζει «φιλτράρισμα» βασισμένο στο εικονικό πόρο.

Είναι σαφές ότι θα μπορούσαμε να καθορίσουμε περισσότερες διεπαφές και URIs που θα αντιπροσωπεύουν διαφορετικές όψεις των πόρων βασισμένων στις σχέσεις που περιγράφονται από το μοντέλο δεδομένων που παρουσιάστηκε, αλλά δεδομένου ότι επικεντρωνόμαστε στη διαχείριση των πόρων επίβλεψης απέχουμε από να περιπλέξουμε το API για τώρα. Καθώς οι υπηρεσίες REST δεν αποθηκεύουν την κατάσταση των πόρων, η καταγραφή των πόρων επίβλεψης στο σύστημά μας πραγματοποιείται μέσω μιας βάσης δεδομένων που υιοθετεί το μοντέλο δεδομένων που σχεδιάσαμε και αποθηκεύει τις λαμβανόμενες πληροφορίες καθώς επίσης και τις σχέσεις μεταξύ των διαφορετικών οντοτήτων. Η βάση δεδομένων χρησιμεύει επίσης ως μια αποθήκευση των ιστορικών στοιχείων που μπορούν να χρησιμοποιηθούν για την επεξεργασία από άλλα συστατικά.

Τα αρχικά συμπεράσματα συνοψίζονται στα παρακάτω σημεία:

- Η προτεινόμενη υποδομή επίβλεψης είναι ευέλικτη και προσαρμόσιμη σε διαφορετικές υλοποιήσεις περιβαλλόντων Νεφών. Η χρήση του libvirt API επιτρέπει την αλληλεπίδραση με μεγάλο αριθμό διαφορετικών μεσολογισμικών Νέφους (hypervisors). Από την άλλη, η δομή του Nagios βασισμένη στα ανεξάρτητα δομικά στοιχεία (plugins) επιτρέπει την συλλογή δεδομένων από διαφορετικές πηγές (φυσικούς, εικονικούς πόρους και εφαρμογές).

- Τα τεχνικά χαρακτηριστικά του μηχανισμού διαμορφώνουν ελαστικότητα όχι μόνο στον αριθμό της συλλεχθείσας πληροφορίας αλλά και στον τύπο αυτής. Η ύπαρξη της οντότητας του παραγωγού πληροφορίας η οποία παρέχεται ως πόρος της *RESTful* υπηρεσίας επίβλεψης, υποστηρίζει την εισαγωγή καινούριος πηγών πληροφορίας. Επιπρόσθετα, η ίδια η υπηρεσία επίβλεψης θα μπορούσε να φιλοξενηθεί σε Υπολογιστικό Νέφος, εκμεταλλευόμενη την ελαστικότητα που αυτό παρέχει σε διαφορετικές συνθήκες λειτουργίας.
- Υψηλή απόδοση λειτουργίας: όπως παρουσιάστηκε και σε προηγούμενο κεφάλαιο, η απόδοση του μηχανισμού του Nagios είναι πολύ υψηλή όσον αφορά τον επιπρόσθετο φόρτο εργασίας που επιφέρει στο σύστημα. Παρόλα αυτά, η συχνή λειτουργία του στοιχείου NEB2REST και ως εκ τούτου η συχνή επικοινωνία του στοιχείου αυτού με την υπηρεσία Monitoring Service, θα μπορούσε να προκαλέσει αυξημένη κίνηση στο δίκτυο της υποδομής.
- Οι *RESTful* υπηρεσίες είναι αποδοτικές και απλές στην ανάπτυξη σε σχέση με τις τεχνολογίες SOAP και WSRF. Η αρχιτεκτονική που παρουσιάστηκε μπορεί εύκολα να επεκταθεί και προσαρμοστεί σε νέες ανάγκες σχεδιασμού απλά και μόνο με την εισαγωγή καινούριο πόρων στην υλοποίηση της υπηρεσίας Monitoring Service.

6.2 Ελαστική φιλοξενία διαδικτύου σε ιδιωτικά Νέφη (Elastic Web hosting)

Ο όρος «ελαστικότητα» μπορεί να έχει χρησιμοποιηθεί και στο παρελθόν έμμεσα ή άμεσα, αλλά η επικρατούσα, σύγχρονη χρήση της στον τομέα των υπολογιστών προέρχεται από το προϊόν της Amazon, "*Elastic Compute Cloud*" (EC2). Γενικά, ελαστικότητα (*elasticity*) σημαίνει «η δυνατότητα κάποιου να προσαρμοστεί» (λεξικό της Οξφόρδης), επομένως η ικανότητα να προσαρμοστεί σε νέους όρους και συνθήκες. Στον χώρο των υπολογιστών και των υπολογιστικών συστημάτων συγκεκριμένα, η ελαστικότητα μεταφράζεται ως ενεργοποίηση και διάθεση πόρων κατ' απαίτηση. Αυτό το χαρακτηριστικό γίνεται εύκολα εφικτό μέσω της χρήσης της τεχνολογίας της εικονικοποίησης (*virtualization*): οι χρήστες δεν έχουν πρόσβαση στην φυσική υποδομή αλλά τους εικονικούς πόρους που μπορούν να τρέξουν οπουδήποτε υπάρχει ικανοποιητική υπολογιστική χωρητικότητα. Είναι κατανοητό πώς η έννοια της ελαστικότητας υποδηλώνει μια δυναμικότητα της υποδομής σε αντίθεση με τον στατικό τρόπο διαχείρισης των πόρων: εάν ένας χρήστης φιλοξενεί μια ιστοσελίδα σε κάποιον εξυπηρετητή του οποίου η χωρητικότητα υπερβεί το μέγιστο, ένας νέος κεντρικός υπολογιστής πρέπει να αγοραστεί, να εγκατασταθεί και να διαμορφωθεί. Οι ελαστικές λύσεις, εντούτοις, παρέχουν υποδομή που δεν είναι ορατή στο χρήστη και μπορεί να μεγαλώσει ή να μικρύνει ανάλογα με την τρέχουσα απαίτηση (φυσικά, εάν η ικανότητα της πραγματικής υποδομής είναι στο μέγιστο, το ίδιο πρόβλημα εμφανίζεται, αλλά υποτίθεται ότι η πραγματική ικανότητα υποδομής είναι αρκετά υψηλή). Εξίσου σημαντική είναι η δυνατότητα να μειωθεί το μέγεθος της εικονικής υποδομής μόλις ανιχνευθεί η πλεονάζουσα χωρητικότητα.

Με τον ίδιο τρόπο που εφαρμόζεται η ελαστικότητα στην δυναμική διαχείριση των πόρων, μπορεί να εφαρμοστεί και στην περίπτωση της φιλοξενίας διαδικτύου (*web hosting*): δεδομένου ότι οι κεντρικοί εξυπηρετητές του διαδικτύου (*web servers*) λειτουργούν χωρίς αποθήκευση της κατάστασης (*stateless*), με την έννοια ότι δεν αποθηκεύουν κανένα στοιχείο από ένα αίτημα σε άλλο, πρόσθετοι κεντρικοί εξυπηρετητές διαδικτύου μπορούν να

παρασχεθούν εάν ο αριθμός των εισερχόμενων αιτημάτων αυξηθεί απότομα. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας εικονικές μηχανές (VMs) που προετοιμάζονται αναλόγως, παραδείγματος χάριν έχοντας εγκατεστημένο τον εξυπηρετητή διαδικτύου, και από την στιγμή που μια εικονική μηχανή μπει σε λειτουργία, θα μπορεί να εξυπηρετήσει τα αιτήματα και μπορεί να αφοπλιστεί μόλις μειωθεί πάλι το φορτίο. Στο κεφάλαιο αυτό, θα προτείνουμε και θα παρουσιάσουμε μια λύση βασισμένη σε εργαλεία ελεύθερου λογισμικού για την πραγματοποίηση της ελαστικής φιλοξενίας διαδικτύου σε μια υποδομή ιδιωτικού Νέφους.

6.2.1 Προτεινόμενη αρχιτεκτονική

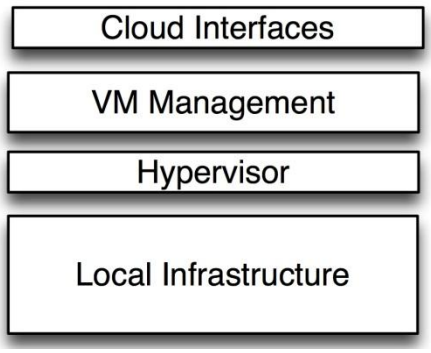
Σε αυτό το υποκεφάλαιο περιγράφεται η αρχιτεκτονική της προτεινόμενης λύσης σε τρία βήματα. Αρχικά παρουσιάζεται συνοπτικά η αρχιτεκτονική του ιδιωτικού Νέφους τόσο η γενική του δομή όσο και με την εισαγωγή του σεναρίου της ελαστικής φιλοξενίας διαδικτύου σε αυτήν. Στη συνέχεια, αναλύεται ο ρόλος της εξισορρόπησης φορτίου (*load balancing*) στην αρχιτεκτονική του συστήματός μας. Τέλος, παρουσιάζεται η τεχνική δυναμικής διάθεσης των εικονικών μηχανών με στόχο την επίτευξη δυναμικότητας στην αρχιτεκτονική.

