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

Μηχανισμός Αξιολόγησης και Επιλογής Υπολογιστικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Β. Παπαγεωργίου

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Οκτώβριος 2009

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

Μηχανισμός Αξιολόγησης και Επιλογής Υπολογιστικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Β. Παπαγεωργίου

Επιβλέπων : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Οκτωβρίου 2009.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Βασίλειος Λούμος	Ελευθέριος Καγιάφας

Αθήνα, Οκτώβριος 2009

.....

Κωνσταντίνος Β. Παπαγεωργίου

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

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία σχεδιάζεται και υλοποιείται ένα μηχανισμός αξιολόγησης και επιλογής υπολογιστικών νεφών. Ο μηχανισμός αυτός σχεδιάστηκε σαν υπηρεσία ιστού και υλοποιήθηκε σε γλώσσα προγραμματισμού Java και με την πλατφόρμα Gria Development Kit. Η χρήση του μπορεί να γίνει με την βοήθεια του Gria Client με την προσθήκη ειδικού plug-in που επίσης υλοποιήθηκε στα πλαίσια της διπλωματικής.

Ο σχεδιασμός του συγκεκριμένου μηχανισμού έχει σαν στόχο την συνολική εξυπηρέτηση των χρηστών που επιθυμούν να εκτελέσουν κάποια εργασία σε υπολογιστικό νέφος, από την φάση της αναζήτησης και δέσμευσης πόρων μέχρι την εκτέλεση της εργασίας και τη λήψη των αποτελεσμάτων. Ο χρήστης σε ειδικό περιβάλλον θέτει τις απαιτήσεις του όσον αφορά τις παραμέτρους του QoS και ο μηχανισμός ανάμεσα σε πληθώρα νεφών που διατηρεί σε μια βάση δεδομένων επιλέγει εκείνο που ταιριάζει καλύτερα στις ανάγκες του χρήστη.

Ο μηχανισμός φιλοξενείται σε ένα εξυπηρετητή και εκτελείται σαν υπηρεσία ιστού, ενώ το κομμάτι της υλοποίησης που αφορά τον πελάτη (Gria Client) εξυπηρετεί τις ανάγκες αλληλεπίδρασης με τον χρήστη, δηλαδή προσδιορισμού των προτιμήσεων για το QoS και παρουσίασης των αποτελεσμάτων εκτέλεσης του μηχανισμού.

Λέξεις- Κλειδιά: Υπολογιστικό νέφος, μηχανισμός επιλογής, αξιολόγηση αξιοπιστίας, cloud computing, Gria, Utility Computing, Eucalyptus, Amazon EC2

ABSTRACT

The objective of this study is the design and implementation of an evaluation and selection mechanism for computing clouds. The mechanism was designed as a web service and was implemented in Java programming language with the platform Gria Development Kit. It can be used with Gria Client including a special plug-in which was also implemented for this purpose.

The design of this mechanism aims to help user that want to conduct a computing task in a computing cloud, starting from searching and allocating resources till the execution of the task and getting the results. The user sets his QoS preferences regarding the cloud and the mechanism selects the one that best fits his needs, out of a variety of computing clouds that are stored in a local database.

The mechanism is hosted on a server as a web service, while the part that regards the client (Gria Client) is used to interact with the user, which is settings the QoS preferences and showing the selection mechanism results.

Keywords: Computing cloud, selection mechanism, reliability evaluation, cloud computing, Gria, Utility Computing, Eucalyptus, Amazon EC2

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εκπονήθηκε υπό την επίβλεψη του διδάκτορα Ε.Μ.Π. Ανδρέα Μενύχτα. Πρωτίστως λοιπόν θα ήθελα να ευχαριστήσω τον κ. Μενύχτα για την βοήθεια και καθοδήγηση του κατά την διάρκεια της συγγραφής της διπλωματικής η οποία αποτελεί και δική του έμπνευση. Θα ήθελα επίσης να ευχαριστήσω την καθηγήτρια Ε.Μ.Π. Θεοδώρα Βαρβαρίγου που μου έδωσε την δυνατότητα να ασχοληθώ με τον συγκεκριμένο τομέα της επιστήμης υπολογιστών.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια παρατηρείται μια αλλαγή στον τρόπο αντιμετώπισης των υπηρεσιών υπολογιστών. Προγράμματα και δεδομένα πλέον απομακρύνονται από τους προσωπικούς υπολογιστές και εκτελούνται και αποθηκεύονται σε συστοιχίες απομακρυσμένων υπολογιστών, τα αποκαλούμενα και υπολογιστικά νέφη. Η στροφή σε αυτό τον τύπο οργάνωσης είναι ακόμα σε αρχικό στάδιο , η αγορά λογισμικού για προσωπικούς υπολογιστές δεν πρόκειται να εξαφανιστεί σύντομα, αλλά οι συνεχείς καινοτομίες πάνω στον τομέα μας οδηγούν στο συμπέρασμα ότι οδηγούμαστε με σταθερά βήματα προς αυτή την κατεύθυνση.

Η αλλαγή αυτή αναμένεται επηρεάσει όλους όσους εμπλέκονται στην αγορά υπολογιστών από τους απλούς χρήστες, μέχρι τους προγραμματιστές, τους IT managers ακόμα και τους κατασκευαστές hardware για υπολογιστές. Βρισκόμαστε μπροστά σε μια σημαντική αλλαγή. Παρόμοια αλλαγή συνέβη για πρώτη φορά πριν από 50 χρόνια, με την δημιουργία των πρώτων time-sharing συστημάτων τα οποία επέτρεπαν πρόσβαση στις υπολογιστικές δυνατότητες των mainframes ακόμα και σε χρήστες που δεν διέθεταν ένα δικό τους. Ιδιώτες με χρήση τερματικών επικοινωνούσαν μέσω τηλεφωνικών γραμμών με ένα υπολογιστικό κέντρο όπου πραγματοποιούνται όλες οι εργασίες. Η άφιξη των προσωπικών υπολογιστών(PC) την δεκαετία του 1980 έφερε μαζί της την ελπίδα της «απελευθέρωσης» των προγραμμάτων και δεδομένων από τα κέντρα αυτά. Παρόλα αυτά τα PC είχαν μια εμφανή αδυναμία: δεν υπήρχε εύκολος τρόπος για συνεργασία και διαμοίραση των δεδομένων.

Με την νέα τάση, οι υπολογιστικές λειτουργίες μεταφέρονται σε απομακρυσμένα datacenters που είναι προσβάσιμα μέσω του διαδικτύου. Η διαφορά με τα time-sharing συστήματα είναι ότι πλέον ένας χρήστης μπορεί μέσω του διαδικτύου να επικοινωνεί με πολλά νέφη ταυτόχρονα, κάποια από τους οποίους είναι δυνατόν να ανταλλάσουν πληροφορίες μεταξύ τους.

Από τα πιο σημαντικά και δημοφιλή υπολογιστικά νέφη που είναι διαθέσιμα για τους χρήστες αυτή τη στιγμή είναι το νέφος της Amazon, ενώ παράλληλα ειδικό λογισμικό ανοιχτό ανοιχτού κώδικα με την ονομασία Eucalyptus, προσομοιώνει την λειτουργία του Amazon και επιτρέπει σε οποιοδήποτε χρήστη να δημιουργήσει το δικό του υπολογιστικό νέφος και να το θέσει στην διάθεση του κοινού.

1.1. ΣΚΟΠΟΣ

Η διπλωματική αφορά στην μελέτη των σύγχρονων τάσεων όσον αφορά τα υπολογιστικά νέφη και την υλοποίηση υπηρεσίας ιστού η οποία παρακολουθεί και αξιολογεί διαθέσιμα υπολογιστικά νέφη και κατόπιν αιτήματος ενός χρήστη, μπορεί και προτείνει αυτό που ταιριάζει στις ανάγκες του.

Αρχικά για τους σκοπούς της διπλωματικής έγινε μια μελέτη γύρω από τα υπολογιστικά νέφη. Μελετήθηκαν οι κατηγορίες στις οποίες διακρίνονται και οι διαφορές μεταξύ τους. Καταγράφηκαν τα πιο σημαντικά νέφη που υπάρχουν αυτή τη στιγμή στην αγορά και πραγματοποιήθηκε ένα σύντομο συγκριτικό τους.

Ακολούθως γίνεται παρουσίαση της πλατφόρμας ανοιχτού κώδικα για υπολογιστικά νέφη, του Eucalyptus. Μελετήθηκε η δομή του, οι διαχειριστικές δυνατότητες που προσφέρει και οι τρόποι αλληλεπίδρασης μαζί του. Η κατανόηση του τρόπου λειτουργίας του Eucalyptus είναι κρίσιμη καθώς με αυτόν τον τρόπο γίνεται δυνατή η αξιολόγηση των διαφορετικών νεφών που χρησιμοποιούν την πλατφόρμα αυτή.

Στην συνέχεια γίνεται ο σχεδιασμός του συστήματος. Οι σχεδιαστικές επιλογές που έγιναν οδήγησαν στην δημιουργία του μηχανισμού αξιολόγησης και επιλογής υπολογιστικού νέφους. Ο συγκεκριμένος μηχανισμός έχει σαν σκοπό την πλήρη αξιολόγηση των υπηρεσιών που είναι σε θέση να παρέχει ένα νέφος , ενώ παράλληλα επιβάλλεται να είναι όσο το δυνατόν πιο απλοϊκός και εύκολος στην χρήση, ώστε να μπορεί να το χρησιμοποιήσει ο οποιοσδήποτε χωρίς ειδικές γνώσεις πάνω σε υπολογιστικά νέφη.

Το σχεδιασμό ακολούθησε η υλοποίηση για υπολογιστικό περιβάλλον Linux. Κάποια από τα υποσυστήματα του μηχανισμού χρησιμοποιούν εργαλεία τα οποία είναι platform dependent και ως εκ τούτου δεν ήταν δυνατή η δημιουργία υλοποίησης που να λειτουργεί κάτω από οποιοδήποτε λειτουργικό σύστημα.

1.2. ΔΟΜΗ ΤΟΥ ΚΕΙΜΕΝΟΥ

Για την πλήρη περιγραφή του μηχανισμού , είναι απαραίτητη η εξέταση ζητημάτων τα οποία καθόρισαν την ανάγκη δημιουργίας του μηχανισμού αυτού. Η περιγραφή ξεκινά από το δεύτερο κεφάλαιο το οποίο παρέχει όλες τις πληροφορίες γύρω από τις σύγχρονες τάσεις στα υπολογιστικά νέφη.

Πιο συγκεκριμένα το πρώτο κομμάτι του κεφαλαίου αφορά την περιγραφή των σύγχρονων τάσεων για παροχή δικτυακών υπηρεσιών που βασίζονται σε υπολογιστικά νέφη, του επονομαζόμενου cloud computing. Στο δεύτερο κομμάτι γίνεται αναφορά στους σύγχρονους παρόχους cloud computing και επιχειρείται μια σύγκριση μεταξύ των κυριοτέρων από αυτούς.

Στο τρίτο κεφάλαιο περιγράφεται η πλατφόρμα Eucalyptus, πλατφόρμα πάνω στην οποία βασιστήκαμε για την δημιουργία του μηχανισμού αξιολόγησης και επιλογής. Περιγράφεται η δομή και ο τρόπος αλληλεπίδρασης μαζί της.

Στο τέταρτο κεφάλαιο περιγράφεται ο σχεδιασμός του μηχανισμού. Περιγράφονται όλα τα επιμέρους υποσυστήματα του μηχανισμού και οι λόγοι που μας οδήγησαν στην συγκεκριμένη δομή, ενώ στο επόμενο κεφάλαιο παρουσιάζουμε αποτελέσματα εκτέλεσης του μηχανισμού αξιολόγησης και επιλογής σε δεδομένο dataset.

Στο έκτο κεφάλαιο έχουμε μια επίδειξη χρήσης της υπηρεσίας συνοδευόμενη από τα απαραίτητα screenshots, και τέλος στο έβδομο κεφάλαιο μετά από μια σύνοψη της εργασίας γίνεται αναφορά σε πιθανές προοπτικές επέκτασης του μηχανισμού.

ΚΕΦΑΛΑΙΟ 2

ΣΥΓΧΡΟΝΕΣ ΤΑΣΕΙΣ ΣΤΗΝ ΠΑΡΟΧΗ ΥΠΗΡΕΣΙΩΝ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΑ ΝΕΦΗ

Το κεφάλαιο αυτό έχει ως σκοπό την εισαγωγή στην έννοια του υπολογιστικού νέφους. Πραγματοποιείται επεξήγηση και διασαφήνιση όρων όπως cloud computing, utility computing, Software as a Service, που χρησιμοποιούνται ευρέως στις σύγχρονες υπηρεσίες ιστού και επιχειρείται μια προσπάθεια επεξήγησης των συνθηκών που δημιούργησαν αυτή τη νέα γενιά υπηρεσιών. Αναφέρονται τα πλεονεκτήματα του υπολογιστικού νέφους, καταγράφονται οι διαφορετικές κατηγορίες υπηρεσιών υπολογιστικού νέφους και τέλος γίνεται μια περιγραφή των μεγαλύτερων σύγχρονων παρόχων και μια σύγκρισή τους.

2.1. CLOUD COMPUTING

Ο όρος *cloud computing* ή *on-demand computing* είναι ένας νέος όρος[1] ο οποίος αναφέρεται τόσο στις εφαρμογές που είναι διαθέσιμες μέσω Internet με την μορφή μιας υπηρεσίας ιστού όσο και στο hardware και το λογισμικό που βρίσκεται σε datacenters και χρησιμοποιείται για την παροχή αυτών των υπηρεσιών. Οι εφαρμογές συνήθως αναφέρονται ως *Software as a Service (SaaS)* ενώ το λογισμικό και το hardware στα datacenters είναι αυτό που αποκαλούμε νέφος-*Cloud*[10].Το cloud computing μπορεί να χαρακτηριστεί αν όχι απλά συγγενές, απόγονος του grid computing[48,49,50,51].

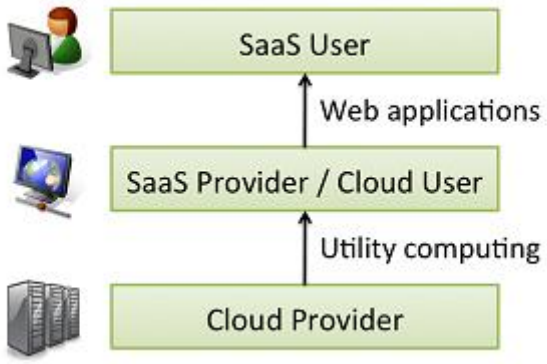
Τα υπολογιστικά νέφη διακρίνονται σε τρείς κύριες κατηγορίες ανάλογα την οργάνωση τους και τον σκοπό που εξυπηρετούν[2].Όταν το νέφος είναι διαθέσιμο επί πληρωμή στο ευρύ καταναλωτικό κοινό αποκαλείται δημόσιο νέφος-*Public Cloud*, ενώ η υπηρεσία που πωλείται αποκαλείται *Utility Computing*. Τα κυριότερα παραδείγματα public cloud που υπάρχουν σήμερα είναι το Amazon Web Services[12], το Google AppEngine[13] και το Microsoft Azure[14]. Στην περίπτωση που το νέφος δεν είναι διαθέσιμο στο κοινό αλλά αποτελεί το datacenter μιας επιχείρησης ή ενός οργανισμού τότε αποκαλείται ιδιωτικό νέφος-*Private Cloud*. Πέραν των private και public cloud, υπάρχουν και τα υβριδικά νέφη, τα αποκαλούμενα hybrid cloud. Τα hybrid cloud προκύπτουν από συνδυασμό private και public cloud. Ο πιο συνηθισμένος λόγος ύπαρξης hybrid cloud είναι όταν το

private cloud ενός οργανισμού δεν μπορεί να ανταποκριθεί σε απότομες αυξομειώσεις workflow , οπότε μέρος της εργασίας κατανέμεται σε κάποιο εξωτερικό pubic cloud για ικανοποίηση των προσωρινών αυξημένων απαιτήσεων. Το cloud computing περιλαμβάνει το SaaS και Utility Computing αλλά δεν περιλαμβάνει τα private cloud.

Είναι εύκολο να διαπιστώσουμε τα πλεονεκτήματα του SaaS τόσο για του χρήστες όσο και για τους παρόχους των υπηρεσιών[3,4,5]. Τους παρόχους εξυπηρετεί η εύκολη εγκατάσταση του λογισμικού, η εύκολη συντήρηση και η κεντρική διαχείριση. Οι χρήστες από την άλλη, έχουν την δυνατότητα πρόσβασης στην υπηρεσία από οποιοδήποτε σημείο, οποιαδήποτε στιγμή , τη δυνατότητα να μοιράζονται εύκολα δεδομένα με άλλους χρήστες και αποθήκευσης τους σε ασφαλές σημείο. Το cloud computing δεν προσφέρει κάτι διαφορετικό , απλά δίνει τη δυνατότητα στους παρόχους να προσφέρουν το προϊόν τους σαν SaaS χωρίς να έχουν δικό τους datacenter.

Πιο αναλυτικά όσον αφορά τη χρήση του hardware το Cloud computing παρέχει τις παρακάτω δυνατότητες[17]:

- Το Cloud computing δημιουργεί την ψευδαίσθηση της ύπαρξης άπειρων υπολογιστικών πόρων. Οι χρήστες έχουν την δυνατότητα να αυξομειώνουν τους πόρους που χρησιμοποιούν κάθε στιγμή και έτσι δεν απαιτείται από τους χρήστες να κάνουν μακροπρόθεσμο σχεδιασμό των απαιτήσεων τους σε hardware.



ΕΙΚΟΝΑ 1:ΧΡΗΣΤΕΣ ΚΑΙ ΠΑΡΟΧΟΙ CLOUD COMPUTING

- Οι χρήστες του Cloud computing δεν δεσμεύονται με μακρόχρονες συμφωνίες όσον αφορά την χρήση των πόρων[9]. Με αυτό τον τρόπο δίνεται η δυνατότητα σε εταιρείες να ξεκινούν με μικρές συμφωνίες και να αυξάνουν τους χρησιμοποιούμενους υπολογιστικούς πόρους όσο αυξάνονται και οι απαιτήσεις τους.

- Η δυνατότητα που υπάρχει να χρεώνεται η χρήση του υπολογιστικών πόρων με βάση βραχυπρόθεσμες συμφωνίες(π.χ. χρέωση επεξεργαστών /ώρα χρήσης) οδηγεί στην αποδέσμευση υπολογιστικών πόρων όταν πλέον δεν υπάρχει ανάγκη χρήσης τους.

Ένα επιτυχημένο παράδειγμα χρήσης των παραπάνω είναι το Elastic Compute Cloud (EC2) που παρέχεται από την Amazon Web Services[6]. Το EC2 παρέχει έναν υπολογιστικό κόμβο με επεξεργαστή 1.0 GHz 2007 Opteron, 1.7GB RAM και 160GB αποθηκευτικό χώρο προς $0.10 την ώρα. Το Amazon Simple Storage Service (S3) χρεώνει $0.12-$0.15 ανά GB για κάθε μήνα χρήσης ενώ $0.10-$0.17 ανά GB εισερχόμενων ή εξερχόμενων δεδομένων.

2.2. CLOUD COMPUTING: ΓΙΑΤΙ ΤΩΡΑ;

Αν και η δημιουργία και λειτουργία μεγάλης κλίμακας datacenters ήταν καταλυτικός παράγοντας για το cloud Computing, είναι οι τρέχουσες τεχνολογικές τάσεις και νέα επιχειρηματικά μοντέλα που το ώθησαν στο να επικρατήσει αυτή τη χρονική περίοδο[7,8,11].

Κάποια χρόνια πριν, η πραγματοποίηση πληρωμής με χρήση πιστωτικής κάρτας απαιτούσε μια συμφωνία με μια υπηρεσία επεξεργασία πληρωμών όπως η VeriSign ή η Authorize.net. Η συμφωνία ήταν κομμάτι μιας μεγαλύτερης εμπορικής συμφωνίας, καθιστώντας το αδύνατο για ένα ιδιώτη ή μια μικρή επιχείρηση να κάνει διαδικτυακές αγορές με χρήση πιστωτικής κάρτας. Με τον ερχομό όμως του Web2.0 και τη δημιουργία του PayPal[15], κάθε ιδιώτης μπορεί να πραγματοποιήσει αγορές χωρίς να είναι απαραίτητη η ύπαρξη κάποιας μακροχρόνιας συμφωνίας , με μηδαμινές επιπλέον χρεώσεις. Για παράδειγμα με τη χρήση του Google AdSense[16], οποιοσδήποτε ιδιώτης διαθέτει δικιά του ιστοσελίδα, μπορεί αποκτήσει έσοδα από διαφημίσεις. Το Amazon Web Services εδραιώθηκε το 2006 με χρήση αυτής της δυνατότητας: οι χρήστες δεν απαιτείται να έχουν συμβόλαιο ή κάποια μακροχρόνια συμφωνία με την Amazon. Το μόνο που απαιτείται να έχουν είναι μια πιστωτική κάρτα.

Αν και με την έλευση του Cloud Computing πιστεύεται ότι θα δημιουργηθούν νέοι τύποι εφαρμογών, είναι πολύ πιθανό ορισμένοι ήδη υπάρχοντες τύποι να αξιοποιήσουν το cloud computing , συμβάλλοντας με αυτό τον τρόπο στην περαιτέρω εδραίωσή του.

- ***Φορητές διαδραστικές εφαρμογές:*** Έχει διατυπωθεί η άποψη ότι «το μέλλον ανήκει στις υπηρεσίες που ανταποκρίνονται σε πραγματικό χρόνο σε πληροφορίες που παρέχονται είτε από χρήστες είτε από μη-ανθρώπινους παράγοντες»[18]. Τέτοιες υπηρεσίες εξυπηρετούνται καλύτερα από υποδομές cloud όχι μόνο γιατί πρέπει να είναι διαθέσιμες όσο το δυνατόν περισσότερο χρόνο αλλά γιατί τέτοιου τύπου υπηρεσίες χρησιμοποιούν μεγάλο όγκο δεδομένων που εύκολα αποθηκεύονται στα μεγάλα datacenters. Κάτι τέτοιο ισχύει σε μεγαλύτερο βαθμό για εφαρμογές που

χρησιμοποιούν πολλές πηγές δεδομένων ή ακόμα και άλλες υπηρεσίες όπως π.χ. τα mashups.

- ***Παράλληλη επεξεργασία δεδομένων:*** Το cloud computing παρέχει μια μοναδική ευκαιρία για εφαρμογές επεξεργασίας δεδομένων οι οποίες επεξεργάζονται Terabyte δεδομένων και απαιτούν πολλές ώρες για να ολοκληρωθούν. Αν υπάρχει δυνατότητα παραλληλοποίησης της εφαρμογής , οι χρήστες μπορούν να εκμεταλλευτούν τις χρεώσεις των παρόχων : η χρήση εκατοντάδων υπολογιστών για μια μικρή περίοδο κοστίζει όσο η χρήση μερικών υπολογιστών για μεγαλύτερη περίοδο. Ορισμένα εργαλεία όπως το MapReduce της Google[19] και το πρόγραμμα ανοιχτού κώδικα Hadoop[20], επιτρέπουν σε χρήστες να υλοποιούν τέτοιες εφαρμογές χωρίς να γνωρίζουν λεπτομέρειες για την πραγματική κατανομή των εργασιών αυτών στους εκατοντάδες εξυπηρετητές του υπολογιστικού νέφους. Η ανάλυση κόστους/απόδοσης πρέπει να λαμβάνει υπ' όψιν το κόστος της μεταφοράς δεδομένων στο νέφος σε σχέση με την πιθανή επιτάχυνση στην επεξεργασία των δεδομένων.

- ***Επέκταση εμπορικών εφαρμογών:*** Οι τελευταίες εκδόσεις των εφαρμογών Mathematica και Matlab επιτρέπουν την χρήση Cloud Computing για την πραγματοποίηση υπολογισμών με μεγάλο υπολογιστικό κόστος. Άλλες εφαρμογές θα μπορούσαν με τον ίδιο τρόπο να ωφεληθούν ενσωματώνοντας την δυνατότητα να «τρέχουν» σε νέφος. Και σε αυτή την περίπτωση είναι απαραίτητο να ληφθεί υπ' όψιν το κόστος της μεταφοράς δεδομένων όπως και της χρήσης επεξεργαστικών κόμβων στο νέφος σε σχέση με το κέρδος σε χρόνο. Τα symbolic mathematics παρουσιάζουν ιδιαίτερο ενδιαφέρον καθώς απαιτούν μεγάλο επεξεργαστικό κόστος ανά μονάδα δεδομένων. Ένα ενδιαφέρον εναλλακτικό μοντέλο θα ήταν να διατηρούμε τα δεδομένα στο νέφος και να βασιζόμαστε στο διαθέσιμο εύρος ζώνης για να παρουσιάζουμε ένα αποκρίσιμο GUI στον χρήστη. Ένα παρόμοιο παράδειγμα είναι το rendering ενός 3D animation, όπου το επεξεργαστικό κόστος ανά μονάδα δεδομένων είναι πολύ μεγάλο.

- ***Εφαρμογές που απαιτούν άμεση απόκριση:*** Ορισμένες εφαρμογές που θα μπορούσαν να αξιοποιήσουν τον παραλληλισμό του νέφους , περιορίζονται λόγο του κόστους μεταφοράς δεδομένων και το χρόνο απόκρισης του. Για παράδειγμα, ενώ τα υπολογιστικά μοντέλα που χρησιμοποιούνται για την αξιολόγηση μακροπρόθεσμων οικονομικών αποφάσεων είναι κατάλληλα για χρήση σε νέφος, η αγοραπωλησία μετοχών που απαιτεί ακρίβεια δεκάτων του δευτερολέπτου δεν είναι . Τέτοιες εφαρμογές πιθανότατα δεν θα αξιοποιήσουν τις συγκεκριμένες υποδομές

τουλάχιστον μέχρι το κόστος μεταφοράς δεδομένων στο νέφος αλλά και η ταχύτητα απόκρισης του να μειωθούν .

2.3. ΠΟΙΟΣ ΜΠΟΡΕΙ ΝΑ ΓΙΝΕΙ ΠΑΡΟΧΟΣ CLOUD COMPUTING;

Ενώ δεν φαίνεται δύσκολο να βρεθούν χρήστες cloud computing, ποιος θα γινόταν πάροχος cloud computing και γιατί; Η δημιουργία και η συντήρηση τέτοιων εγκαταστάσεων είναι μια πολυδάπανη και για αυτό μόνο μεγάλες εταιρείες που δραστηριοποιούνται στο διαδίκτυο όπως η Amazon, Google, eBay και η Microsoft διαθέτουν δικές τους. Παράλληλα οι εταιρείες αυτές έπρεπε να αναπτύξουν το κατάλληλο λογισμικό(MapReduce, GoogleFileSystem, BigTable, Dynamo[19,21,22,23]) και υποδομές το οποίο χρησιμεύει για την διαχείρισή τους αλλά και τους βοηθά να προστατέψουν την επένδυσή τους από φυσικές και ηλεκτρονικές επιθέσεις.

Αυτός είναι και ο λόγος ότι προκειμένου μια εταιρεία να γίνει πάροχος cloud computing πρέπει να έχει επενδύσει όχι μόνο σε πολύ μεγάλα datacenters αλλά και στο κατάλληλο λογισμικό και να διαθέτει την απαραίτητη τεχνογνωσία που απαιτείται για να διατηρείς τέτοιες υποδομές. Με αυτές τις προϋποθέσεις, κάποιοι λόγοι που μπορεί να ωθήσουν μια εταιρεία να γίνει πάροχος cloud computing είναι :

- ***Σημαντικό κέρδος:*** Αν και $0.10 την ώρα(Amazon Web Services) μοιάζει μικρό ποσό, υπολογίζεται ότι πολύ μεγάλα datacenters (δεκάδων χιλιάδων υπολογιστών) μπορούν να αγοράσουν hardware, σύνδεση στο διαδίκτυο και ηλεκτρική παροχή στο 1/5 ή ακόμα και στο 1/7 της τιμής[24] που προσφέρεται σε μεσαίου μεγέθους datacenters (εκατοντάδων ή χιλιάδων υπολογιστών). Το κόστος για την ανάπτυξη και εγκατάσταση του απαραίτητου λογισμικού μπορεί να αποσβεστεί λόγω του μεγάλου αριθμού υπολογιστών.

- ***Ενίσχυση υπάρχουσας επένδυσης:*** Η προσθήκη υπηρεσιών Cloud Computing σε μια υπάρχουσα υποδομή δημιουργεί μια νέα πηγή εσόδων με (ιδανικά) μικρό κόστος, βοηθώντας στην εξόφληση μεγάλων επενδύσεων σε datacenters. Για παράδειγμα πολλές από τις τεχνολογίες του Amazon Web Services είχαν δημιουργηθεί αρχικά για την εξυπηρέτηση εσωτερικών λειτουργιών της Amazon[25].

- ***Επίθεση στο κατεστημένο:*** Μια εταιρεία με το απαιτούμενο datacenter και λογισμικό , μπορεί να επιθυμεί να προσφέρει κάτι εναλλακτικό σε σχέση με τα υπάρχοντα νέφη. Για παράδειγμα η Google με το GoogleApps παρέχει τη δυνατότητα αυτόματου scalability και load balancing διευκολύνοντας τους χρήστες του νέφους, οι οποίοι σε άλλους παρόχους θα έπρεπε να υλοποιήσουν τις συγκεκριμένες δυνατότητες μόνοι τους.

- ***Ενίσχυση των υφιστάμενων σχέσεων με τους πελάτες:*** Εταιρείες παροχής υπηρεσιών πληροφορικής έχουν εκτεταμένες σχέσεις με τους πελάτες τους μέσω των υπηρεσιών που προσφέρουν. Η επιπλέον παροχή cloud computing υπηρεσιών θα βοηθούσε στην διατήρηση και ενίσχυση των σχέσεων αυτών.

2.4. ΚΑΤΗΓΟΡΙΕΣ CLOUD COMPUTING ΚΑΙ ΠΑΡΟΧΟΙ

Κάθε εφαρμογή απαιτεί ένα συγκεκριμένο υπολογιστικό μοντέλο, ένα μοντέλο για την αποθήκευση δεδομένων και ,στην περίπτωση διαμοιρασμένων εφαρμογών, ένα μοντέλο επικοινωνίας. Στην περίπτωση του cloud computing η χρήση virtual machines είναι απαραίτητη καθώς οι διεργασίες διαχείρισης των υπολογιστικών πόρων που εκτελούνται στο νέφος πρέπει να είναι κρυφές από τον χρήστη. Ένας τρόπος διαχωρισμού των utility computing προσφορών που υπάρχουν αυτή τη στιγμή θα ήταν με κριτήριο το πόσο ουσιαστικό έλεγχο δίνουν στον χρήστη στην διαχείριση των υπολογιστικών πόρων.

Σε αυτή την περίπτωση πρώτο στη λίστα είναι το *Amazon EC2* και *Enomaly*[34]. Ένα instance EC2 είναι όσο το δυνατόν πιο κοντά στο hardware δίνοντας τη δυνατότητα στο χρήστη να ελέγχει ακόμα και τον πυρήνα του λειτουργικού που θα χρησιμοποιήσει. Το ορατό στον χρήστη API είναι ελάχιστο, περιλαμβάνοντας μόνο ορισμένες εντολές για την ρύθμιση του hardware. Δεν υπάρχει όριο στις εφαρμογές που μπορεί να χρησιμοποιήσει ο χρήστης αλλά από την άλλη η Amazon δεν προσφέρει δυνατότητα για auto-scaling και επαναφοράς σε περίπτωση ανεπιθύμητου τερματισμού της εφαρμογής, καθώς αυτές είναι δυνατότητες που εξαρτώνται από την εκάστοτε εφαρμογή.

Στο άλλο άκρο από το Amazon EC2 είναι τα *Google AppEngine, Force.com[26] , Rackspace Cloud[27], Engine Yard Cloud[28].* Οι συγκεκριμένες πλατφόρμες δεν στοχεύουν σε γενικού τύπου εφαρμογές, αλλά είναι ιδανικές για web εφαρμογές παρέχοντας δυνατότητες auto-scaling. Ειδικά το Force.com έχει σχεδιαστεί ειδικά για εμπορικές εφαρμογές οι οποίες χρησιμοποιούν την salesforce.com βάση δεδομένων και τίποτα περαιτέρω.

Η ενδιάμεση λύση είναι το *Microsoft Azure*. Οι εφαρμογές του Azure στηρίζονται στις βιβλιοθήκες .NET , και γίνονται compile με τη βοήθεια του Common Language Runtime. Το σύστημα της Microsoft εξυπηρετεί εφαρμογές γενικού σκοπού, δίνοντας την δυνατότητα στους χρήστες να επιλέξουν την γλώσσα προγραμματισμού που θα χρησιμοποιήσουν αλλά όχι και το λειτουργικό σύστημα. Οι βιβλιοθήκες παρέχουν ως ένα βαθμό τη δυνατότητα για auto-scaling και αυτόματης ρύθμισης του δικτύου, κατόπιν κάποιων ρυθμίσεων από τον χρήστη.

