ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Προσομοίωση Υπολογιστικών Νεφών** **Simulation of Cloud Computing Infrastructures**
Ονοματεπώνυμο Φοιτητή	**Μέγκη Ντίμο**
Πατρώνυμο	**Ηρακλή**
Αριθμός Μητρώου	**ΜΠΣΠ19037**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**

Οκτώβριος 2022

Τριμελής Εξεταστική Επιτροπή

Χρήστος Δουληγέρης Καθηγητής	Δημήτριος Βέργαδος Καθηγητής	Παναγιώτης Κοτζανικολάου Αναπληρωτής Καθηγητής

Περίληψη

Η διεξαγωγή πειραμάτων για τη διαχείριση μιας υποδομής νέφους σε ένα πραγματικό περιβάλλον μπορεί να μην είναι βιώσιμη και πρακτική λόγω κόστους και χρονικών περιορισμών και μπορεί να μην παρέχει όλη την ευελιξία που απαιτείται για μια δοκιμή. Αντίθετα, ένας προσομοιωτής νέφους μπορεί να αποτελέσει μια ικανοποιητική και βιώσιμη επιλογή, καθώς επιτρέπει στους ερευνητές και στους παρόχους νέφους να εξετάζουν και να προβλέπουν από κοινού πώς μπορεί να επηρεαστεί η λειτουργία ενός τέτοιου συστήματος από την τροποποίηση των επιμέρους στοιχείων του. Εντούτοις, το παρόν έγγραφο επικεντρώνεται στην ανάλυση πλαισίων προσομοίωσης που αποσκοπούν στην ευκολότερη και ταχύτερη διεξαγωγή πειραματικών μελετών με βάση το νέφος. Μέχρι σήμερα έχουν προταθεί αρκετοί προσομοιωτές νέφους με διάφορα χαρακτηριστικά, που είναι διαθέσιμοι προς χρήση και μας επιτρέπουν να δοκιμάζουμε τις υπηρεσίες μας σε ένα ελεγχόμενο και επαναλαμβανόμενο περιβάλλον χωρίς κόστος. Ωστόσο, εξακολουθεί να είναι δύσκολη η επιλογή του καταλληλότερου εργαλείου για την αξιολόγηση μιας προτεινόμενης έρευνας. Στα πλαίσια της παρούσας εργασίας εγκαταστάθηκαν και αναλύθηκαν τρία εργαλεία για περιβάλλοντα νέφους: το Cloudsim, μια de facto βασική πλατφόρμα προσομοίωσης, το GreenCloud για προσομοίωση με ενεργειακό προσανατολισμό, και το CloudAnalyst για προσομοίωση εφαρμογών διαδικτύου μεγάλης κλίμακας. Οι τρεις προσομοιωτές αναλύονται ως προς την αρχιτεκτονική, τα στοιχεία μοντελοποίησης και τη διαδικασία προσομοίωσης.

Λέξεις Κλειδιά

Υπολογιστικό Νέφος, Προσομοιωτές Υπολογιστικού Νέφους, Cloudsim, GreenCloud, Cloud Analyst

Abstract

Performing experiments to manage a cloud infrastructure in a real environment may not be practical due to cost and time limits and it might not provide all the flexibility needed for a test. Therefore, a cloud simulator could be a more viable choice, since it allows both researchers and cloud providers alike to examine and predict the influence of modifying individual components on the operation of such a system. This paper focuses on the analysis of simulation frameworks designed to make cloud-based experimental studies easier and faster. To this day, several simulators with different attributes have been proposed and are available for use, allowing us to try out our services in a controlled and repeatable environment at no cost. However, it is still difficult to choose the most appropriate tool to evaluate the proposed research. In this paper, we have installed and analyzed three tools for cloud environments: Cloudsim, which is a de facto core simulation platform, GreenCloud for energy-oriented simulation, and CloudAnalyst for large-scale web application simulation. The three simulators are analyzed in terms of architecture, modeling elements, and simulation process.