6.2.1.1 Αρχιτεκτονική ιδιωτικού Νέφους

Όπως αναφέρθηκε προηγουμένως, όλο και περισσότεροι ενδιαφέρον εκδηλώνεται για την δημιουργία υποδομών ιδιωτικού Νέφους. Δεδομένου ότι αυτή η λύση κερδίζει έδαφος, διάφορα συστήματα ανοιχτού κώδικα και APIs έχουν αναπτυχθεί που επιτρέπουν τη διαχείριση εικονικών μηχανών (VM) και πραγματοποιούν το μοντέλο IaaS. Το Eucalyptus [113] είναι μια λύση που αναπτύσσεται στο πανεπιστήμιο Καλιφόρνιας και προσφέρει τη δυνατότητα να εγκατασταθούν και να ελεγχθούν οι στιγμιότυπα VM μέσω διαφορετικών μεσολογισμικών (*hypervisors*, Xen ή KVM) πέρα από τους φυσικούς πόρους. Το Nimbus [114] είναι παρόμοιο με το Eucalyptus σύστημα και παρέχει τις απαραίτητες διεπαφές για να δώσει τον έλεγχο χρηστών πάνω σε VMs. Ένα άλλο σύστημα διαχείρισης εικονικών μηχανών που επιτρέπει την δημιουργία ιδιωτικών ή υβριδικών υποδομών Νεφών είναι το OpenNebula

[115] που υποστηρίζεται κυρίως από το πανεπιστήμιο της Μαδρίτης και έχει χρησιμοποιηθεί ήδη σε διάφορες ερευνητικές πρωτοβουλίες [116] [117].

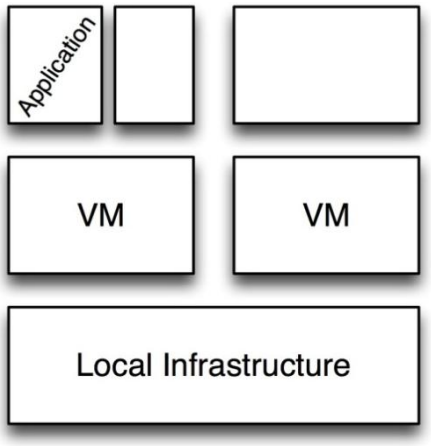
Ανεξάρτητα από τις διαφορές, όλες οι προαναφερθείσες λύσεις μοιράζονται μια κοινή, υψηλού επιπέδου αρχιτεκτονική για την υλοποίηση ενός Ιδιωτικού Νέφους. Μια αφηρημένη αρχιτεκτονική προσέγγιση ενός ιδιωτικού σύννεφου παρουσιάζεται στο Σχήμα 57.



Σχήμα 57: Γενική αρχιτεκτονική Ιδιωτικού Νέφους

Η πολυστρωματική αρχιτεκτονική αποτελείται από τη φυσική (τοπική) υποδομή, πάνω από αυτήν βρίσκεται το μεσολογισμικό (Xen, KVM κ.λπ.) ενώ πάνω από το στρώμα αυτό βρίσκεται το διοικητικό πλαίσιο της εικονικοποίησης (OpenNebula, Eucalyptus κ.λπ.). Το τελευταίο εκθέτει τις κατάλληλες διεπαφές για τους χρήστες/διαχειριστές έτσι ώστε να έχουν πρόσβαση και να ελέγχουν το Ιδιωτικό Νέφος. Με βάση αυτό το αρχιτεκτονικό πρότυπο μπορούμε να προσδιορίσουμε σαφώς δύο διακριτικές υποδομές: τη φυσική και την εικονική υποδομή (Σχήμα 58).

Με την εισαγωγή των τεχνολογιών εικονικοποίησης που έχουν χρησιμοποιηθεί από τους παρόχους Νεφών, ο καταναλωτής δεν ενδιαφέρεται πλέον για τους φυσικούς πόρους αλλά για τα VM στα οποία έχει πρόσβαση. Η φυσική υποδομή γίνεται διαφανής για αυτόν, και μέσω της υποδομής-ως-υπηρεσία (IaaS) διαπραγματεύεται την πρόσβαση σε ένα εικονικό περιβάλλον με συγκεκριμένες προδιαγραφές. Αυτή η εικονική υποδομή είναι δυναμική, εύκαμπτη και εξατομικεύσιμη σύμφωνα με την εφαρμογή ή την υπηρεσία που κάθε χρήστης θέλει να εκτελέσει.



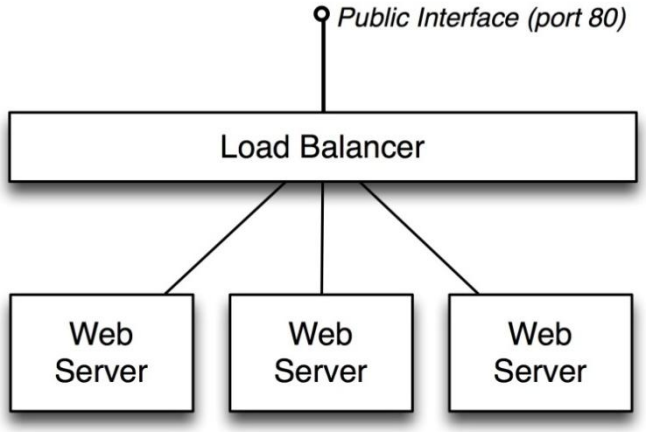
Σχήμα 58: Εικονική και φυσική υποδομή

Επιστρέφοντας στο σενάριό μας και σύμφωνα με την αρχιτεκτονική ιδιωτικού Νέφους που παρουσιάστηκε, η ελαστική φιλοξενία διαδικτύου μπορεί να πραγματοποιηθεί όταν ένας εξυπηρετητής διαδικτύου (*web server*) εκτελείται ως εφαρμογή και φιλοξενείται σε μια εικονική μηχανή (VM) μιας υποδομής Νέφους. Θα πρέπει να γίνει σαφές σε αυτό το σημείο ότι η λύση που προτείνεται δεν είναι κάποια απόλυτα καινοτόμα υπηρεσία ή κάτι το οποίο δεν προσφέρεται από τα υπάρχοντα εμπορικά Νέφη (π.χ Amazon EC2). Η πρόταση που εκφράζεται σε αυτό το κεφάλαιο δίνει βαρύτητα στην τεχνική υλοποίησης ενός περιβάλλοντος ελαστικής φιλοξενίας διαδικτύου για Ιδιωτικά ή Υβριδικά Νέφη χρησιμοποιώντας αποκλειστικά εργαλεία και μηχανισμούς ανοιχτού κώδικα (open source).

6.2.1.2 Εξισορρόπηση Φορτίου (Load balancing)

Η χρησιμοποίηση τεχνικών εξισορρόπησης φορτίων (*load balancing*) σε υπηρεσίες φιλοξενίας του διαδικτύου είναι ένα σχετικά παλαιό θέμα [118]. Στην προτεινόμενη προσέγγισή θα χρησιμοποιήσουμε εξυπηρετητές διαδικτύου σε μια υποδομή ιδιωτικού Νέφους, και θα επωφεληθούμε από την ευελιξία της εικονικοποίησης και ενός αυτό-ελεγχόμενου μηχανισμού διαχείρισης της υποδομής. Η κατανομή των εισερχόμενων αιτημάτων ρυθμίζεται από ένα δομικό συστατικό που εκτελεί την εξισορρόπηση φορτίων. Όπως φαίνεται στο Σχήμα 59 κανένας από τους διαθέσιμους εξυπηρετητές δικτύου δεν είναι άμεσα προσβάσιμος από τους χρήστες. Ο εξισορροπιστής φορτίων θα λάβει τα εισερχόμενα

αιτήματα στην πόρτα 80 και θα τα διανείμει εξίσου προς τους κεντρικούς υπολογιστές λαμβάνοντας υπόψη την κατάσταση κάθε κόμβου.



Σχήμα 59: Εξισορρόπηση φορτίου στην διαδικτυακή φιλοξενία

Η εισαγωγή αυτού του συστατικού ασκεί πολύ σημαντική επίδραση στην αξιοπιστία της παρεχόμενης υπηρεσίας. Έχοντας τον έλεγχο των εισερχομένων αιτημάτων, μέσω του *load balancer* μπορούμε να βελτιστοποιήσουμε τη χρησιμοποίηση των πόρων μας και συνολικά την ποιότητα της παρεχόμενης υπηρεσίας (QoS). Επιπλέον, αυτή η δομή μας βοηθά να αυξήσουμε φαινομενικά την χωρητικότητα της υποδομής μας χωρίς την επένδυση σε καινούριους εξυπηρετητές με εξεζητημένες τεχνικές προδιαγραφές. Επίσης, το προτεινόμενο μοντέλο πραγματοποιεί ένα σύστημα διαχείρισης σφαλμάτων: σε περίπτωση που ένας κεντρικός υπολογιστής διαδικτύου δεν είναι προσβάσιμος ή εκτός λειτουργίας, τα αιτήματα HTTP θα συνεχίσουν να εξυπηρετούνται από τους υπόλοιπους κεντρικούς υπολογιστές.