Παρά τον διαχωρισμό που παρατηρείται, δεν μπορεί κάποιος να ισχυριστεί ότι μια πλατφόρμα είναι καλύτερη από μια άλλη. Ισχύει αναλογικά ότι ισχύει και στις γλώσσες προγραμματισμού και τα frameworks. Οι γλώσσες χαμηλού επιπέδου όπως η C και η assembly επιτρέπουν τον πλήρη έλεγχο του διαθέσιμου hardware, αλλά αν κάποιος επιθυμεί να γράψει μια εφαρμογή Web, οι διαχείριση των sockets και των requests είναι δύσκολη ακόμα και με χρήση έτοιμων βιβλιοθηκών. Από την άλλη frameworks υψηλού επιπέδου όπως το Ruby on Rails,

δεν απαιτούν από τον προγραμματιστή να ασχοληθεί καθόλου με τη διαχείριση των requests, αλλά σε περίπτωση που η εφαρμογή δεν είναι απόλυτα συμβατή με το framework , ο προγραμματισμός μπορεί να γίνει αρκετά πολύπλοκος. Κανένας προγραμματιστής Ruby on Rails δεν θα αρνούνταν την υπεροχή της C σε συγκεκριμένες περιπτώσεις και το αντίστροφο. Με τον ίδιο τρόπο και στην περίπτωσή μας, διαφορετικές εφαρμογές θα απαιτήσουν και διαφορετικές πλατφόρμες utility computing.

Επικεντρώνοντας στις 3 μεγάλες πλατφόρμες utility computing (Amazon Web Services, Microsoft Azure, Google AppEngine) παρουσιάζουμε αναλυτικά στοιχεία για τις δυνατότητες τους στον παρακάτω πίνακα:

	Amazon Web Services	Microsoft Azure	Google AppEngine
Υπολογιστικό μοντέλο(VM)	• x86 Αρχιτεκτονική με χρήση Xen VM • Η ελαστικότητα στους υπολογιστικούς πόρους επιτρέπει scalability, την οποία όμως πρέπει να υλοποιήσει ο χρήστης ή να την εξασφαλίσει μέσω κάποιοι μεσάζοντα(π.χ.Rightscale)	• Microsoft Common Language Runtime (CLR) VM • Οι VM παρέχονται ανάλογα τις προδιαγραφές που έχει θέσει ο χρήστης. Δίνεται η δυνατότητα για automatic load balancing.	• Προκαθορισμένο framework και δομή εφαρμογής: όλα τα εργαλεία του προγραμματιστή γραμμένα σε Python. • Αυτόματο scale up και scale down των υπολογιστικών πόρων και των πόρων αποθήκευσης
Μοντέλο Αποθήκευσης	• Ποικίλει από block store(EBS) σε επαυξημένο key/blob store(SimpleDB) • Το automatic scaling ποικίλει από καθόλου scaling ή διαμοίραση(EBS) ως πλήρως αυτόματο (SimpleDB,S3), ανάλογα ποιο μοντέλο χρησιμοποιείται • Εγγυήσεις για την συνοχή των δεδομένων ποικίλλουν πολύ ανάλογα το μοντέλο που χρησιμοποιείται • Τα API ποικίλουν από ευρέως χρησιμοποιούμενα (EBS) ως πιο ειδικά	• SQL Data Services • Azure Storage Service	• MegaStore/BigTable
Μοντέλο	• Δημοσιοποίηση της IP τοπολογίας. Οι	• Αυτόματο ανάλογα την	• Σταθερή τοπολογία για

Δικτύου	λεπτομέρειες αποκρύβονται από τον χρήστη. • Η ύπαρξη security groups μας δίνει την δυνατότητα να ελέγχουμε ποιοι κόμβοι μπορούν να επικοινωνούν • Οι Availability Zones μας δίνουν τη δυνατότητα απομόνωσης πιθανών βλαβών του δικτύου • Οι ελαστικότητα στις IP διευθύνσεις	περιγραφή των components της εφαρμογής	να στηρίξει τη δομή των Web εφαρμογών • Το scale up ή scale down είναι αυτόματο και κρυμμένο από τον προγραμματιστή

2.5. ΑΝΑΓΚΗ ΓΙΑ ΔΗΜΙΟΥΡΓΙΑ ΙΔΙΩΤΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΝΕΦΩΝ

Μέχρι στιγμής έχουμε δει ότι σε περίπτωση που μια επιχείρηση το θελήσει, υπάρχουν στην αγορά μεγάλοι πάροχοι cloud computing που μπορεί να εξυπηρετήσουν στην δημιουργία της κατάλληλης IT υποδομής. Ωστόσο, μεγάλοι IT οργανισμοί προτιμούν να επωφεληθούν των πλεονεκτημάτων του cloud computing δημιουργώντας υποδομές οι οποίες θα λειτουργούν εντός του οργανισμού[29,30]. Οι λόγοι είναι εν τέλει η αίσθηση ασφάλειας και ελέγχου, καθώς οι περισσότεροι μηχανικοί υπολογιστών προτιμούν να στηρίζονται στις δυνατότητες τους για την πραγματοποίηση και διεκπεραίωση κρίσιμων εργασιών παρά να εξαρτώνται από – όσο οργανωμένοι και καλοπροαίρετοι και αν είναι αυτοί- ξένους.

Στο παρελθόν οι εφαρμογές έτρεχαν σε dedicated servers και ειδικά συστήματα αποθήκευσης. Πλέον οι εφαρμογές μπορούν να μεταφερθούν σε μεγάλο αριθμό εξυπηρετητών και συστήματα αποθήκευσης χάρη στην ανάπτυξη των virtual machines. Η διαχείριση των virtual machines δεν είναι κάτι δύσκολο, αλλά η διαχείριση του live migration των virtual machines αποτελεί μια τροχοπέδη στην δημιουργία του private cloud.

Τη λύση σε αυτό το πρόβλημα το έδωσε η cloud computing πλατφόρμα ανοιχτού κώδικα, *Eucalyptus*[31,32]. Το Eucalyptus είναι λογισμικό που σχεδιάστηκε από την αρχή ώστε να είναι εύκολο στην εγκατάσταση και όσο το δυνατόν λιγότερο παρεμβατικό στο υπάρχον σύστημα, χωρίς να απαιτεί από τους κόμβους στους οποίους τρέχει να του αφιερώνουν αποκλειστικούς πόρους. Το εξωτερικό interface του Eucalyptus βασίζεται στο API του πολύ δημοφιλούς Amazon EC2, ενώ παρέχει ένα στρώμα εικονικού δικτύου το οποίο επιτρέπει την απομόνωση της δικτυακής κίνησης διαφορετικών χρηστών.

ΚΕΦΑΛΑΙΟ 3

EUCALYPTUS

Το Eucalyptus πρόκειται για λογισμικό ανοιχτού κώδικα το οποίο δημιουργήθηκε για να προσφέρει τη δυνατότητα υλοποίησης cloud computing υπηρεσίας με χρήση clusters υπολογιστών. Το τρέχον interface του ,είναι συμβατό με τα interface των EC2, S3, EBS[39,40] της Amazon. Το Eucalyptus υλοποιήθηκε χρησιμοποιώντας κοινά εργαλεία του Linux και βασικές τεχνολογίες υπηρεσιών ιστού, κάτι που επιτρέπει την εύκολη εγκατάσταση και συντήρηση του [33]. Ξεκίνησε αρχικά σαν πανεπιστημιακό project στο " Computer Science Department at the University of California , Santa Barbara", αλλά πλέον υποστηρίζεται από την εταιρεία "Eucalyptus Systems"[31,46,47] –μια εταιρεία που ιδρύθηκε από τους αρχικούς δημιουργούς για να συνεχιστεί η προσπάθεια.

3.1. ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ EUCALYPTUS

Η αρχιτεκτονική του Eucalyptus είναι απλή, ευέλικτη με ιεραρχική σχεδίαση[44,45]. Ουσιαστικά το σύστημα επιτρέπει σε ένα χρήστη να ξεκινάει, να ελέγχει , να έχει πρόσβαση σε και να τερματίζει virtual machines[41,42,43] χρησιμοποιώντας μια προσομοίωση του Amazon EC2 interface. Αυτό σημαίνει ότι οι χρήστες του Eucalyptus αλληλεπιδρούν με το σύστημα χρησιμοποιώντας τα ακριβώς ίδια εργαλεία και interface που χρησιμοποιούν για να αλληλεπιδράσουν με το Amazon EC2. Προς το παρόν υποστηρίζει virtual machines που τρέχουν σε Xen[35] και KVM[36] hypervisor , αλλά στο μέλλον αναμένεται να προστεθεί υποστήριξη και για VMware[37] καθώς και άλλους.

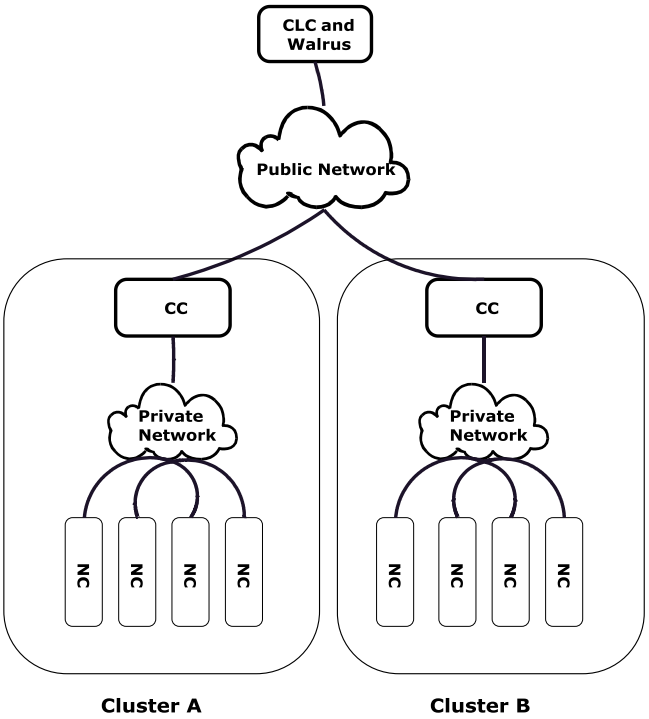
Το Eucalyptus υλοποιεί κάθε υποσύστημα του συστήματος με την μορφή μιας αυτοτελούς υπηρεσίας ιστού. Υπάρχουν συνολικά 4 βασικά υποσυστήματα, το καθένα με το δικό του Web-service interface, τα οποία απαρτίζουν το Eucalyptus:

- ***Node Controller:*** είναι υπεύθυνο για την εκτέλεση, επιτήρηση και τερματισμό των VM instances στον κόμβο όπου τρέχει
- ***Cluster Controller:*** συλλέγει πληροφορίες και χρονοδρομολογεί την εκτέλεση των VM σε συγκεκριμένους node controllers, ενώ διαχειρίζεται και το virtual instance δίκτυο

- ***Storage Controller (Walrus):*** είναι μια υπηρεσίας ιστού η οποία χρησιμοποιείται για την αποθήκευση και ανάκτηση δεδομένων υλοποιώντας το Amazon S3 interface, παρέχοντας ένα μηχανισμό για την αποθήκευση και πρόσβαση σε virtual machine images και δεδομένων του χρήστη.

- ***Cloud Controller:*** είναι το σημείο «εισόδου» στο νέφος τόσο για τους χρήστες του όσο και για τον διαχειριστή. Παίρνει πληροφορίες από τους κόμβους για τους διαθέσιμους πόρους, παίρνει αποφάσεις σχετικά με την χρονοδρομολόγηση διεργασιών , τις οποίες υλοποιεί με αιτήσεις του προς τους cluster controllers.

Όσα αναφέρουμε, φαίνονται περιγραφικά και στην Εικόνα 2:



ΕΙΚΟΝΑ 2: ΙΕΡΑΡΧΙΚΗ ΔΟΜΗ ΤΟΥ EUCALYPTUS

3.2. NODE CONTROLLER

Ο *Node Controller (NC)* τρέχει σε κάθε κόμβο στον οποίο προορίζεται να τρέχουν virtual machine instances. Ένας NC ελέγχει και χρησιμοποιεί το λογισμικό στον κόμβο(π.χ. το λειτουργικό σύστημα και τον hypervisor) ανάλογα τις εντολές που λαμβάνει από τον Cluster Controller.

Ένας NC συλλέγει πληροφορίες για τους πόρους τους κόμβου που βρίσκεται – τον αριθμό των πυρήνων, το μέγεθος της μνήμης, τον διαθέσιμο χώρο στο δίσκο- αλλά ελέγχει και την κατάσταση των virtual machine instances που βρίσκονται στον κόμβο (αν και ο NC παρακολουθεί τις instances που ελέγχει, είναι πιθανό instances να ξεκινήσουν και να τερματιστούν χωρίς την παρέμβαση του NC

με χρήση άλλων μηχανισμών). Οι πληροφορίες αυτές συλλέγονται και προωθούνται στο Cluster Controller κατόπιν αιτήματός του.

Ο Cluster Controller ελέγχει τις virtual machine instances σε ένα κόμβο με χρήση των εντολών *runInstance* και *terminateInstance* στον NC του κόμβου. Κατόπιν εξακρίβωσης του αποστολέα της αίτησης –π.χ. μόνο ο administrator και ο χρήστης που την ξεκίνησε είναι σε θέση να τερματίσει μια instance- και μετά τον έλεγχο των διαθέσιμων πόρων, ο NC εκτελεί την εντολή με την βοήθεια του hypervisor. Για να ξεκινήσει μια instance, ο NC κάνει ένα αντίγραφο στον κόμβο των αρχείων image του instance (του πυρήνα του λειτουργικού, το συστήματος αρχείων και της μνήμης), είτε από κάποιο απομακρυσμένο repository είτε από τοπική cache , δημιουργεί ένα σημείο πρόσβασης στο στρώμα του εικονικού δικτύου και δίνει εντολή στον hypervisor να ξεκινήσει την instance. Για να σταματήσει μια instance, ο NC δίνει εντολή στον hypervisor να τερματίσει την VM, καταστρέφει το σημείο πρόσβασης στο εικονικό δίκτυο και διαγράφει τα αρχεία που σχετίζονται με την instance. Αξίζει να αναφερθεί ότι στο Eucalyptus, οι NC που ανήκουν στο ίδιο cluster αποτελούν μια ξεχωριστά διαχειριζόμενη οντότητα, μια *Availability Zone*.

3.3. CLUSTER CONTROLLER

Ο Cluster Controller (CC) συνήθως τρέχει στο front-end υπολογιστή ενός cluster ή σε οποιοδήποτε υπολογιστή έχει δικτυακή σύνδεση τόσο με τους κόμβους που τρέχουν NC όσο και με τον υπολογιστή που τρέχει τον Cloud Controller (CLC). Πολλές από τις λειτουργίες του CC είναι παρόμοιες με τις λειτουργίες του NC αλλά είναι γενικά στον πληθυντικό (π.χ. *runInstances, terminateInstances*) . Ο CC έχει τρεις βασικές διεργασίες να εκτελέσει: να δρομολογήσει τις ερχόμενες αιτήσεις για έναρξη instances σε συγκεκριμένους NC , να ελέγχει το στρώμα εικονικού δικτύου[52] των instances και να συλλέγει/ αναφέρει πληροφορίες για μια ομάδα NC. Όταν ο CC λαμβάνει μια σειρά από instances που πρέπει να τρέξουν, επικοινωνεί με κάθε NC μέσω της εντολής describeResources και στέλνει την εντολή runInstances στον πρώτο NC που έχει αρκετούς ελεύθερους πόρους για να φιλοξενήσει το instance. Όταν ένας CC λαμβάνει μια εντολή describeResources, λαμβάνει παράλληλα και μια σειρά από χαρακτηριστικά (επεξεργαστικοί πυρήνες, μνήμη και χώρος δίσκου) που περιγράφουν τους πόρους που απαιτεί ένα instance. Με χρήση αυτής της πληροφορίας , ο CC υπολογίζει πόσες παράλληλες instances του συγκεκριμένου τύπου μπορούν να εκτελεστούν στους NC που ελέγχει και αναφέρει το νούμερο αυτό στον CLC.

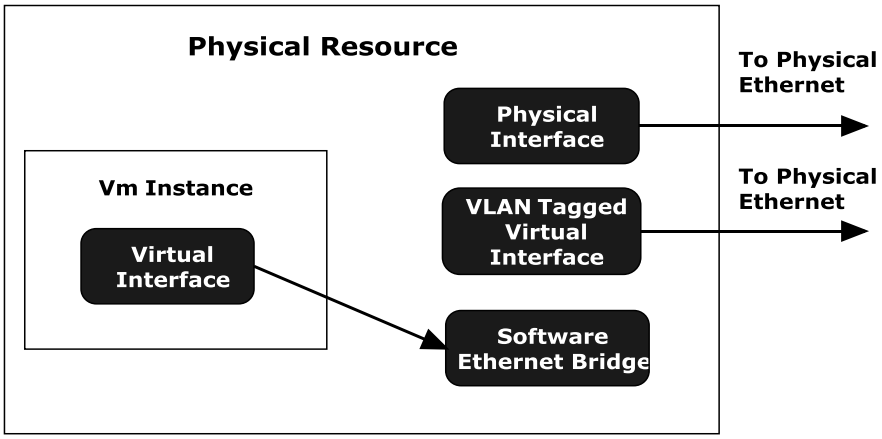
3.4. VIRTUAL NETWORK OVERLAY

Στο Eucalyptus, όλες οι virtual machine έχουν δικτυακή σύνδεση μεταξύ τους ενώ τουλάχιστον μια έχει πρόσβαση στο internet έτσι ώστε ο ιδιοκτήτης να μπορεί να συνδεθεί και να αλληλεπιδράσει μαζί τους. Σε ένα υπολογιστικό νέφος που μπορεί παράλληλα να χρησιμοποιούν πολλοί χρήστες, οι virtual machines που ανήκουν σε μια κατανομή μπορούν να επικοινωνούν ενώ virtual machines

διαφορετικών κατανομών είναι απομονωμένες για λόγους ασφαλείας, ώστε να μην είναι δυνατόν ένας χρήστης να παρεμβληθεί στις machines ενός άλλου χρήστη. Το Eucalyptus το επιτυγχάνει αυτό ,διατηρώντας παράλληλα την επικοινωνία μεταξύ των virtual machines όσο το δυνατόν πιο απλοϊκή, ώστε να μπορεί να εκμεταλλευτεί στο μέγιστο την υψηλή απόδοση των σύγχρονων εικονικών δικτυακών interfaces[53].

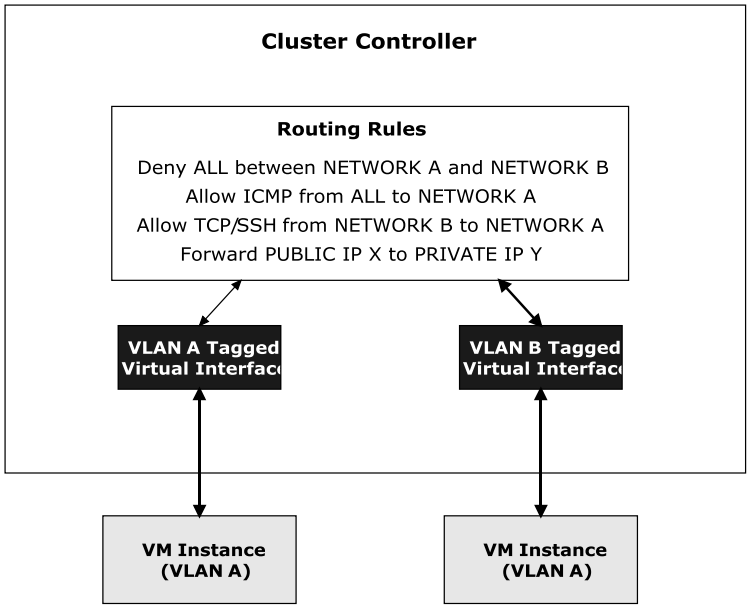
Στο Eucalyptus, το CC ελέγχει την έναρξη και τον τερματισμό των δικτυακών interface των instances με τέσσερις διαφορετικούς τρόπους που ορίζονται από τον διαχειριστή του συστήματος. Στην πρώτη περίπτωση το Eucalyptus προσαρτά το interface της VM κατευθείαν στην Ethernet bridge που συνδέεται στο πραγματικό δίκτυο του κόμβου επιτρέποντας στον διαχειριστή να ελέγχει την DHCP κίνηση του εικονικού δικτύου με τον ίδιο τρόπο που θα διαχειρίζονταν DHCP κίνηση αντικειμένων που δεν ανήκουν στο Eucalyptus. Η δεύτερη επιλογή επιτρέπει στον διαχειριστή να ορίσει ζεύγη MAC και IP διευθύνσεων. Σε αυτή την επιλογή , σε κάθε νέο instance που δημιουργείται από το σύστημα ορίζεται ένα ζεύγος , το οποίο ελευθερώνεται όταν το instance τερματίζεται. Σε αυτές τις δύο περιπτώσεις η απόδοση της διασύνδεσης μεταξύ των VM είναι σχεδόν η βέλτιστη όταν οι VM τρέχουν στο ίδιο cluster αλλά δεν υπάρχει απομόνωση μεταξύ των δικτύων τον διαφορετικών VM .

Στην τρίτη επιλογή το Eucalyptus διαχειρίζεται και ελέγχει πλήρως τα δίκτυα VM , παρέχοντας απομόνωση της κίνησης μεταξύ των VM και τον ορισμό firewalls μεταξύ ομάδων VM , ενώ υποστηρίζει τη δυναμική παραχώρηση IP διευθύνσεων στις VM κατά την εκκίνηση και την λειτουργία τους. Σε αυτή την περίπτωση στους χρήστες επιτρέπεται κατά την εκκίνηση των VM να τις προσαρτούν σε ένα δίκτυο που ορίζεται από τον χρήστη. Σε κάθε τέτοιο δίκτυο δίνεται ένα μοναδικό VLAN χαρακτηριστικό από το Eucalyptus (Εικόνα 3) , καθώς και ένα μοναδικό υποδίκτυο IP από ένα εύρος υποδικτύων που έχει ορίσει ο διαχειριστής του Eucalyptus από πριν. Με αυτό τον τρόπο κάθε ομάδα VM που ανήκουν σε ένα υποδίκτυο απομονώνεται από τις VM που ανήκουν σε άλλο υποδίκτυο, όπως αυτό ορίζεται από το VLAN χαρακτηριστικό και την IP.



ΕΙΚΟΝΑ 3:ΣΕ ΚΑΘΕ VM INSTANCE ΤΟΥ EUCALYPTUS ΔΙΝΕΤΑΙ ΕΝΑ VIRTUAL INTERFACE ΠΟΥ ΣΥΝΔΕΕΤΑΙ ΜΕ ΤΗΝ ETHERNET BRIDGE ΤΟΥ ΚΟΜΒΟΥ, ΣΤΟ ΟΠΟΙΟ ΤΟ VLAN INTERFACE ΣΥΝΔΕΕΤΑΙ ΠΕΡΑΙΤΕΡΩ.

Το CC λειτουργεί σαν δρομολογητής μεταξύ των υποδικτύων VM με πρωταρχική επιλογή να μπλοκάρει την IP κίνηση μεταξύ των υποδικτύων VM. Αν ο χρήστης το επιθυμεί μπορεί να επιτραπεί για παράδειγμα η κίνηση πακέτων ICMP(ping) από και προς το δημόσιο Internet επιτρέποντας μόνο SSH συνδέσεις μεταξύ των VM που ελέγχει. Το CC χρησιμοποιεί το σύστημα φιλτραρίσματος πακέτων του Linux iptables[54] . Τέλος να σημειώσουμε ότι όλες οι VM σε αυτό τον τρόπο λειτουργίας παίρνουν IP διεύθυνση από μια προκαθορισμένη λίστα από εσωτερικές IP διευθύνσεις και για αυτό το λόγο δεν μπορούν να συνδεθούν με εξωτερικά συστήματα. Προκειμένου να διαχειριστεί αυτή την κατάσταση το CC επιτρέπει στον διαχειριστή να ορίσει μια λίστα από δημόσιες IPv4 διευθύνσεις που δεν χρησιμοποιούνται παρέχοντας την δυνατότητα στους χρήστες να αιτούνται IP από αυτή τη λίστα για τις VM τους, όταν αυτές ξεκινούν. Σε αυτή την περίπτωση χρησιμοποιείται το Linux iptables προκειμένου να ορίσει NAT, DNAT και SNAT κανόνες για την μετάφραση δημοσίων IP διευθύνσεων σε ιδιωτικές IP διευθύνσεις. (Εικόνα 4). Η τελευταία διαθέσιμη επιλογή είναι παρόμοια με αυτή που μόλις περιγράφηκε με την διαφορά ότι δεν παρέχει απομόνωση των δικτύων των VM.



ΕΙΚΟΝΑ 4:ΤΟ CC ΧΡΗΣΙΜΟΠΟΙΕΙ ΤΟ LINUX IPTABLES ΣΥΣΤΗΜΑ ΦΙΛΤΡΑΡΙΣΜΑΤΟΣ ΠΑΚΕΤΩΝ ΓΙΑ ΝΑ ΕΠΙΤΡΕΠΕΙ ΣΕ ΧΡΗΣΤΕΣ ΝΑ ΟΡΙΖΟΥΝ ΚΑΝΟΝΕΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕΤΑΞΥ ΤΩΝ VM ΔΙΚΤΥΩΝ ΚΑΙ ΝΑ ΑΝΑΘΕΤΕΙ IP ΔΙΕΥΘΥΝΣΕΙΣ ΔΥΝΑΜΙΚΑ ΚΑΤΑ ΤΗΝ ΕΚΚΙΝΗΣΗ ΤΩΝ VM

Από άποψη ταχύτητας επικοινωνίας, δύο VM επιτυγχάνουν την μέγιστη δυνατή όταν βρίσκονται στο ίδιο cluster και ανήκουν στο ίδιο εικονικό δίκτυο. Στην περίπτωση που VM που ανήκουν σε διαφορετικά εικονικά δίκτυα θέλουν να επικοινωνήσουν, όλη η κίνηση θα περάσει μέσα από τον CC ο οποίος λειτουργεί σαν δρομολογητής IP. Με αυτό τον τρόπο δίνεται η δυνατότητα στον

χρήστη να επιλέξει , ανάλογα τις ανάγκες του, είτε μέγιστη απόδοση χωρίς περιορισμούς στην επικοινωνία μεταξύ των VM, είτε ελαφρά μειωμένη απόδοση ελέγχοντας ωστόσο την επικοινωνία μεταξύ των VM.

Όταν οι VM είναι διαμοιρασμένες σε διάφορα clusters, το Eucalyptus παρέχει μηχανισμό σύνδεσης των CC με την χρήση tunneling(για παράδειγμα με χρήση του VTUN[38]. Σε αυτή την περίπτωση, τα πακέτα VLAN από το ένα cluster με tunneling μεταφέρονται στο άλλο με χρήση TCP ή UDP σύνδεσης. Η απόδοση αυτής της επικοινωνίας όπως είναι λογικό εξαρτάται κυρίως από την ταχύτητα διασύνδεσης μεταξύ των cluster. Παρόλα αυτά σε περίπτωση πολύ καλή διασύνδεσης μεταξύ των cluster, η απόδοση του tunnel είναι που μπορεί να επηρεάσει την ταχύτητα επικοινωνίας.

3.5. STORAGE CONTROLLER (WALRUS)

Το Eucalyptus περιλαμβάνει το Walrus , μια service αποθήκευσης δεδομένων που χρησιμοποιεί τεχνολογίες υπηρεσιών ιστού (Axis2, Mule) και έχει συμβατό interface με το Simple Storage Service (S3) της Amazon. Το Walrus υλοποιεί μέσω HTTP το interface REST καθώς και SOAP interface τα οποία είναι συμβατά με το S3. Το Walrus παρέχει δύο ειδών λειτουργίες:

- Οι χρήστες που έχουν πρόσβαση στο Eucalyptus μπορούν να χρησιμοποιήσουν το Walrus για να μεταφέρουν δεδομένα από και προς το νέφος καθώς και από instances που έχουν ξεκινήσει σε κόμβους

- Επιπλέον το Walrus λειτουργεί σαν service αποθήκευσης VM images. Οι images του σύστημα αρχείων, του πυρήνα και της μνήμης που χρησιμοποιούνται για να δημιουργηθούν instance VM σε κόμβους μπορούν να μεταφερθούν στο Walrus και από εκεί και έπειτα να χρησιμοποιηθούν από τους κόμβους.

Οι χρήστες χρησιμοποιούν τα εργαλεία του S3 (είτε τις Amazon είτε άλλων δημιουργών) για να μεταφέρουν δεδομένα προς και από το Walrus. Το σύστημα χρησιμοποιεί τα πιστοποιητικά της βάσης δεδομένων του Cloud Controller.

Όπως και το S3, το Walrus υποστηρίζει παράλληλες και σειριακές μεταφορές δεδομένων. Προκειμένου ενισχυθεί η παραλληλοποίηση το Walrus δεν αναλαμβάνει τον έλεγχο της εγκυρότητας των δεδομένων. Όπως και στο S3 οι χρήστες πρέπει να βεβαιώσουν ότι τα δεδομένα που μεταφέρονται στο Walrus είναι έγκυρα και ότι δεν υπάρχουν διπλές εγγραφές σε ίδια αντικείμενα. Σε περίπτωση που επιχειρηθεί εγγραφή σε ένα αντικείμενο στο οποίο μια άλλη εγγραφή είναι ήδη σε εξέλιξη, η πρώτη εγγραφή ακυρώνεται. Το Walrus αποστέλλει ένα MD5 checksum του αντικειμένου που αποθηκεύτηκε. Από την στιγμή που ο χρήστης έχει εξακριβωθεί σαν έγκυρος χρήστης του Eucalyptus, όλες οι αιτήσεις για εγγραφή και ανάγνωση αντικειμένων αποστέλλονται μέσω HTTP.

Το Walrus επίσης λειτουργεί σαν service αποθήκευσης και διαχείρισης των VM image. Οι images του συστήματος αρχείων, του πυρήνα του λειτουργικού και της μνήμης της VM ενσωματώνονται σε πακέτα και μεταφέρονται με χρήση των εργαλείων EC2 που προσφέρονται από την Amazon. Αυτά τα εργαλεία συμπιέζουν τα image, τα κρυπτογραφούν με χρήση των πιστοποιητικών και τα χωρίζουν σε πολλά κομμάτια που περιγράφονται στο "image description file" ή αλλιώς *manifest*. Το Walrus έχει το καθήκον να επαληθεύσει την εγκυρότητα και να αποκρυπτογραφήσει τα image που έχουν μεταφερθεί από τους χρήστες. Όταν ένα node controller ζητά κάποιο image από το Walrus, προτού το κάνει instantiate, στέλνει μια αίτηση για μεταφορά η οποία πιστοποιείται με χρήση κατάλληλων πιστοποιητικών. Έπειτα το image επαληθεύεται ως προς την εγκυρότητά του ,αποκρυπτογραφείται και τέλος αποστέλλεται στον κόμβο. Για λόγους βελτίωσης της απόδοσης και επειδή συνήθως τα VM image είναι αρκετά μεγάλα, το Walrus διατηρεί μια cache με τα image που έχουν ήδη αποκρυπτογραφηθεί. Η cache διαγράφεται μετά την πάροδο κάποιου διαστήματος ή όταν το manifest του image ξαναγραφτεί.

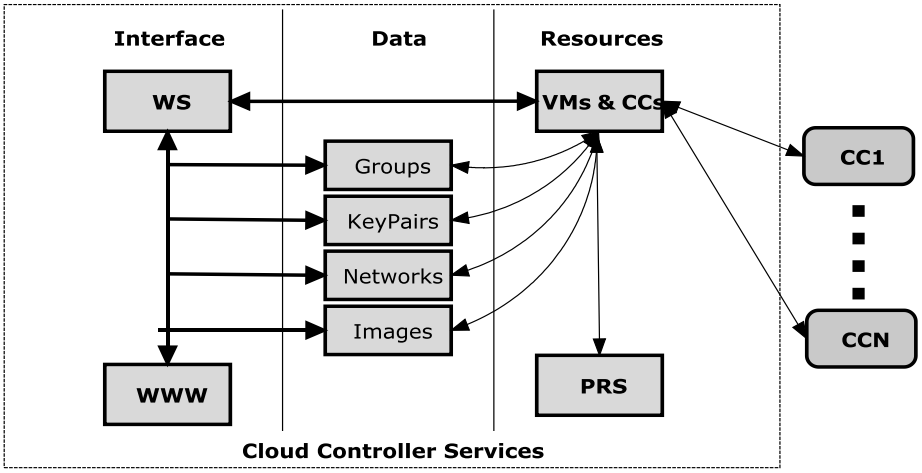
3.6. CLOUD CONTROLLER

Τα στοιχεία που αποτελούν το Eucalyptus και τα οποία περιγράφηκαν παραπάνω διαχειρίζονται από τον Cloud Controller (CLC). Το CLC είναι μια συλλογή από υπηρεσίες ιστού που ανάλογα τον ρόλο του χωρίζονται σε τρεις κατηγορίες:

- ***Resource Services:*** Είναι οι υπηρεσίες που διαχειρίζονται την διανομή των πόρων του συστήματος στις διάφορες λειτουργίες, επιτρέπουν στους χρήστες να διαχειρίζονται τις ιδιότητες των virtual machines, και καταγράφουν τόσο τα components του συστήματος όσο και τους εικονικούς πόρους.

- ***Data Services:*** Ελέγχουν τα δεδομένα του συστήματος και των χρηστών και παρέχουν ένα παραμετροποιήσιμο περιβάλλον στο χρήστη για την πραγματοποίηση των αιτημάτων για κατανομή πόρων.