Keywords

Cloud Computing, Cloud Simulators, Cloudsim, GreenCloud, CloudAnalyst

Ευχαριστίες

Πρώτα απ΄ όλα, θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Χρήστο Δουληγέρη, Καθηγητή Πανεπιστημίου Πειραιώς για την επίβλεψη της διπλωματικής μου εργασίας. Παράλληλα θέλω να ευχαριστήσω θερμά τους συμφοιτητές μου, Μάρθα, Σόνια, Άγγελο και Γρηγόρη για την ομαδική συνεργασία καθ΄ όλη τη διάρκεια του μεταπτυχιακού. Επίσης, θα ήθελα να ευχαριστήσω την αδερφή μου Άλντα, την φίλη μου Ιωάννα και τον Μάρκο για την ευγενική υποστήριξη και την ενθάρρυνσή τους. Τέλος, θα ήθελα να ευχαριστήσω την μητέρα μου για την ατελείωτη αγάπη της, την άνευ όρων εμπιστοσύνη και τη συνεχή υποστήριξη της.

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Συντομογραφίες

BWs	Balance Workloads
CIS	Cloud Information Service
$CIWs$	Intensive Workloads
CPU	Central Processing Unit
DCN	Data Center Network
$DIWs$	Data Intensive Workloads
$DVFS$	Dynamic Voltage and Frequency Scaling
$ESCE$	Equally Spread Current Execution
$FCFS$	First-Come-First-Serve
GUI	Graphical User Interface
GE	Gigabit Ethernet
JDK	Java Development Kit
$MIPS$	Million Instructions per Second
PUE	Power Usage Effectiveness
RAM	Random Access Memory
RR	Round Robin
SJF	Shortest Job First
VM	Virtual Machine

Κεφάλαιο 1

Εισαγωγή

1.1 Δήλωση προβλήματος και στόχος της μελέτης

Το υπολογιστικό νέφος εφαρμόζεται σε πολλούς τομείς της πληροφορικής, καθώς μειώνει σημαντικά το κόστος και μεγιστοποιεί τα οφέλη των σύγχρονων επιχειρήσεων και οργανισμών. Έχει φθάσει στο στάδιο όπου πολλές επιχειρήσεις εξετάζουν το ενδεχόμενο να υιοθετήσουν αυτή την τεχνολογία ή τη χρησιμοποιούν ήδη εκτενώς. Αν και ο τομέας του υπολογιστικού νέφους προσφέρει πολλές ευκαιρίες, παρουσιάζει επίσης αρκετές προκλήσεις. Τα μη επαναλαμβανόμενα πειράματα αποτελούν μια γενική πρόκληση στην έρευνα για την εφαρμογή του υπολογιστικού νέφους, καθώς υπάρχουν τεράστιες δυσκολίες στη χρήση πραγματικών υποδομών χωρίς τη χρήση μιας προσομοιωτικής εφαρμογής όπως το CloudSim. Η χρήση του προσομοιωτή παρουσιάζει πολλά πλεονεκτήματα και καθιστά δυνατή την εύκολη επανάληψη και τον έλεγχο των πειραμάτων, καθώς τα πειράματα προσομοιώνονται και το φυσικό σύστημα δεν χρειάζεται να αναδιαμορφωθεί και να εγκατασταθεί εκ νέου. Πολλοί προσομοιωτές είναι διαθέσιμοι και είναι εξειδικευμένοι για διαφορετικούς φόρτους εργασίας και τύπους πειραμάτων, επομένως, είμαστε σε θέση να επιλέξουμε ανάμεσα σε ένα ευρύ φάσμα προσομοιωτών για χρήση τόσο σε εμπορικά όσο και σε ερευνητικά έργα. Το παρόν έγγραφο επικεντρώνεται στην ανάλυση τριών γνωστών προσομοιωτών για διάφορους κοινούς τύπους πειραμάτων. Στόχος είναι να καταδειχθεί ο τρόπος λειτουργίας τους και να πραγματοποιηθούν προσομοιώσεις για κάθε προσομοιωτή ξεχωριστά σύμφωνα με ορισμένες μελέτες, προκειμένου να επιβεβαιωθούν

τα αποτελέσματα που προκύπτουν.