6.2.1.3 Υλοποιώντας την επεκτασιμότητα (scalability)

Η ελαστικότητα της προτεινόμενης λύσης επιτυγχάνεται μέσω της ενεργοποίησης ενός νέου VM που θα φιλοξενήσει έναν πρόσθετο εξυπηρετητή διαδικτύου όταν η χωρητικότητα του τρέχοντος συστήματος φτάνει σε ένα όριο. Για αυτόν τον λόγο, το σύστημα διαχείρισης και ελέγχου που εξυπηρετεί τα εισερχόμενα αιτήματα θα πρέπει να εμπεριέχει συγκεκριμένες πολιτικές για να εφαρμόσει την κατανομή του φορτίου. Συνολικά η υψηλού επιπέδου λειτουργία του συστήματος όταν φθάνει ένα νέο εισερχόμενο αίτημα περιγράφεται από τον ακόλουθο ψευδοκώδικα:

```
If (web_server_capacity < limit)
{
    forward(request)
 } else {
   instantiate_new_VM()
   reset_load_balancer()
   forward(request)
}
```

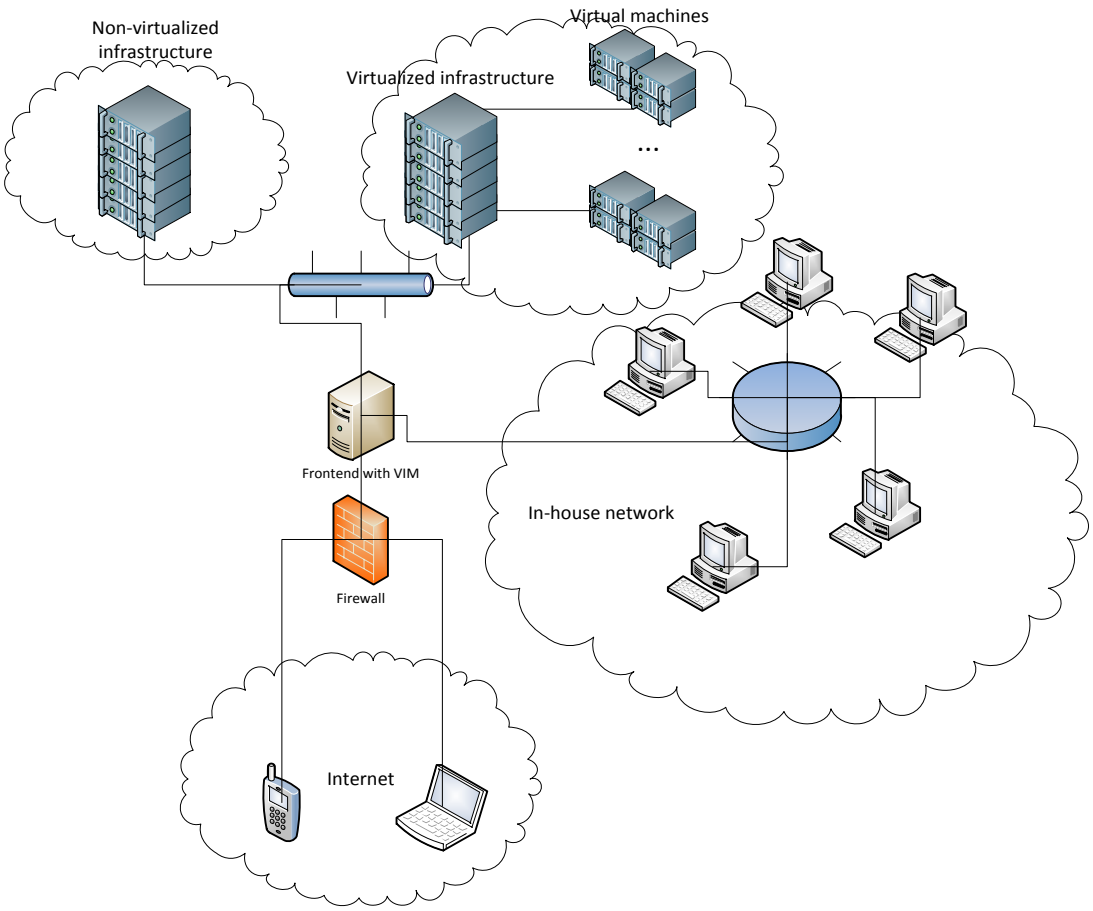
Μέσω αυτής της λειτουργίας μπορούμε να εγγυηθούμε ότι πάντα τα εισερχόμενα αιτήματα θα εξυπηρετηθούν και στον ίδιο χρόνο θα πετύχουμε στην κατοχή της καλής χρησιμοποίησης των πόρων. Αυτή η δυναμικότητα μας επιτρέπει να χρησιμοποιήσουμε τους ελεύθερους πόρους με άλλες εφαρμογές και δραστηριότητες και να μην αφιερώσουμε ολόκληρη την υποδομή μας με έναν στατικό τρόπο για τη λειτουργία του εξυπηρετητή διαδικτύου.

6.2.2 Υλοποίηση

Σε αυτή την ενότητα παρουσιάζεται η υλοποίηση της αρχιτεκτονικής του συστήματος ελαστικής φιλοξενίας διαδικτύου σε περιβάλλον ιδιωτικού Νέφους. Όλα τα μέρη λογισμικού που χρησιμοποιήθηκαν για αυτήν την λύση είναι λογισμικό ανοιχτού κώδικα. Παρακάτω παρουσιάζουμε την φυσική υποδομή που χρησιμοποιήσαμε για την πειραματική εφαρμογή αυτού του συστήματος, αλλά όπως είναι κατανοητό η προτεινόμενη λύση μπορεί να εφαρμοστεί και σε διαφορετικές υλοποιήσεις υποδομής Ιδιωτικού Νέφους.

6.2.2.1 Φυσική υποδομή

Η φυσική υποδομή που χρησιμοποιήθηκε κατά την αξιολόγηση της προτεινόμενης λύσης παρουσιάζεται στο Σχήμα 60. Οι φυσικοί πόροι, που χρησιμοποιούνται ως υπολογιστικοί κόμβοι, λειτουργούν σε ένα απομονωμένο υποδίκτυο και είναι προσβάσιμοι μόνο από τον κεντρικό κόμβο εισόδου (*frontend node*). Τα αιτήματα κατατίθενται στον κόμβο αυτό ο οποίος μέσω του συστατικού *Load Balancer* τα κατανέμει στους εξυπηρετητές των εικονικών μηχανών.



Σχήμα 60: Συνύπαρξη φυσικής και εικονικής υποδομής

6.2.2.2 Λειτουργικό σύστημα

Το CentOS (αρχικά για *Community ENTerprise Operating System*) είναι μια διανομή GNU/Linux βασισμένη στην Red Hat Enterprise Linux (RHEL) διανομή. Ενώ το RHEL είναι μια εμπορική διανομή Linux, το Cent OS αποτελεί λογισμικό ανοιχτού κώδικα. Η συγκεκριμένη έκδοση χρησιμοποιείται σχεδόν από το 30% όλων των εξυπηρετητών διαδικτύου που βασίζονται σε Linux [119]. Η επιλογή του Cent OS ως το λειτουργικό σύστημα για στην πειραματική μας εγκατάσταση έγινε καθώς παρέχει απευθείας τόσο πλήρη εικονικοποίηση (*virtualization* – οι εφαρμογές πελάτη μπορούν να εκτελούνται στον κόμβο χωρίς κάποια μετατροπή) όσο και παρα-εικονικοποίηση (*para-virtualization* – ένα ενδιάμεσο στρώμα μεσολογισμικού παρεμβαίνει μεταξύ του υλικού και της εφαρμογής πελάτη). Ως εκ τούτου, επιλέχθηκε η χρήση αυτής της διανομής Linux ως λειτουργικό σύστημα για τους φυσικούς κόμβους της υποδομής. Επιπρόσθετα, η ευκολία δημιουργίας εικόνων (images)

λειτουργικού συστήματος Cent OS, μας οδήγησε στην χρήση αυτής της έκδοσης και για το λειτουργικό σύστημα των VMs του Ιδιωτικού Νέφους της πειραματικής εγκατάστασής μας.

6.2.2.3 Μεσολογισμικό Νέφους

Για την πραγματοποίηση ενός Ιδιωτικού Νέφους, υπάρχουν διάφορες επιλογές διαχειριστών εικονικής υποδομής (*Virtual Infrastructure Managers - VIM*) : Eucalyptus, Nimbus και OpenNebula, είναι τα τρία πιο δημοφιλή. Το OpenStack, ένα VIM που αναπτύσσεται από την Rackspace και τη NASA [120], και είναι ένα ακόμη ελπιδοφόρο πρόγραμμα που βρίσκεται υπό ανάπτυξη ενώ άλλα τα τρία VIMs μπορούν να ληφθούν ήδη για χρήση παραγωγής. Υπάρχουν ευδιάκριτα πλεονεκτήματα και μειονεκτήματα σε κάθε ένα από αυτά τα τρία VIMs και η επιλογή ενός από αυτά δεν είναι εύκολη διαδικασία. Για την περίπτωσή μας, δεδομένου ότι υλοποιούμε ένα Ιδιωτικό Νέφος γύρω από έναν κεντρικό κόμβο και έχουμε ένα μικρό σύνολο μηχανών που είναι προσβάσιμα μόνο από έμπιστους χρήστες, αποφασίσαμε να χρησιμοποιήσουμε το OpenNebula. Η πιο πρόσφατη έκδοση του OpenNebula, έκδοση 2.0 (Οκτώβριος 2010), παρέχει υποστήριξη για Xen, KVM και VMWare, προσθέτει ένα καταχωρητή εικόνων μηχανών (*VM images repository*) για τη διαχείριση των εικόνων VM και έτσι η βασική εγκατάστασή στον κεντρικό κόμβο είναι εύκολη και αρκετά σύντομη.

6.2.2.4 Εξισορροπιστής φορτίου (Load balancer)

Όπως και με το λειτουργικό σύστημα και τον λογισμικό VIM, υπάρχουν πολλαπλές επιλογές και για την εξισορρόπηση φορτίων. Στην περίπτωσή μας, δεν έχουμε ισχυρές απαιτήσεις φορτίων: θέλουμε το πρόγραμμα εξισορρόπησης να εκτελείται σε μια VM, να διανέμει τα εισερχόμενα αιτήματα σε πολλαπλούς κεντρικούς εξυπηρετητές, να είμαστε σε θέση να ενημερωθούμε για την παρούσα κατάσταση και να αλληλεπιδράσουμε με αυτόν κατά τη διάρκεια της εκτέλεσης. Με το λογισμικό *balance* [121], βρήκαμε μια απλή, εύχρηστη λύση που καλύπτει αυτές τις προδιαγραφές. Το *balance* είναι ένας μικρός (περίπου 2k γραμμές κώδικα) γενικό *TCP proxy* που υποστηρίζει *round-robin* εξισορρόπηση φορτίου και διαθέτει μηχανισμούς διαχείρισης σφαλμάτων. Μπορεί εύκολα να ελεγχθεί κατά τη διάρκεια της εκτέλεσης μέσω μιας απλής διεπαφής γραμμών εντολών. Η απλότητά του σημαίνει ότι

κάποιος μπορεί να υλοποιήσει και να επεκτείνει την εξισορρόπηση εύκολα αλλά και να χρησιμοποιήσει άμεσα τις παρεχόμενες δυνατότητες.