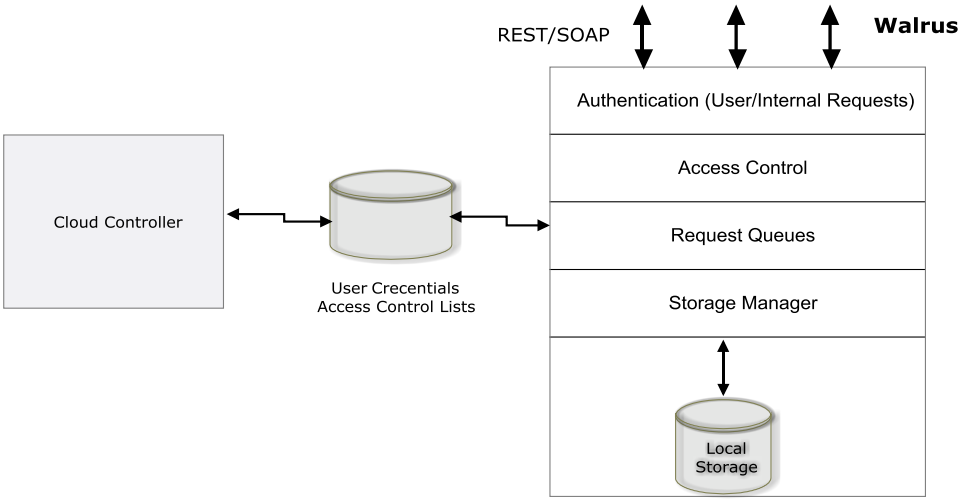
- ***Interface Services:*** Διαχειρίζονται την μετατροπή πρωτοκόλλων, την πιστοποίηση των χρηστών μέσων κατάλληλων interface και παρέχουν εργαλεία διαχείρισης του συστήματος.



ΕΙΚΟΝΑ 5: ΟΙ SERVICES ΤΟΥ CLOUD CONTROLLER. ΟΙ ΜΑΥΡΕΣ ΓΡΑΜΜΕΣ ΔΕΙΧΝΟΥΝ ΤΑ ΜΗΝΥΜΑΤΑ ΧΡΗΣΤΩΝ ΕΝΩ ΟΙ ΑΝΟΙΧΤΟΧΡΩΜΕΣ ΤΑ ΜΗΝΥΜΑΤΑ ΜΕΤΑΞΥ ΤΩΝ SERVICES ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Οι Resource services τις επεξεργάζονται τις αιτήσεις που κάνει ο χρήστης για έλεγχο των VM και επικοινωνούν με τα CC για την δέσμευση και την αποδέσμευση των φυσικών πόρων. Μια εικόνα της κατάστασης των πόρων του συστήματος(System Resource State- SRS) διατηρείται με την κατάλληλη επικοινωνία με τους CC και χρησιμοποιείται για να επαληθευτεί το κατά πόσο είναι δυνατή η πραγματοποίηση των αιτήσεων του χρήστη (με χρήση Service Level Agreements – SLA). Ο έλεγχος του SRS πραγματοποιείται σε δύο στάδια: όταν φτάνουν οι αιτήσεις του χρήστη οι πληροφορίες SRS χρησιμοποιούνται για να αποφασιστεί κατά πόσο θα γίνει αποδεκτή η αίτηση του χρήστη με βάση τα SLA. Η δημιουργία VM κατόπιν περιλαμβάνει την *δέσμευση* των πόρων στο SRS, μεταφορά του αιτήματος για την δημιουργία VM η οποία ακολουθείται από την *επιβεβαίωση* της κατάστασης των πόρων στο SRS σε περίπτωση επιτυχίας ή την ακύρωσή της σε περίπτωση που προκύψει κάποιο σφάλμα.

Το SRS στην συνέχεια παρακολουθεί την κατάσταση των πόρων και είναι υπεύθυνο για την αλλαγή της κατάστασης των VM που τρέχουν. Οι πληροφορίες του SRS τροποποιούνται από γεγονότα που συμβαίνουν στο σύστημα με βάση ένα σύστημα κανόνων που στηρίζεται στα υπάρχοντα SLA. Η εφαρμογή του SLA πυροδοτείται από κάποιο γεγονός (π.χ. αλλάζει μια ιδιότητα του δικτύου ή λήγει κάποιος χρονομέτρης) , το οποίο αξιολογεί και στην συνέχεια τροποποιεί την αίτηση του χρήστη(π.χ. μπορεί να απορρίψει την αίτηση του χρήστη σε περίπτωση που αυτή δεν είναι δυνατόν να ικανοποιηθεί), ή προκαλεί αλλαγές στην κατάσταση του συστήματος (π.χ. κατανομές με χρονικό περιορισμό). Ενώ η αναπαράσταση του συστήματος στο SRS μπορεί να μην αντανακλά πάντα τους πραγματικούς πόρους, η φύση και η πιθανότητα των ανακριβειών επιτρέπει την διόρθωσή τους όταν



ΕΙΚΟΝΑ 6:ΤΟ EUCALYPTUS ΠΕΡΙΛΑΜΒΑΝΕΙ ΤΟ WALRUS, ΕΝΑ SERVICE ΔΙΑΧΕΙΡΙΣΗΣ ΔΕΔΟΜΕΝΩΝ ΣΥΜΒΑΤΟ ΜΕ ΤΟ S3, ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ ΓΙΑ ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΠΡΟΣΒΑΣΗ ΣΕ ΔΕΔΟΜΕΝΑ ΧΡΗΣΤΩΝ ΑΛΛΑ ΚΑΙ IMAGES

εφαρμόζεται το SLA. Επιπλέον ο έλεγχος αποδοχής και τα SLA λειτουργούν μαζί για δύο λόγους: να βεβαιώσουν ότι δεν γίνονται αποδεκτές περισσότερες αιτήσεις χρηστών από όσες μπορεί το σύστημα να ικανοποιήσει και να μειωθεί η πιθανότητα αποτυχίας ικανοποίησης των SLA.

Το Eucalyptus επιτρέπει στους χρήστες να επιλέξουν πιο cluster θέλουν να χρησιμοποιήσουν για να εκκινήσουν τις VM προσδιορίζοντας την “Availability Zone” του νέφους που θέλουν να χρησιμοποιήσουν. Επιπλέον γίνεται να αφήσουν την δυνατότητα επιλογής cluster στο Eucalyptus. Για παράδειγμα με χρήση της Availability Zone “any” , το Eucalyptus θα τοποθετήσει τις καινούργιες Vm στο πιο άδειο cluster , αλλά στην περίπτωση ελλείψεων σε πόρους θα διαμοιράσει τις VM σε πολλαπλά cluster.

Οι Data Services διαχειρίζονται την δημιουργία, τροποποίηση, έλεγχο και αποθήκευση των δεδομένων συστήματος και των χρηστών. Οι χρήστες συνδέονται με αυτά τα services για να ανακαλύψουν πληροφορίες για τους διαθέσιμους πόρους (images και clusters) και για να διαχειριστούν διάφορες παραμέτρους που αφορούν τις virtual machines και την κατανομή των δικτύων. Τα Resource Services αλληλεπιδρούν με τα Data Services για να αποκωδικοποιήσουν παραμέτρους που εισάγουν οι χρήστες (π.χ. τα keys που σχετίζονται με την δημιουργία ενός VM instance) . Παρόλα αυτά οι παράμετροι services αυτών δεν είναι στατικές. Για παράδειγμα οι χρήστες μπορούν να αλλάξουν τους κανόνες που λειτουργούν τα firewall κάτι που επηρεάζει την κίνηση του δικτύου. Οι αλλαγές μπορούν να γίνουν είτε όταν σύστημα λειτουργεί είτε όταν αυτό δεν λειτουργεί αλλά με την χρήση κατάλληλων παραμέτρων στην αίτηση για κατανομή πόρων. Σαν αποτέλεσμα, οι υπηρεσίες που διαχειρίζονται το δίκτυο και την συνοχή των δεδομένων πρέπει κατόπιν απαίτησης του χρήστη να τροποποιούν την κατάσταση μιας ομάδας virtual machines και τους εικονικού δικτύου που αυτές βρίσκονται.

Επιπλέον των interfaces που χρησιμοποιούν μια γλώσσα προγραμματισμού (SOAP και “Query”), οι Interface services προσφέρουν ένα web interface για χρήστες και διαχειριστές του νέφους. Χρησιμοποιώντας ένα web

browser, οι χρήστες μπορούν να αιτηθούν για πρόσβαση στο νέφος, να κατεβάσουν τα κρυπτοποιημένα πιστοποιητικά που απαιτούνται για τα προγραμματιστικά interface, να αλληλεπιδράσουν με το σύστημα ,π.χ. για να δούνε τις διαθέσιμες images δίσκου. Οι διαχειριστές μπορούν επιπλέον να διαχειριστούν τους λογαριασμούς χρηστών και να επιβλέπουν την διαθεσιμότητα και λειτουργία των συστατικών του συστήματος.

Τέλος , τα interface των υπηρεσιών ιστού μπορεί να χρησιμοποιηθούν από χρήστες για να εκτελέσουν τα αιτήματα τους με διάφορους τρόπους (π.χ. EC2 SOAP και “Query”, S3 SOAP και REST). Επειδή οι χρήστες μπορούν να χρησιμοποιήσουν και EC2 SOAP και EC2 Query πρωτόκολλα[12], πολλά εργαλεία που είχαν σχεδιαστεί αρχικά για να δουλεύουν με τα EC2 και S3 μπορούν να δουλέψουν και με το Eucalyptus χωρίς τροποποιήσεις.

ΚΕΦΑΛΑΙΟ 4

ΣΧΕΔΙΑΣΜΟΣ ΜΗΧΑΝΙΣΜΟΥ ΑΞΙΟΛΟΓΗΣΗΣ ΚΑΙ ΕΠΙΛΟΓΗΣ ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΝΕΦΟΥΣ

Όπως παρατηρήσαμε ήδη, στην αγορά υπάρχει πλήθος παρόχων cloud computing ενώ η έλευση και εδραίωση του Eucalyptus θα δώσει την δυνατότητα σε περισσότερες εταιρείες να ασχοληθούν με τον συγκεκριμένο τομέα παροχής υπηρεσιών[55,56]. Τα οφέλη της ύπαρξης πολλών παρόχων είναι εμφανή για τους χρήστες των συστημάτων αυτών , ωστόσο το πρόβλημα που δημιουργείται είναι ότι πολλές φορές οι χρήστες συστημάτων cloud computing δεν είναι σε θέση να γνωρίζουν το σύνολο των διαθέσιμων υποδομών και ποίες από τις υποδομές ταιριάζουν καλύτερα στις ανάγκες τους. Η λύση που προτείνουμε είναι μια υπηρεσία ιστού η οποία επιβλέπει τα διαθέσιμα νέφη, καταγράφει πληροφορίες για αυτά και κατόπιν αιτήματος του χρήστη επιλέγει το καλύτερο για αυτόν με βάση τις ανάγκες του.

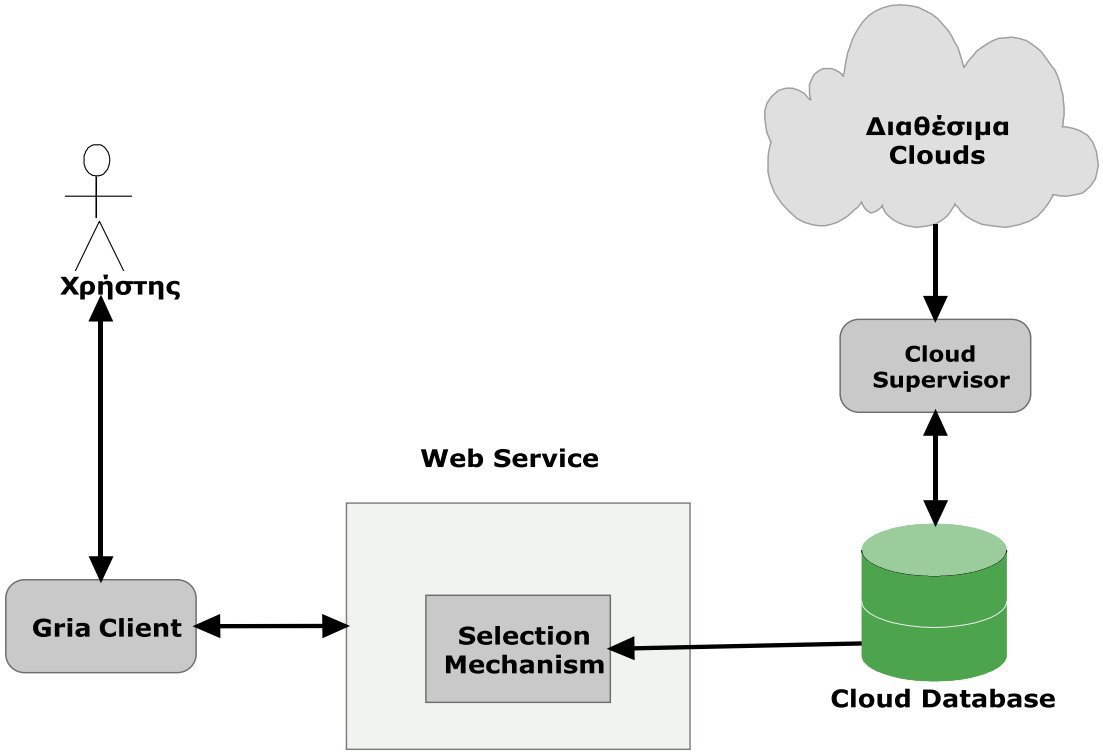
4.1. ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Η αρχιτεκτονική του συστήματος είναι απλή και αποτελείται από 4 ανεξάρτητα υποσυστήματα τα οποία υλοποιούν κοινά πρωτόκολλα επικοινωνίας. Τα υποσυστήματα αυτά είναι τα ακόλουθα:

- ***Cloud Database:*** Είναι η βάση δεδομένων στην οποία περιέχονται αναλυτικά στοιχεία για κάθε νέφος.
- ***Cloud Supervisor:*** Ουσιαστικά πρόκειται για έναν daemon ο οποίος επιτηρεί τα νέφη που υπάρχουν στην βάση ενημερώνοντας την για οποιαδήποτε αλλαγή συμβαίνει.
- ***Web Service:*** Η υπηρεσία ιστού η οποία υλοποιήθηκε με την βοήθεια του Gria Development Kit. Αναμένει συνδέσεις από τους χρήστες, επικοινωνεί με την βάση για να συλλέξει στοιχεία για τα νέφη και υλοποιεί τον μηχανισμό επιλογής.

- ***Gria Client:*** Ο client με τον οποίο ο χρήστης συνδέεται με την υπηρεσία ιστού είναι ο Gria Client με την προσθήκη ενός plug-in για την αλληλεπίδραση με την υπηρεσία ιστού που δημιουργήσαμε.

Η αρχιτεκτονική του συστήματος φαίνεται και στο παρακάτω διάγραμμα:



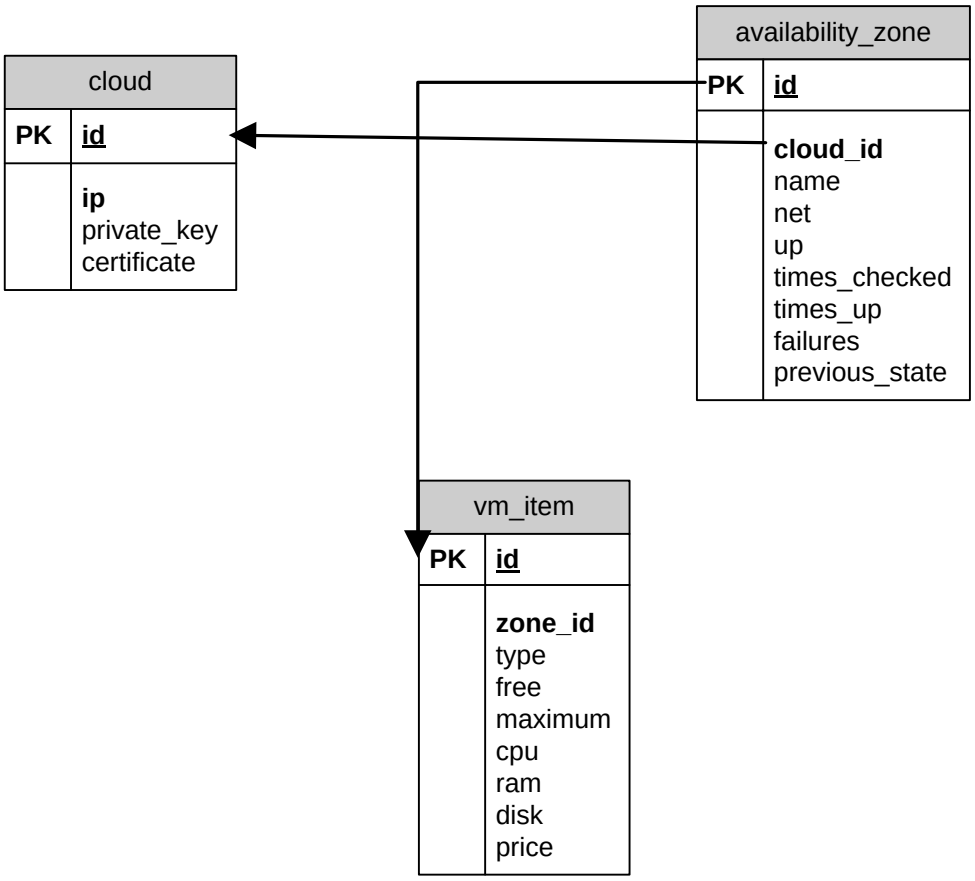
ΕΙΚΟΝΑ 7: ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ ΑΞΙΟΛΟΓΗΣΗΣ ΚΑΙ ΕΠΙΛΟΓΗΣ CLOUD

4.2. CLOUD DATABASE

Η βάση δεδομένων που διατηρεί στοιχεία για τα νέφη που υπάρχουν διαθέσιμα είναι βασικότατο στοιχείο του συστήματος. Η βάση διατηρεί τα βασικά στοιχεία κάθε νέφους τα οποία πρέπει να διαθέτουμε για να μπορέσουμε να επικοινωνήσουμε μαζί του : IP, το μοναδικό certificate και το private key που απαιτούνται για την εγκατάσταση ασφαλούς σύνδεσης μαζί του.

Όπως είδαμε και προηγουμένως, κάθε νέφος με την πλατφόρμα Eucalyptus αποτελείται από ανεξάρτητα availability zones για τα οποία πρέπει να συλλέγουμε πληροφορίες. Οι πληροφορίες αυτές αφορούν τόσο τα χαρακτηριστικά στοιχεία της availability zone όσο και στατιστικά στοιχεία λειτουργίας της. Οι πληροφορίες λοιπόν που συλλέγονται αφορούν : το όνομα του availability zone, το αν το availability zone είναι διαθέσιμο για χρήση, το εύρος ζώνης του τοπικού δικτύου διασύνδεσης μεταξύ των κόμβων του availability zone ,στατιστικά στοιχεία για το πόσο καιρό το συγκεκριμένο availability zone βρίσκεται υπό επιτήρηση, τη χρονική περίοδο που είναι σε κατάσταση λειτουργίας και τον αριθμό των βλαβών που έχουν εμφανιστεί όσο καιρό παρακολουθείται.

Η βάση τέλος κρατάει αναλυτικά στοιχεία και για κάθε τύπο virtual machine που είναι διαθέσιμος ανά availability zone. Πιο συγκεκριμένα για κάθε τύπο virtual machine φυλάσσονται στοιχεία για τον τύπο της (αυτή τη στιγμή οι διαθέσιμοι τύποι διακρίνονται σε m1.small,c1.small,m1.large,c1.large,m1.xlarge [45]), πόσες instances αυτού του τύπου που μπορεί να υποστηρίξει αυτό το availability zone την δεδομένη χρονική στιγμή και πόσες το μέγιστο, τον αριθμό τον επεξεργαστών, το μέγεθος της μνήμης RAM, το μέγεθος του σκληρού δίσκου και το κόστος χρήσης του συγκεκριμένου τύπου VM ανά ώρα. Το σχεσιακό μοντέλο της απλής αυτής βάσης φαίνεται στο ακόλουθο διάγραμμα:



ΕΙΚΟΝΑ 8: ΣΧΕΣΙΑΚΟ ΜΟΝΤΕΛΟ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Όπως παρατηρούμε το σχήμα της συγκεκριμένης βάσης δεδομένων δεν είναι ιδιαίτερα πολύπλοκο, αλλά απαιτούμε υψηλές ταχύτητες πρόσβασης στα δεδομένα και όσο το δυνατόν πιο εύκολη διασύνδεση με τον υπόλοιπο σύστημα.

Για τους ανωτέρω λόγους προτιμήθηκε η μηχανή HSQLDB 1.8[57]. Η HSQLDB είναι ένα σύστημα διαχείρισης βάσης δεδομένων γραμμένο σε Java. Η HSQLDB υποστηρίζει τα πρότυπα SQL–92, SQL-1999 και SQL-2008 ενώ οι πίνακες αποθηκεύονται στην μνήμη και όχι στον σκληρό δίσκο για λόγους ταχύτητας πρόσβασης και με δεδομένο ότι η βάση μας δεν διαχειρίζεται πολύ μεγάλο όγκο δεδομένων. Η επιλογή της συγκεκριμένης μηχανής έγινε εξαιτίας των χαμηλών

απαιτήσεών της σε hardware και της εύκολης διαχείρισης της με χρήση του οδηγού JDBC μέσα από το πρόγραμμά μας, καθότι η υλοποίησή τόσο του Cloud Supervisor όσο και του Web Service με τα οποία αλληλεπιδρά η βάση έχει γίνει σε Java.

Τα στοιχεία με τα οποία αλληλεπιδρά η βάση είναι ο Cloud Supervisor και το Web Service. Η ενημέρωση των στοιχείων της βάσης γίνεται από τον Cloud Supervisor ανά τακτά χρονικά διαστήματα, ενώ το Web Service έχει μόνο το δικαίωμα ανάγνωσης των δεδομένων που υπάρχουν ήδη στην βάση.

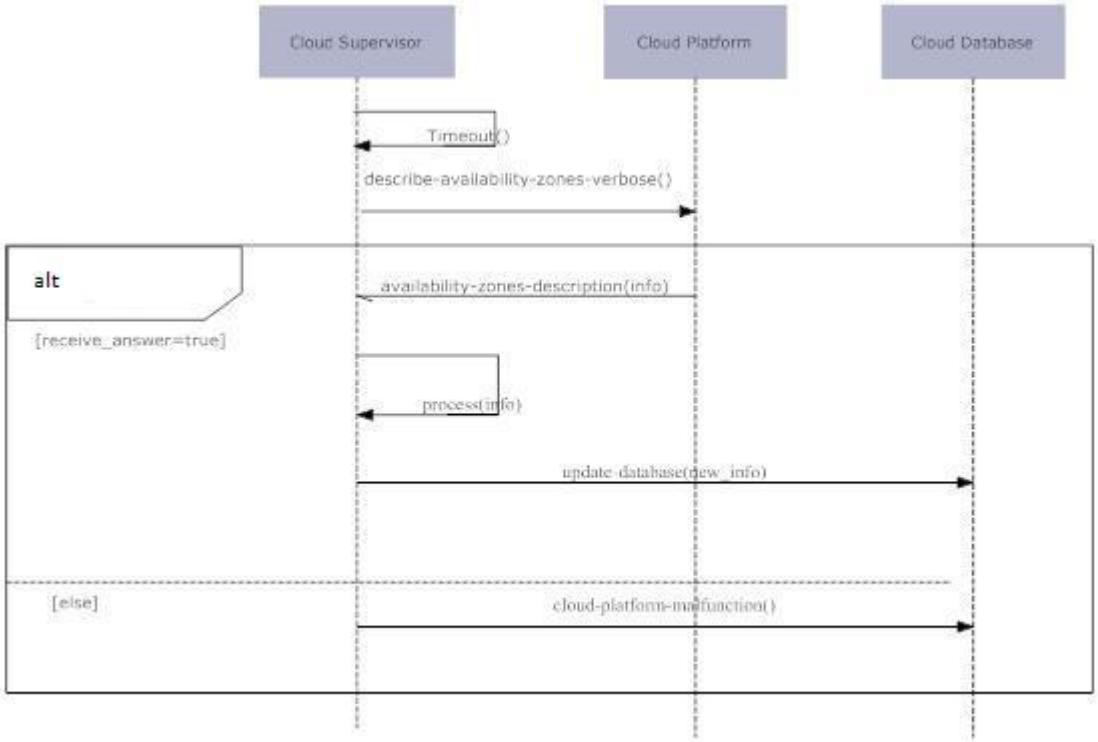
4.3. CLOUD SUPERVISOR

Προκειμένου η cloud database να είναι συνεχώς ενημερωμένη για την τρέχουσα κατάσταση στα νέφη, κρίθηκε απαραίτητη η δημιουργία ενός προγράμματος το οποίο θα τα επιτηρεί ανά τακτά χρονικά διαστήματα και θα ενημερώνει την βάση με τα νεώτερα στοιχεία. Επιπλέον αυτού, το πρόγραμμα πρέπει να επεξεργάζεται την κατάσταση των availability zones των νεφών και να ενημερώνει τα στατιστικά στοιχεία που φυλάσσονται για αυτές στην βάση. Το πρόγραμμα αυτό είναι ο Cloud Supervisor.

Ο Cloud Supervisor πρόκειται ουσιαστικά για ένα daemon γραμμένο σε Java ο οποίος επικοινωνεί με τη διαθέσιμα νέφη και πραγματοποιεί την ανανέωση της Cloud Database. Αυτή τη στιγμή ο daemon είναι σχεδιασμένος ώστε να συνεργάζεται με τα νέφη που υλοποιούν το interface επικοινωνίας του Amazon EC2. Όπως είδαμε και προηγουμένως, ανάμεσα στις πλατφόρμες που υποστηρίζουν το συγκεκριμένο interface είναι και το Eucalyptus.

Ο τρόπος που επικοινωνεί ο cloud supervisor με τα νέφη είναι με ανταλλαγή SOAP μηνυμάτων. Για να επιτύχουμε την επικοινωνία αυτή κάνουμε χρήση των εργαλείων API του Eucalyptus , έκδοση 1.5.2(api tools.v1.5.2). Η έκδοση αυτή των API tools του Eucalyptus προσομοιώνει τη λειτουργία των API tools έκδοσης 2008-12-01 του Amazon EC2. Τα εργαλεία αυτά είναι ουσιαστικά ο μοναδικός τρόπος επικοινωνίας με τα νέφη και μας δίνουν δυνατότητες διαχείρισής τους αλλά και άντλησης πληροφοριών για αυτά. Ειδικά το τελευταίο είναι πολύ σημαντικό και είναι η δυνατότητα που αξιοποιούμε και στην δικιά μας υλοποίηση.

Η εργασία που επιτελεί ο cloud supervisor είναι η εξής: ανά τακτό χρονικό διάστημα και σε κάθε νέφος που βρίσκεται στην βάση δεδομένων, με χρήση των αποθηκευμένων certificates και private keys, εκτελεί την εντολή "*describe-availability-zones verbose*". Η συγκεκριμένη εντολή ζητά από το Cloud Controller να δημοσιεύσει τις Availability zones οι οποίες λειτουργούν σε αυτό καθώς και λεπτομέρειες για τις ζώνες αυτές. Ο cloud supervisor λαμβάνει την απάντηση του cloud, επεξεργάζεται τις πληροφορίες και ανανεώνει ανάλογα τα περιεχόμενα της Cloud database. Σε περίπτωση μη έγκαιρης απάντησης, το νέφος θεωρείται ότι είναι εκτός λειτουργίας. Οι περιπτώσεις στις οποίες το νέφος ή συγκεκριμένο availability zone θεωρήθηκε εκτός λειτουργίας καταγράφονται στην βάση και αξιοποιούνται στον υπολογισμό της αξιοπιστίας. Η ανταλλαγή των μηνυμάτων αυτών φαίνονται και στο παρακάτω ακολουθιακό διάγραμμα:



ΕΙΚΟΝΑ 9:ΑΚΟΛΟΥΘΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΛΕΙΤΟΥΡΓΙΑΣ CLOUD SUPERVISOR

4.4. WEB SERVICE

Η Web Service αποτελεί τον πυρήνα της υλοποίησής μας. Μια σημαντική σχεδιαστική απόφαση που λάβαμε ήταν η χρήση του Gria[58] για την υλοποίηση της. Το Gria είναι μία service-oriented υποδομή (SOI- Service Oriented Infrastructure) η οποία σχεδιάστηκε ώστε να υποστηρίζει business 2 business συνεργασίες. Το Gria χρησιμοποιεί τα πιο πρόσφατα πρωτόκολλα για υπηρεσίες ιστού και ασφάλεια Public Key Infrastructure (PKI)[59] εξασφαλίζοντας με αυτόν τον τρόπο ασφαλή επικοινωνία μεταξύ του χρήστη και της υπηρεσίας ιστού. Το Gria είναι λογισμικό ανοιχτού κώδικα και διατίθεται δωρεάν. Η έκδοση του Gria που χρησιμοποιούμε είναι η 5.3.

Η υπηρεσία ιστού που υλοποιήσαμε λοιπόν αναπτύχθηκε με την βοήθεια του Gria Development Kit, και μπορεί να συνδεθεί οποιοσδήποτε χρήστης με την βοήθεια του Gria Client. Είναι μια δωρεάν υπηρεσία ιστού και δεν προϋποθέτει την ύπαρξη κάποιου λογαριασμού ή SLA. Πρόκειται για μια υπηρεσία ιστού η οποία μπορεί να γίνει deploy στον Tomcat και επιτρέπει σε πόρους να δημιουργηθούν ή να καταστραφούν . Οι πόροι στην περίπτωσή μας αφορούν τον μηχανισμό επιλογής availability zone και προστατεύονται με χρήση PBAC 2 (Process-Based Access Control 2)[60].

Αφού συνδεθεί ο χρήστης και καταθέσει το αίτημά του για εξεύρεση νέφους, η υπηρεσία ιστού συνδέεται με την Cloud Database , εκτελεί τον μηχανισμό για την επιλογή νέφους και κατόπιν αποστέλλει τα αποτελέσματα της

επιλογής πίσω στον Gria Client. Όπως είναι κατανοητό, το κυριότερο στοιχείο της Web Service είναι ο μηχανισμός επιλογής, ο οποίος αναλύεται στην συνέχεια.

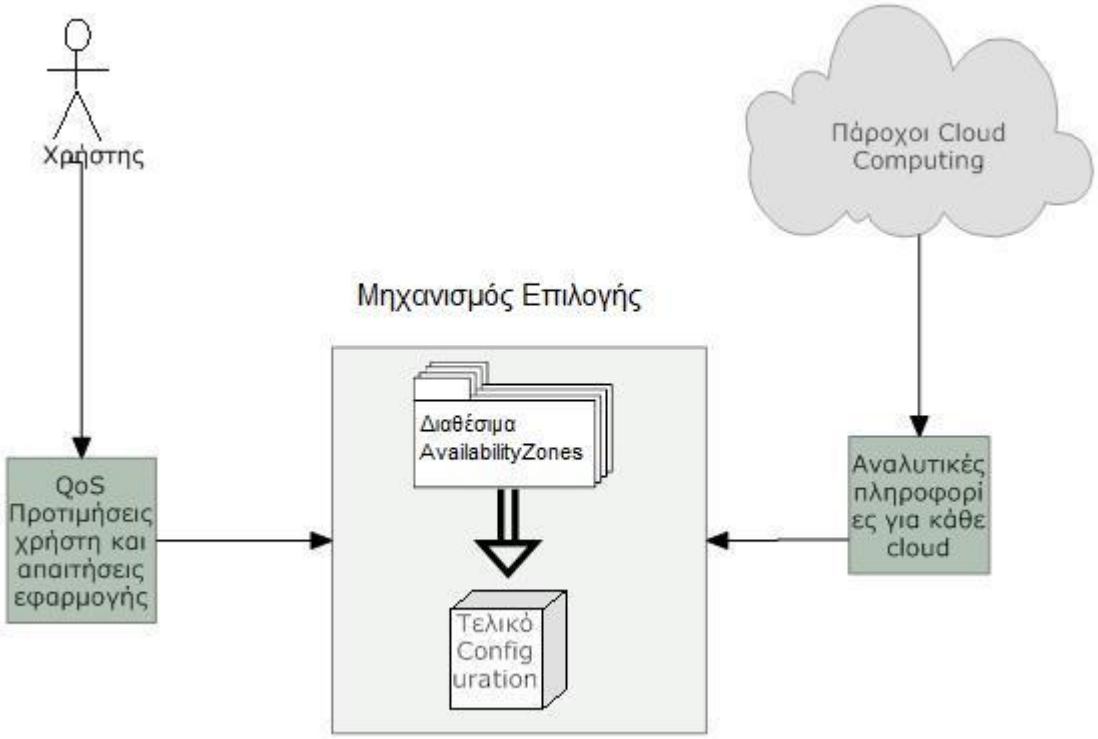
4.4.1. ΕΠΙΣΚΟΠΗΣΗ ΜΗΧΑΝΙΣΜΟΥ ΕΠΙΛΟΓΗΣ ΝΕΦΟΥΣ

Σκοπός του συγκεκριμένου μηχανισμού επιλογής, είναι να επιλέξει ανάμεσα στο πλήθος των διαθέσιμων νεφών, το availability zone του νέφους που ταιριάζει περισσότερο στις ανάγκες του χρήστη. Πέρα από αυτό, ο μηχανισμός δημιουργεί και αναλυτικό configuration για κάθε availability zone, δηλαδή πόσες virtual machines και τι τύπου θα χρησιμοποιηθούν. Συνολικά, τα δεδομένα που χρησιμοποιούνται στον αλγόριθμος επιλογής είναι:

- ***Πληροφορίες για τις απαιτήσεις της εφαρμογής σε hardware.*** Ο χρήστης δηλώνει τις ζητούμενες από την εφαρμογή απαιτήσεις σε υπολογιστικούς πόρους. Οι πληροφορίες αυτές είναι κρίσιμης σημασίας καθώς καθορίζουν σε μεγάλο την τελική επιλογή.

- ***QoS περιορισμοί και προτιμήσεις του χρήστη.*** Ο χρήστης ορίζει τις προτιμήσεις του όσον αφορά περιορισμούς στο QoS, ενώ μπορεί να δηλώσει και βαρύτητα που πρέπει να δοθεί σε συγκεκριμένους παράγοντες QoS.