1.2 Προσομοίωση υπολογιστικού νέφους

Οι προσομοιώσεις νέφους χρησιμοποιούνται για την αξιολόγηση αρχιτεκτονικών, αλγορίθμων, τοπολογιών και στρατηγικών που βρίσκονται υπό έρευνα και ανάπτυξη, αντιμετωπίζοντας πολλά ζητήματα όπως η διαχείριση πόρων, ο προγραμματισμός εφαρμογών, η εξισορρόπηση φορτίου, η εκτέλεση φόρτου εργασίας και η βελτιστοποίηση της κατανάλωσης ενέργειας [1]. Μεχρι σήμερα, έχουν αναπτυχθεί πολλοί προσομοιωτές νέφους οι οποίοι αξιοποιούνται ενεργά για τη διεξαγωγή ερευνών. Αυτοί οι προσομοιωτές διαφέρουν ως προς τα χαρακτηριστικά τους, όπως η διαθεσιμότητα του γραφικού περιβάλλοντος χρήστη, η βασική γλώσσα προγραμματισμού, η αδειοδότηση, η επεκτασιμότητα και ούτω καθεξής. Τα οφέλη που προσφέρουν οι προσομοιωτές σε σχέση με τη δημιουργία ενός φυσικού νέφους είναι τα εξής:

- Το ελάχιστο κόστος από την αγορά υλικού και ιδιόκτητου λογισμικού. Επίσης, πολλοί προσομοιωτές διατίθενται δωρεάν. Πολλοί προσομοιωτές διατίθενται επίσης δωρεάν. Επιπλέον, οι προσομοιωτές νέφους δεν απαιτούν κόστος συντήρησης.
- Επαναλαμβανόμενο και ελεγχόμενο: Μπορούμε να δοκιμάσουμε την πειραματική διαδικασία (προσομοίωση) όσο απαιτείται μέχρι να έχουμε την επιθυμητή έξοδο.
- Περιβάλλον: ένας προσομοιωτής παρέχει ένα περιβάλλον για την αξιολόγηση διαφορετικών σεναρίων υπό διαφορετικούς φόρτους εργασίας.

1.3 Διάρθρωση της μελέτης

Το πρώτο κεφάλαιο έχει ως στόχο να εξοικειώσει τον αναγνώστη με τις βασικές έννοιες που σχετίζονται με το θέμα μας. Αρχικά, στην ενότητα 2.1 αναλύεται η έννοια του υπολογιστικού νέφους περιγράφοντας τα βασικά χαρακτηριστικά του καθώς και τις τεχνολογίες που το καθιστούν εφικτό, ιδίως την τεχνική της εικονικοποίησης. Στη συνέχεια παρουσιάζουμε τα διάφορα μοντέλα υπηρεσιών που παρέχονται, καθώς και τη χρησιμότητά τους ανά

κατηγορία χρηστών. Στην ενότητα 2.2, συζητάμε τη σημασία των προσομοιωτών στο υπολογιστικό νέφος και περιγράφουμε εν συντομία ορισμένους προσομοιωτές νέφους, όπως οι CloudSim, CloudAnalyst, EMUSIM, IcanCloud, GreenCloud, NetworkCloudSim, GroudSim, SimIC, MDCSim και DsSim. Στα επόμενα τρία κεφάλαια, περιγράφονται λεπτομερώς οι τρεις προσομοιωτές όσον αφορά την αρχιτεκτονική, τα στοιχεία μοντελοποίησης και τη διαδικασία προσομοίωσης και παρουσιάζονται τα πειραματικά αποτελέσματα των προσομοιώσεων, η ανάλυσή τους και τα συμπεράσματα. Οι προσομοιώσεις πραγματοποιήθηκαν με την εκτέλεση διαφορετικών σεναρίων για τη δοκιμή των δυνατοτήτων του κάθε εργαλείου και παρέχονται οδηγίες χρήσης για κάθε προσομοιωτή για την κατανόηση της εγκατάστασης και της χρήσης του. Εν κατακλείδι, επισημαίνονται τα συμπεράσματα της συνολικής μας μελέτης.