Η διαμόρφωση της εξισορρόπησης των εισερχόμενων αιτημάτων στην πόρτα 80 για δυο εξυπηρετητές, γίνεται με την παρακάτω εντολή:

 # balance www host1 host2

Το λογισμικό *balance* θα εκτελείται στο παρασκήνιο και θα εξισορροπεί τα εισερχόμενα αιτήματα δυναμικά μεταξύ των host1 και host2. Στην πειραματική μας εφαρμογή επιλέξαμε να περιορίσουμε, για λόγους απλότητας, τον αριθμό των συνδέσεων που ένας εξυπηρετητής μπορεί να δεχθεί έτσι ώστε να προχωρήσουμε με την αξιολόγηση. Η ρύθμιση αυτή επιτυγχάνεται με την παρακάτω εντολή [122]:

 # balance www host1::8 host2::8

Όπως παρουσιάζεται και στην εντολή, ορίσαμε ότι και οι δύο κόμβοι (host1, host2) μπορούν να εξυπηρετήσουν ταυτόχρονα 8 συνδέσεις. Είναι προφανές ότι ι διαμόρφωση του λογισμικού εξισορρόπησης θα πρέπει να είναι αντίστοιχο με τις δυνατότητες των υπαρχόντων εξυπηρετητών (για παράδειγμα, δεν θα πρέπει το balance να έχει διαμορφωθεί να επιτρέπει μέχρι 250 ταυτόχρονες συνδέσεις σε ένα κόμβο όταν ο εξυπηρετητής αυτού επιτρέπει μέχρι 150 συνδέσεις).

Η κατάσταση των συνδέσεων μπορεί να παρουσιαστεί με την παρακάτω εντολή:

 # balance http -i -c show

GRP Type # S ip-address port c totalc maxc sent rcvd

0 RR 0 ENA 10.0.0.12 80 1 4 8 0 0

0 RR 1 ENA 10.0.0.13 80 0 4 8 0 0

Κάθε γραμμή προσδιορίζει ένα κόμβο, η κολόνα “c” τον αριθμό των συνδέσεων αυτή τη στιγμή, “totalc” τον συνολικό αριθμό των συνδέσεων μέχρι τώρα και “maxc” τον μέγιστο αριθμό των συνδέσεων που κάθε κόμβος μπορεί να εξυπηρετήσει.

Η προσθήκη ενός κόμβου δυναμικά πραγματοποιείται με τις παρακάτω εντολές:

 # balance http -i -c "create 10.0.0.14 80"

 # balance –i –c "enable 2"

Η απενεργοποίηση ενός κόμβου γίνεται με:

 # balance –i –c "disable 2"

Όπως γίνεται κατανοητό, το *balance* επιτρέπει όλες τις απαραίτητες λειτουργίες που θα χρειαστούμε για την διαχείριση του φορτίου στο Νέφος με δυναμικό τρόπο. Σε αυτό το πλαίσιο λειτουργίας, το συστατικό της αρχιτεκτονικής μας μπορεί με αυτοματοποιημένο τρόπο να διαχειριστεί τους κόμβους της υποδομής μας.

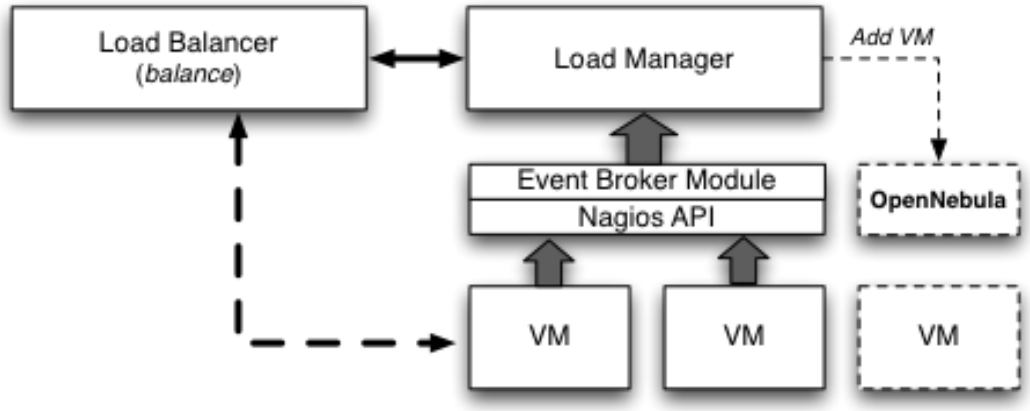
6.2.2.5 Επίβλεψη και διαχείριση

Καθώς η ελαστικότητα είναι το βασικό χαρακτηριστικό της προτεινόμενης αρχιτεκτονικής, η επίβλεψη του φορτίου και η διαχείριση της εικονικής υποδομής είναι δύο κρίσιμες λειτουργίες αυτού του συστήματος. Σε αυτή την βάση, έχουμε δημιουργήσει έναν μηχανισμό που βασίζεται σε δύο λογισμικά ανοιχτού κώδικα: το Nagios για την επίβλεψη των πόρων και το OpenNebula για την διαχείριση των εικονικών μηχανών. Ο μηχανισμός αυτός παρακολουθεί τους πόρους και το φορτίο τους και σε συνδυασμό με το λογισμικό εξισορρόπησης κατανέμει τα εισερχόμενα αιτήματα ή/και ενεργοποιεί καινούριες εικονικές μηχανές εάν κριθεί απαραίτητο.

Ο μηχανισμός διαχείρισης αποτελείται από τρία κύριο συστατικά (Σχήμα 61):

- *Nagios API*: είναι ένας μηχανισμός επί΄βλεψης πόρων μέσω του οποίου μπορούμε να αποκτήσουμε τη κατάσταση πολλαπλών κόμβων σχετικά με τη διαθεσιμότητα, το φορτίο, τη μνήμη και διάφορες άλλες μετρικές. Στην περίπτωσή μας, έχουμε εφαρμόσει συγκεκριμένα έναν έλεγχο υπηρεσιών Nagios προκειμένου να ελεγχθεί ο αριθμός συνδέσεων για την υπηρεσία httpd που τρέχει σε κάθε κόμβο, έναν έλεγχο PING για τη διαθεσιμότητα και έναν έλεγχο που μας επιστρέφει το φορτίου της CPU

- *Event Broker Module*: αυτό είναι ένα συστατικό γραμμένο σε C γλώσσα προγραμματισμού και χρησιμοποιεί το *Nagios Event Broker API* (NEB) [111]. Η λειτουργία αυτού του λογισμικού βασίζεται στις ειδοποιήσεις που το Nagios παράγει αυτόματα κάθε φορά που ένας έλεγχος πραγματοποιείται. Με αυτόν τον τρόπο το συστατικό αυτό αναγνωρίζει τα γεγονότα που συμβαίνουν στην υποδομή και προωθεί τις απαραίτητες πληροφορίες στο συστατικό *Load Manager*.
- *Load Manager*: το συστατικό αυτό λαμβάνει ως δεδομένα εισόδου τις πληροφορίες επίβλεψης που προέρχονται από το Nagios σχετικά με την κατάσταση των εικονικών μηχανών (VMs). Στην περίπτωση που ο εξυπηρετητής σε κάποια VM πλησιάζει κάποια όρια διαθεσιμότητας (χωρητικότητα, φορτίο CPU, μνήμη κ.α.), τότε ο *Load Manager* δίνει εντολή στο μεσολογισμικό (*OpenNebula API*) να ενεργοποιήσει μια καινούρια εικονική μηχανή και ταυτόχρονα αναδιαμορφώνει τον εξισορροπηστή (*balance API*) φορτίου για τον νέο κόμβο.



Σχήμα 61: Επίβλεψη και διαχείριση φορτίου

Τα δεδομένα επίβλεψης που διατίθεται στον *Load Manager* προέρχονται τόσο από τον εξισορροπηστή φορτίου (*Load Balancer*), όσο και από το *Event Broker Module*. Ως εκ τούτου, ο τύπος της συλλεχθείσας πληροφορίας και οι πολιτικές που εφαρμόζονται πάνω σε αυτές μπορούν να τροποποιηθούν αναλόγως τις ανάγκες της εκάστοτε υποδομής και υλοποίησης. Αυτή η δυνατότητα της προτεινόμενης λύσης επαυξάνει την δυναμικότητα και ελαστικότητα της αρχιτεκτονικής.

6.2.3 Πειραματική εφαρμογή και αποτελέσματα

Η υποδομή που χρησιμοποιήσαμε για την πειραματική εφαρμογή του συστήματός μας αποτελείται από έναν εμπρόσθιο κόμβο (*front-end*) και δύο κόμβους εργασίας (*workers*) με τεχνικά χαρακτηριστικά που παρουσιάζονται στον παρακάτω πίνακα:

Front-end

CPU

2x Intel(R) Xeon(TM) CPU 3.20GHz

16kiB L1 cache

1MiB L2 cache

RAM

8GiB system memory

Composed of 4 x 2GiB DIMM DDR Synchronous 333 MHz (3.0 ns)

Disk (SATA)

2 x 250GB HDS722525VLSA80

Raid 1 setup (mirrored)

Network

2x 82546GB Gigabit Ethernet Controller

Workers

CPU

2x Intel(R) Xeon(TM) CPU 3.20GHz

16kiB L1 cache

1MiB L2 cache

RAM

8GiB system memory

Composed of 4 x 2GiB DIMM DDR Synchronous 333 MHz (3.0 ns)

Disk (SATA)
2 x 250GB HDS722525VLSA80
Raid 1 setup (mirrored)
Network
82546GB Gigabit Ethernet Controller