- ***Πληροφορίες όσον αφορά τα διαθέσιμα clouds.*** Κάθε ένα από τα διαθέσιμα clouds αντιστοιχίζεται με πληροφορίες για το QoS , πληροφορίες οι οποίες θα χρησιμοποιηθούν κατά την διαδικασία της επιλογής. Οι πληροφορίες αφορούν παραμέτρους του QoS οι οποίες δημοσιοποιούνται από τους παρόχους cloud computing, καθώς και πληροφορίες οι οποίες εξάγονται για τα clouds από μηχανισμούς επιτήρησης του συστήματος.



ΕΙΚΟΝΑ 10:ΔΙΑΓΡΑΜΜΑ ΛΕΙΤΟΥΡΓΙΑΣ ΜΗΧΑΝΙΣΜΟΥ ΕΠΙΛΟΓΗΣ

Από την στιγμή που όλες οι πληροφορίες δοθούν στον μηχανισμό επιλογής , ξεκινά η εκτέλεση του αλγορίθμου η οποία θα μας δώσει σαν αποτέλεσμα το βέλτιστο configuration για την εργασία που επιθυμεί να τρέξει ο χρήστης.

4.4.2. ΠΑΡΑΜΕΤΡΟΙ QOS

Όπως αναφέρθηκε και παραπάνω, για να κάνουμε την βέλτιστη τελική επιλογή configuration για τον χρήστη, εξετάζουμε διάφορους παραμέτρους του QoS. Παρακάτω παρουσιάζουμε τις παραμέτρους που χρησιμοποιούμε και οι οποίοι θεωρούμε ότι παίζουν το σπουδαιότερο ρόλο για την επιλογή ενός νέφους. Οι παράμετροι αυτοί χωρίζονται σε δύο κύριες κατηγορίες:

- *Παράμετροι υπολογιστικών πόρων.* Είναι οι παράμετροι που σχετίζονται με το hardware. Περιλαμβάνουν:
    - τον αριθμό επεξεργαστικών μονάδων ανά κόμβο,
    - το μέγεθος την μνήμης RAM ανά κόμβο,
    - το μέγεθος του σκληρού δίσκου ανά κόμβο,
    - το διαθέσιμο εύρος ζώνης του τοπικού δικτύου
    - το πλήθος των υπολογιστικών κόμβων.

- *Παράμετροι αξιοπιστίας συστήματος* .Είναι οι παράμετροι που σχετίζονται με την αξιοπιστία του συστήματος. Οι παράμετροι που χρησιμοποιούμε είναι :
    - Availability: Η πιθανότητα να είναι το availability zone διαθέσιμο οποιαδήποτε στιγμή θελήσουμε να το χρησιμοποιήσουμε.
    - Continuous availability: Η πιθανότητα το availability zone να είναι διαθέσιμο για τη χρονική περίοδο που χρησιμοποιούμε το νέφος.
    - MTTR: ο μέσος χρόνος που απαιτείται για την επαναφορά availability zone σε λειτουργία μετά από βλάβη.
- *Προτιμήσεις χρήστη.* Είναι οι παράμετροι που σχετίζονται με περιορισμούς που θέτει ο χρήστης για την εκτέλεση της εργασίας του και ιδιαίτερες προτιμήσεις όσον αφορά την επιλογή του τελικού configuration. Οι παράμετροι αυτές είναι :
    - Χρόνος εκτέλεσης: Η χρονική περίοδος για την οποία επιθυμεί ο χρήστης να κάνει χρήση του νέφους.
    - Κόστος: Ο μέγιστο ποσό που είναι διατεθειμένος να δώσει ο χρήστης για την χρήση της υπηρεσίας.
    - Reliability Gravity: Η βαρύτητα που πρέπει να δοθεί από τον μηχανισμό επιλογής στην αξιοπιστία του availability zone.
    - Cost Gravity: Η βαρύτητα που πρέπει να δοθεί από τον μηχανισμό επιλογής στον περιορισμό του τελικό κόστους για την χρήση της υπηρεσίας.

4.4.3. ΜΕΤΡΗΣΗ ΑΞΙΟΠΙΣΤΙΑΣ ΝΕΦΟΥΣ

Ένας κρίσιμος παράγοντας για την επιλογή του κατάλληλου νέφους είναι η αξιοπιστία του και για αυτό θεωρήθηκε σημαντικό να βρεθεί μια κατάλληλη μετρική αξιολόγησής της. Μελετήσαμε ένα πλήθος βιβλιογραφίας για την αξιοπιστία συστημάτων[62,63,64,65,66] . Στην περίπτωσή μας ωστόσο υπάρχει ο περιορισμός ότι δεν είμαστε σε θέση να γνωρίζουμε την εσωτερική δομή ενός νέφους και η μόνη πληροφορία όσον αφορά την αξιοπιστία του αποτελεί η ένδειξη ότι ένα availability zone είναι σε κατάσταση λειτουργίας ή όχι. Με αυτά τα δεδομένα υπόψη καταλήξαμε ότι σαν παράμετροι αξιοπιστίας μπορούν να θεωρηθούν:

- *Η διαθεσιμότητα* : Η πιθανότητα το σύστημα να είναι σε λειτουργία κάποια τυχαία χρονική στιγμή.
- *Η συνεχής διαθεσιμότητα*: Η πιθανότητα το σύστημα να είναι σε λειτουργία συνεχώς για κάποια χρονική περίοδο t.

- *Ο Μέσος Χρόνος για Επαναφορά(MTTR)*: Ο μέσος όρος του χρόνου που απαιτείται προκειμένου να σύστημα να επανέλθει έπειτα από μια βλάβη.

Όπως έχουμε δει κάθε availability zone ενός νέφους έχει ανεξάρτητη λειτουργία από τα υπόλοιπα του ίδιου νέφους. Επομένως αυτές οι παράμετροι μπορούν να χαρακτηρίσουν μοναδικά την αξιοπιστία του κάθε availability zone. Για τον σκοπό αυτό δημιουργήσαμε το μέγεθος *Reliability Score*. Ο τρόπος υπολογισμού του Reliability Score κάθε availability zone είναι ο ακόλουθος:

1. Ανάλογα την διαθεσιμότητα, με βάση την κατηγοριοποίηση όπως γίνεται και στο [61] υπολογίζεται το availability score του κάθε availability zone. Οι τιμές του availability score φαίνονται στην συνέχεια :

Διαθεσιμότητα(%)	Availability score	Χαρακτηρισμός
[0, 90)	1	unmanaged
[90, 95)	2	managed
[95, 99)	4	well managed
[99, 99.9)	8	fault tolerant
[99.9, 99.99)	12	high availability
[99.99, 100]	16	ultra high availability

2. Υπολογίζουμε την συνεχή διαθεσιμότητα του availability zone. Θεωρούμε ότι η βλάβες είναι τυχαίες και επομένων ακολουθούν την κατανομή Poisson. Επομένως η continuous availability θα υπολογίζεται σύμφωνα με τον τύπο :

$$Continuous\ Availability = e^{-t*\lambda}$$

, όπου λ ο λόγος του ολικού αριθμού των παρατηρούμενων βλαβών προς το συνολικό χρόνο λειτουργίας του availability zone, t ο χρόνος που ο χρήστης θα χρησιμοποιήσει την υπηρεσία.

3. Υπολογισμός του μέσου χρόνου για επιδιόρθωση βλάβης(MTTR). Ο υπολογισμός του MTTR γίνεται με την βοήθεια του τύπου:

$$MTTR = \frac{Χρόνος\ που\ το\ Availability\ Zone\ δεν\ είναι\ διαθέσιμο}{Χρόνος\ που\ το\ Availability\ Zone\ παρακολουθείται}$$

4. Έχοντας υπολογίσει τις παραπάνω παραμέτρους, ο υπολογισμός του “Reliability score” γίνεται με βάση τον τύπο:

$$ReliabilityScore = \frac{AvailabilityScore * ContinousAvailability}{\exp(MTTR)}$$

Ο δείκτης αυτός είναι σημαντικότατος για την επιλογή ενός νέφους από τον μηχανισμό μας και όσο μεγαλύτερο το Reliability score ενός availability zone, τόσο μεγαλύτερη είναι η πιθανότητα να μην εμφανίσει σοβαρό πρόβλημα κατά την χρονική διάρκεια που θα το χρησιμοποιεί ο χρήστης.

4.4.4. ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Στην συνέχεια παρουσιάζουμε τον αλγόριθμο που χρησιμοποιείται στο μηχανισμό επιλογής πόρων για να επιλέξουμε σε ποιο availability zone, ποιού νέφους και με τι κατανομή VM instances είναι προτιμότερο να τρέξει ο χρήστης την εφαρμογή του. Ο σκοπός του αλγορίθμου είναι να καταλήξουμε σε ένα τελικό configuration λαμβάνοντας υπόψη τις προτιμήσεις του χρήστη για τις QoS παραμέτρους και την ποιότητα των υπηρεσιών που προσφέρονται από τους cloud computing παρόχους.

Καθώς δεν υπάρχει αρκετή βιβλιογραφία στον συγκεκριμένο τομέα της επιλογής νέφους, μελετήσαμε κυρίως βιβλιογραφία που αφορά στην επιλογή workflows σε grid αρχιτεκτονικές[67,68,69,70]. Με βάση τις ανάγκες του μηχανισμού μας, εν τέλει προτιμήθηκε μια παραλλαγή του αλγορίθμου που παρουσιάζεται στο [71]. Ακολουθήθηκε η ίδια λογική, προσαρμοσμένη ωστόσο στις QoS παραμέτρους του προβλήματός μας και στις ειδικές συνθήκες των υπολογιστικών νεφών.

Η στρατηγική που ακολουθούμε στον αλγόριθμο είναι ότι αρχικά αποκλείουμε τα προβληματικά και τα availability zones των νεφών τα οποία δεν πληρούν τα QoS standards που έχει θέσει ο χρήστης. Δημιουργούμε τα πιθανά configuration για τα availability zones που έχουν μείνει και τα οποία καλύπτουν τις απαιτήσεις του χρήστη σε υπολογιστικούς πόρους, υπολογίζοντας το κόστος τους και τις μετρικές της αξιοπιστίας τους. Δημιουργούμε μια σχετική κατάταξη των διαθέσιμων configurations και επιλέγουμε το configuration με την εν τέλει καλύτερη απόδοση.

Μέσα στον αλγόριθμο οι προτιμήσεις του χρήστη όσον αφορά την βαρύτητα στους παράγοντες κόστος(CostSlope) και αξιοπιστία(ReliabilitySlope) εκφράζονται με τη μορφή δεκαδικών αριθμών που παίρνουν τιμές στο διάστημα [0,1]. Όσο μεγαλύτερος ο αριθμός τόσο πιο σημαντική είναι η παράμετρος για τον χρήστη. Σαν αποτέλεσμα έχουμε :

$$CostSlope + ReliabilitySlope \leq 2$$

Ακολούθως περιγράφονται αναλυτικά όλα τα βήματα του αλγορίθμου.

Βήμα 1: Επιλογή των availability zones που είναι σε κατάσταση λειτουργίας και μπορούν να διαθέσουν τον απαιτούμενο αριθμό κόμβων οι οποίοι να πληρούν τα χαρακτηριστικά που έχει θέσει ο χρήστης σχετικά με : αριθμό επεξεργαστών, μέγεθος RAM, μέγεθος σκληρού δίσκου, εύρος ζώνης δικτύου. Τα υπόλοιπα availability zones απορρίπτονται καθώς δεν μπορούν να καλύψουν τις ανάγκες του χρήστη.

Βήμα 2: Υπολογισμός του configuration του κάθε availability zone, δηλαδή υπολογισμός της κατανομής των VM instances στους διαφορετικούς τύπους VM που προσφέρονται από το νέφος, με κριτήριο την μικρότερη δυνατή σπατάλη πόρων.

Βήμα 3: Υπολογισμός του συνολικού κόστους χρήσης για κάθε configuration. Το κόστος χρήσης του configuration x ορίζεται σαν *initialCost(x)*. Τα configurations με μεγαλύτερο κόστος από το μέγιστο ποσό που μπορεί να διαθέσει ο χρήστης απορρίπτονται.

Βήμα 4: Υπολογισμός του *reliability score* για το configuration της εκάστοτε availability zone. Το reliability score του configuration x ορίζεται σαν *initialReliabilityScore(x)* και γίνεται με τον τρόπο που αναφέρθηκε προηγουμένως.

Βήμα 5: Υπολογισμός μέγιστου κόστους (*maxCost*), ελάχιστου κόστους (*minCost*), μέγιστου Reliability score (*maxReliabilityScore*) και ελάχιστου Reliability score (*minReliabilityScore*) των configurations που έχουμε δημιουργήσει σε προηγούμενα βήματα.

Βήμα 6: Για κάθε configuration x ,υπολογισμός των τιμών $F_{Cost}(x)$ και $F_{Reliability}(x)$ με χρήση των συναρτήσεων:

$$F_{Cost}(x) = \exp(CostSlope * \frac{initialCost(x) - minCost}{maxCost - minCost}$$

και

$$F_{Reliability}(x) = \exp(ReliabilitySlope * \frac{initialReliabilityScore(x) - minReliabilityScore}{maxReliabilityScore - minReliabilityScore})$$

Βήμα 7: Για κάθε configuration x, υπολογισμός των νέων κανονικοποιημένων τιμών Cost και ReliabilityScore με βάση τους τύπους:

$$Cost(x) = initialCost(x) * F_{Cost}(x)$$

$$ReliabilityScore(x) = initialReliabilityScore(x) * F_{Reliability}(x)$$

Βήμα 8: Υπολογισμός του δείκτη *ConvertedIndex(x)* του κάθε configuration x με βάση τον ακόλουθο τύπο:

$$ConvertedIndex(x) = \frac{Cost(x)}{ReliabilityScore(x)}$$

Ο δείκτης ConvertedIndex είναι σημαντικότατος καθώς μας δείχνει την αξιοπιστία ενός configuration σε σχέση με το κόστος χρήσης του. Όσο χαμηλότερος ο δείκτης τόσο καλύτερη η ποιότητα υπηρεσιών που παρέχει στον χρήστη καθώς για την εκτέλεση μιας εργασίας προσφέρει πιο αξιόπιστη υπηρεσία σε μικρότερο κόστος. Ο συγκεκριμένος δείκτης χρησιμοποιείται για την κατάταξη των configurations και εν τέλει επιλογή του βέλτιστου, δηλαδή του configuration με την μικρότερη τιμή ConvertedIndex.

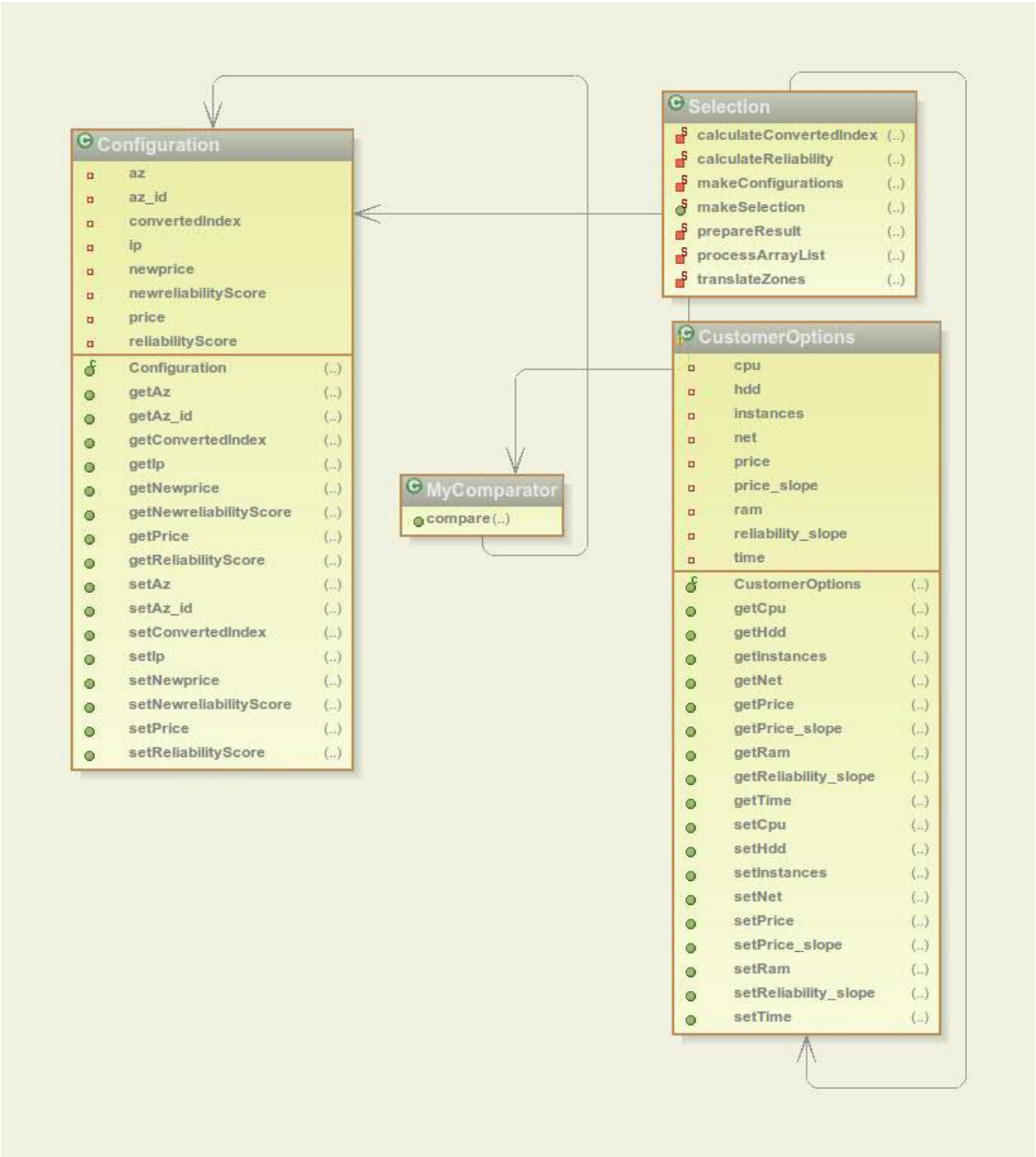
4.5. GRIA CLIENT

Όπως είδαμε και προηγουμένως , η Web Service υλοποιήθηκε με την βοήθεια του Gria Development Kit. Για την χρήση μιας τέτοιας υπηρεσίας ιστού απαιτείται και ο αντίστοιχος client, ο Gria Client. Ο Gria client πρόκειται για ένα πρόγραμμα το οποίο παρέχει ένα πλήρες γραφικό περιβάλλον για αλληλεπίδραση με τις υπηρεσίες ιστού. Το γεγονός επιπλέον ότι υποστηρίζει plug-ins μας επιτρέπει να ενσωματώσουμε τις νέες δυνατότητες που θέλουμε για την υπηρεσία μας χωρίς να χρειαστεί να γίνει recompile ολόκληρος ο client.

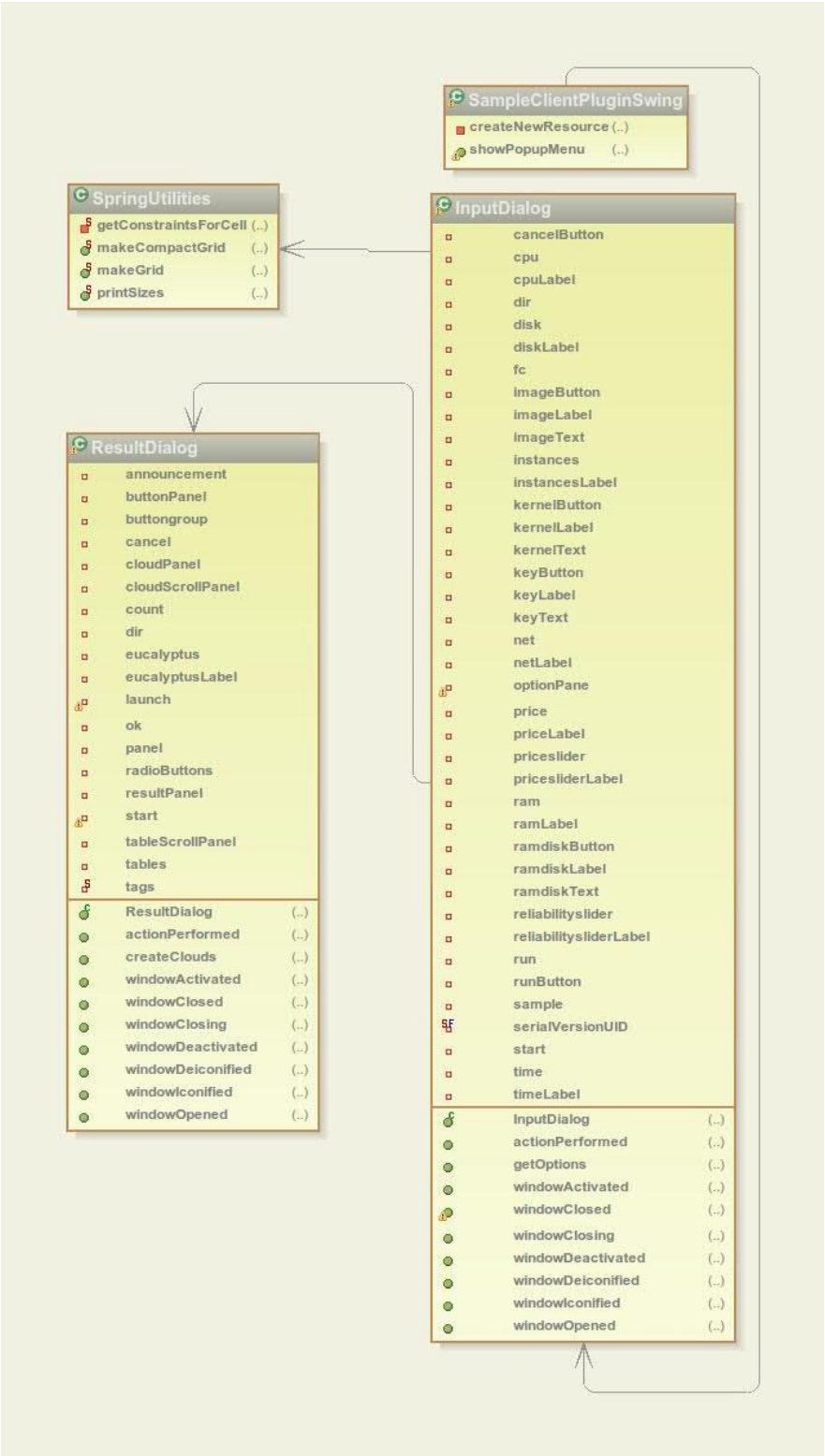
Για την αξιοποίηση λοιπόν της υπηρεσίας που δημιουργήσαμε εμφανίστηκε η ανάγκη για την δημιουργία του κατάλληλου plug-in. Το plug-in που δημιουργήσαμε για την περίπτωσή μας δημιουργεί το κατάλληλο γραφικό περιβάλλον ώστε ο χρήστης αφότου συνδεθεί με την υπηρεσία, με την χρήση του κατάλληλου γραφικού περιβάλλοντος να μπορεί εύκολα εισάγει αναλυτικά τις απαιτήσεις του όσον αφορά τις QoS απαιτήσεις του. Τα δεδομένα που εισήγαγε ο χρήστης μεταφέρονται στην Web Service η οποία κατόπιν στέλνει τα αποτελέσματα της επιλογής στον Gria client. Ο Gria client με την βοήθεια του plug-in παρουσιάζει τα αποτελέσματα σε ένα όμορφο γραφικό περιβάλλον. Το plug in είναι γραμμένο σε γλώσσα προγραμματισμού Java με την βοήθεια του SDK 1.6 ενώ για την δημιουργία των γραφικών έγινε χρήση του Java Swing. Στην συνέχεια της διπλωματικής περιέχεται tutorial για τον τρόπο χρήσης της υπηρεσίας με screenshots από τα γραφικό περιβάλλον του Gria client.

4.6. ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ

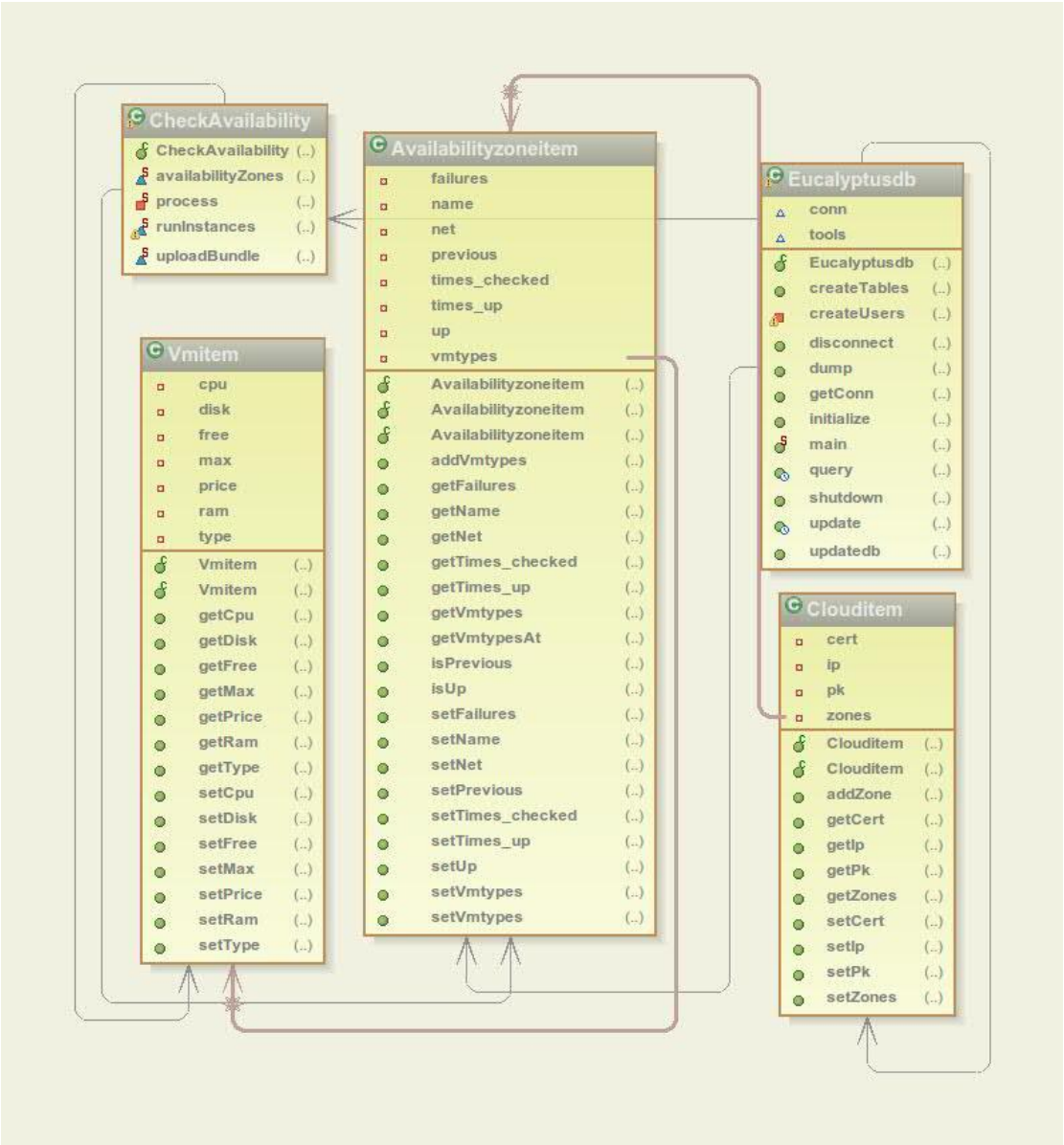
Τα διαγράμματα κλάσεων της υλοποίησης δίνονται ακολούθως.



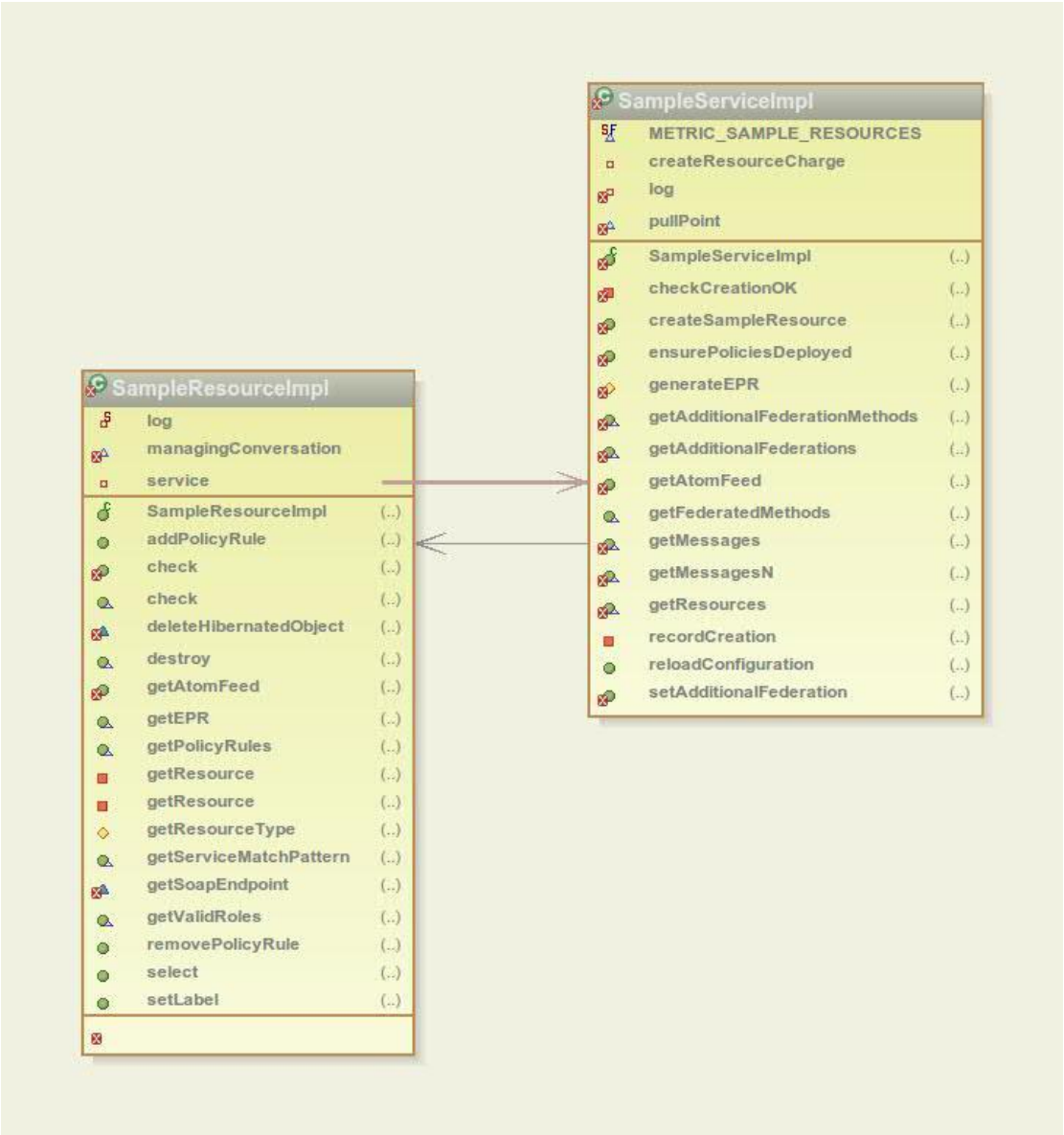
ΕΙΚΟΝΑ 11: ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΜΗΧΑΝΙΣΜΟΥ ΕΠΙΛΟΓΗΣ.



ΕΙΚΟΝΑ 12:ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΓΙΑ ΔΗΜΙΟΥΡΓΙΑ USER INTERFACE.



ΕΙΚΟΝΑ 13:ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ CLOUD SUPERVISOR ΚΑΙ CLOUD DB.



ΕΙΚΟΝΑ 14:ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΥΛΟΠΟΙΗΣΗΣ WEB SERVICE.

ΚΕΦΑΛΑΙΟ 5

ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΚΤΕΛΕΣΗΣ ΑΛΓΟΡΙΘΜΟΥ ΕΠΙΛΟΓΗΣ ΣΕ ΔΕΔΟΜΕΝΟ DATASET

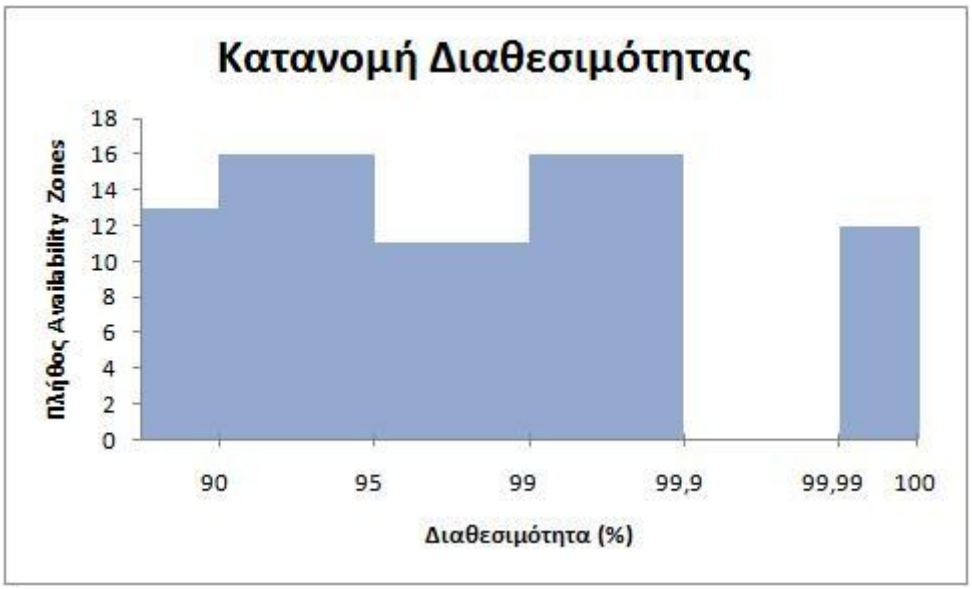
Στο προηγούμενο κεφάλαιο περιγράψαμε τον αλγόριθμο που χρησιμοποιείται στον μηχανισμό επιλογής για την αξιολόγηση των availability zones των υπολογιστικών νεφών. Αυτός ο αλγόριθμος υλοποιήθηκε σε μια υπηρεσία ιστού και πραγματοποιήσαμε διάφορες εκτελέσεις του προκειμένου να δούμε τον τρόπο λειτουργίας του. Για να γίνει αυτό προχωρήσαμε στην δημιουργία ενός dataset από διαθέσιμα νέφη περιλαμβανομένων των availability zones και λεπτομερειών για τους τύπους των VM. Στην συνέχεια «τρέξαμε» την Web Service για συγκεκριμένες προτιμήσεις χρήστη και μελετήσαμε τα αποτελέσματα του αλγορίθμου.