Κεφάλαιο 2

Τεχνικό Υπόβαθρο

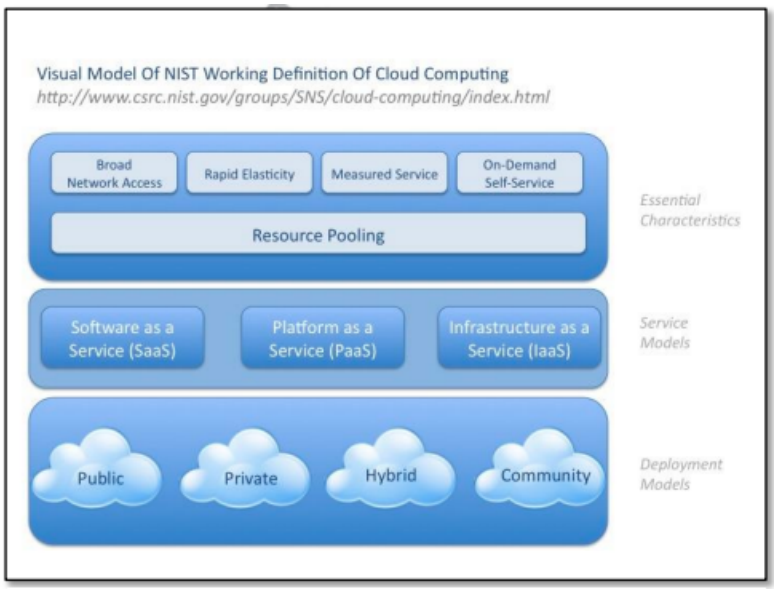
2.1 Υπολογιστικό Νέφος

Το υπολογιστικό νέφος αποτελεί μια εξαιρετικά δημοφιλή προσέγγιση τα τελευταία χρόνια στον τομέα των κατανεμημένων συστημάτων με ορισμένα ιδιαίτερα χαρακτηριστικά που το τοποθετούν στο επίκεντρο των σημερινών εξελίξεων όσον αφορά την παροχή υπολογιστικών πόρων μέσω του διαδικτύου [2]. Με την εξέλιξη της εικονικοποίησης, την πρόσβαση στο διαδίκτυο με υψηλές ταχύτητες και, κυρίως, την υποστήριξη κορυφαίων εταιρειών πληροφορικής, το μακροπρόθεσμο όραμα της "πληροφορικής ως υπηρεσία" (utility computing) έχει επιτευχθεί και το υπολογιστικό νέφος έχει γίνει ένας από τους ταχύτερα αναπτυσσόμενους τομείς της πληροφορικής [3]. Αυτό οφείλεται στην αποδοτικότητα και την ευελιξία που προσφέρει, επιτρέποντας στους χρήστες να έχουν γρήγορη πρόσβαση σε πόρους από οπουδήποτε και ανά πάσα στιγμή, με βάση την πληρωμή ανά χρήση. Στο μοντέλο υπηρεσιών νέφους, όλες οι υπηρεσίες είναι διαθέσιμες μέσω του διαδικτύου. Οι υπηρεσίες νέφους παρέχονται με τέτοιο τρόπο ώστε ο τελικός χρήστης να μην μπορεί να διακρίνει τεχνικές λεπτομέρειες. Είναι σημαντικό να κατανοήσουμε τα κύρια χαρακτηριστικά των υπηρεσιών νέφους, τα πλεονεκτήματα και τους περιορισμούς τους, ώστε οι χρήστες να μπορούν να λάβουν τις σωστές αποφάσεις και να επωφεληθούν πλήρως από αυτή την τεχνολογία. Αυτό αποτελεί το επίκεντρο της παρούσας ενότητας, η οποία εξετάζει τις κύριες πτυχές του υπολογιστικού νέφους.

2.1.1 Ορισμός και βασικά χαρακτηριστικά

Ο ορισμός του NIST (National Institute of Standards and Technology)[4], που παρουσιάζεται στο Σχήμα 2.1, υπογραμμίζει τα βασικά χαρακτηριστικά που διακρίνουν το cloud computing από άλλες παραδοσιακές υπηρεσίες πληροφορικής και καλύπτει σημαντικές έννοιες που επιτρέπουν την κατανόηση της ορολογίας του, συμπεριλαμβανομένων των τύπων υπηρεσιών cloud και των μοντέλων ανάπτυξης.

«Το Υπολογιστικό Νέφος (Cloud Computing) είναι ένα μοντέλο που επιτρέπει ευέλικτη, κατόπιν ζήτησης δικτυακή πρόσβαση σε ένα κοινόχρηστο σύνολο διαμορφώσιμων πόρων (π.χ. δίκτυα, κεντρικοί υπολογιστές, αποθηκευτικοί χώροι, εφαρμογές και υπηρεσίες), το οποίο μπορεί να τροφοδοτηθεί γρήγορα και να διατεθεί με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδρασης με τον πάροχο της υπηρεσίας. Το μοντέλο του Υπολογιστικού Νέφους προωθεί την διαθεσιμότητα και αποτελείται από πέντε βασικά χαρακτηριστικά, τρία μοντέλα παροχής υπηρεσιών, και τέσσερα μοντέλα ανάπτυξης.»