Πίνακας 24: Υποδομή αξιολόγησης εξισορρόπησης φορτίου

Χρησιμοποιήσαμε το OpenNebula για να εκκινήσουμε 4 VMs στους κόμβους εργασίας και καθένα από αυτά τα στιγμιότυπα φιλοξένησε έναν εξυπηρετητή διαδικτύου *HTTP Apache web server*. Το πειραματικό σενάριο που εφαρμόσαμε ήταν μέσω του λογισμικού *curl-loader* να δημιουργήσουμε φορτίο αιτημάτων στους διαθέσιμους εξυπηρετητές διαδικτύου και να αναγκάσουμε τον *Load Manager* να ενεργοποιήσει ένα νέο VM όταν προσεγγίσουμε το καθορισμένο όριο **100 συνδέσεων**, ενώ η μέγιστη χωρητικότητα του εξυπηρετητή είναι **150 συνδέσεις.**

Για τις δοκιμές μας, χρησιμοποιήσαμε το πρωτόκολλο μεταφοράς SSH, το οποίο αντιγράφει τις εικόνες VM στον κόμβο που πρόκειται να εκτελεσθούν χρησιμοποιώντας το ασφαλές πρωτόκολλο δικτύων *Secure Shell* (ssh). Για ένα VM με περίπου 2GB μέγεθος, η μεταφορά μέσω μια σύνδεσης *Gigabit Ethernet* που χρησιμοποιεί πήρε στην περίπτωσή μας περίπου **90 δευτερόλεπτα**. Αποφασίσαμε να χρησιμοποιήσουμε το διευθυντή μεταφοράς SSH δεδομένου ότι αυτό δεν απαιτεί καμία περαιτέρω μετατροπή εκτός από τη διαμόρφωση SSH στον κόμβο *front-end* και στους κόμβους *workers.*

Δεδομένου ότι τα 90 δευτερόλεπτα προαναφερθέντα είναι ο χρόνος που απαιτείται για την καθαρή μεταφορά του αρχείου εικόνας VM, ο χρόνος που απαιτήθηκε για την συνολική ενεργοποίηση (εκκίνηση κλπ) του VM πρέπει να συνυπολογιστεί. Για την εικόνα VM των 2GB, ο χρόνος για την εκκίνηση ήταν περίπου **50 δευτερόλεπτα**, έχοντας συνολικό χρόνο για την εγκατάσταση ενός VM σε **140 δευτερόλεπτα.**

Αυτός είναι ένας αρκετά καλός χρόνος. Παρόλα αυτά εάν δεν ρυθμίσουμε κατάλληλα τις πολιτικές και λειτουργία του *Load Manager* υπάρχει κίνδυνος κάποια αιτήματα να μην

εξυπηρετηθούν. Έχουμε αναφέρει προηγουμένως το γεγονός ότι το κατώτατο όριο, στο οποίο ένα νέο VM πρέπει να ενεργοποιηθεί, πρέπει να είναι χαμηλότερο από τη μέγιστη ικανότητα στην οποία τα VM μπορούν να λειτουργήσουν. Ανάλογα με τον αριθμό συνδέσεων που επιτρέπονται στο σύνολο και τη συχνότητα άφιξης των αιτημάτων, το κατώτατο όριο μπορεί να τεθεί αναλόγως. Φυσικά αυτό δεν εξασφαλίζει απόλυτα την εξυπηρέτηση κάθε αιτήματος αφού με υψηλό ρυθμό άφιξης αιτημάτων θα μπορούσαμε να έχουμε σφάλματα. Εντούτοις, για αργούς ρυθμούς άφιξης συνδέσεων αυτή η λύση μπορεί να εφαρμοστεί.

Επεκτείνοντας τις πολιτικές διαχείρισης του *Load Manager* με σκοπό την καλύτερη και ασφαλέστερη εξυπηρέτηση των αιτημάτων εφαρμόσαμε μια συνδυαστική τεχνική: όταν η χωρητικότητα ενός εξυπηρετητή προσεγγίζει ένα όριο (χαμηλότερο του προηγούμενου σεναρίου) τότε ενεργοποιείται ένα νέο VM αλλά παραμένει σε «παγωμένη» κατάσταση (*paused mode*), εωσότου ο αριθμός των συνδέσεων προσεγγίσει ένα νέο όριο (μεγαλύτερο του προηγουμένου αλλά μικρότερο της μέγιστης δυνατότητας εξυπηρέτησης) όπου και ενεργοποιείται σε πλήρη λειτουργία η εικονική μηχανή. Με την τεχνική αυτή καταναλώνουμε μεν περισσότερους πόρους (συγκεκριμένα μνήμη RAM και όχι CPU) αλλά βελτιώνουμε κατά πολύ τον χρόνο ενεργοποίησης ενός νέου εξυπηρετητή και έτσι την αξιοπιστία του συστήματός μας συνολικά. Στην πειραματική μας διάταξη, διαμορφώσαμε το πρώτο όριο για την ενεργοποίηση των «παγωμένων» εικόνων στις 80 συνδέσεις, ενώ το δεύτερο όριο στις 130. Με την τεχνική αυτή καταφέραμε χρόνο ενεργοποίησης της VM, στην δεύτερη φάση, περίπου **15 δευτερόλεπτα.** Φυσικά, με την υλοποίηση αυτή επιτυγχάνουμε καλύτερη και ασφαλέστερη εξυπηρέτηση των αιτημάτων αλλά υπό συνθήκες μεγάλου ρυθμού άφιξης παραμένει μια μικρή πιθανότητα αποτυχίας εξυπηρέτησης.

6.2.4 Αρχικά συμπεράσματα

Σε αυτή την ενότητα συζητήσαμε τα πλεονεκτήματα ενός ιδιωτικού Νέφους σε αντίθεση με ένα δημόσιο. Επιπλέον, επιδείξαμε πώς αυτό το ιδιωτικό Νέφος μπορεί να υλοποιηθεί χρησιμοποιώντας μόνο ελεύθερο λογισμικό. Σε αυτό το πλαίσιο, παρουσιάσαμε πώς

μπορούμε να εφαρμόσουμε και να πραγματοποιήσουμε την ελαστική φιλοξενία Ιστού σε μια ιδιωτική υποδομή Νεφών χρησιμοποιώντας μόνο ελεύθερο λογισμικό επίσης. Η λύση που αναπτύχθηκε επικυρώθηκε μέσω μιας ελαστικής χρήσης φιλοξενίας Ιστού σε μια μικρής κλίμακας δοκιμή αλλά η αρχιτεκτονική. Αυτό είναι μόνο μια χρήση της εικονικής υποδομής, η οποία μπορεί να χρησιμοποιηθεί για διάφορους λόγους. Παρουσιάσαμε, εντούτοις, πόσο εύκολη και αποδοτική είναι η οργάνωση και η λειτουργία ενός ιδιωτικού Νέφους. Το πλεονέκτημα αυτής της λύσης είναι ότι η πλήρης υποδομή - φυσική και εικονική – παραμένει στην διαχείριση του ιδιοκτήτη αλλά μπορεί, λόγω της δυνατότητας του OpenNebula να εφαρμοστούν και μοντέλα υβριδικών Νεφών. Ένα δημόσιο Νέφος μπορεί, εντούτοις, να χρησιμοποιηθεί εάν η υπάρχουσα φυσική υποδομή δεν είναι αρκετά μεγάλη προκειμένου να γεφυρωθεί το χρονικό χάσμα όταν συνειδητοποιείται αυτό έως ότου μπορούν να αποκτηθούν περισσότεροι φυσικοί πόροι.

7 *Συμπεράσματα και μελλοντική εργασία*

Στο συγκεκριμένο κεφάλαιο περιλαμβάνεται η σύνοψη της διατριβής και τα συμπεράσματα που εξήχθησαν κατά την εκπόνησή της, η συνεισφορά και η καινοτομία που επιδεικνύει στον αντίστοιχο ερευνητικό χώρο, και συζητούνται θέματα μελλοντικής εργασίας και επέκτασης των ερευνητικών αποτελεσμάτων.

7.1 Σύνοψη και συμπεράσματα

Στην παρούσα διατριβή παρουσιάστηκαν γενικά στοιχεία για τα περιβάλλοντα Υπολογιστικών Νεφών, καθώς και τα συγκεκριμένα χαρακτηριστικά για τα συστήματα διαχείρισής του. Η ανάλυση στην συνέχεια επικεντρώθηκε στους μηχανισμούς επίβλεψης και παρακολούθησης υπηρεσιών και πόρων που διατίθενται από αυτές τις υποδομές. Μελετήθηκαν οι διαθέσιμες τεχνολογίες και καταγράφηκαν τα χαρακτηριστικά και ο προδιαγραφές σχεδιασμού ενός σύγχρονου συστήματος επίβλεψης για περιβάλλοντα Νεφών.

Η πρώτη προτεινόμενη αρχιτεκτονική λύση επίβλεψης πόρων που παρουσιάστηκε αποτελεί ένα υπηρεσιοστρεφές μηχανισμό ο οποίος μπορεί να εφαρμοστεί σε περιβάλλοντα Νεφών και επεκτείνεται σε όλα τα στρώματα της αρχιτεκτονικής αυτής: SaaS, PaaS και IaaS. Αποτελείται από υπηρεσίες συλλογής, αποθήκευσης, συνάθροισης και διαχείρισης

δεδομένων, ακλουθώντας μια ιεραρχική αρχιτεκτονική εξάγοντας δεδομένα από την εφαρμογή στην εικονική υποδομή προς την πλατφόρμα του Νέφους. Τεχνικά, ο μηχανισμός έχει υλοποιηθεί με γλώσσα προγραμματισμού Java και βασίστηκε στις βιβλιοθήκες και τις παρεχόμενες λειτουργίες του Globus MDS. Ένα ιδιαίτερο χαρακτηριστικό αυτού του μηχανισμού επίβλεψης είναι η δυνατότητα αναδιαμόρφωσης των παραμέτρων και συνθηκών της παρακολούθησης. Παρέχεται η δυνατότητα εφαρμογής πολιτικών διαμόρφωσης της επίβλεψης, δυναμικά βασιζόμενοι στα συλλεχθέντα δεδομένα.