5.1. ΣΤΑΤΙΣΤΙΚΑ ΣΤΟΙΧΕΙΑ DATASET

Στην ενότητα αυτή θα παρουσιάσουμε ορισμένα στατιστικά στοιχεία για το dataset που χρησιμοποιήσαμε. Να αναφέρουμε ότι το dataset αποτελείται από 10 νέφη και συνολικά 100 ανεξάρτητα availability zones. Με δεδομένο ότι κάθε availability zone σύμφωνα με το μοντέλο που εξετάζουμε πρέπει να υποστηρίξει 5 διαφορετικούς τύπους VM instance, υπάρχουν συνολικά 500 ξεχωριστά τύποι VM που υποστηρίζονται.

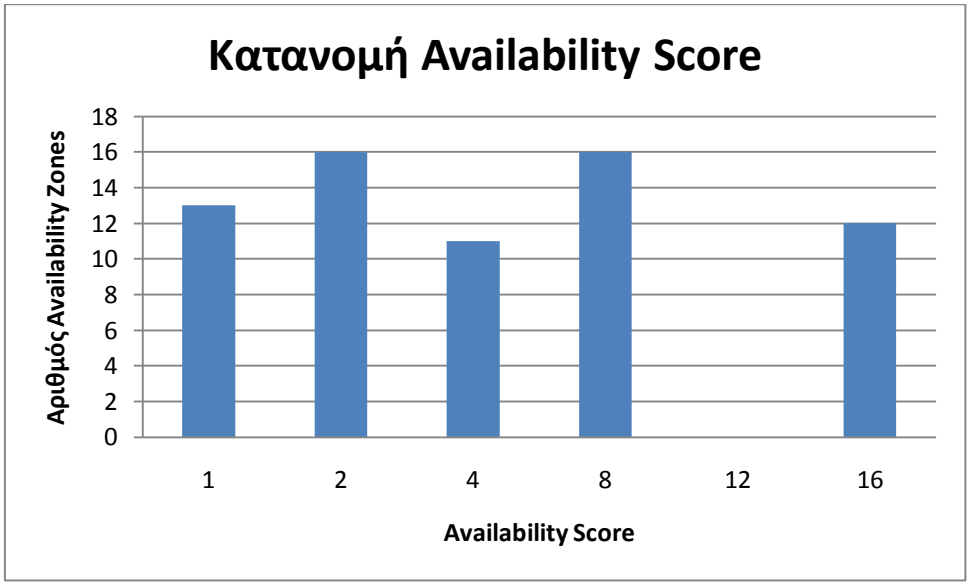
5.1.1. ΣΤΑΤΙΣΤΙΚΑ AVAILABILITY ZONES

Όπως αναφέρθηκε και προηγουμένως στο κεφάλαιο 4, τα χαρακτηριστικά των availability zones που μας ενδιαφέρουν είναι η διαθεσιμότητα, το availability score, το MTTR, το εύρος ζώνης δικτύου, ο αριθμός βλαβών και αν είναι σε κατάσταση λειτουργίας ή όχι. Το dataset περιλαμβάνει μόνο availability zones που είναι εν λειτουργία. Σε αυτά, η διαθεσιμότητα είναι η ακόλουθη:



ΕΙΚΟΝΑ 15:ΚΑΤΑΝΟΜΗ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ΤΩΝ AVAILABILITY ZONES.

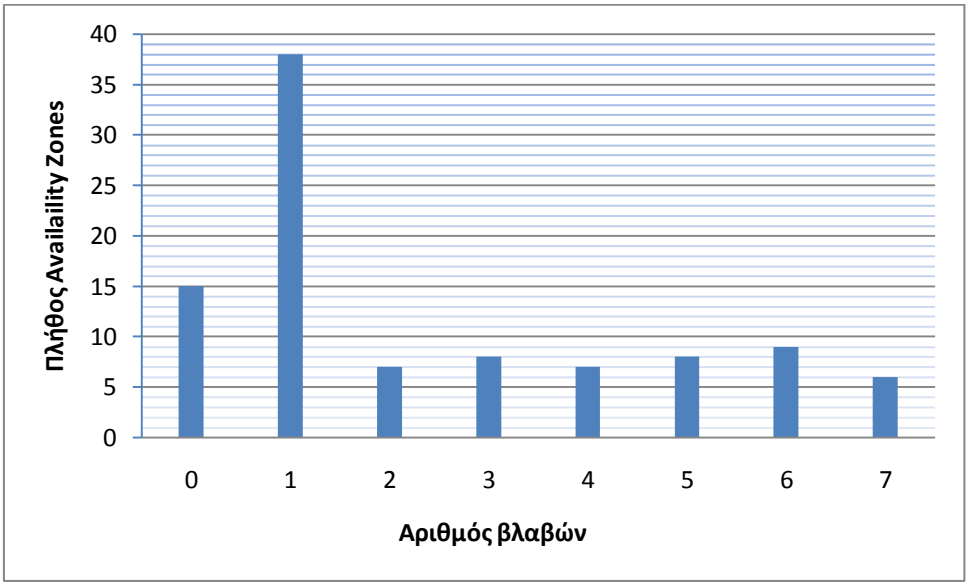
Με βάση την διαθεσιμότητα της, η κάθε availability zone παίρνει το ανάλογο availability score. Η κατανομή των availability scores είναι η ακόλουθη:



ΕΙΚΟΝΑ 16:ΚΑΤΑΝΟΜΗ AVAILABILITY SCORE ΤΩΝ AVAILABILITY ZONES.

Βλέπουμε ότι υπάρχει σχεδόν ίσος αριθμός availability zones για κάθε availability score. Εξαίρεση παρατηρείται στον availability score ίσο με 12 όπου δεν υπάρχει ούτε ένα availability score. Εντυπωσιακό στοιχείο είναι ότι υπάρχουν 12 zones με τέλειο availability score.

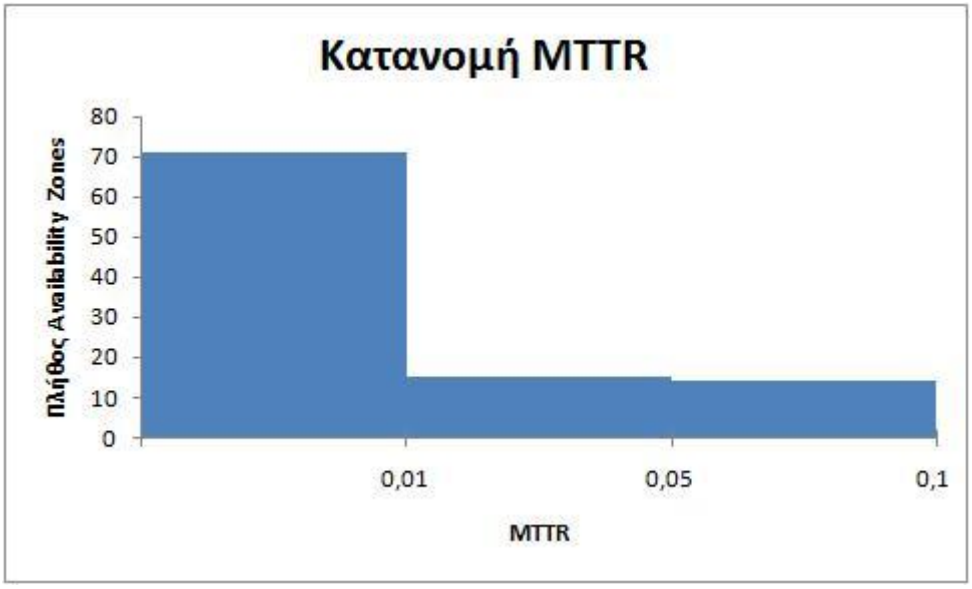
Ο αριθμός των βλαβών που έχει εμφανίσει μια availability zone όσο την επιτηρούμε είναι σημαντικός παράγοντας. Τα αντίστοιχα στατιστικά φαίνονται στο παρακάτω διάγραμμα.



ΕΙΚΟΝΑ 17:ΚΑΤΑΝΟΜΗ ΑΡΙΘΜΟΥ ΒΛΑΒΩΝ AVAILABILITY ZONES.

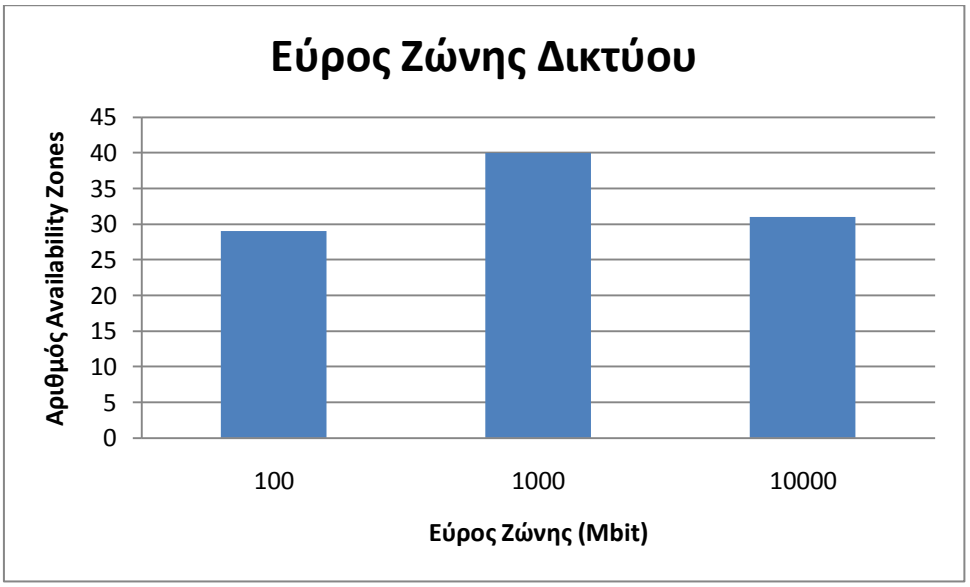
Ο μέγιστος αριθμός βλαβών που έχουν εμφανίσει οι availability zones που παρατηρούμε είναι εφτά, ενώ τα περισσότερα έχουν εμφανίσει μόλις μία βλάβη.

Από τα παραπάνω δεδομένα μπορεί να προκύψει και το MTTR του κάθε availability zone. Η αντίστοιχη κατανομή φαίνεται στο διάγραμμα:



ΕΙΚΟΝΑ 18:ΚΑΤΑΝΟΜΗ MTTR AVAILABILITY ZONES.

Ένα τελευταίο χαρακτηριστικό είναι το εύρος ζώνης του δικτύου της κάθε availability zone. Η αντίστοιχη κατανομή φαίνεται στο ακόλουθο διάγραμμα:



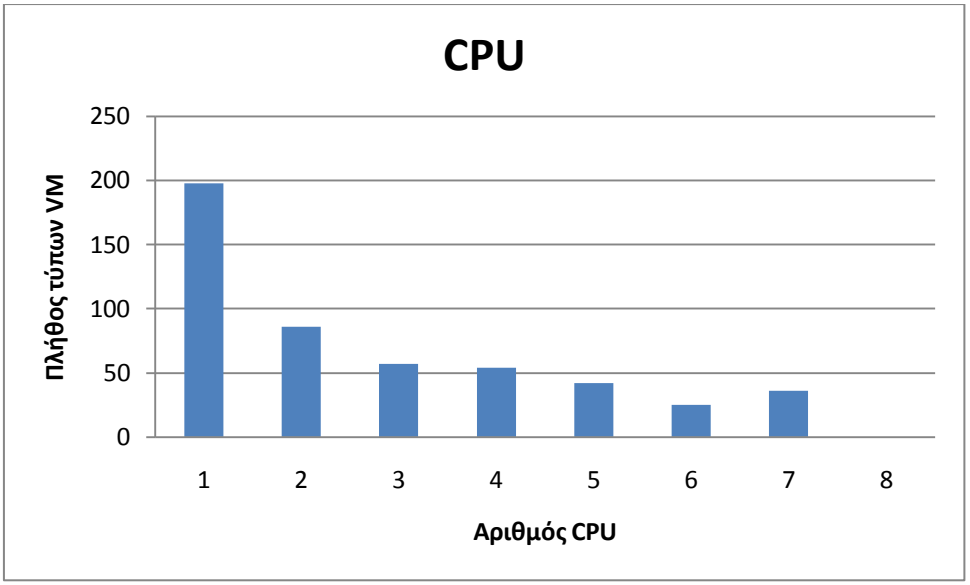
ΕΙΚΟΝΑ 19:ΚΑΤΑΝΟΜΗ ΕΥΡΟΥΣ ΖΩΝΗΣ ΔΙΚΤΥΟΥ AVAILABILITY ZONES.

Παρατηρούμε ότι υπάρχουν τοπικά δίκτυα με τρία διαφορετικά εύρη ζώνης. 100MBit, 1GBit, 10GBit, με την δεύτερη κατηγορία να είναι η πιο συνηθισμένη.

5.1.2. ΣΤΑΤΙΣΤΙΚΑ ΤΥΠΩΝ VM

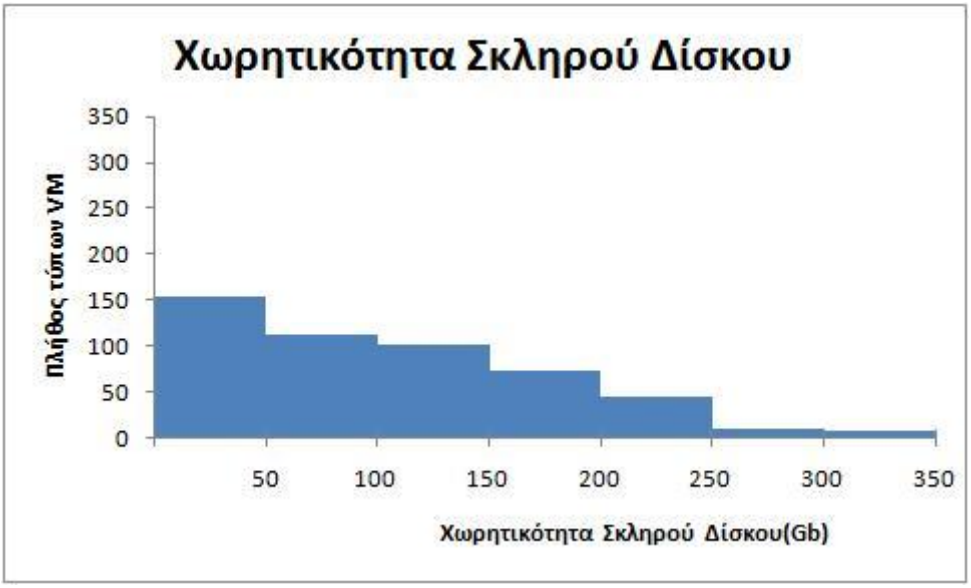
Παραπάνω είδαμε τα στατιστικά των availability zones του dataset. Πολλές σημαντικές πληροφορίες για το QoS αφορούν τους τύπους των VM instances που μπορούν να εκκινηθούν σε ένα νέφος. Τα αντίστοιχα στατιστικά του dataset φαίνονται στα ακόλουθα διαγράμματα.

Αρχικά σημαντική παράμετρος είναι ο αριθμός των επεξεργαστών που διαθέτει ένα VM instance. Η κατανομή του αριθμού επεξεργαστών φαίνεται στο παρακάτω διάγραμμα:



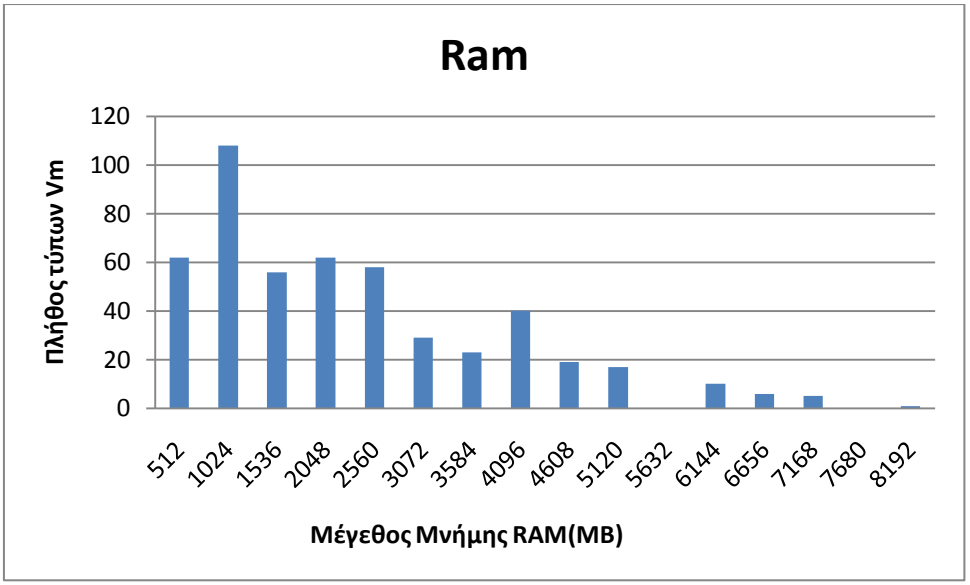
ΕΙΚΟΝΑ 20:ΚΑΤΑΝΟΜΗ CPU ΤΩΝ ΤΥΠΩΝ VM.

Η χωρητικότητα του σκληρού δίσκου επίσης αποτελεί μια σημαντική παράμετρο. Η αντίστοιχη κατανομή φαίνεται στο ακόλουθο διάγραμμα:



ΕΙΚΟΝΑ 21:ΚΑΤΑΝΟΜΗ ΧΩΡΗΤΙΚΟΤΗΤΑΣ ΣΚΛΗΡΟΥ ΔΙΣΚΟΥ ΤΩΝ ΤΥΠΩ VM.

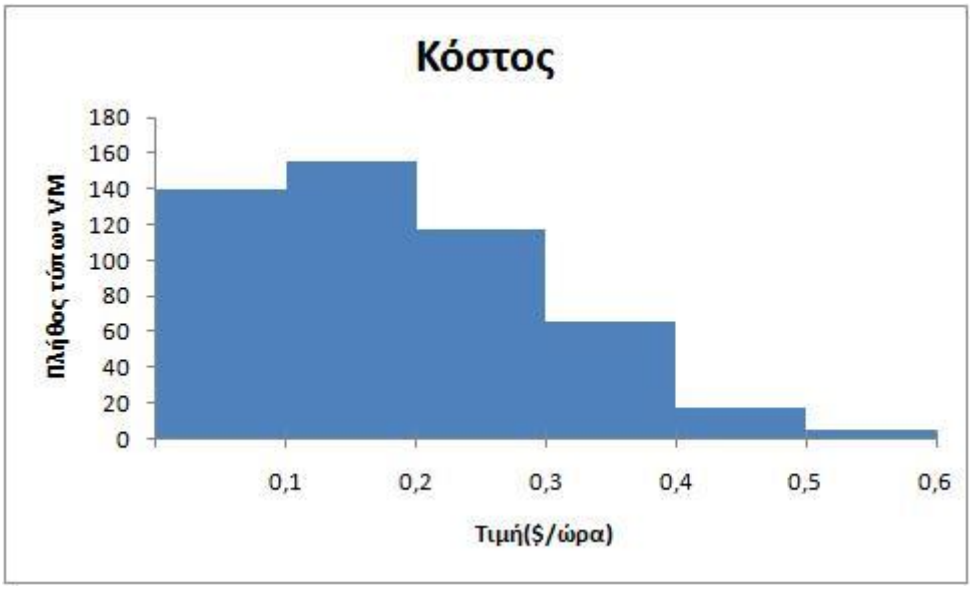
Στην συνέχεια παρουσιάζεται το μέγεθος της μνήμης RAM που διαθέτει κάθε τύπος VM.



ΕΙΚΟΝΑ 22:ΚΑΤΑΝΟΜΗ ΜΕΓΕΘΟΥΣ RAM ΤΥΠΩΝ VM.

Από τα παραπάνω διαγράμματα που αφορούν CPU, HDD , RAM βλέπουμε ότι υπάρχει μεγαλύτερος αριθμός διαθέσιμων virtual machines για μικρότερα μεγέθη των παραπάνω χαρακτηριστικών. Κάτι τέτοιο μπορεί να σημαίνει ότι όσο χαμηλότερες οι απαιτήσεις μας σε hardware τόσο περισσότερες επιλογές θα έχουμε κατά την αναζήτηση κατάλληλου cloud.

Τέλος μια πολύ σημαντική παράμετρος αποτελεί και το κόστος ανά ώρα χρήσης ενός VM instance. Η κατανομή φαίνεται στο ακόλουθο διάγραμμα:



ΕΙΚΟΝΑ 23:ΚΑΤΑΝΟΜΗ ΚΟΣΤΟΥΣ ΧΡΗΣΗΣ ΤΥΠΩΝ VM.

5.2. ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΚΤΕΛΕΣΗΣ ΑΛΓΟΡΙΘΜΟΥ

Έχοντας ως δεδομένο το dataset που περιγράφηκε ανωτέρω, προχωρήσαμε στην εκτέλεση του αλγορίθμου. Οι παράμετροι QoS που θέσαμε ήταν οι ακόλουθες:

Παράμετροι QoS	Τιμή
Αριθμός instances που θέλουμε να εκκινήσουμε	50
Αριθμός CPU	2
Μέγεθος RAM	2048
Χωρητικότητα Σκληρού Δίσκου	100
Εύρος Ζώνης Δικτύου	1000
Μέγιστο κόστος ($/ώρα)	10.00
Χρόνος χρήσης νέφους(ώρες)	50

5.2.1. ΕΚΤΕΛΕΣΗ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΣΤΑΘΕΡΕΣ ΠΑΡΑΜΕΤΡΟΥΣ QoS

Αρχικά εκτελούμε τον αλγόριθμο επιλογής για τις παραπάνω παραμέτρους QoS. Επιπλέον οι επιλογές «Βαρύτητα Αξιοπιστίας» και «Βαρύτητα Κόστους» θέτονται στην τιμή 50%. Τα πρώτα 7 αποτελέσματα που επιστρέφει ο αλγόριθμος είναι τα ακόλουθα:

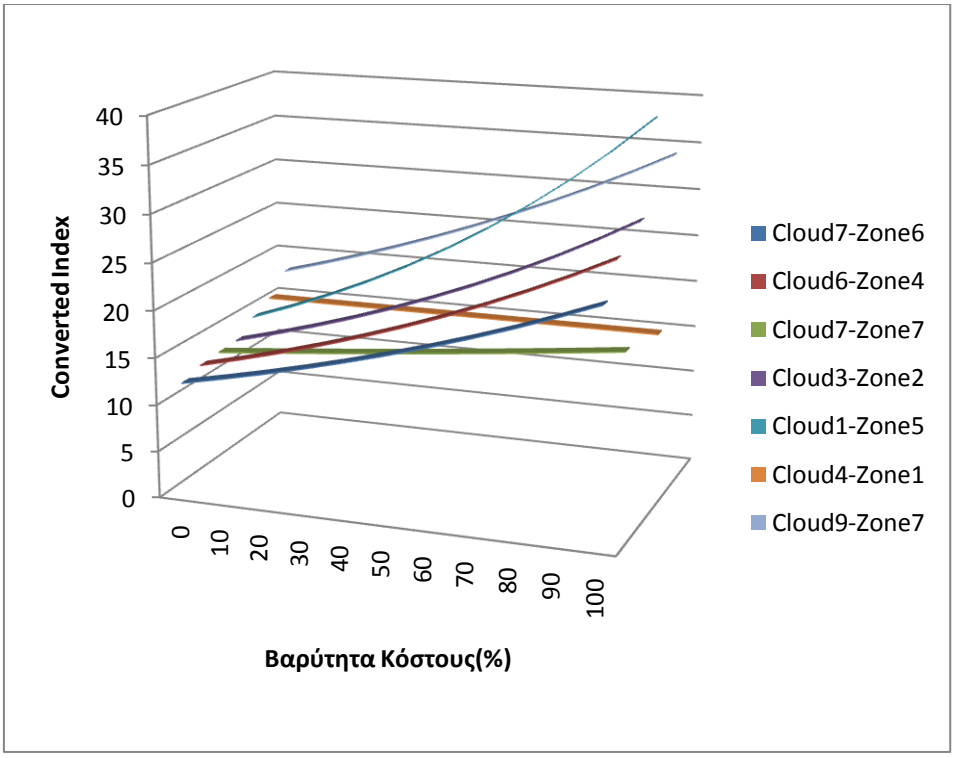
Διεύθυνση IP νέφους	Όνομα Availability Zone	Κόστος ($/ώρα)	Reliability Score	ConvertedIndex
Cloud7	Zone7	226,66	12	15,113
Cloud4	Zone1	159,54	7,99	15,786
Cloud7	Zone6	321,38	16	17,253
Cloud6	Zone4	339,15	16	18,918
Cloud3	Zone2	348,3	16	19,815
Cloud1	Zone5	387,97	16	24,033
Cloud9	Zone7	309,26	12	24,629

Βλέπουμε ότι ο αλγόριθμος επέλεξε πρώτο το availability zone με το μικρότερο ConvertedIndex. Με την βαρύτητα αξιοπιστίας και βαρύτητα κόστους να είναι στα ίδια επίπεδα, ουσιαστικά επιλέχθηκε η «μέση» λύση, δηλαδή το availability zone που ικανοποιεί στον ίδιο βαθμό και τα δύο κριτήρια. Αυτό είναι ο λόγος που αξιόπιστες λύσεις όπως η Zone 6 στο νέφος Cloud7 είναι στην τρίτη θέση , καθώς έχει αρκετά μεγάλο κόστος. Αντίθετα στην δεύτερη θέση είναι η Zone1 Cloud4 η οποία αν και όχι ιδιαίτερα αξιόπιστη είναι ιδιαίτερα οικονομική. Τέλος να παρατηρήσουμε ότι στην περίπτωση των availability zones με το ίδιο σκορ αξιοπιστίας, κατατάσσονται σε σειρά προτίμησης καθαρά με βάση το κόστος χρήσης τους.

5.2.2. ΕΚΤΕΛΕΣΗ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΜΕΤΑΒΛΗΤΕΣ ΤΙΜΕΣ ΠΑΡΑΜΕΤΡΩΝ QoS

Προκειμένου να μελετήσουμε τον βαθμό που επηρεάζουν οι μεταβλητές «βαρύτητα αξιοπιστίας» και «βαρύτητα κόστους» το τελικό αποτέλεσμα προχωρήσαμε σε μια σειρά εκτελέσεων του αλγορίθμου. Στις εκτελέσεις αυτές διατηρούσαμε σταθερή την μία από αυτές τις μεταβλητές και μεταβάλλαμε την άλλη από το 0% έως το 100%.

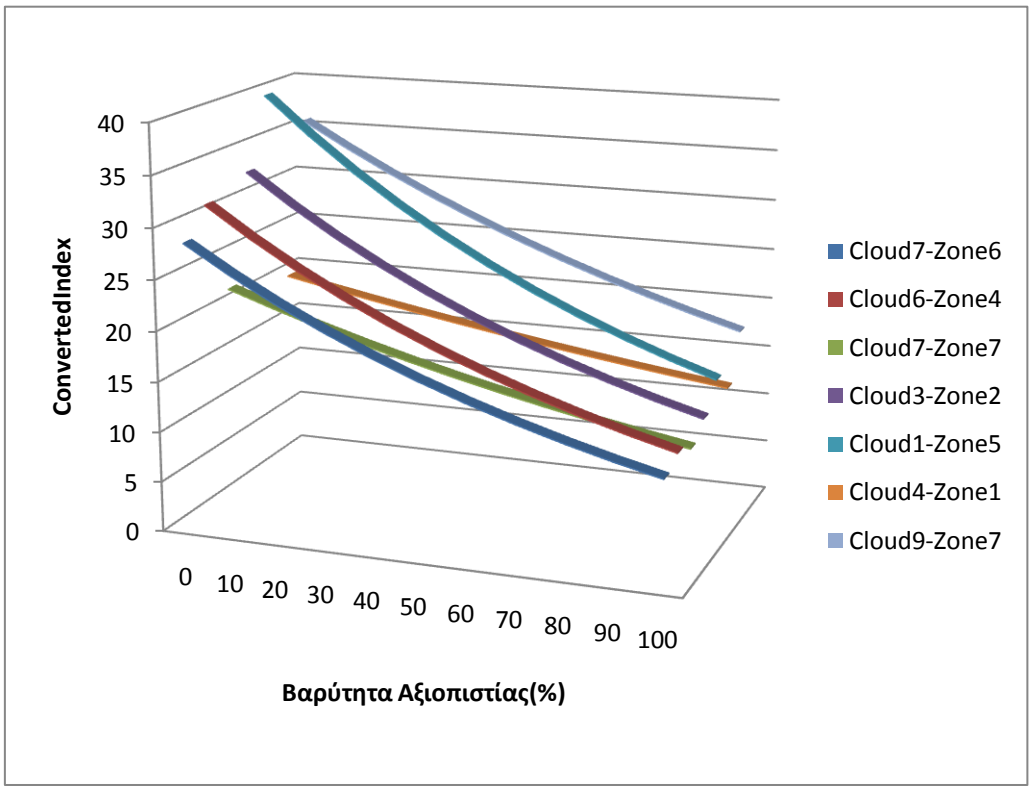
Αρχικά, θέσαμε την μεταβλητή «βαρύτητα αξιοπιστίας» στο 50%. Στο παρακάτω διάγραμμα φαίνεται η μεταβολή των δεικτών ConvertedIndex για τα 7 πρώτα availability zones που επέλεξε ο αλγόριθμος , καθώς η μεταβλητή «βαρύτητα κόστους» μεταβάλλεται από το 0% ως το 100%.



ΕΙΚΟΝΑ 24:ΔΙΑΓΡΑΜΜΑ ΜΕΤΑΒΟΛΗΣ CONVERTEDINDEX ΓΙΑ ΚΥΜΑΙΝΟΜΕΝΗ ΒΑΡΥΤΗΤΑ ΚΟΣΤΟΥΣ.

Παρατηρούμε ότι όσο αυξάνεται η βαρύτητα κόστους αυξάνεται και το ConvertedIndex όλων των availability zones. Μάλιστα, το ConvertedIndex των Zone7 Cloud7 και Zone1 Cloud4 αυξάνεται με αισθητά μικρότερο ρυθμό σε σχέση με των υπόλοιπων. Βλέποντας το κόστος χρήσης της κάθε availability zone γίνεται κατανοητός ο λόγος, καθώς το κόστος των δύο αυτών availability zones είναι έως και 50% χαμηλότερο από των υπολοίπων. Χαρακτηριστικό είναι ότι το Converted Index του Cloud1 Zone5 σχεδόν τριπλασιάζεται για βαρύτητα κόστους στο 100%. Αν και αρκετά αξιόπιστο, το συγκεκριμένο availability zone καταλήγει να είναι το τελευταίο σε σειρά επιλογής , καθώς είναι το πιο ακριβό από όλα τα διαθέσιμα.

Η δεύτερη σειρά μετρήσεων που κάναμε ήταν θέτοντας την «βαρύτητα κόστους» στο 50% και μεταβάλλοντας την «βαρύτητα αξιοπιστίας» από το 0% στο 100%. Τα αποτελέσματα που πήραμε φαίνονται ακολούθως.



ΕΙΚΟΝΑ 25:ΔΙΑΓΡΑΜΜΑ ΜΕΤΑΒΟΛΗΣ CONVERTEDINDEX ΓΙΑ ΚΥΜΑΙΝΟΜΕΝΗ ΒΑΡΥΤΗΤΑ ΑΞΙΟΠΙΣΤΙΑΣ.

Στο διάγραμμα αυτό βλέπουμε ότι όσο αυξάνεται η βαρύτητα αξιοπιστίας μειώνεται το Converted Index όλων των availability zones. Η μείωση είναι μεγαλύτερη για availability zones που διαθέτουν τον μέγιστο βαθμό αξιοπιστίας, ενώ για availability zones που δεν είναι αξιόπιστα η μείωση είναι σχεδόν μηδενική. Χαρακτηριστικά, ενώ στα περισσότερα availability zones η μείωση του ConvertedIndex είναι τουλάχιστον 50%, στο Zone1-Cloud4 η μείωση είναι σχεδόν μηδενική, καθώς το συγκεκριμένο είναι το πιο αναξιόπιστο από όλα τα availability zones.