Σχήμα 2.1: NIST: Ο ορισμός του υπολογιστικού νέφους [4]

Τα πέντε χαρακτηριστικών όπως αυτά έχουν οριστεί από το NIST είναι:

i) *η άμεση πρόσβαση σε υπολογιστικούς πόρους, όταν ζητηθεί (on-demand self service)*, ένα ιδιαίτερα επιθυμητό χαρακτηριστικό από ένα

μεγάλο αριθμό επιχειρήσεων, καθώς εξαλείφει την ανάγκη για πρόβλεψη των μελλοντικών αναγκών, την αγορά εξοπλισμού για την ικανοποίηση των αναγκών αυτών και βεβαίως την συντήρηση του [5]. Αυτό δίνει τη δυνατότητα στον πελάτη να αποφύγει μια περιττή αρχική επένδυση σε εξοπλισμό (π.χ. servers), από τη μία πλευρά, και βοηθά στην αποφυγή εξόδων από ανεπαρκώς χρησιμοποιούμενους πόρους, από την άλλη.
ii)*η ευρεία πρόσβαση στο δίκτυο (broad network access)*, είναι επίσης ένα θεμελιώδης χαρακτηριστικό του υπολογιστικού νέφους. Οι δυνατότητες που προσφέρονται από το νέφος μπορεί να είναι διαθέσιμες μέσω του διαδικτύου και προσβάσιμες μέσω τυποποιημένων μηχανισμών οι οποίοι επιτρέπουν τη χρήση των υπηρεσιών του από ετερογενείς συσκευές και πλατφόρμες [5].
iii) *η διαθεσιμότητα των πόρων (resource pooling)*, η δυνατότητα να χρησιμοποιείται το ίδιο σύνολο υπολογιστικών πόρων από έναν μεγάλο αριθμό πελατών την ίδια χρονική στιγμή [5]. Το σκεπτικό είναι ότι ο πάροχος παρέχει στους πελάτες-χρήστες, οι οποίοι είναι πρόθυμοι να πληρώσουν για την υπηρεσία αυτή, πρόσβαση σε μια μεγάλη κοινή δεξαμενή υπολογιστικών πόρων, εξασφαλίζοντας παράλληλα ότι η απόδοση ανταποκρίνεται στις απαιτήσεις τους.
iv)*η ευελιξία της επεκτασιμότητας (rapid elasticity)*,όσο αφορά την ταχύτητα δέσμευσης / αποδέσμευσης πόρων με αποτέλεσμα την δυναμική προσαρμογή των πληροφοριακών συστημάτων στις εκάστοτε ανάγκες. Στο μυαλό του πελάτη καταναλωτή οι δυνατότητες του σύννεφου είναι απεριόριστες, ενώ οι πόροι που διαθέτει τείνουν στο άπειρο. Επιπλέον είναι πάντοτε διαθέσιμες και ο πελάτης μπορεί να προμηθευτεί κάθε φορά όσες ακριβώς χρειάζεται. Η ταχύτητα που κάθε φορά ο πάροχος νεφοϋπολογιστικής προσαρμόζεται στις απαιτήσεις του πελάτη εξαρτάται από τις συμβάσεις (SLA's – Service Level Agreements) που έχει συνάψει μαζί του.
Και τέλος η v) *τιμολόγηση βάσει χρήσης (a measured service)*, μια ακόμα καινοτομία που φέρνει το υπολογιστικό νέφος είναι η εισαγωγή ενός διαφορετικού μοντέλου τιμολόγησης. Ο πελάτης δηλαδή πληρώνει ανάλογα με την χρήση των πόρων που ζητά να χρησιμοποιήσει. Το ακριβές μοντέλο τιμολόγησης μπορεί να διαφέρει από υπηρεσία σε υπηρεσία ή και από πάροχο σε πάροχο [5]. Η χρήση των υπολογιστικών πόρων καταμετράται, ελέγχεται και αποτυπώνεται σε αναφορές (reports) παρέχοντας σαφή στοιχεία στα συναλλασσόμενα μέρη (πάροχος – πελάτης). Έτσι ο πάροχος μπορεί να τιμολογεί με απόλυτη διαφάνεια και ακρίβεια τον πελάτη του για τις υπηρεσίες που χρησιμοποίησε και ο πελάτης μπορεί να ελέγχει τι καλείται να πληρώσει ακριβώς, για ποιες υπηρεσίες και πότε τις χρησιμοποίησε.