Όσον αφορά την αξιολόγηση του μηχανισμού, έγιναν επανειλημμένα πειράματα επιβλέποντας την λειτουργία και απόδοση μιας εφαρμογής διόρθωσης χρώματος και επεξεργασίας εικόνων βίντεο. Κατά την διαδικασία εκτέλεσης του σεναρίου ολοκληρώθηκε επιτυχώς η συλλογή, συνάθροιση και αποθήκευση των δεδομένων τόσο από την εφαρμογή όσο και από την υποδομή (εικονική και φυσική). Η τεχνική αναδιαμόρφωσης του μηχανισμού βασίστηκε στην τυπική απόκλιση του ρυθμού μετάδοσης των πλαισίων εικόνας. Ορίστηκε το όριο του ρυθμού απώλειας πλαισίων στο 5% της συμφωνηθέντας στο SLA τιμής. Η παραβίαση αυτής της συνθήκης ενεργοποιεί την αναδιαμόρφωση είτε αλλάζοντας την συχνότητα συλλογής δεδομένων είτε τις παραμέτρους επίβλεψης είτε και τα δύο. Μέσω των πειραμάτων που εκτελέστηκαν αποδείχθηκε ότι ο μηχανισμός λειτούργησε αποδοτικά ακόμα και με πολύ μικρό διάστημα συλλογής. Επιπρόσθετα, η επιβάρυνση στο φορτίο CPU που επέφερε η λειτουργία του μηχανισμού επίβλεψης ήταν αμελητέα (λιγότερο από 1%) σε όλες τις συνθήκες λειτουργίας, αποδεικνύοντας την αποτελεσματική υλοποίηση του μηχανισμού. Σχετικά με τον χρόνο απόκρισης της υπηρεσίας επίβλεψης του μηχανισμού, τα αποτελέσματα της αξιολόγησης ήταν πολύ ενθαρρυντικά αφού ακόμα και με πολλαπλές συνδέσεις ο χρόνος απόκρισης ήταν κοντά στο 1 sec.

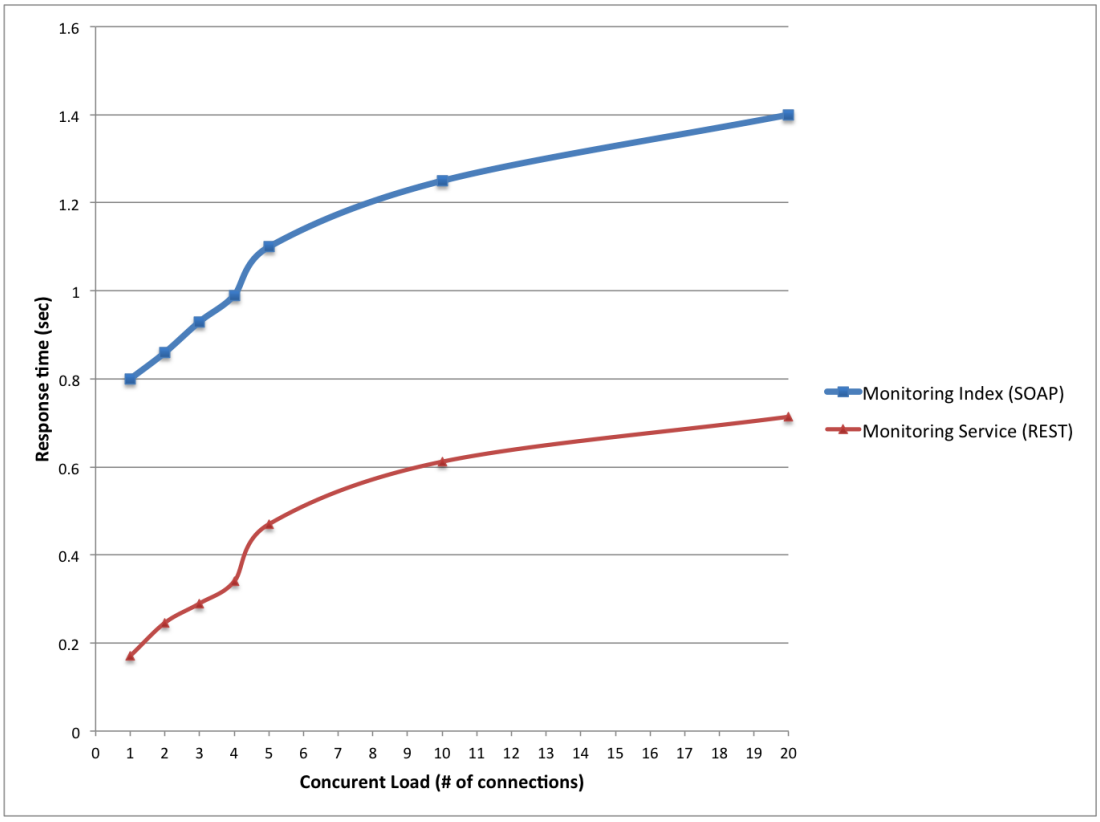
Η δεύτερη λύση που παρουσιάστηκε σε αυτή την διατριβή αφορά ένα σύστημα επίβλεψης πολλαπλών πηγών που δίνει βαρύτητα στον υπολογισμό και την αξιολόγηση της ενεργειακής απόδοσης της υποδομής με στόχο την βέλτιστη διαχείριση των πόρων αυτής. Η υλοποίηση της αρχιτεκτονικής αυτής βασίζεται στα συστατικά στοιχεία των συλλεκτών δεδομένων τα οποία συγκεντρώνουν πληροφορίες από την εφαρμογή, την φυσική και

εικονική υποδομή και την προωθούν στην πλατφόρμα του Νέφους και μέσω της υπηρεσίας επίβλεψης αποθηκεύονται τα δεδομένα σε μια τοπική βάση δεδομένων. Οι τεχνολογίες που χρησιμοποιήθηκαν ήταν RESTful Java υπηρεσίες, στο επίπεδο της πλατφόρμας, εκτελέσιμα αρχεία σε Python, για την συλλογή δεδομένων από το σύστημα επίβλεψης πόρων Nagios και τους αισθητήρες κατανάλωσης ενέργειας, και προγράμματα σε Java για την συλλογή δεδομένων σχετικά με την εφαρμογή και το εικονικό περιβάλλον. Επιπρόσθετα, ορίστηκε η ενεργειακή απόδοση των διαφόρων οντοτήτων μιας υποδομής Νέφους και αναλύθηκαν οι τεχνικές και οι πολιτικές που κάποιος πάροχος IaaS θα μπορούσε να χρησιμοποιήσει έτσι ώστε να βελτιώσει την ενεργειακή απόδοση της υποδομής του.

Όσον αφορά την αξιολόγηση του μηχανισμού αυτού, έγινε πειραματική εφαρμογή του συστήματος σε περιβάλλον Νέφους μικρής κλίμακας (τέσσερις κόμβοι) και αξιολογήθηκε τόσο για την απόδοση του μηχανισμού συλλογής δεδομένων, όσο και για την διαχείριση της υποδομής βασισμένοι στην ενεργειακή απόδοση. Συγκεκριμένα, και οι τέσσερις συλλέκτες δεδομένων επέδειξαν πολύ μικρή παρεμβατικότητα στο σύστημα (μέγιστη μέση τιμή 0.4%), ενώ ο χρόνος απόκρισης του συλλέκτη ενεργειακών δεδομένων ήταν 0.5 sec ανά αίτηση με μέγιστη τιμή 3 sec όταν «πιέσαμε» το σύστημα με πολλαπλές παράλληλες αιτήσεις. Παρόλα αυτά, τα αποτελέσματα είναι ικανοποιητικά αν αναλογιστεί κανείς ότι η ενεργειακή κατανάλωση είναι μια παράμετρος που δεν μεταβάλλεται πολύ απότομα. Σχετικά με την πρόβλεψη της ενεργειακής απόδοσης, το προτεινόμενο μοντέλο βασίστηκε σε μια γραμμική σχέση μεταξύ της κατανάλωσης ενέργειας και της χρησιμοποίησης CPU. Οι τιμές που υπολογίστηκαν από το μοντέλο αυτό παρείχαν ικανοποιητικές προσεγγίσεις της πρόβλεψης της ενεργειακής απόδοσης. Οι μικρές αποκλίσεις που παρατηρούνται θα μπορούσαν να ελαχιστοποιηθούν με την χρήση πιο σύνθετου μοντέλου λαμβάνοντας υπόψη την χρησιμοποίηση του δίσκου και την μνήμης εκτός από την CPU. Στην συνέχεια, προσομοιώθηκε η λειτουργία μιας οικολογικής πολιτικής διαχείρισης πόρων σε περιβάλλον Νέφους, σε σύγκριση με μια πολιτική βασισμένη στο κόστος. Μετά από μια εβδομάδα εκτέλεσης του πειράματος και υπολογισμού της ενεργειακής απόδοσης και στις δύο περιπτώσεις, παρουσιάστηκε βελτίωση της ενεργειακής αποδοτικότητας της

υποδομής του προμηθευτή κατά 14% στην περίπτωση χρήσης της οικολογικής πολιτικής. Τα συνολικά ευρήματα της αξιολόγησης του μηχανισμού αυτού, μας καθιστούν σαφές τις μεγάλες δυνατότητες και προοπτικές που διαμορφώνονται από μια αποτελεσματική διαχείριση της πληροφορίας σε περιβάλλοντα Νεφών.