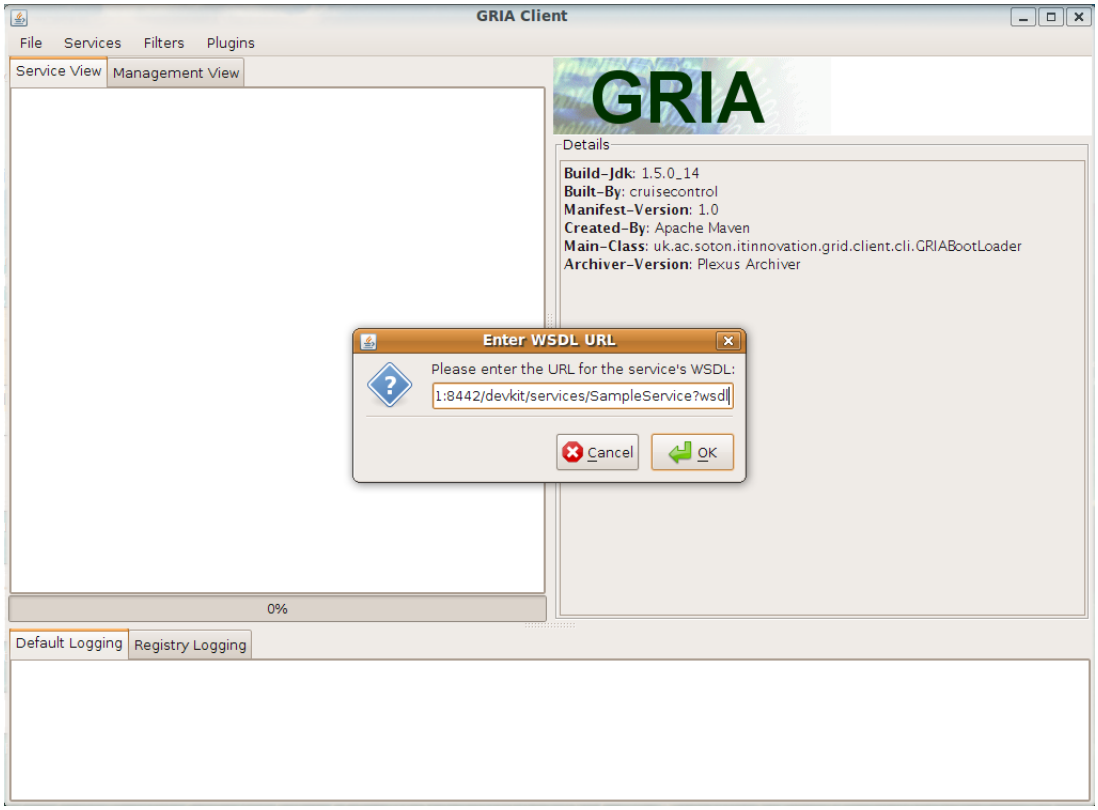
Να παρατηρήσουμε ότι καμία από τις δύο αυτές μεταβλητές δεν επηρεάζει σε απόλυτο βαθμό την τελική επιλογή νέφους. Σε κάθε περίπτωση ο αλγόριθμος θα επιλέξει μια ολοκληρωμένη λύση για τον χρήστη, η οποία θα του επιτρέψει να εκτελέσει την εργασία του απροβλημάτιστα.

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΔΕΙΞΗ ΤΗΣ ΧΡΗΣΗΣ ΤΗΣ ΥΠΗΡΕΣΙΑΣ

Ακολούθως θα παρουσιαστούν στιγμιότυπα χρήσης της υπηρεσίας συνοδευόμενα από διευκρινιστικά σχόλια.

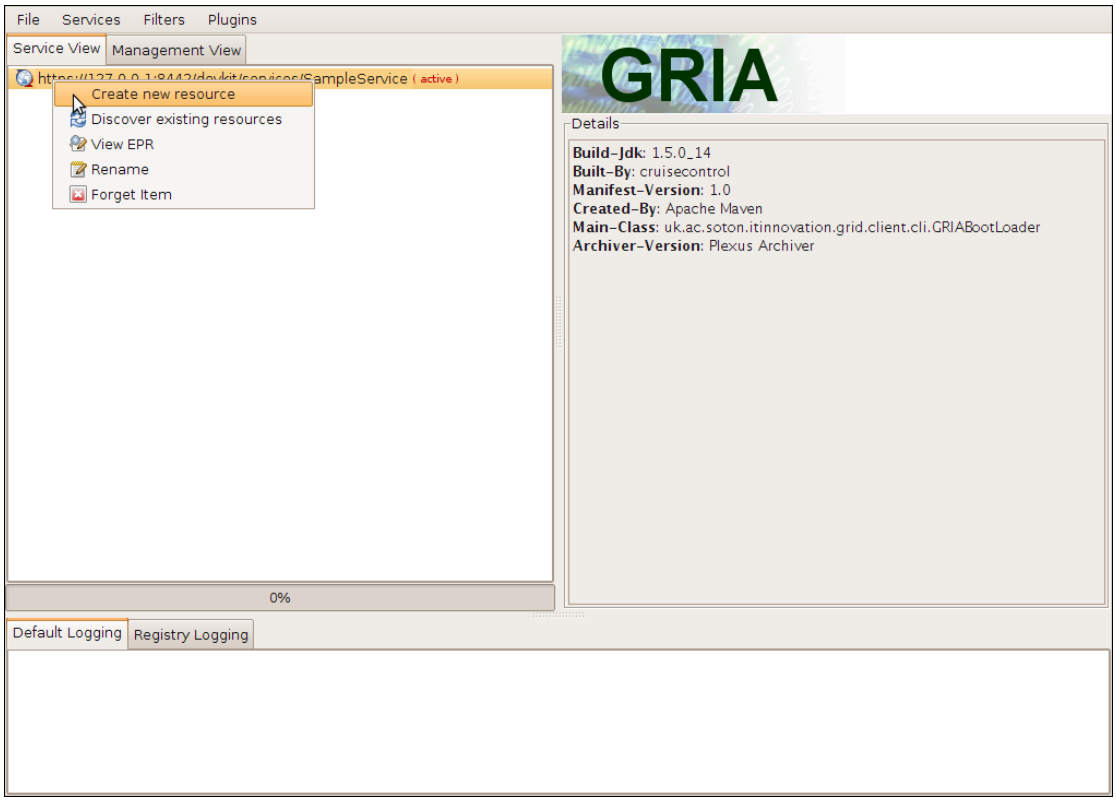
Αρχικά ο χρήστης εκκινεί τον Gria Client με το ειδικό plug-in για χρήση της υπηρεσίας ιστού μας. Προκειμένου το client να εντοπίσει την υπηρεσία, είναι απαραίτητο ο χρήστης να εισάγει στον client το wsdl της υπηρεσίας μας. Αυτό μπορεί να γίνει είτε με drag&drop του URL από την σελίδα της υπηρεσίας στο κεντρικού παράθυρο του client , είτε επιλέγοντας ***Services->Add Service*** και εισάγοντας το URL στο ειδικό παράθυρο όπως φαίνεται παρακάτω:



ΕΙΚΟΝΑ 26:ΕΙΣΑΓΩΓΗ URL ΥΠΗΡΕΣΙΑΣ ΣΤΟ ΕΙΔΙΚΟ POP UP ΠΑΡΑΘΥΡΟ.

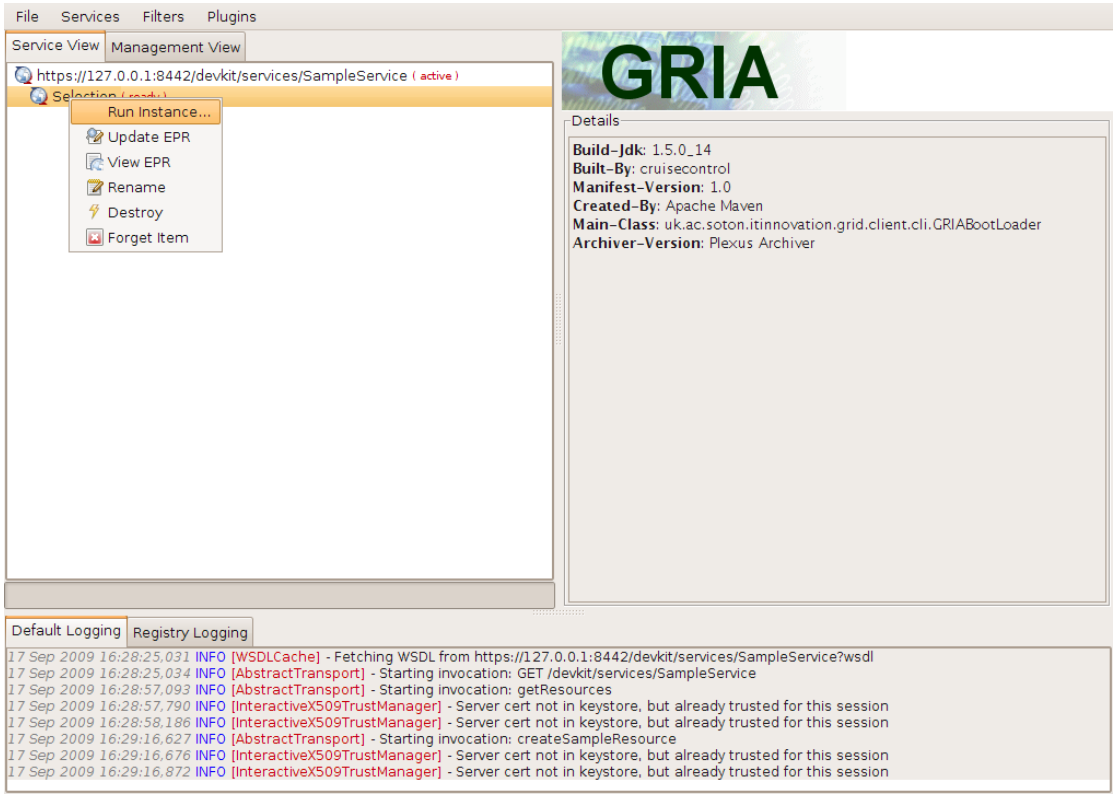
Κατά την προσθήκη της υπηρεσίας στον client θα παρουσιαστεί το ειδικό certificate της υπηρεσίας που έχουμε δημιουργήσει με την επιλογή να το απορρίψουμε ή να το δεχτούμε. Φυσικά το δεχόμαστε, και η σύνδεση με τον εξυπηρετητή έχει εγκατασταθεί.

Όπως είδαμε και κατά την παρουσίαση της δομής του συστήματος, προκειμένου να χρησιμοποιήσουμε τις δυνατότητες της Web Service πρέπει να δεσμεύσουμε πόρους, πόρους που θα χρησιμοποιηθούν για την εκτέλεση του μηχανισμού επιλογής. Αυτό μπορεί να γίνει επιλέγοντας την υπηρεσία και κάνοντας ***Δεξί Κλικ-> Create New Resource*** . Ονομάζουμε την resource όπως επιθυμούμε (εδώ "Selection") και είμαστε έτοιμοι να χρησιμοποιήσουμε την υπηρεσία.



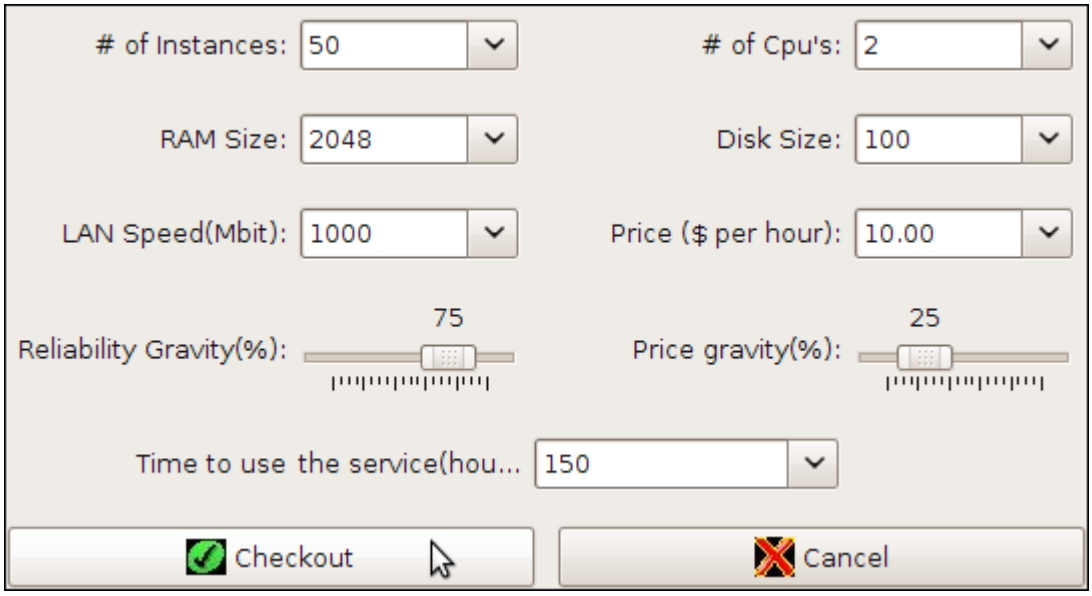
ΕΙΚΟΝΑ 27:ΔΗΜΙΟΥΡΓΙΑ ΚΑΙΝΟΥΡΓΙΟΥ RESOURCE.

Με δεξί κλικ πάνω στο resource που δημιουργήσαμε, εμφανίζονται κάποιες γενικές επιλογές που υπάρχουν για όλες τις Gria Web Services (***"Update EPR"***, ***"View EPR"***, ***"Rename"***, ***"Destroy"***,***"Forget item"***) καθώς και η επιλογή ***"Run Instance"*** η οποία υλοποιεί τον μηχανισμό μας. Όσα περιγράφουμε φαίνονται στην παρακάτω εικόνα.



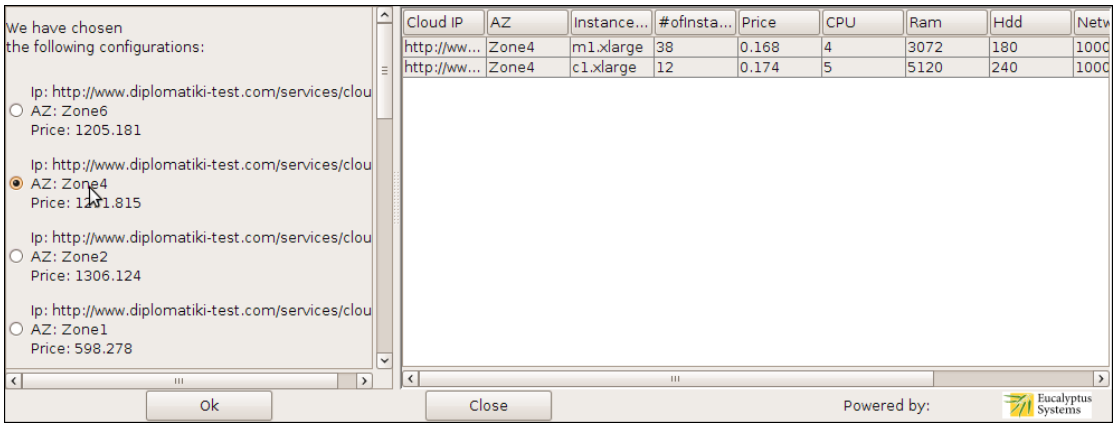
ΕΙΚΟΝΑ 28:ΕΚΤΕΛΕΣΗ ΤΟΥ ΜΗΧΑΝΙΣΜΟΥ ΕΠΙΛΟΓΗΣ.

Επιλέγοντας “Run Instances” εμφανίζεται ένα ειδικό frame, στο οποίο μπορούμε να θέσουμε τις επιλογές που επιθυμούμε για το configuration. Μπορούμε να επιλέξουμε αριθμό instances από VM που επιθυμούμε να εκκινήσουμε, το μέγεθος της μνήμης RAM, το πλήθος των επεξεργαστών και το μέγεθος του σκληρού δίσκου ανά instance, το εύρος ζώνης της διασύνδεσης μεταξύ των instances , το μέγιστο ποσό ανά ώρα που είμαστε διατεθειμένοι να πληρώσουμε και την ώρα που επιθυμούμε να χρησιμοποιήσουμε το νέφος. Επίσης θέτουμε και τι βαρύτητα πρέπει να δοθεί στους παράγοντες αξιοπιστία και κόστος (“Reliability Gravity” και “Price Gravity” αντίστοιχα).



ΕΙΚΟΝΑ 29:ΕΙΣΑΓΩΓΗ ΠΑΡΑΜΕΤΡΩΝ QOS.

Επιλέγοντας “Checkout” , τα δεδομένα που εισάγαμε στέλνονται στην Web Service και εν συνεχεία λαμβάνουμε την απάντηση της με τα αποτελέσματα της εκτέλεσης του αλγορίθμου επιλογής στα clouds της βάσης δεδομένων που διατηρούμε. Τα αποτελέσματα παρουσιάζονται με τον τρόπο που φαίνεται παρακάτω:



ΕΙΚΟΝΑ 30:ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΚΤΕΛΕΣΗΣ ΜΗΧΑΝΙΣΜΟΥ ΕΠΙΛΟΓΗΣ.

Στο αριστερό τμήμα του παραθύρου παρουσιάζονται κατά σειρά προτίμησης από το καλύτερο στο χειρότερο τα νέφη με τα αντίστοιχα Availability Zones και την συνολική τιμή για χρήση της υποδομής. Επιλέγοντας κάθε ένα από αυτά, στο δεξί τμήμα της οθόνης παρουσιάζονται τα αναλυτικά στοιχεία του configuration, δηλαδή επιπλέον πόσες instances και σε τι τύπου VM θα τις τρέξουμε, των αριθμό των διαθέσιμων CPU, τη διαθέσιμη RAM κτλ.

ΚΕΦΑΛΑΙΟ 7

ΕΠΙΛΟΓΟΣ

7.1. ΣΥΝΟΨΗ

Στην παρούσα εργασία περιγράφουμε ένα μηχανισμό αξιολόγησης και επιλογής υπολογιστικών νεφών. Στόχος της εργασίας είναι να καθοριστούν τα στοιχεία που μπορούν να χαρακτηρίσουν ένα νέφος και λαμβάνοντας αυτά τα στοιχεία να αποφανθούμε κατά πόσο μπορεί να ανταποκριθεί στις απαιτήσεις μιας εφαρμογής.

Αρχικά δίνεται το θεωρητικό υπόβαθρο για να κατανοήσει τον σχεδιασμό και την υλοποίηση του μηχανισμού. Αναλύονται τα βασικότερα χαρακτηριστικά των πιο δημοφιλών υπολογιστικών νεφών και οι βασικές παράμετροι σχεδιασμού τους με ιδιαίτερη έμφαση την ελευθερία που δίνουν στον χρήστη όσον αφορά την διαχείριση των υπολογιστικών πόρων.

Στην συνέχεια γίνεται ειδική παρουσίαση της πλατφόρμας υπολογιστικού νέφους Eucalyptus. Η υλοποίηση του μηχανισμού έγινε ειδικά για την συγκεκριμένη πλατφόρμα, επομένως είναι σημαντικό να γίνει κατανοητή η οργάνωση, ο τρόπος λειτουργίας και οι διαχωριστικές δυνατότητες τις συγκεκριμένης πλατφόρμας.

Στην συνέχεια παρουσιάζουμε τον σχεδιασμό του μηχανισμού μας. Γίνεται γενική περιγραφή της αρχιτεκτονικής του αλλά και ανάλυση του τρόπου λειτουργίας των υποσυστημάτων που το απαρτίζουν .

Προκειμένου να αξιολογήσουμε τον μηχανισμό που δημιουργήσαμε πραγματοποιούμε κάποιες δοκιμές πάνω τον αλγόριθμο για συγκεκριμένο dataset και παρουσιάζουμε τα στατιστικά των αποτελεσμάτων εκτέλεσής του. Τα αποτελέσματα που πήραμε σε κάθε περίπτωση επιβεβαίωσαν τις αρχικές εκτιμήσεις.

7.2. ΕΠΕΚΤΑΣΕΙΣ

Η παρούσα διπλωματική εργασία πέρα της περιγραφής του μηχανισμού αξιολόγησης και επιλογής υπολογιστικών νεφών, σκοπό έχει και την καταγραφή της κατάστασης που διαμορφώνεται στον χώρο του cloud computing. Είναι δυνατόν επομένως να αποτελέσει ένα «οδηγό» για τον συγκεκριμένο τομέα της επιστήμης των υπολογιστών, ο οποίος μπορεί να ανανεώνεται συνεχώς με τα νέα δεδομένα που προκύπτουν στον χώρο.

Όσον αφορά τώρα τον μηχανισμό που υλοποιήσαμε, είναι δυνατόν να δεχτεί τροποποιήσεις που θα του δώσουν νέες δυνατότητες και θα τον κάνουν πιο αποτελεσματικό στην επιλογή υπολογιστικού νέφους.

Το σύστημα που κατασκευάσαμε έχει σχεδιαστεί ώστε να συνεργάζεται με υπολογιστικά νέφη που χρησιμοποιούν το interface επικοινωνίας του Amazon EC2, δηλαδή αυτά που χρησιμοποιούν την πλατφόρμα Eucalyptus και φυσικά με το ίδιο το Amazon EC2. Μια ενδιαφέρουσα εργασία θα ήταν η επέκταση του συστήματος ώστε να επικοινωνεί και με άλλα δημοφιλή νέφη όπως το Google AppEngine ή το Microsoft Azure.

Ο μηχανισμός αυτή τη στιγμή , μετά την επιλογή του κατάλληλου υπολογιστικού νέφους για τον χρήστη, του παρουσιάζει το αναλυτικό configuration για να τρέξει τα instances. Πιθανή επέκταση θα ήταν η δημιουργία ενός πιο ολοκληρωμένου συστήματος ώστε μετά την διαδικασία της επιλογής να δίνεται η δυνατότητα στον χρήστη να εκκινήσει τα instances στο επιλεγμένο νέφος μέσα από τον Gria Client καθώς και η δυνατότητα να τα διαχειριστεί και να λάβει αποτελέσματα , χωρίς να εμπλέκεται άμεσα με το interface του κάθε νέφους.

Όσον αφορά τον αλγόριθμο επιλογής , θα μπορούσε να εξελιχτεί ώστε να περιλαμβάνει περισσότερες επιλογές για την αξιολόγηση και επιλογή του νέφους. Αυτή τη στιγμή ο αλγόριθμος λαμβάνει υπόψη τους παράγοντες κόστος και αξιοπιστία του νέφους. Μια πιθανή επέκταση θα ήταν να προστεθεί κάποια μετρική της απόδοσης του νέφους , ανάλογα του hardware. Μια αξιολόγηση των διαθέσιμων υπολογιστικών πόρων.

Επιπλέον, μια μεθοδολογία που χρησιμοποιείται ευρέως στην αξιολόγηση υπηρεσιών θα μπορούσε να χρησιμοποιηθεί και στην συγκεκριμένη περίπτωση αφορά την υλοποίηση κάποιας μορφής feedback των χρηστών. Μετά την εκτέλεση της εργασίας του σε ένα υπολογιστικό νέφος, ο χρήστης έχει την δυνατότητα να αξιολογήσει την ποιότητα υπηρεσιών που του παρείχε το νέφος. Η γνώση αυτή θα μπορούσε να χρησιμοποιηθεί από τον αλγόριθμο επιλογής σε μελλοντικές αξιολογήσεις του νέφους.

Θα μπορούσαν να προστεθούν περισσότερες παράμετροι QoS τις οποίες θα ήταν δυνατό να ελέγχει ο χρήστης. Καθορισμός Min – Max τιμών, αξιολόγηση βαρύτητας παραμέτρων, γενικά περισσότερες επιλογές που θα δίνουν τη δυνατότητα στον χρήστη να περιγράφει πιο αναλυτικά τις απαιτήσεις του.

Όπως είδαμε, ο όλος μηχανισμός υλοποιήθηκε με την βοήθεια της πλατφόρμας Gria Development Kit και έχει μορφή υπηρεσίας ιστού που είναι προσβάσιμη με τη βοήθεια του Gria Client. Μια εναλλακτική προσέγγιση θα ήταν η υλοποίηση του μηχανισμού σαν ιστοσελίδα. Με αυτό τον τρόπο θα γίνονταν πιο απλοϊκή η χρήση του, θυσιάζοντας κάποιες από τις δυνατότητες ασφαλείας του Gria.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, “Above the Clouds: A Berkley View on Cloud Computing”, Electrical Engineering and Computer Sciences University of California at Berkeley, February 10 2009

[2] J. Carolan, S. Gaede, “Introduction to Cloud Computing Architecture”, Sun Microsystems, June 2009

[3] TC3 Health Case Study: Amazon Web Services [online]. Available from: http://aws.amazon.com/solutions/case-studies/tc3-health/.

[4] Washington Post Case Study: Amazon Web Services [online]. Available from: http://aws.amazon.com/solutions/case-studies/washington-post/.

[5] Amazon.com CEO Jeff Bezos on Animoto [online]. April 2008. Available from: http://blog.animoto.com/2008/04/21/amazon-ceo-jeff-bezos-on-animoto/.

[6] Amazon EC2 Features [online]. Available from: http://aws.amazon.com/ec2/.

[7] Brian Hayes, “Cloud Computing”, Communications of the ACM, Vol. 51, No.7, July 2008

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities” , 2008

[9] GARFINKEL, S. An Evaluation of Amazon’s Grid Computing Services: EC2, S3 and SQS . Tech. Rep. TR-08-07, Harvard University, August 2007.

[10] Twenty Experts Define Cloud Computing, 18 July 2008 [online],Available from: http://cloudcomputing.sys-con.com/read/612375_p.htm .

[11] S. London. INSIDE TRACK: The high-tech rebels. Financial Times, 06 Sept. 2002[online]. Available from: http://search.ft.com/nonFtArticle?id=020906000371

[12] Amazon Web Services (WS), http:// aws.amazon.com/ ,21 September 2009

[13] Google App Engine, http://code.google.com/intl/el-GR/appengine/, 21 September 2009

[14] Microsoft Azure, http://www.microsoft.com/azure/ , 21 September 2009

[15] Paypal, https://www.paypal.com/gr , 21 September 2009

[16] Google AdSense , https://www.google.com/adsense/login/el/ , 21 September 2009

[17] VOGELS, W. A Head in the Clouds—The Power of Infrastructure as a Service. In First workshop on Cloud Computing and in Applications (CCA '08) (October 2008).

[18] SIEGELE, L. Let It Rise: A Special Report on Corporate IT. The Economist (October 2008).

[19] DEAN, J., AND GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation (Berkeley, CA, USA, 2004), USENIX Association, pp. 10–10

[20] BIALECKI, A., CAFARELLA, M., CUTTING, D., AND O'MALLEY, O. Hadoop: a framework for running applications on large clusters built of commodity hardware. Wiki at http://lucene. apache. org/hadoop.

[21] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The google file system. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles (New York, NY, USA, 2003), ACM, pp. 29–43. Available from: http://portal.acm.org/ft_gateway.cfm?id=945450&type=pdf&coll=Portal&dl=GUIDE&CFID=19219697&CFTOKEN=50259492.

[22] CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W., WALLACH, D., BURROWS, M., CHANDRA, T., FIKES, A., AND GRUBER, R. Bigtable: A distributed storage system for structured data. In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06) (2006).

[23] DECANDIA, G., HASTORUN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN,S., VOSSHALL, P., AND VOGELS, W. Dynamo: Amazon's highly available key-value store. In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles (2007), ACM Press New York, NY, USA, pp. 205–220.

[24] HAMILTON, J. Cost of Power in Large-Scale Data Centers [online]. November 2008. Available from: http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx.

[25] VOGELS, W. A Head in the Clouds—The Power of Infrastructure as a Service. In First workshop on Cloud Computing and in Applications (CCA '08) (October 2008).

[26] Salesforce.com , http://www.salesforce.com/platform/ , 21 September 2009

[27] Rackspace Cloud, http://www.rackspace.com/index.php , 21 September 2009

[28] EngineYard Cloud, http://www.engineyard.com/ , 21 September 2009

[29] Michael Vizard, "The need to build private clouds", 30 April 2009[online]. Available from : http://blogs.eweek.com/masked_intentions/content/cloud_computing/the_need_to_build_private_clouds.html

[30] Chris Preimesberger , "Get off my Cloud: Private Cloud Computing Takes Shape", 4 November 2008[online]. Available from : http://www.eweek.com/c/a/Cloud-Computing/Why-Private-Cloud-Computing-Is-Beginning-to-Get-Traction/

[31] Eucalyptus System Inc, http://www.eucalyptus.com/ , 21 September 2009

[32] Eucalyptus - Elastic Utility Computing Architecture Linking Your Programs To Useful Systems, http://open.eucalyptus.com/ , 21 September 2009

[33] Canonical, "Ubuntu: An introduction to cloud computing", 21 September 2009

[34] Enomalism elastic computing infrastructure. http://www.enomaly.com , 21 September 2009

[35] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164–177, New York, NY, USA, 2003. ACM.

[36] F. Bellard. QEMU, a Fast and Portable Dynamic Translator. Proceedings of the USENIX Annual Technical Conference, FREENIX Track, pages 41–46, 2005.

[37] Vmware home page – http://www.vmware.com

[38] M. Krasnyansky. VTun-Virtual Tunnels over TCP/IP networks, 2003

[39] Amazon simple storage service api (2008-12-01) http://docs.amazonwebservices.com/AmazonS3/2008-12-01/.

[40] Amazon Elastic Compute Cloud(EC2) api (2008-12-01) http://docs.amazonwebservices.com/AWSEC2/2008-12-01/

[41] J. Chase, D. Irwin, L. Grit, J. Moore, and S. Sprenkle. Dynamic virtual clusters in a grid site manager. High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on, pages 90–100, 2003.

[42] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. Sci. Program., 13(4):265–275, 2005.

[43] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher: An Extensible Framework for Managing Clusters of Virtual Machines. In Proceedings of the 21st Large Installation System Administration Conference (LISA),November 2007.

[44] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, "Eucalyptus : A Technical Report on an Elastic Utility Computing Archietcture Linking Your Programs to Useful Systems",UCSB Computer Science Technical Report Number 2008-10, Octomber 2008

[45] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, "The Eucalyptus Open-source Cloud-computing System", 21 September 2009

[46] Todd Hoff, " Is Eucalyptus Ready to be your private Cloud?", 29 May 2009[online]. Available from : http://highscalability.com/eucalyptus-ready-be-your-private-cloud

[47] Eli Lilly, " Nasa builds Eucalyptus Clouds", 5 June 2009[online]. Available from: http://www.informationweek.com/cloud-computing/blog/archives/2009/06/eli_lilly_nasa.html

[48] F. Berman, G. Fox, and T. Hey. Grid Computing: Making the Global Infrastructure a Reality. Wiley and Sons, 2003.

[49] I. Foster and C. Kesselman, editors. The Grid – Blueprint for a New Computing Infrastructure. Morgan Kaufmann,1998.

[50] NSF TeraGrid Project. http://www.teragrid.org/.

[51] T. Tannenbaum and M. Litzkow. The condor distributed processing system. Dr. Dobbs Journal, February 1995

[52] VDE- Virtual Distributed Ethernet, http://vde.sourceforge.net/

[53] Casazza, J.P., Green,eld, M., Shi, K., Redening server performance characterization for virtualization benchmarking. Intel Technology Journal 10 ,2006,

[54] Gregor N. Purdy, "Linux Iptables pocket reference", 2004

[55] Rightscale, "Rightscale+ Ubuntu+Eucalyptus= cloud in a box", 20 April 2009 [online]. Available from : http://blog.rightscale.com/2009/04/20/rightscale-ubuntu-eucalyptus-cloud-in-a-box/

[56] John Foley , “Eli Lilly on What’s Next in Cloud Computing”, 14 January 2009[online]. Available from : http://www.informationweek.com/cloud-computing/blog/archives/2009/01/whats_next_in_t.html

[57] HSQL Database Engine, http://hsqldb.org/

[58] Gria , Grid Resources for industrial applications http://www.gria.org/

[59] R. Perlman, “An Overview of PKI trust models”,ieee.explore.ieee.org November 1999

[60] M Surridge, S Taylor, D De Roure, E Zaluska, “Experiences with GRIA—Industrial Applications on a Web Services Grid”,2005

[61] Jari Koistinen , “Dimensions for Reliability Contracts in Distributed Object Systems”, Octomber 1997

[62] Gray J., Reuter S. Transaction Processing: Concepts and Techniques, 1993

[63] Cristian F., “Understanding fault-tolerant distributed systems Commun”. ACM 34 (2) 56–78, 1991

[64] Reibman A L and Veeraraghavan M 1991 Reliability modeling: an overview for system designers IEEE, 49–57 , April 1991

[65] Birman K. “ ISIS: a system for fault-tolerant distributed computing Technical Report TR86-744”, Department of Computer Science, Cornell University, 1986

[66] Littlewood B. “Software reliability modelling Software Engineer’s Reference book”, section 31, Butterworth-Heinemann, 1991

[67] Min-You Wu, Wei Shu, H.Zhang, Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems, in: Heterogeneous Computing Workshop, 2000.

[68] Shanshan Song, Yu-Kwong Kwok, Kai Hwang, Security-driven heuristics and a fast genetic algorithm for trusted Grid job scheduling parallel and distributed processing symposium, in : 19th IEEE International 04-08 April, 2005

[69] R.Buyya, M. Murshed, D. Abramson, A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global Grids, in : Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA702, 2002.