Επιπρόσθετα, συγκρίνοντας την απόδοση των δύο συστημάτων που παρουσιάστηκαν (Κεφάλαιο 4 & Κεφάλαιο 5), όσον αφορά τον χρόνο απόκρισης της υπηρεσίας επίβλεψης και παροχής δεδομένων, λάβαμε τα αποτελέσματα που φαίνονται στο Σχήμα 62. Η υπηρεσία Monitoring Index (υλοποιημένη με Java SOAP), του μηχανισμού επίβλεψης πολλαπλών επιπέδων αποδίδει λίγο παραπάνω από 1 δευτερόλεπτο ακόμα και με πολλαπλές παράλληλες αιτήσεις. Από την άλλη, η υπηρεσία του δεύτερου μηχανισμού με υλοποίηση Java REST επιδεικνύει πολύ μεγαλύτερη απόδοση, με τιμές ακόμα και μικρότερες από το μισό. Καταλήγουμε λοιπόν ότι χρήση του *RESTful* προγραμματιστικού παραδείγματος μπορεί να βελτιώσει σημαντικά την απόδοση μια υπηρεσίας δεδομένων και όπως θα συζητήσουμε παρακάτω, είναι στα μελλοντικά σχέδιά μας να το υιοθετήσουμε και για το πρώτο σύστημα.



Σχήμα 62: Σύγκριση χρόνου απόκρισης των δύο υπηρεσιών επίβλεψης

7.2 Μελλοντικά βήματα

Οι μεθοδολογίες και υλοποιήσεις που αναπτύχθηκαν στο πλαίσιο αυτής της διατριβής μπορούν να επεκταθούν ή να συνδυαστούν ώστε να προκύψουν νέα ενισχυμένα συστήματα. Όσον αφορά τον μηχανισμό επίβλεψης πολλαπλών στρωμάτων, είναι μέσα στα μελλοντικά σχέδιά μας να ενισχύσουμε και επεκτείνουμε το προτεινόμενο σύστημα προκειμένου να περιληφθεί ένας πιο εύκαμπτος και φιλικός προς το χρήστη μηχανισμός επιβολής πολιτικών (*policy enforcement*). Με τη χρήση ενός καταλόγου πολιτικών (*registry*) ο οποίος δυναμικά θα μπορούσε να διαμορφώνεται, θα επεκτείναμε τις δυνατότητες λήψης αποφάσεων διατηρώντας τον ευέλικτο τρόπο λειτουργίας του συστήματος. Επιπρόσθετα, όπως αποδείχθηκε σε προηγούμενη ενότητα, η απόδοση της υπηρεσίας δεδομένων που παρέχει τις πληροφορίες στους καταναλωτές είναι πολύ μεγαλύτερη χρησιμοποιώντας τεχνολογία REST. Έτσι λοιπόν, είναι στα μελλοντικά μας βήματα η υλοποίηση των βασικών υπηρεσιών του συστήματος επίβλεψης βάσει του αρχιτεκτονικού παραδείγματος REST έτσι ώστε να βελτιώσουμε την απόδοση του σημαντικά και να επιτύχουμε καλύτερες (*real-time*) συνθήκες λειτουργίας.

Σχετικά με τον υπολογισμό και την αξιολόγηση της ενεργειακής απόδοσης που παρουσιάστηκε στην δεύτερη προτεινόμενη λύση επίβλεψης, σκοπεύουμε να επικεντρωθούμε στην βελτίωση της πρόβλεψης της ενεργειακής κατανάλωσης. Συγκεκριμένα, για να ελαχιστοποιήσουμε την απόκλιση που παρατηρήθηκε στο Σχήμα 50, χρειάζεται να συνυπολογίσουμε τη χρησιμοποίηση της μνήμης και του δίσκου κατά τη διαδικασία υπολογισμού της μελλοντικής κατανάλωσης ενεργείας. Έτσι θα επιτύχουμε καλύτερες τιμές πρόβλεψης και για την κατανάλωση αλλά και για την ενεργειακή απόδοση έπειτα. Επίσης, ένα ακόμα θέμα μελλοντικής εργασίας και έρευνας είναι η εισαγωγή τεχνολογιών κατανεμημένης αποθήκευσης δεδομένων για την αποθήκευση των πληροφοριών επίβλεψης. Ο όγκος των ιστορικών δεδομένων αυξάνεται σημαντικά κατά τη διάρκεια εκτέλεσης μιας εφαρμογής και έτσι η χρήση τοπικών βάσεων δεδομένων δεν είναι η βέλτιστη λύση. Σκοπεύουμε να εξετάσουμε τεχνολογίες όπως το *Apache Hadoop Distributed Filesystem*

(HDFS) το οποίο επιτρέπει την κατανεμημένη αποθήκευση και επεξεργασία δεδομένων με πολύ υψηλό ρυθμό απόδοσης.

8 *Βιβλιογραφικές αναφορές*

[1] C. C. E. Group, The future of cloud computing: Opportunities for european cloud computing beyond 2010, Tech. rep., Information Society and Media European Commission (2010).

[2] WSRF Specification, http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

[3] R. T. Fielding. "Architectural styles and the design of network-based software architectures". PhD thesis, University of California, Irvine, 2000.

[4] Amazon AWS, http://aws.amazon.com/

[5] Amazon EC2 API, http://docs.amazonwebservices.com/AWSEC2/2009-11-30/APIReference/

[6] Amazon EC2, http://aws.amazon.com/ec2/

[7] Amazon S3, http://aws.amazon.com/s3/

[8] Amazon SQS, http://aws.amazon.com/sqs/

[9] CloudFront, http://aws.amazon.com/cloudfront/

[10] Simple DB, http://aws.amazon.com/simpledb/

[11] VMware vCloud, http://www.vmware.com/products/vcloud/

[12] vCloud API Programming Guide, http://communities.vmware.com/servlet/JiveServlet/previewBody/12463-102-1-13007/vCloud_API_Guide.pdf

[13] Google App Engine, http://code.google.com/appengine/

[14] Google App Engine Java API, http://code.google.com/appengine/docs/java/overview.html

[15] Google App Engine Python API, http://code.google.com/appengine/docs/python

[16] Milojičić, Dejan; Llorente, Ignacio M.; Montero, Ruben S.; , "OpenNebula: A Cloud Management Tool," Internet Computing, IEEE , vol.15, no.2, pp.11-14, March-April 2011

[17] Rochwerger, B.; Breitgand, D.; Levy, E.; Galis, A.; Nagin, K.; Llorente, I. M.; Montero, R.; Wolfsthal, Y.; Elmroth, E.; Caceres, J.; Ben-Yehuda, M.; Emmerich, W.; Galan, F.; , "The Reservoir model and architecture for open federated cloud computing," IBM Journal of Research and Development , vol.53, no.4, pp.4:1-4:11, July 2009

[18] OCCI, http://www.occi-wg.org

[19] OCCI Specification, http://forge.ogf.org/sf/docman/do/listDocuments/projects.occi-wg/docman.root.drafts.occi_specification

[20] Windows Azure, http://www.microsoft.com/windowsazure/windowsazure/

[21] SUN Cloud, http://developers.sun.com/cloud/

[22] jClouds, http://code.google.com/p/jclouds/

[23] Deltacloud, http://www.deltacloud.org/

[24] libCloud, http://ci.apache.org/projects/libcloud

[25] Elastic Block Storage, http://aws.amazon.com/ebs/

[26] CloudWatch, http://aws.amazon.com/cloudwatch/

[27] Relational Database Service, http://aws.amazon.com/rds/

[28] Auto Scaling Service, http://aws.amazon.com/autoscaling/

[29] Elastic Load Balancing Service, http://aws.amazon.com/elasticloadbalancing/

[30] Azure Diagnostic Manager , http://www.cerebrata.com/Products/AzureDiagnosticsManager/Default.aspx

[31] Azure Subscription Manager, http://communities.quest.com/docs/DOC-9911

[32] CloudStatus, http://www.hyperic.com/products/cloud-status-monitoring

[33] Sacerdoti, F.D. et al., 2003, “Wide area cluster monitoring with Ganglia”, Proceedings IEEE Cluster Computing.

[34] Ganglia Monitoring System, http://ganglia.info.

[35] XDR, External Data Representation Standard, Sun Microsystems 1987, http://tools.ietf.org/html/rfc1014

[36] RRD, Round Robin Database tool, http://oss.oetiker.ch/rrdtool/

[37] Nagios 1996, http://www.nagios.org.

[38] Nagios plug-in development guidelines 2000, http://nagiosplug.sourceforge.net/developer-guidelines.html.

[39] Nagios Enterprises - Nagios XI. http://www.nagios.com/products/nagiosxi.

[40] Hyperic HQ, 2004, http://www.hyperic.com.

[41] HQ Inventory Model, 2009, http://support.hyperic.com/display/DOC/HQ+Inventory+Model

[42] Hyperic Community, http://www.hyperic.com/community.

[43] Lattice Monitoring Framework, 2009, http://clayfour.ee.ucl.ac.uk/lattice.

[44] Stuart Clayman et. al., 2010, Monitoring Virtual Networks with Lattice, IEEE Network Operations and Management Symposium Workshops.

[45] AutoI project, 2008, http://ist-autoi.eu/autoi/index.php.

[46] Zenoss, 2005, Open Source IT management. http://community.zenoss.org/docs/DOC-2614.

[47] Zope, http://www.zope.org.

[48] 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.

[49] Zanikolas, S., & Sakellariou, R. (2005). "A taxonomy of grid monitoring systems", Future Generation Computer Systems 21, 163-188

[50] 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.

[51] H. B. Newman, I. Legrand, P. Galvez, R. Voicu, C. Cirstoiu, "Monalisa : A distributed monitoring service architecture", CoRR cs.DC/0306096

[52] 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", MCS Preprint #ANL/MCS-P1315-0106, http://www.mcs.anl.gov/uploads/cels/papers/P1315.pdf (2006).

[53] S. Microsystems, Jini architecture specification, version 1.2 (2001).

[54] OASIS wsrf v1.2, http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf, (2009).

[55] OASIS WS Notifation 1.2, http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf, 2004.

[56] OGF Glue specification, http://www.ogf.org/documents/GFD.147.pdf, (2009).

[57] 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).

[58] 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.

[59] S. Gogouvitis, K. Konstanteli, S. Waldschmidt, G. Kousiouris, G. Katsaros, A. Menychtas,D. Kyriazis, T. Varvarigou, Workflow management for soft real-time interactive applications in virtualized environments, Future Generation Computer Systems 28 (1) (2012) 193– 209. doi:10.1016/j.future.2011.05.017.

[60] 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

[61] Foster, I. (2005), "Globus Toolkit Version 4: Software for Service-Oriented Systems", in 'IFIP International Conference on Network and Parallel Computing' , Springer-Verlag, pp. 2-13.

[62] 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.

[63] 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.

[64] G. Heward, I. Muller, J. Han, J.-G. Schneider, S. Versteeg, Assessing the performance impact of service monitoring, Software Engineering Conference, 2010.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] T. Cucinotta, F. Checconi, G. Kousiouris, D. Kyriazis, T. Varvarigou, A. Mazzetti, Z. Zlatev, J. Papay, M. Boniface, S. Berger, D. Lamp, T. Voith, M. Stein, “Virtualised e-learning with real-time guarantees on the Irmos platform”, 2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2010, pp. 1–8.

[71] M. Boniface, B. Nasser, J. Papay, S. Phillips, A. Servin, Z. Zlatev, K. X. Yang, G. Katsaros,K. Konstanteli, G. Kousiouris, A. Menychtas, D. Kyriazis, S. Gogouvitis, “Platform-as-a-service architecture for real-time quality of service management in clouds”, in: Internet and Web Applications and Services, ICIW 2010 Fiftih International Conference, 2010. URL http://eprints.ecs.soton.ac.uk/21078/

[72] US Environmental Protection Agency, Report to Congress on Server and Data Center Energy Efficiency, Tech. rep., Public Law (2007).

[73] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L. M. Vaquero, K. Nagin, B. Rochwerger, Monitoring Service Clouds in the Future Internet, IOS Press, 2010.

[74] M. Lindner, F. Galán, C. Chapman, S. Clayman, D. Henriksson, E. Elmroth, The Cloud Supply Chain: A Framework for Information, Monitoring, Accounting and Billing, in: 2nd International ICST Conference on Cloud Computing (CloudComp 2010), 2011.

[75] G. Heward, I. Muller, J. Han, J.-G. Schneider, S. Versteeg, Assessing the performance impact of service monitoring, Software Engineering Conference, Australian 0 (2010) 192–201. doi:http://doi.ieeecomputersociety.org/10.1109/ASWEC.2010.28.

[76] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The Computer Journal 53 (7) (2010) 1045–1051. http://comjnl.oxfordjournals.org/content/53/7/1045.full.pdf+html, doi:10.1093/comjnl/bxp080.

[77] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems.

[78] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, 2010, pp. 826 –831. doi:10.1109/CCGRID.2010.46.

[79] Y. Lee, A. Zomaya, Energy efficient utilization of resources in cloud computing systems, The Journal of Supercomputing (2006) 1–1310.1007/s11227-010-0421-3. URL http://dx.doi.org/10.1007/s11227-010-0421-3

[80] A. Kansal, F. Zhao, J. Liu, N. Kothari, A. A. Bhattacharya, Virtual machine power metering and provisioning, in: Proceedings of the 1st ACM symposium on Cloud computing, SoCC '10, ACM, New York, NY, USA, 2010, pp. 39–50. doi:http://doi.acm.org/10.1145/1807128.1807136.

[81] C. Liu, J. Huang, Q. Cao, S. Wan, C. Xie, Evaluating energy and performance for server-class hardware configurations, in: Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on, 2011, pp. 339 –347. doi:10.1109/NAS.2011.23.

[82] D. Pfoser, N. Tryfona, V. Verykios, Services-based data management in a global computing environment, in: Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on, 2003, pp. 45–53. http://dx.doi.org/10.1109/WISEW.2003.1286785 doi:10.1109/WISEW.2003.1286785.

[83] L. Field, S. Andreozzi, B. Konya, Grid information system interoperability: The need for a common information model, in: eScience, 2008. eScience '08. IEEE Fourth International Conference on, 2008, pp. 501–507. http://dx.doi.org/10.1109/eScience.2008.159 doi:10.1109/eScience.2008.159.

[84] S. Andreozzi, M. Canaparo, M. Carpene, Glueman: A web-based framework for information providers in grid services, in: Enterprise Distributed Object Computing Conference Workshops, 2008 12th, 2008, pp. 377–384. http://dx.doi.org/10.1109/EDOCW.2008.34 doi:10.1109/EDOCW.2008.34.

[85] OGF, Glue Working Group, GLUE Specification v. 2.0, www.ogf.org/documents/GFD.147.pdf, (2009).

[86] OGF, JSDL Working Group, Activity Instance Description Specification v1.0, (2009).

[87] D-GRID Project Consortium, D-MON Project, http://www.d-grid-gmbh.de/index.php?id=52&L=1, (2009).

[88] NORDU GRID, NORDU Grid Project, http://www.nordugrid.org/, (2011).

[89] Open Virtualization Format Specification - OVF, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.0.0.pdf, (2009)

[90] Badia Sala, R., Goiri Presa,Í., Guitart Fernández, J., Mac´ ıas Lloret, M., Torres Viˆ nals, J., Ayguadé Parra, E., Ejarque, J., Sirvent Pardell, R., Lezzi, D.: Emotive: the BSC"s engine for cloud solutions (2009)

[91] libvirt: The virtualization api (July 2011), http://libvirt.org/

[92] Katsaros, G., Kübert, R., Gallizo, G.: Building a service-oriented monitoring framework with rest and nagios. In: 2011 IEEE International Conference on Services Computing (SCC) (2011)

[93] Racktivity energy switch family, URL http://www.racktivity.com/products/energyswitch.php, 2012

[94] A. Josang, http://folk.uio.no/josang/sl/Subjective logic (2011). http://folk.uio.no/josang/sl/

[95] M. Stepnicka, B. De Baets, L. Noskova, Arithmetic fuzzy models, Fuzzy Systems, IEEE Transactions on 18 (6) (2010) 1058–1069. http://dx.doi.org/10.1109/TFUZZ.2010.2062522

[96] F. J. Nieto, A Trust Model for Services in Federated Platforms, Vol. 76 of Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, 2011, pp. 118–131.

[97] A. Josang, Prospectives for online trust management http://www.dmtf.org/sites/default/files/standards/documents/DSP0243P0243_1.0.0.pdf, (2007).

[98] G. Gary Stoneburner, A. Goguen, A. Feringa, Risk management guide for information technology systems (2002).

[99] Leadership in Energy and Environmental Design (LEED), USGB (2011).

[100] S. Rivoire, M. A. Shah, P. Ranganathan, C. Kozyrakis, Joulesort: A balanced energy efficiency benchmark, in: Proceedings of the ACM SIGMOD international conference on Management of data, SIGMOD '07, ACM, New York, NY, USA, 2007, pp. 365–376. doi:10.1145/1247480.1247522.

[101] X. Fan, W.D. Weber, L. A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th annual international symposium on Computer architecture, ISCA '07, ACM, New York, NY, USA, 2007, pp. 13–23. doi:10.1145/1250662.1250665.

[102] L. Liu, H. Wang, X. Liu, et al., GreenCloud: a New Architecture for Green Data Center, in: 6th International Conference on Autonomic Computing and Communications, Barcelona, Spain, 2009, pp. 29–38.

[103] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the 2008 conference on Power aware computing and systems, Berkeley, CA, USA, 2008, pp. 10–10.

[104] OPTIMIS EU project, Optimis - Optimized infrastructure services (2012). URL http://www.optimis-project.eu/

[105] A. J. Ferrer, F. Hernandez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgo, T. Sharif, C. Sheridan, Optimis: A holistic approach to cloud service provisioning, Future Generation Computer Systems 28 (1) (2012) 66 – 77. doi:10.1016/j.future.2011.05.022.

[106] Unixbench benchmark suite, URL http://code.google.com/p/byte-unixbench/, 2011

[107] stress Workload Generator, URL http://weather.ou.edu/ apw/projects/stress/. 2009

[108] NDOUTILS Documentation Version 1.4, 2007, nagios.sourceforge.net/docs/ndoutils/NDOUtils.pdf

[109] Ethan Galstad, NRPE Documentation, 2007, nagios.sourceforge.net/docs/nrpe/NRPE.pdf

[110] Nagvis Project, 2011, http://www.nagvis.org.

[111] Robert W. Ingraham, "The Nagios 2.X Event Broker Module API", http://nagios.sourceforge.net/download/contrib/documentation/misc/NEB%202x%20Module%20API.pdf, 2006.

[112] cURL and Libcurl API, http://curl.haxx.se/libcurl/

[113] Baun, C., & Kunze, M. , Building a private cloud with Eucalyptus, 2009

[114] Freeman, T. L. D. M. P. B. J. K. K. Nimbus Elastic Scaling in the Clouds. Retrieved from http://www.nimbusproject.org/files/epu_poster4.pdf.

[115] Borja Sotomayor, Ruben S. Montero, Ignacio M. Llorente, & Ian Foster (2009). Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing, 13, 14–22. doi: 10.1109/MIC.2009.119.

[116] BonFire Project (2010). Bonfire | BUILDING SERVICE TESTBEDS ON FIRE. Retrieved from http://www.bonfire-project.eu/.

[117] Reservoir Project (2010). Reservoir fp7. Retrieved from http://62.149.240.97/.

[118] Ludmila Cherkasova (1999). FLEX: Design and Management Strategy for Scalable Web Hosting Service. HP LABORATORIES REPORT, 1999-64.

[119] Matthias Gelbman (2010). Highlights of web technology surveys, July 2010: CentOS is now the most popular Linux distribution on web servers. Retrieved from http://w3techs.com/blog/entry/highlights_of_web_technology_surveys_july_2010.

[120] OpenStack project (2010). OpenStack Open Source Cloud Computing Software. Retrieved from http://www.openstack.org/.

[121] Inlab Software GmbH (2010). Balance. Retrieved from http://www.inlab.de/balance.html.

[122] Costa Walcott (2005). Taking a load off: Load balancing with balance. Retrieved from http://www.linux.com/archive/feature/46735.