[70] R.Buyya, D. Abramson, S.Venugopal, The Grid economy, Proceedings of the IEEE 93(3) (2005) 698-714

[71] D. Kyriazis, K. Tserpes, A. Menychtas, A. Litke, T. Varvarigou, An innovative workflow mapping mechanism for Grids in the frame of Quality of Service, 27 July 2007

ΠΑΡΑΡΤΗΜΑ

ΚΩΔΙΚΑΣ ΚΛΑΣΕΩΝ

Στο παράρτημα αυτό περιέχεται ο κώδικας της υλοποίησης του μηχανισμού. Όπως αναφέρθηκε και προηγουμένως κάναμε χρήση του Gria Development Kit και συγκεκριμένα στηριχτήκαμε σε ένα παράδειγμα υπηρεσίας(Sample Service) που ήταν διαθέσιμο μέσω στο development kit. Ο κώδικας που παρουσιάζουμε αποτελεί προσθήκη στον ήδη υπάρχοντα κώδικα της υπηρεσίας.

Αρχικά παρουσιάζουμε τον κώδικα του plug-in του client ο οποίος χωρίζεται σε δύο πακέτα. Το πρώτο πακέτο είναι το gui. Οι κλάσεις και ο κώδικας που περιέχονται σε αυτό δίνονται στην συνέχεια.

INPUTDIALOG.JAVA

```
package gui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.File;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import sample.SampleResource;
import sample.CustomerOptions;
```

```
import java.rmi.RemoteException;


public class InputDialog extends JDialog implements
ActionListener,WindowListener{

	/**
	 *
	 */
	private static final long serialVersionUID =
7034221373288599186L;
	private boolean start;
	private JLabel
kernelLabel,ramdiskLabel,imageLabel,keyLabel,instancesLabel,ramLabel,
diskLabel,cpuLabel,netLabel,priceLabel,reliabilitysliderLabel,pricesl
iderLabel,timeLabel;
	private JTextField kernelText,ramdiskText,imageText,keyText;
	private JButton
kernelButton,ramdiskButton,imageButton,keyButton,runButton,cancelButt
on;
	private JComboBox instances,ram,disk,cpu,net,price,time;
	private JFileChooser fc;
	private JOptionPane optionPane ;
	private boolean run;
	private Object sample;
	private JSlider reliabilityslider,priceslider;
	private String dir;

	public InputDialog(Object sample){


		this.sample = sample;
		run=false;
		start= false;
		setLayout(new BorderLayout());
		dir="/home/kostas/gria/gria-client-cli-5.3/";

		kernelLabel = new JLabel("Enter kernel manifest
file:",JLabel.TRAILING);
		ramdiskLabel = new JLabel("Enter ramdisk manifest
file:",JLabel.TRAILING);
		imageLabel = new JLabel("Enter image manifest
file:",JLabel.TRAILING);
		keyLabel = new JLabel("Enter key
file",JLabel.TRAILING);
		instancesLabel = new JLabel("# of
Instances:",JLabel.TRAILING);
		cpuLabel = new JLabel("# of
Cpu's:",JLabel.TRAILING);
		ramLabel = new JLabel("RAM
Size:",JLabel.TRAILING);
		diskLabel = new JLabel("Disk
Size:",JLabel.TRAILING);
		netLabel = new JLabel("LAN
Speed(Mbit):",JLabel.TRAILING);
		priceLabel = new JLabel("Price ($ per
hour):",JLabel.TRAILING);
		reliabilitysliderLabel = new JLabel("Reliability
Gravity(%):",JLabel.TRAILING);
		pricesliderLabel= new JLabel("Price
gravity(%):",JLabel.TRAILING);
```

```
            timeLabel          =        new        JLabel("Time        to
use",JLabel.TRAILING);
            JLabel       timeLabel2       =       new       JLabel("the
service(hours):",JLabel.LEADING);


            kernelText   = new JTextField();
            ramdiskText  = new JTextField();
            imageText    = new JTextField();
            keyText      = new JTextField();

            fc = new JFileChooser();
            fc.setFileSelectionMode(JFileChooser.FILES_ONLY);

            kernelButton       = new JButton("Browse...");
            kernelButton.addActionListener(this);




            ramdiskButton      = new JButton("Browse...");
            ramdiskButton.addActionListener(this);

            imageButton        = new JButton("Browse...");
            imageButton.addActionListener(this);

            keyButton          = new JButton("Browse...");
            keyButton.addActionListener(this);
            cpu                =            new            JComboBox(new
String[]{"1","2","3","4","5","6","7","8"});
            cpu.setEditable(false);

            reliabilityslider                  =                  new
JSlider(JSlider.HORIZONTAL,0,100,50);
            reliabilityslider.setMajorTickSpacing(20);
            reliabilityslider.setMinorTickSpacing(5);
            reliabilityslider.setPaintTicks(true);


            priceslider = new JSlider(JSlider.HORIZONTAL,0,100,50);
            priceslider.setMajorTickSpacing(20);
            priceslider.setMinorTickSpacing(5);
            priceslider.setPaintTicks(true);


            time = new JComboBox();
            time.setEditable(true);
            for(int i=1;i<300;i++)
                  time.addItem(i+"");
            instances    = new JComboBox();
            instances.setEditable(false);
            for(int i =1;i<=256;i++)
                  instances.addItem(""+i);

            ram                = new JComboBox();
            ram.setEditable(true);
            for(int i=128;i<=8192;i+=128)
                  ram.addItem(""+i);

            disk         = new JComboBox();
            disk.setEditable(true);
```

```
            for(int i=1;i<=500;i++)
                  disk.addItem(""+i);

            net                =new JComboBox();
            net.setEditable(false);
            net.addItem("100");
            net.addItem("1000");
            net.addItem("10000");
            net.addItem("100000");

            price        =new JComboBox();
            price.setEditable(true);
            price.addItem("0.10");
            price.addItem("0.20");
            price.addItem("0.30");
            price.addItem("0.40");
            price.addItem("0.50");
            price.addItem("0.60");


            JPanel insidePanel = new JPanel(new GridLayout(1,1,5,5));
            JPanel  browsePanel  =  new  JPanel(new  SpringLayout());
//use FlowLayout
            JPanel infoPanel = new JPanel(new SpringLayout());


            browsePanel.add(kernelLabel);
            kernelLabel.setLabelFor(kernelText);
            browsePanel.add(kernelText);
            browsePanel.add(kernelButton);
            browsePanel.add(ramdiskLabel);
            ramdiskLabel.setLabelFor(ramdiskText);
            browsePanel.add(ramdiskText);
            browsePanel.add(ramdiskButton);
            browsePanel.add(imageLabel);
            imageLabel.setLabelFor(imageText);
            browsePanel.add(imageText);
            browsePanel.add(imageButton);
            browsePanel.add(keyLabel);
            keyLabel.setLabelFor(keyText);
            browsePanel.add(keyText);
            browsePanel.add(keyButton);
            SpringUtilities.makeCompactGrid(browsePanel,
                        4, 3, //rows, cols
                        6, 6,        //initX, initY
                        6, 6);       //xPad, yPad

            infoPanel.add(instancesLabel);
            instancesLabel.setLabelFor(instances);
            infoPanel.add(instances);
            infoPanel.add(cpuLabel);
            cpuLabel.setLabelFor(cpu);
            infoPanel.add(cpu);
            infoPanel.add(ramLabel);
            ramLabel.setLabelFor(ram);
            infoPanel.add(ram);
            infoPanel.add(diskLabel);
            diskLabel.setLabelFor(disk);
            infoPanel.add(disk);
            infoPanel.add(netLabel);
            netLabel.setLabelFor(net);
```

```
            infoPanel.add(net);
            infoPanel.add(priceLabel);
            priceLabel.setLabelFor(price);
            infoPanel.add(price);
            infoPanel.add(reliabilitysliderLabel);
            reliabilitysliderLabel.setLabelFor(reliabilityslider);
            infoPanel.add(reliabilityslider);
            infoPanel.add(pricesliderLabel);
            pricesliderLabel.setLabelFor(priceslider);
            infoPanel.add(priceslider);
            infoPanel.add(timeLabel);
            infoPanel.add(timeLabel2);
            infoPanel.add(time);
            infoPanel.add(new JLabel());

            SpringUtilities.makeCompactGrid(infoPanel,
                        5, 4, //rows, cols
                        6, 6,        //initX, initY
                        6, 20);       //xPad, yPad

            insidePanel.add(infoPanel);
            setDefaultCloseOperation(DISPOSE_ON_CLOSE);



            JPanel       buttonPanel       =       new       JPanel(new
GridLayout(1,2,10,10));
            runButton    =          new          JButton("Checkout",new
ImageIcon(dir+"resources/yes.jpg"));
            runButton.addActionListener(this);
            cancelButton       =        new        JButton("Cancel",new
ImageIcon(dir+"resources/no.jpg"));
            cancelButton.addActionListener(this);
            buttonPanel.add(runButton);
            buttonPanel.add(cancelButton);

            add(infoPanel,BorderLayout.CENTER);
            add(buttonPanel,BorderLayout.SOUTH);
            setTitle("Set Options");

            Image                         img                         =
Toolkit.getDefaultToolkit().getImage(dir+"resources/pyrforos.gif");
            setIconImage(img);
            setMinimumSize(new Dimension(550,320));
            Dimension dim = getToolkit().getScreenSize();
            Rectangle abounds = getBounds();
            setLocation((dim.width - abounds.width) / 2,
                        (dim.height - abounds.height) / 2);
            setVisible(true);
            addWindowListener(this);


      }


      public String getOptions(){
            if(run){
                  String result="";
                  result+=instances.getSelectedItem().toString()+" ";
                  result+=cpu.getSelectedItem().toString()+" ";
```

```
                    result+=ram.getSelectedItem().toString()+" ";
                    result+=disk.getSelectedItem().toString()+" ";
                    result+=price.getSelectedItem().toString()+" ";
                    result+=net.getSelectedItem().toString()+" ";
                    result+=priceslider.getValue()+ " ";
                    result+=reliabilityslider.getValue()+" ";
                    result+=time.getSelectedItem().toString();
                    return result;


              }
              return null;

        }



        public void actionPerformed(ActionEvent e) {


              if (e.getSource() == kernelButton) {
                    int                    returnVal                    =
fc.showOpenDialog(InputDialog.this);

                    if (returnVal == JFileChooser.APPROVE_OPTION) {
                          File file = fc.getSelectedFile();
                          kernelText.setText(file.getAbsolutePath());
                    }
              }

              if (e.getSource() == ramdiskButton) {
                    int                    returnVal                    =
fc.showOpenDialog(InputDialog.this);

                    if (returnVal == JFileChooser.APPROVE_OPTION) {
                          File file = fc.getSelectedFile();
                          ramdiskText.setText(file.getAbsolutePath());
                    }
              }

              if (e.getSource() == imageButton) {
                    int                    returnVal                    =
fc.showOpenDialog(InputDialog.this);

                    if (returnVal == JFileChooser.APPROVE_OPTION) {
                          File file = fc.getSelectedFile();
                          imageText.setText(file.getAbsolutePath());
                    }
              }

              if (e.getSource() == keyButton) {
                    int                    returnVal                    =
fc.showOpenDialog(InputDialog.this);

                    if (returnVal == JFileChooser.APPROVE_OPTION) {
                          File file = fc.getSelectedFile();
                          keyText.setText(file.getAbsolutePath());
                    }
              }

              if (e.getSource() == runButton){
```

```
                    run=true;
                    dispose();
                    setVisible(false);

              }

              if( e.getSource() == cancelButton){
                    dispose();
                    setVisible(false);

              }

        }


        public void windowClosed(WindowEvent arg0) {
              if(run&&!start){
                    start=true;
                    String options=getOptions();
                    String result=null;
                    try{

                          result=((SampleResource)
sample).check(options);
                    }
                    catch(RemoteException e){
                          e.printStackTrace();
                    }
                    ResultDialog out=new ResultDialog(result);

              }
        }
        public void windowActivated(WindowEvent arg0) {
        }
        public void windowClosing(WindowEvent arg0) {
        }
        public void windowDeactivated(WindowEvent arg0) {
        }
        public void windowDeiconified(WindowEvent arg0) {
        }
        public void windowIconified(WindowEvent arg0) {
        }
        public void windowOpened(WindowEvent arg0) {
        }

}
```

SPRINGUTILITIES.JAVA

```
package gui;

import javax.swing.*;
import javax.swing.SpringLayout;
import java.awt.*;
```

```
public class SpringUtilities {

      public static void printSizes(Component c) {
            System.out.println("minimumSize        =        "        +
c.getMinimumSize());
            System.out.println("preferredSize       =       "       +
c.getPreferredSize());
            System.out.println("maximumSize        =        "        +
c.getMaximumSize());
      }


      public static void makeGrid(Container parent,
                  int rows, int cols,
                  int initialX, int initialY,
                  int xPad, int yPad) {
            SpringLayout layout;
            try {
                  layout = (SpringLayout)parent.getLayout();
            } catch (ClassCastException exc) {
                  System.err.println("The first argument to makeGrid
must use SpringLayout.");
                  return;
            }

            Spring xPadSpring = Spring.constant(xPad);
            Spring yPadSpring = Spring.constant(yPad);
            Spring initialXSpring = Spring.constant(initialX);
            Spring initialYSpring = Spring.constant(initialY);
            int max = rows * cols;


            Spring                  maxWidthSpring                  =
layout.getConstraints(parent.getComponent(0)).
            getWidth();
            Spring                  maxHeightSpring                 =
layout.getConstraints(parent.getComponent(0)).
            getWidth();
            for (int i = 1; i < max; i++) {
                  SpringLayout.Constraints           cons           =
layout.getConstraints(
                              parent.getComponent(i));

                  maxWidthSpring     =     Spring.max(maxWidthSpring,
cons.getWidth());
                  maxHeightSpring    =    Spring.max(maxHeightSpring,
cons.getHeight());
            }


            for (int i = 0; i < max; i++) {
                  SpringLayout.Constraints           cons           =
layout.getConstraints(
                              parent.getComponent(i));

                  cons.setWidth(maxWidthSpring);
                  cons.setHeight(maxHeightSpring);
            }
```

```
            SpringLayout.Constraints lastCons = null;
            SpringLayout.Constraints lastRowCons = null;
            for (int i = 0; i < max; i++) {
                  SpringLayout.Constraints           cons           =
layout.getConstraints(
                              parent.getComponent(i));
                  if (i % cols == 0) {
                        lastRowCons = lastCons;
                        cons.setX(initialXSpring);
                  } else {

      cons.setX(Spring.sum(lastCons.getConstraint(SpringLayout.EAST),
                                    xPadSpring));
                  }

                  if (i / cols == 0) {
                        cons.setY(initialYSpring);
                  } else {

      cons.setY(Spring.sum(lastRowCons.getConstraint(SpringLayout.SOU
TH),
                                    yPadSpring));
                  }
                  lastCons = cons;
            }

            SpringLayout.Constraints              pCons              =
layout.getConstraints(parent);
            pCons.setConstraint(SpringLayout.SOUTH,
                        Spring.sum(
                                    Spring.constant(yPad),

      lastCons.getConstraint(SpringLayout.SOUTH)));
            pCons.setConstraint(SpringLayout.EAST,
                        Spring.sum(
                                    Spring.constant(xPad),

      lastCons.getConstraint(SpringLayout.EAST)));
      }

      private static SpringLayout.Constraints getConstraintsForCell(
                  int row, int col,
                  Container parent,
                  int cols) {
            SpringLayout layout = (SpringLayout) parent.getLayout();
            Component c = parent.getComponent(row * cols + col);
            return layout.getConstraints(c);
      }

      public static void makeCompactGrid(Container parent,
                  int rows, int cols,
                  int initialX, int initialY,
                  int xPad, int yPad) {
            SpringLayout layout;
            try {
                  layout = (SpringLayout)parent.getLayout();
            } catch (ClassCastException exc) {
                  System.err.println("The     first     argument     to
makeCompactGrid must use SpringLayout.");
                  return;
            }
```

```
            Spring x = Spring.constant(initialX);
            for (int c = 0; c < cols; c++) {
                  Spring width = Spring.constant(0);
                  for (int r = 0; r < rows; r++) {
                        width = Spring.max(width,
                                    getConstraintsForCell(r,       c,
parent, cols).
                                    getWidth());
                  }
                  for (int r = 0; r < rows; r++) {
                        SpringLayout.Constraints constraints =
                              getConstraintsForCell(r,    c,    parent,
cols);
                        constraints.setX(x);
                        constraints.setWidth(width);
                  }
                  x       =       Spring.sum(x,       Spring.sum(width,
Spring.constant(xPad)));
            }

            Spring y = Spring.constant(initialY);
            for (int r = 0; r < rows; r++) {
                  Spring height = Spring.constant(0);
                  for (int c = 0; c < cols; c++) {
                        height = Spring.max(height,
                                    getConstraintsForCell(r,       c,
parent, cols).
                                    getHeight());
                  }
                  for (int c = 0; c < cols; c++) {
                        SpringLayout.Constraints constraints =
                              getConstraintsForCell(r,    c,    parent,
cols);
                        constraints.setY(y);
                        constraints.setHeight(height);
                  }
                  y       =       Spring.sum(y,       Spring.sum(height,
Spring.constant(yPad)));
            }

            SpringLayout.Constraints              pCons              =
layout.getConstraints(parent);
            pCons.setConstraint(SpringLayout.SOUTH, y);
            pCons.setConstraint(SpringLayout.EAST, x);
      }
}
```

RESULTDIALOG.JAVA

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
```

```
import java.awt.event.WindowListener;
import java.util.ArrayList;
import java.awt.Rectangle;
import java.awt.Dimension;

import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTable;

public class ResultDialog extends JDialog implements
ActionListener,WindowListener{

	private JRadioButton[] radioButtons;
	private ButtonGroup buttongroup;
	private JScrollPane cloudScrollPanel,tableScrollPanel;
	private JPanel panel,cloudPanel,buttonPanel;
	private JSplitPane resultPanel;
	private JTable[] tables;
	private JButton ok,cancel;
	private int count;
	private boolean launch,start;
	private JLabel eucalyptus,eucalyptusLabel;
	private JLabel announcement;
	private static String[] tags = {"Cloud IP","AZ","Instance
Type","#ofInstances","Price","CPU","Ram","Hdd","Network","ConvertedIn
dex","Overall Price"};
	private String dir;

	public ResultDialog(String result){
		dir="/home/kostas/gria/gria-client-cli-5.3/";
		launch=false;
		start=false;
		if(result!=null){


			Image im = (new
ImageIcon(dir+"resources/eucalyptus.png")).getImage();

			eucalyptusLabel = new JLabel("Powered by:");
			eucalyptus = new JLabel(new
ImageIcon(im.getScaledInstance(80, -1, Image.SCALE_SMOOTH)));
			eucalyptus.setSize(60,60);
			eucalyptusLabel.setLabelFor(eucalyptus);


			String[] tmp;
			ArrayList<String[]> configuration=new
ArrayList<String[]>();
			tmp=result.split(";");
			for(int i=0;i<tmp.length&&i<16;i++){
				configuration.add(tmp[i].split("#"));
			}
```

```
                    count = configuration.size();

                    panel = new JPanel(new BorderLayout());
                    resultPanel                    =                    new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
                    cloudPanel          =          new          JPanel(new
GridLayout(configuration.size()+1,1,10,10));
                    buttonPanel          =          new          JPanel(new
GridLayout(1,8,10,10));


                    buttongroup = new ButtonGroup();

                    radioButtons                    =                    new
JRadioButton[configuration.size()];
                    tables = new JTable[configuration.size()];

                    announcement    =    new    JLabel("<html>We    have
chosen<br></br>the following configurations:</html>");
                    cloudPanel.add(announcement);

                    for(int i=0;i<configuration.size()&&i<15;i++){
                          createClouds(i, configuration.get(i));
                    }

                    radioButtons[0].setSelected(true);


                    cloudScrollPanel = new JScrollPane(cloudPanel);

                    tableScrollPanel = new JScrollPane(tables[0]);


                    resultPanel.setLeftComponent(cloudScrollPanel);
                    resultPanel.setRightComponent(tableScrollPanel);

                    ok = new JButton("Ok");

                    ok.addActionListener(this);
                    cancel = new JButton("Close");
                    cancel.addActionListener(this);

                    buttonPanel.add(new JLabel(""));
                    buttonPanel.add(ok);
                    buttonPanel.add(new JLabel(""));
                    buttonPanel.add(cancel);
                    buttonPanel.add(new JLabel(""));
                    buttonPanel.add(new JLabel(""));
                    buttonPanel.add(eucalyptusLabel);
                    buttonPanel.add(eucalyptus);

                    panel.add(resultPanel,BorderLayout.CENTER);
                    panel.add(buttonPanel,BorderLayout.PAGE_END);
                    add(panel);
                    setSize(1000,400);
                    setDefaultCloseOperation(DISPOSE_ON_CLOSE);

                    Image                    img                    =
Toolkit.getDefaultToolkit().getImage(dir+"resources/pyrforos.gif");
                    setIconImage(img);
                    Dimension dim = getToolkit().getScreenSize();
```

```
                    Rectangle abounds = getBounds();
                    setLocation((dim.width - abounds.width) / 2,
                                (dim.height - abounds.height) / 2);

                    setVisible(true);
                    setTitle("Results");
              }

              else{
                    JOptionPane.showMessageDialog(this,
                                "No Cloud could be found to match your
configuration",
                                "Sorry",
                                JOptionPane.WARNING_MESSAGE);
                    dispose();
                    setVisible(false);

              }

        }

        public void createClouds(int index,String[] cloud){

              String[][] details = new String[cloud.length][];
              for(int i =0;i<cloud.length;i++){
                    details[i] = cloud[i].split(" ");
              }


              //create the radio button with the name of the cloud
              radioButtons[index]=      new      JRadioButton("<html>Ip:
"+details[0][0]+"<br></br>AZ:    "+details[0][1]+    "<br></br>Price:
"+details[0][details[0].length-1]+"</html>");
              buttongroup.add(radioButtons[index]);
              cloudPanel.add(radioButtons[index]);
              radioButtons[index].addActionListener(this);

              //create the table with details about the configuration
of the cloud
              tables[index]= new JTable(details,tags);
              tables[index].setEnabled(false);
              tables[index].setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

        }

        public void actionPerformed(ActionEvent arg0) {

              if(arg0.getSource()==ok){
                    launch=true;
                    dispose();
                    setVisible(false);

              }

              else if(arg0.getSource()==cancel){
                    dispose();
                    setVisible(false);

              }
              else{
```

```
                    for(int i=0;i<count;i++){
                          if(arg0.getSource()==radioButtons[i]){

                                resultPanel.remove(tableScrollPanel);
                                tableScrollPanel          =          new
JScrollPane(tables[i]);
                                resultPanel.add(tableScrollPanel);

                          }

                    }

              }

        }



        public void windowActivated(WindowEvent arg0) {


        }

        public void windowClosed(WindowEvent arg0) {
              if(launch&&!start){
                    start=true;
              }

        }


        public void windowClosing(WindowEvent arg0) {
        }

        public void windowDeactivated(WindowEvent arg0) {
        }

        public void windowDeiconified(WindowEvent arg0) {
        }
        public void windowIconified(WindowEvent arg0) {
        }
        public void windowOpened(WindowEvent arg0) {

        }

}
```

Η μόνη αλλαγή που κάναμε ήταν στον υπόλοιπο κώδικα αφορά την κλάση SampleClientPluginSwing του πακέτου sample.client. Ο κώδικας της κλάσης αυτής φαίνεται παρακάτω.

SAMPLECLIENTPLUGINSWING.JAVA

```
package sample.client;

import gui.*;
```

```
import uk.ac.soton.itinnovation.grid.client.proxy.WebProxy;
import uk.ac.soton.itinnovation.grid.types.ConversationID;
import uk.ac.soton.itinnovation.grid.client.events.RegistryEvent;
import
uk.ac.soton.itinnovation.grid.client.swing.AbstractConversationBrowse
rPlugin;
import
uk.ac.soton.itinnovation.grid.client.plugins.GridClientPluginManager;
import uk.ac.soton.itinnovation.grid.client.proxy.ProxyHelpers;
import uk.ac.soton.itinnovation.grid.utils.ResourceTypeRegistry;
import
uk.ac.soton.itinnovation.grid.client.swing.ResourcePropertiesPanel;
import uk.ac.soton.ecs.iam.grid.comms.client.helpers.UserAbort;
import
uk.ac.soton.ecs.iam.grid.comms.client.StorableInStateRepository;
import java.rmi.RemoteException;
import java.util.Map;
import uk.ac.soton.itinnovation.grid.types.ThrowingRunnable;
import javax.swing.JOptionPane;
import sample.client.RemoteSampleService;
import uk.ac.soton.ecs.iam.grid.client.swing.Action;
import javax.swing.JPopupMenu;
import javax.swing.Icon;
import org.apache.axis.message.addressing.EndpointReferenceType;
import sample.SampleResource;
import uk.ac.soton.ecs.iam.grid.client.swing.ConversationBrowser;
import
uk.ac.soton.ecs.iam.grid.comms.client.StorableInStateRepository;
import
uk.ac.soton.itinnovation.grid.client.plugins.GridClientPluginProvider
;

import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.util.ArrayList;

/** Grid client plugin for the sample service.
 * This plugin extends the Grid client to allow it to use the Grid
sample service.
 * It registers the objects relating to the service and also extends
the client's
 * Swing interface to add menu actions for these resources.
 *
 * @see GridClientPluginProvider for details of the client's plugin
system
 */
public class SampleClientPluginSwing extends
AbstractConversationBrowserPlugin {
    public void showPopupMenu(JPopupMenu menu, Class itemClass,
EndpointReferenceType[] items) {
        // The user is trying to open the pop-up menu. Add any
extra menu items

        if
(RemoteSampleService.class.isAssignableFrom(itemClass)) {
            // The select object(s) are sample services...
            menu.add(new Action(browser, items, "Create new
resource") {
                @Override
                public void dispatch(EndpointReferenceType
epr, Object service) throws Exception {
```

```
                                    createNewResource((RemoteSampleService)
service);
                              }
                        });
                  }                                  else                          if
(SampleResource.class.isAssignableFrom(itemClass)) {


                        menu.add(new     Action(browser,     items,     "Run
Instance...") {
                              public   void   dispatch(EndpointReferenceType
epr, Object sample) throws RemoteException {

                                    InputDialog       tsa       =       new
InputDialog(sample);
                              }
                        });


                  }
            }

            /** Create a new sample resource. */
            private    void    createNewResource(final    RemoteSampleService
service) throws Exception {
                  /* Get the user to choose a name for the new resource. */
                  final         String         name         =         (String)
JOptionPane.showInputDialog(browser,
                              "Please enter a name for the new resource:",
                              "Name",
                              JOptionPane.QUESTION_MESSAGE,
                              null,        // icon
                              null,
                  "My resource");
                  if (name == null)
                        throw new UserAbort();

                  /* Invoke the service operation to create the resource
(in a new thread). */
                  browser.runThread(new ThrowingRunnable() {
                        public void run() throws RemoteException {
                              SampleResource            resource            =
service.createSampleResource(name);
                              // Add the new resource to the Swing GUI
                              EndpointReferenceType    epr    =    ((WebProxy)
resource).proxyGetTarget();
                              ConversationID.setParent(epr,      ((WebProxy)
service).proxyGetTarget().getAddress().toString());

            browser.broadcast(RegistryEvent.createRegistryAddedEvent(epr));
                        }
                  }, "Create resource");
            }
}
```

Ο υπόλοιπος κώδικας έχει σχέση με το service αυτό καθαυτό.

Τροποποιήσαμε την κλάση SampleResource του πακέτου sample.common.

SAMPLECOMMON.JAVA

```
package sample;

import uk.ac.soton.itinnovation.grid.types.IdentifiableResource;
import uk.ac.soton.itinnovation.grid.comms.wsdl.WebMethod;
import uk.ac.soton.itinnovation.grid.types.ResourceMetadata;
import uk.ac.soton.itinnovation.grid.comms.wsrf.WSResourceLifetime;
import uk.ac.soton.itinnovation.grid.comms.wsdl.SoapBindingName;
import uk.ac.soton.itinnovation.grid.comms.wsdl.WebService;
import uk.ac.soton.itinnovation.grid.types.PolicyManagement;

import java.rmi.RemoteException;
import java.util.ArrayList;

/** The public interface for the Sample objects.
 * Each method in this interface is a SOAP operation on the
SampleResource service.
 * @see SampleService
 * @see sample.client.SampleConversation
 */
@WebService(targetNamespace = "http://www.it-
innovation.soton.ac.uk/2006/grid/sample",
            name = "SampleResource",
            wsdlLocation = "sample/sampleresource.wsdl",
            serviceName = "SampleResource")
            @SoapBindingName("SampleResourceSoapBinding")
            public interface SampleResource extends PolicyManagement,
WSResourceLifetime, ResourceMetadata, IdentifiableResource {
      /** A unique string to represent a type of resource accessed
through the service. The PBAC policy with
       * this name is used to protect operations on the service which
quote a context with this type.
       * When discovering resources, they have this type in their
metadata.
       *                                                          @see
uk.ac.soton.itinnovation.grid.types.ConversationID#getResourceType
       */
      final String SAMPLE_RESOURCE_TYPE =
"http://SAMPLE/2006/SampleResourceType";

      /* Add additional operations here. You must also add them to
the PBAC resource policy
       * (sample-policy.xml).
       * After changing the policy, you must undeploy the old one
using the web interface.
       * Returning the main admin screen will then automatically
deploy the new version.
       */


      @WebMethod
      String check(String options) throws RemoteException;
}
```

Στην συνέχεια παρουσιάζονται οι κλάσεις του πακέτου ec2. Οι κλάσεις του πακέτου αυτού αφορούν την δημιουργία της Cloud Database και του Cloud Supervisor.

CLOUDITEM.JAVA

```
package ec2;

import java.util.ArrayList;

public class Clouditem {

	private String ip;
	private String pk,cert;
	private ArrayList<Availabilityzoneitem> zones;

	public Clouditem()
	{
		zones=new ArrayList<Availabilityzoneitem>();
	}

	public  Clouditem(String  ip,  ArrayList<Availabilityzoneitem>
zones){
		this.ip=ip;
		this.zones=zones;
	}

	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}

	public ArrayList<Availabilityzoneitem> getZones() {
		return zones;
	}

	public void setZones(ArrayList<Availabilityzoneitem> zones) {
		this.zones = zones;
	}

	public void addZone(Availabilityzoneitem zone){
		zones.add(zone);
	}

	public String getPk() {
		return pk;
	}

	public void setPk(String pk) {
		this.pk = pk;
	}

	public String getCert() {
		return cert;
	}

	public void setCert(String cert) {
		this.cert = cert;
	}



}
```

AVAILABILITYZONESITEM.JAVA

```
package ec2;

import java.util.ArrayList;

public class Availabilityzoneitem {

	private String name;
	private boolean up,previous;
	private int net;
	private int times_up;
	private int times_checked,failures;
	private ArrayList<Vmitem> vmtypes;

	public Availabilityzoneitem() {
		vmtypes = new ArrayList<Vmitem>();
	}

	public Availabilityzoneitem(String name, boolean up){
		this.name=name;
		this.up=up;
		vmtypes = new ArrayList<Vmitem>();
	}

	public Availabilityzoneitem( String name, boolean up, Vmitem
m1small, Vmitem c1medium, Vmitem m1large, Vmitem m1xlarge, Vmitem
c1xlarge){

		this.name=name;
		this.up=up;
		vmtypes = new ArrayList<Vmitem>();
		vmtypes.add(m1small);
		vmtypes.add(c1medium);
		vmtypes.add(m1large);
		vmtypes.add(m1xlarge);
		vmtypes.add(c1xlarge);


	}


	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public boolean isUp() {
		return up;
	}

	public void setUp(boolean up) {
		this.up = up;
	}

	public ArrayList<Vmitem> getVmtypes() {
		return vmtypes;
```

```
        }

        public Vmitem getVmtypesAt(int index) {
                return vmtypes.get(index);
        }

        public void setVmtypes(ArrayList<Vmitem> vmtypes) {
                this.vmtypes = vmtypes;
        }
        public void setVmtypes(int index,Vmitem item){
                vmtypes.set(index,item);
        }

        public void addVmtypes(Vmitem item){
                vmtypes.add(item);
        }

        public int getNet() {
                return net;
        }

        public void setNet(int net) {
                this.net = net;
        }

        public int getTimes_up() {
                return times_up;
        }

        public void setTimes_up(int times_up) {
                this.times_up = times_up;
        }

        public int getTimes_checked() {
                return times_checked;
        }

        public void setTimes_checked(int times_checked) {
                this.times_checked = times_checked;
        }

        public boolean isPrevious() {
                return previous;
        }

        public void setPrevious(boolean previous) {
                this.previous = previous;
        }

        public int getFailures() {
                return failures;
        }

        public void setFailures(int failures) {
                this.failures = failures;
        }

}
```

VMITEM.JAVA

```
package ec2;

public class Vmitem {
	private int free;
	private int type;
	private int max;
	private int cpu;
	private int ram;
	private int disk;
	private double price;

	public Vmitem(){

	}

	public Vmitem(int type,int free,int max,int cpu,int ram, int
disk,double price){
		this.type=type;
		this.free=free;
		this.max=max;
		this.cpu=cpu;
		this.ram=ram;
		this.disk=disk;
		this.price= price;
	}

	public int getFree() {
		return free;
	}

	public void setFree(int free) {
		this.free = free;
	}

	public int getMax() {
		return max;
	}

	public void setMax(int max) {
		this.max = max;
	}

	public int getCpu() {
		return cpu;
	}

	public void setCpu(int cpu) {
		this.cpu = cpu;
	}

	public int getRam() {
		return ram;
	}

	public void setRam(int ram) {
		this.ram = ram;
	}
```

```
	public int getDisk() {
		return disk;
	}

	public void setDisk(int disk) {
		this.disk = disk;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public int getType() {
		return type;
	}

	public void setType(int type) {
		this.type = type;
	}


}
```

CHECKAVAILABILITY.JAVA

```
package ec2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.StringTokenizer;



public class CheckAvailability {


	public CheckAvailability(){
		super();
	}


	static void runInstances(String[] configuration){
		String image = uploadBundle(configuration);

		String precmd="";
		String cbuf;
		String output="";
		String	cmd	=	configuration[0]+"/bin/ec2-describe-
availability-zones verbose";
		if(!configuration[1].isEmpty())
			cmd+=" -U " + configuration[1];
		if(!configuration[2].isEmpty())
			cmd+=" -K "+ configuration[2];
```

```
            if(!configuration[3].isEmpty())
                  cmd+=" -C " + configuration[3];

            Runtime r = Runtime.getRuntime();
            String[] nargs = { "bash" ,"-i","-c", precmd+cmd};
            Process p;
            try {
                  p = r.exec(nargs);
                  p.waitFor();
            } catch (IOException e) {
                  e.printStackTrace();
            }catch (InterruptedException e){
                  e.printStackTrace();
            }

      }



      static String uploadBundle(String[] options){
            String result ="";

            return result;
      }

      /* params: api directory, url of the service, private key,
certificate,
       * returns ArrayList of Availability Zones, NULL if cloud is
unaccessible*/
      static ArrayList<Availabilityzoneitem> availabilityZones(String
apidir,String url, String pk, String crt){
            ArrayList<Availabilityzoneitem>                result=new
ArrayList<Availabilityzoneitem>();


            //String precmd="source ~/.euca/eucarc;";
            String precmd="";
            String cbuf;
            String output="";
            String cmd = apidir+"/bin/ec2-describe-availability-zones
verbose";
            if(!url.isEmpty())
                  cmd+=" -U " + url;
            if(!pk.isEmpty())
                  cmd+=" -K "+ pk;
            if(!crt.isEmpty())
                  cmd+=" -C " + crt;
            try{
                  Runtime r = Runtime.getRuntime();
                  String[] nargs = { "bash" ,"-i","-c", precmd+cmd};
                  Process p = r.exec(nargs);
                  p.waitFor();
                  BufferedReader is =
                        new                        BufferedReader(new
InputStreamReader(p.getInputStream()));
                  while((cbuf=is.readLine())!=null){
                        output+=cbuf+"\n";

                  }
```

```
            } catch (IOException e){
                  e.printStackTrace();
            }catch (InterruptedException e){
                  e.printStackTrace();
            }
            if(output.contains("AVAILABILITYZONE"))
                  result=process(output);
            else
                  result=null;
            return result;
      }

      private static ArrayList<Availabilityzoneitem> process(String
output){
            Availabilityzoneitem item = new Availabilityzoneitem();
            ArrayList<Availabilityzoneitem>     result     =     new
ArrayList<Availabilityzoneitem>(0);
            ArrayList<String> token=new ArrayList<String>(0);
            StringTokenizer st = new StringTokenizer(output);
            while (st.hasMoreTokens()){
                  token.add(st.nextToken());
            }

            item.setName(token.get(1));
            item.setUp("UP".equals(token.get(2)));
            if(item.isUp()&&token.size()>58){
                  for(int i =0;i<5;i++)
                        item.setVmtypes(i,new
Vmitem(i,Integer.parseInt(token.get(17+i*9)),Integer.parseInt(token.g
et(19+i*9)),Integer.parseInt(token.get(20+i*9)),Integer.parseInt(toke
n.get(21+i*9)),Integer.parseInt(token.get(22+i*9)),0));
            }

            else{
                  for(int i =0;i<5;i++)
                        item.setVmtypes(i,new Vmitem(i,0,0,0,0,0,0));

            }

            result.add(item);
            return result;

      }

}
```

EUCALYPTUSDB.JAVA

```
package ec2;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
```

```
public class Eucalyptusdb {

	Connection conn;
	String tools="~/.euca/ec2";

	public Connection getConn() {
		return conn;
	}




	// we dont want this garbage collected until we are done
	public	Eucalyptusdb(String	db_file_name_prefix,String
username,String password) throws Exception {	// note more general
exception
		if(db_file_name_prefix.equals(""))
			db_file_name_prefix="hsql://localhost/xdb";
		// Load the HSQL Database Engine JDBC driver
		// hsqldb.jar should be in the class path or made part of
the current jar
		Class.forName("org.hsqldb.jdbcDriver");

		// connect to the database.	This will load the db files
and start the
		// database if it is not alread running.
		// db_file_name_prefix is used to open or create files
that hold the state
		// of the db.
		// It can contain directory names relative to the
		// current working directory
		conn = DriverManager.getConnection("jdbc:hsqldb:"
				+ db_file_name_prefix,	// filenames
				username,		// username
				password);		// password
	}

	public void shutdown() throws SQLException {

		Statement st = conn.createStatement();

		// db writes out to files and performs clean shuts down
		// otherwise there will be an unclean shutdown
		// when program ends
		st.execute("SHUTDOWN");
		conn.close();	// if there are no other open connection
	}

	//use for SQL command SELECT
	public synchronized ResultSet query(String expression) throws
SQLException {

		Statement st = null;
		ResultSet rs = null;

		st = conn.createStatement();		// statement objects
can be reused with

		// repeated calls to execute but we
```

```
            // choose to make a new one each time
            rs = st.executeQuery(expression);    // run the query

            // do something with the result set.
            st.close();      // NOTE!! if you close a statement the
associated ResultSet is
            return rs;
            // closed too
            // so you should copy the contents to some other object.
            // the result set is invalidated also  if you recycle an
Statement
            // and try to execute some other query before the result
set has been
            // completely examined.
      }

      //use for SQL commands CREATE, DROP, INSERT and UPDATE
      public  synchronized  void  update(String  expression)  throws
SQLException {

            Statement st = null;

            st = conn.createStatement();    // statements

            int i = st.executeUpdate(expression);    // run the query

            if (i == -1) {
                  System.out.println("db error : " + expression);
            }

            st.close();
      }

      //fills the db for first time
      public void initialize() throws SQLException{

            Clouditem cloud = new Clouditem();

      cloud.setIp("http://192.168.1.2:8773/services/Eucalyptus");
            cloud.setPk("~/.euca/euca2-admin-7a5239f7-pk.pem");
            cloud.setCert("~/.euca/euca2-admin-7a5239f7-cert.pem");
            update("INSERT     INTO     cloud(id,ip,pk,cert)     VALUES
("+cloud.getIp().hashCode()+","+"'"+cloud.getIp()+"',
'"+cloud.getPk()+"', '"+cloud.getCert()+"')");

            ArrayList<Availabilityzoneitem>      zones      =      new
ArrayList<Availabilityzoneitem>();

      zones=CheckAvailability.availabilityZones(tools,cloud.getIp(),c
loud.getPk(),cloud.getCert());
            if(zones==null){
            }
            else{

                  for(int j=0;j<zones.size();j++){
                        update(
                              "INSERT                    INTO
availability_zones(id,cloud_id,name,up,failures,times_checked,times_u
p,previous_state)
VALUES("+(cloud.getIp()+zones.get(j).getName()).hashCode()+","+cloud.
getIp().hashCode()+",'"+zones.get(j).getName()+"',"+zones.get(j).isUp
```

```
()+","+(zones.get(j).isUp()?0:1)+",1,"+(zones.get(j).isUp()?1:0)+","+
zones.get(j).isUp()+")");
                        for(int k =0;k<5;k++)
                              update(
                                          "INSERT                INTO
vm_items(zone_id,type,free,maximum,cpu,ram,disk)               VALUES
("+(cloud.getIp()+zones.get(j).getName()).hashCode()+","+k+","+zones.
get(j).getVmtypesAt(k).getFree()+","+zones.get(j).getVmtypesAt(k).get
Max()+","+zones.get(j).getVmtypesAt(k).getCpu()+","+zones.get(j).getV
mtypesAt(k).getRam()+","+zones.get(j).getVmtypesAt(k).getDisk()+")");
                  }

            }



      }

      //creates the database
      public void createTables() throws SQLException{

            update(
            "DROP TABLE IF EXISTS vm_items");
            update(
            "DROP TABLE IF EXISTS availability_zones");
            update(
            "DROP TABLE IF EXISTS cloud");
            update(
                        "CREATE TABLE cloud (" +
                        "id INTEGER PRIMARY KEY," +
                        "ip VARCHAR(256)," +
                        "pk VARCHAR(256)," +
                        "cert VARCHAR(256))"
            );
            update(
                        "CREATE TABLE availability_zones (" +
                        "id INTEGER PRIMARY KEY, " +
                        "cloud_id INTEGER," +
                        "name VARCHAR(256), " +
                        "up BOOLEAN," +
                        "net INTEGER,"+
                        "failures BIGINT,"+
                        "times_checked BIGINT," +
                        "times_up BIGINT," +
                        "previous_state BOOLEAN,"+
            "FOREIGN KEY (cloud_id) REFERENCES cloud (id) ON DELETE
CASCADE )" );
            update(
                        "CREATE TABLE vm_items (" +
                        "id INTEGER IDENTITY, " +
                        "zone_id INTEGER, " +
                        "type TINYINT,"+
                        "free INTEGER, " +
                        "maximum INTEGER, " +
                        "cpu TINYINT, " +
                        "ram SMALLINT, " +
                        "disk SMALLINT," +
                        "price FLOAT," +
            "FOREIGN KEY (zone_id) REFERENCES availability_zones ( id
) ON DELETE CASCADE )");
```

```
        }

        private void createUsers() throws SQLException{

                update("CREATE USER service PASSWORD none ADMIN");

        }

        public void updatedb() throws SQLException{

                ResultSet rs;

                rs=query("SELECT * FROM CLOUD");

                ArrayList<String>cloud_list=dump(rs);

                ArrayList<Availabilityzoneitem>      zones      =      new
ArrayList<Availabilityzoneitem>();
                for(int i = 0;i<cloud_list.size();i++){
                        zones.clear();
                        Clouditem cloud = new Clouditem();
                        String arguments[] = cloud_list.get(i).split(" ");
                        cloud.setIp(arguments[1]);
                        cloud.setPk(arguments[2]);
                        cloud.setCert(arguments[3]);

        zones=CheckAvailability.availabilityZones(tools,cloud.getIp(),c
loud.getPk(),cloud.getCert());

                        if(zones!=null){

                                for(int j=0;j<zones.size();j++){
                                        rs=query("SELECT         *         FROM
availability_zones                                                    WHERE
id="+(cloud.getIp()+zones.get(j).getName()).hashCode());
                                        ArrayList<String> output = dump(rs);


                                        if(output.size()>0){
                                                String[]            tmp           =
output.get(0).split(" ");

                                                int
failures=Integer.parseInt(tmp[6]);
                                                int         times_checked         =
Integer.parseInt(tmp[7]);
                                                int           times_up            =
Integer.parseInt(tmp[8]);
                                                String previous_state=tmp[9];

                                                update(
                                                        "UPDATE
availability_zones                      SET                     name='"+
zones.get(j).getName()+"',up="+zones.get(j).isUp()+",failures="+((!zo
nes.get(j).isUp())&&previous_state.equalsIgnoreCase("TRUE")?failures+
1:failures)+",times_checked="+times_checked+1+",times_up=
"+(zones.get(j).isUp()?times_up+1:times_up)+",previous_state="+zones.
get(j).isUp()+"                         WHERE                     id="+
(cloud.getIp()+zones.get(j).getName()).hashCode());
                                                for(int k =0;k<5;k++)
                                                        update(
```

```
                                                                "UPDATE
vm_items                          SET                          free="+
zones.get(j).getVmtypesAt(k).getFree()+",maximum="+
zones.get(j).getVmtypesAt(k).getMax()+",cpu="+
zones.get(j).getVmtypesAt(k).getCpu()+",ram="+zones.get(j).getVmtypes
At(k).getRam()+",disk="+      zones.get(j).getVmtypesAt(k).getDisk()+"
WHERE  zone_id="+  (cloud.getIp()+zones.get(j).getName()).hashCode()+"
AND type="+k);
                              }

                              else{

                                    update(
                                                      "INSERT          INTO
availability_zones(id,cloud_id,name,up,failures,times_checked,times_u
p,previous_state)
VALUES("+(cloud.getIp()+zones.get(j).getName()).hashCode()+","+cloud.
getIp().hashCode()+",'"+zones.get(j).getName()+"','"+zones.get(j).isUp
()+","+(zones.get(j).isUp()?0:1)+",1,"+(zones.get(j).isUp()?1:0)+","+
zones.get(j).isUp()+")");
                                    for(int k =0;k<5;k++)
                                          update(
                                                                "INSERT    INTO
vm_items(zone_id,type,free,maximum,cpu,ram,disk)              VALUES
("+(cloud.getIp()+zones.get(j).getName()).hashCode()+","+k+","+zones.
get(j).getVmtypesAt(k).getFree()+","+zones.get(j).getVmtypesAt(k).get
Max()+","+zones.get(j).getVmtypesAt(k).getCpu()+","+zones.get(j).getV
mtypesAt(k).getRam()+","+zones.get(j).getVmtypesAt(k).getDisk()+")");


                              }

                        }
                  }
                  //if the cloud is not accessible
                  else{
                        update(
                                    "UPDATE   availability_zones   SET
up=false WHERE cloud_id="+ (cloud.getIp().hashCode()));

                  }
            }
      }

      public void disconnect(){
            try {
                  conn.close();
            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }

      public ArrayList<String> dump(ResultSet rs) throws SQLException
{

            // the order of the rows in a cursor
            // are implementation dependent unless you use the SQL
ORDER statement
            ArrayList<String> result=new ArrayList<String>(0);
            ResultSetMetaData meta   = rs.getMetaData();
```

```
            int                 colmax = meta.getColumnCount();
            int                 i;
            Object              o = null;

            // the result set is a cursor into the data.  You can
only
            // point to one row at a time
            // assume we are pointing to BEFORE the first row
            // rs.next() points to next row and returns true
            // or false if there is no next row, which breaks the
loop
            for (; rs.next(); ) {
                  String tmp="";
                  for (i = 0; i < colmax; ++i) {
                        o = rs.getObject(i + 1);      // Is SQL the
first column is indexed
                        tmp+=o.toString() + " ";
                        // with 1 not 0

                  }
                  result.add(tmp);
            }
            return result;
      }                                          //void dump( ResultSet
rs )

      public static void main(String[] args) {

            Eucalyptusdb db = null;
            String dir = "hsql://localhost/xdb";
            int count,timeout;
            count=timeout=0;

            if(args.length==1){
                  if(args[0].equalsIgnoreCase("shutdown")){
                        try {
                              db = new Eucalyptusdb(dir,"sa","");
                              db.shutdown();
                              return;
                        } catch (Exception ex1) {
                              ex1.printStackTrace();     // could not
start db
                              return;                    // bye bye
                        }

                  }
                  else if (args[0].equalsIgnoreCase("initialize")){
                        try{
                              db = new Eucalyptusdb(dir,"sa","");
                              db.createTables();
                              db.createUsers();
                              db.initialize();
                              db.getConn().close();
                              return;

                        }
                        catch(Exception e){
                              e.printStackTrace();
                        }

                  }
```

```
                    else{
                          String query = args[0];
                          try{
                                db = new Eucalyptusdb(dir,"sa","");
                                ResultSet rs;
                                rs = db.query(query);
                                ArrayList<String> result = db.dump(rs);
                                for(int i =0;i<result.size();i++){

      System.out.println(result.get(i));
                                }
                          }catch(Exception e){
                                e.printStackTrace();

                          }


                    }



            }
            else if(args.length==2){
                    count = Integer.parseInt(args[0]);
                    timeout = Integer.parseInt(args[1]);
            }
            else{
                    System.out.println("No                        proper
argument.{shutdown,initialize,(count,timeout)}");
                    return;

            }


            for(int j=0;j<count;j++){
                    try {
                          db = new Eucalyptusdb(dir,"service","none");
                          db.updatedb();
                          ResultSet rs;
                          rs=db.query("SELECT * FROM CLOUD");
                          ArrayList<String> tsa = db.dump(rs);
                          for(int i =0;i<tsa.size();i++)
                                System.out.println(tsa.get(i));
                          System.out.println();
                          rs=db.query("SELECT           *           FROM
availability_zones");
                          tsa = db.dump(rs);
                          for(int i =0;i<tsa.size();i++)
                                System.out.println(tsa.get(i));
                          System.out.println();
                          rs=db.query("SELECT * FROM vm_items");
                          tsa = db.dump(rs);
                          for(int i =0;i<tsa.size();i++)
                                System.out.println(tsa.get(i));
                          System.out.println();

                    } catch (Exception e) {
                          e.printStackTrace();
                    }
                    try {
```

```
                        Thread.sleep(timeout);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }


                }
                try {
                    db.getConn().close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }


        }
}
```

Οι κλάσεις που αφορούν τον αλγόριθμο είναι οι παρακάτω και περιέχονται στο πακέτο algorithm.

CONFIGURATION.JAVA

```
package algorithm;

import ec2.Availabilityzoneitem;

public class Configuration {
        private Availabilityzoneitem az;
        private String ip;
        private double newprice,price;
        private int az_id;
        private                                                double
reliabilityScore,newreliabilityScore,convertedIndex;

        public double getReliabilityScore() {
                return reliabilityScore;
        }

        public void setReliabilityScore(double reliabilityScore) {
                this.reliabilityScore = reliabilityScore;
        }

        public int getAz_id() {
                return az_id;
        }

        public void setAz_id(int az_id) {
                this.az_id = az_id;
        }

        public Configuration() {
                super();
        }

        public Availabilityzoneitem getAz() {
                return az;
        }
```

```
	public void setAz(Availabilityzoneitem az) {
		this.az = az;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public double getNewprice() {
		return newprice;
	}

	public void setNewprice(double newprice) {
		this.newprice = newprice;
	}

	public double getNewreliabilityScore() {
		return newreliabilityScore;
	}

	public void setNewreliabilityScore(double newreliabilityScore)
{
		this.newreliabilityScore = newreliabilityScore;
	}

	public double getConvertedIndex() {
		return convertedIndex;
	}

	public void setConvertedIndex(double convertedIndex) {
		this.convertedIndex = convertedIndex;
	}

	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}


}
```

MyComparator.java

```
package algorithm;

import ec2.Availabilityzoneitem;

public class Configuration {
	private Availabilityzoneitem az;
	private String ip;
	private double newprice,price;
```

```
	private int az_id;
	private double
reliabilityScore,newreliabilityScore,convertedIndex;

	public double getReliabilityScore() {
		return reliabilityScore;
	}

	public void setReliabilityScore(double reliabilityScore) {
		this.reliabilityScore = reliabilityScore;
	}

	public int getAz_id() {
		return az_id;
	}

	public void setAz_id(int az_id) {
		this.az_id = az_id;
	}

	public Configuration() {
		super();
	}

	public Availabilityzoneitem getAz() {
		return az;
	}

	public void setAz(Availabilityzoneitem az) {
		this.az = az;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public double getNewprice() {
		return newprice;
	}

	public void setNewprice(double newprice) {
		this.newprice = newprice;
	}

	public double getNewreliabilityScore() {
		return newreliabilityScore;
	}

	public void setNewreliabilityScore(double newreliabilityScore)
{
		this.newreliabilityScore = newreliabilityScore;
	}

	public double getConvertedIndex() {
		return convertedIndex;
	}
```

```
	public void setConvertedIndex(double convertedIndex) {
		this.convertedIndex = convertedIndex;
	}

	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}

}
```

SELECTION.JAVA

```
package algorithm;

import java.sql.ResultSet;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collections;

import ec2.Availabilityzoneitem;
import ec2.Eucalyptusdb;
import ec2.Vmitem;

public class Selection {

	public static String makeSelection(CustomerOptions options){

		ArrayList<Configuration>output = new
ArrayList<Configuration>();

		try {
			Eucalyptusdb db=new
Eucalyptusdb("","service","none");
			ResultSet rs;
			rs=db.query("SELECT
zone_id,ip,name,failures,times_checked,times_up,v.type,v.free,v.cpu,v
.ram,v.disk,v.price,net FROM cloud,vm_items v,availability_zones a
WHERE cloud.id=a.cloud_id AND v.zone_id=a.id AND up=true AND
cpu>="+options.getCpu()+" AND ram>="+options.getRam()+ " AND
disk>="+options.getHdd()+" AND net>="+options.getNet() +"order by
zone_id,type ASC");
			output=processArrayList(db.dump(rs));

		} catch (Exception e) {
			e.printStackTrace();
		}

		output=makeConfigurations(output, options);

		if(output.size()>0){
			calculateReliability(output,options);
			calculateConvertedIndex(output,options);
			System.out.println(prepareResult(output));
			return prepareResult(output);
```

```
            }


            return null;

      }


      private  static  String  prepareResult(ArrayList<Configuration>
input){
            String result="";
            DecimalFormat myFormatter = new DecimalFormat("###.###");

            for(int i=0;i<input.size();i++){
                  Configuration conf=input.get(i);
                  for(int
j=0;j<conf.getAz().getVmtypes().size();j++){
                        Vmitem item = conf.getAz().getVmtypesAt(j);
                        result+=conf.getIp()+"
"+conf.getAz().getName()+"          "+translateZones(item.getType())+"
"+item.getFree()+"             "+myFormatter.format(item.getPrice())+"
"+item.getCpu()+"         "+item.getRam()+"         "+item.getDisk()+"
"+conf.getAz().getNet()+"
"+myFormatter.format(conf.getConvertedIndex())+       "       "       +
myFormatter.format(conf.getPrice());
                        if(j<conf.getAz().getVmtypes().size()-1)
                              result+="#";
                  }
                  if(i<input.size()-1)
                        result+=";";
            }


            return result;
      }

      private              static              ArrayList<Configuration>
processArrayList(ArrayList<String> input){

            ArrayList<Configuration>        result        =        new
ArrayList<Configuration>();
            for(String i: input){

                  String[] tmp = i.split(" ");
                  Configuration conf = new Configuration();
                  conf.setAz_id(Integer.parseInt(tmp[0]));
                  conf.setIp(tmp[1]);
                  Availabilityzoneitem        item        =        new
Availabilityzoneitem();
                  item.setName(tmp[2]);
                  item.setFailures(Integer.parseInt(tmp[3]));
                  item.setTimes_checked(Integer.parseInt(tmp[4]));
                  item.setTimes_up(Integer.parseInt(tmp[5]));
                  item.setNet(Integer.parseInt(tmp[12]));
                  item.addVmtypes(new
Vmitem(Integer.parseInt(tmp[6]),Integer.parseInt(tmp[7]),Integer.pars
eInt(tmp[7]),Integer.parseInt(tmp[8]),Integer.parseInt(tmp[9]),Intege
r.parseInt(tmp[10]),Double.parseDouble(tmp[11])));
                  conf.setAz(item);
                  result.add(conf);
            }
```

```
            return result;

      }

      private static String translateZones(int input){
            String result;
            switch(input){
            case 0:result="m1.small";break;
            case 1:result="c1.medium";break;
            case 2:result="m1.large";break;
            case 3:result="m1.xlarge";break;
            case 4:result="c1.xlarge";break;
            default: result="no default zone";break;

            }
            return result;



      }

      private              static              ArrayList<Configuration>
makeConfigurations(ArrayList<Configuration>     input,CustomerOptions
options){
            ArrayList<Configuration>        result        =        new
ArrayList<Configuration>();

            float tmp_price;
            int tmp,number;
            boolean found,first;
            int zone;
            for(int i=0;i<input.size();){

                  zone=input.get(i).getAz_id();
                  tmp_price=0;
                  tmp=0;
                  found=false;
                  first =true;
                  Configuration conf = new Configuration();

      while(i<input.size()&&zone==input.get(i).getAz_id()){
                        if(!found){

                              if(options.getInstances()-
tmp<=input.get(i).getAz().getVmtypesAt(0).getFree()){
                                    found= true;

                                    number  =  options.getInstances()-
tmp;

      input.get(i).getAz().getVmtypesAt(0).setFree(number);

      tmp_price+=number*input.get(i).getAz().getVmtypesAt(0).getPrice
();

      if(tmp_price<=options.getPrice()){


      conf.setPrice(tmp_price*options.getTime());
```

```
      conf.setIp(input.get(i).getIp());
                                        if(first){

      conf.setAz(input.get(i).getAz());
                                        }
                                        else{

      conf.getAz().addVmtypes(input.get(i).getAz().getVmtypesAt(0));
                                        }
                                        conf.setAz_id(zone);
                                        result.add(conf);
                                  }

                              }
                              else{

      number=input.get(i).getAz().getVmtypesAt(0).getFree();
                                  tmp+=number;

      tmp_price+=number*input.get(i).getAz().getVmtypesAt(0).getPrice
();
                                  if(first){

      conf.setAz(input.get(i).getAz());
                                        first = false;
                                  }
                                  else{

      conf.getAz().addVmtypes(input.get(i).getAz().getVmtypesAt(0));
                                  }
                              }
                        }
                        i++;
                  }
            }



            return result;
      }

      private                         static                        void
calculateReliability(ArrayList<Configuration>   input,CustomerOptions
options){
            for(Configuration i:input){

                  double                  availability                =
(double)i.getAz().getTimes_up()/i.getAz().getTimes_checked();

                  int availability_score;
                  if(availability<0.90)
                        availability_score=1;
                  else if(availability<0.95)
                        availability_score=2;
                  else if(availability<0.99)
                        availability_score=4;
                  else if(availability<0.999)
                        availability_score=8;
                  else if(availability<0.9999)
                        availability_score=12;
```

```
                    else
                          availability_score=16;


                    double   continuous_availability=(double)Math.exp(-
options.getTime()*i.getAz().getFailures()/i.getAz().getTimes_checked(
));
                    double mttr =(double)(Math.exp(1-availability));
                    double                    score                    =
(double)availability_score*continuous_availability/mttr;
                    i.setReliabilityScore(score);

      System.out.println("Continuous:"+continuous_availability+"
mttr:"+mttr+" "+i.getIp()+" "+i.getAz().getName()+":"+score);

            }

      }

      private                        static                        void
calculateConvertedIndex(ArrayList<Configuration>
input,CustomerOptions options){
            double mincost=input.get(0).getPrice();
            double maxcost=input.get(0).getPrice();
            double maxreliability=input.get(0).getReliabilityScore();
            double minreliability=input.get(0).getReliabilityScore();

            for(int i=1;i<input.size();i++){
                  if(input.get(i).getPrice()<mincost)
                        mincost=input.get(i).getPrice();
                  if(input.get(i).getPrice()>maxcost)
                        maxcost=input.get(i).getPrice();

      if(input.get(i).getReliabilityScore()<minreliability)

      minreliability=input.get(i).getReliabilityScore();

      if(input.get(i).getReliabilityScore()>maxreliability)

      maxreliability=input.get(i).getReliabilityScore();
            }

            for(Configuration i: input){
                  double
fcost=(double)Math.exp(options.getPrice_slope()*(i.getPrice()-
mincost)/((maxcost-mincost)*100));
                  double
freliability=(double)Math.exp(options.getReliability_slope()*(i.getRe
liabilityScore()-minreliability)/(100*(maxreliability-
minreliability)));
                  double newprice=(double)fcost*i.getPrice();
                  double
newreliabilityscore=(double)freliability*i.getReliabilityScore();

      i.setConvertedIndex((double)newprice/newreliabilityscore);




            }
            Collections.sort(input,new MyComparator());
```

```
            for(Configuration i: input)
                  System.out.println(i.getIp()+"
"+i.getConvertedIndex());
      }


}
```

Η κλάση τέλος SampleResourceImpl του πακέτου sample.impl είναι η ακόλουθη.

SAMPLERESOURCEIMPL.JAVA

```
package sample.impl;

//Eucalyptus imports
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import ec2.*;
import algorithm.*;

//database imports
import java.sql.ResultSet;
import java.util.ArrayList;




import sample.SampleResource;        //  SOAP  operations  on  sample
resources
import sample.SampleService;         // SOAP operations on the service
import sample.SampleResourceBean;    // Bean holding resource state

//  Client  imports  -  we  act  as  a  client  to  the  account  and  SLA
services
import
uk.ac.soton.ecs.iam.grid.comms.client.StorableInStateRepository;
//import
uk.ac.soton.ecs.iam.grid.comms.client.TradeAccountConversation;
import uk.ac.soton.ecs.iam.grid.comms.client.SLAConversation;
import uk.ac.soton.itinnovation.grid.service.types.Metric;

// Standard Java imports
import java.net.URL;
//import java.math.BigDecimal;
import org.apache.log4j.Logger;
import org.w3c.dom.Document;
import java.rmi.RemoteException;

// Standard Grid service imports
import org.apache.axis.message.addressing.EndpointReferenceType;
import
uk.ac.soton.itinnovation.grid.service.utils.gridservit.GridServiceLit
e;
import uk.ac.soton.itinnovation.grid.utils.AtomUtils;
import uk.ac.soton.itinnovation.grid.types.ConversationID;
```

```
import uk.ac.soton.itinnovation.grid.types.GridErrorException;
import uk.ac.soton.itinnovation.grid.service.types.GridResource;
import uk.ac.soton.ecs.iam.grid.utils.ImplementationFactory;

// Hibernate persistence
import org.hibernate.Session;
import org.hibernate.Transaction;

/** The implementation of SampleService. The mapping from the
interface to this class is in the
 * implementationfactory.properties file.
 */
public class SampleResourceImpl extends GridServiceLite implements
SampleResource {
	private static Logger log =
Logger.getLogger(SampleResourceImpl.class);
	private SampleServiceImpl service = (SampleServiceImpl)
ImplementationFactory.getSingletonInstance(SampleService.class);

	public SampleResourceImpl() {
		super("SampleResource");
	}

	/** Get a SampleResourceBean that was stored using hibernate.
	 * Create a hibernate session automatically. This is not
suitable
	 * for beans containing lazy collections, which require the
session
	 * to still be open when the collection is accessed.
	 * @throws GridErrorException if the resource does not exist
	 */
	private SampleResourceBean getResource(String resourceID) {
		Session session = factory.openSession();
		try {
			return getResource(session, resourceID);
		} finally {
			session.close();
		}
	}

	@Override
	protected Class<? extends GridResource> getResourceType(String
resourceID) {
		return SampleResourceBean.class;
	}

	/** Get a SampleResourceBean that was stored using hibernate.
	 * Use this version if you already have a session (to prevent
deadlock) or if
	 * the session must remain open when accessing the bean
(because it contains
	 * lazy collections that are only fetched from the database
when accessed).
	 * @param session hibernate session to use
	 * @throws GridErrorException if the resource does not exist
	 */
	private SampleResourceBean getResource(Session session, String
resourceID) {
		SampleResourceBean resource = (SampleResourceBean)
session.get(SampleResourceBean.class, resourceID);
		if (resource == null)
```

```
                    throw new GridErrorException("Can't find resource
'" + resourceID + "'!");
            return resource;
      }

      public void destroy() throws RemoteException {
            String conversationID = getConversationFromContext();

            // Look up the resource
            SampleResourceBean resource;
            try {
                  resource = getResource(conversationID);
            } catch (GridErrorException ex) {
                  log.error("Failed to get resource! Removing from
PBAC anyway.", ex);
                  pdp.signal(conversationID, "destroy");
                  throw ex;
            }

            // Notify the SLA manager that usage has finished
            StorableInStateRepository     managingConversation     =
getManagingConversation(resource);
            if (managingConversation instanceof SLAConversation) {
                  // Our use of resources within the SLA drops to
zero
                  EndpointReferenceType            epr            =
ConversationID.getEPR(thisServiceAddress,   resource.getResourceID(),
null);

                  service.pullPoint.addInstantaneousUsageReport(epr,
SampleServiceImpl.METRIC_SAMPLE_RESOURCES, 0.0);
                  service.pullPoint.addInstantaneousUsageReport(epr,
Metric.CURRENT_ACTIVITIES, 0.0);
            }

            /* Do any clean-up required here. If an exception is
thrown here
             * the resource will still exist and be accessible to the
user.
             */

            // Signal to PBAC that the resource is finished
            pdp.signal(conversationID, "destroy");

            // Remove the resource from hibernate
            deleteHibernatedObject(resource);
      }

      /** Used by SampleServiceImpl to when returning EPRs. */
      URL getSoapEndpoint() {
            return thisServiceAddress;
      }

      /** Return notices from the service as an Atom feed.
       * These notices are displayed on the main admin page and can
also be polled
       * by the administrator using a news aggregator. The feed
should be used to alert the
       * admin to configuration problems and any other important
issues that come up.
       */
```

```
	public	Document	getAtomFeed(String	atomFeed,	String
serviceBase) {
		Document	doc	=	AtomUtils.getEmptyAtomFeed("Sample
Resource", "Sample Resource",
				atomFeed, serviceBase);
		return doc;
	}


	public String check(String input) throws RemoteException {

		String[] tmp = input.split(" ");
		CustomerOptions options = new CustomerOptions();
		options.setInstances(Integer.parseInt(tmp[0]));
		options.setCpu(Integer.parseInt(tmp[1]));
		options.setRam(Integer.parseInt(tmp[2]));
		options.setHdd(Integer.parseInt(tmp[3]));
		options.setPrice(Float.parseFloat(tmp[4]));
		options.setNet(Integer.parseInt(tmp[5]));
		options.setPrice_slope(Integer.parseInt(tmp[6]));
		options.setReliability_slope(Integer.parseInt(tmp[7]));
		options.setTime(Integer.parseInt(tmp[8]));

		String result = Selection.makeSelection(options);


		String s ="192.168.1.2 twin0 m1.xlarge 15 0.15 2 12 50
1000 99.9 0.16;a192.168.1.2a twin0a m1.xlargea 15a 0.15a 2a 12a 50a
1000a 99.9%a 0.17a";
		//return s;
		return result;
	}

	public ArrayList<String> select()throws RemoteException{

		ArrayList<String> result = new ArrayList<String>();
		result.add("instances");
		result.add("cpu");
		result.add("ram");
		result.add("disk");
		return result;

	}

}
```