2.1.2 Εικονικοποίηση και υπολογιστικό νέφος

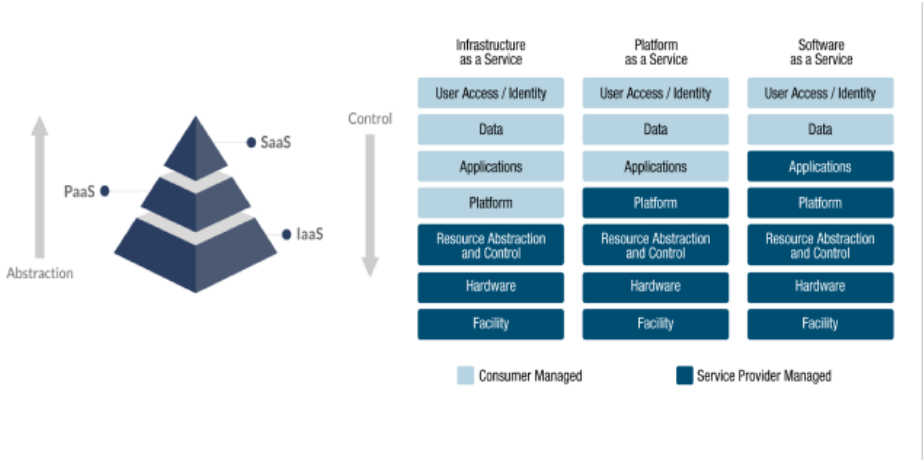
Η εικονικοποίηση είναι το κύριο θεμέλιο του υπολογιστικού νέφους [6]. Η έννοια αυτή αναφέρεται στο σύνολο των εργαλείων υλικού και λογισμικού που επιτρέπουν την αφαίρεση ενός φυσικού πόρου σε διάφορες λογικές μονάδες, οι οποίες μπορούν να χρησιμοποιηθούν χωριστά από διαφορετικά λειτουργικά συστήματα (OS) και εφαρμογές [7]. Η εικονικοποίηση πόρων αποτελεί βασικό χαρακτηριστικό για τους παρόχους υπολογιστικού νέφους για τη δημιουργία αποδοτικών, ευέλικτων και οικονομικά αποδοτικών μοντέλων υπολογισμού που ικανοποιούν ότι ανταποκρίνονται στις προκλήσεις της αγοράς του νέφους. Επιτρέπει την απλή διαχείριση των πόρων και τη δυναμική αλλαγή μεγέθους των πόρων, τη μείωση του κόστους του υλικού λόγω του διαμοιρασμού των πόρων, αυξημένη διαθεσιμότητα και ταχεία ανάκτηση μέσω εύκολης δημιουργίας αντιγράφων ασφαλείας και την ταχεία μετακίνηση. Το αποτέλεσμα είναι η καλύτερη εκμετάλλευση των πόρων συνολικά, μεγάλη εξοικονόμηση ενέργειας και φυσικού χώρου και ευκολότερη διαχείριση της υποδομής.

Η εικονικοποίηση μπορεί να υλοποιηθεί σε διάφορα επίπεδα με τη χρήση διαφορετικών μεθόδων. Ξεχωρίζουμε τρεις κύριες μορφές εικονικοποίησης, δηλαδή την "εικονικοποίηση διακομιστή", την "εικονικοποίηση δικτύου" και την "εικονικοποίηση αποθήκευσης", οι οποίες βασίζονται όλες στην έννοια της αφαίρεσης και διαμοιρασμού υλικού [8]. Η "εικονικοποίηση αποθήκευσης" επιτρέπει την πρόσβαση σε εικονικούς δίσκους ανεξαρτήτως της θέσης των δεδομένων και της αντιστοίχισης τους στη συσκευή αποθήκευσης. Η "εικονικοποίηση δικτύου" αναφέρεται στη δημιουργία απομονωμένων εικονικών δικτύων που επικαλύπτονται στην ίδια φυσική υποδομή και μοιράζονται το διαθέσιμο εύρος ζώνης. Ένα εικονικό δίκτυο μπορεί να συνδυάζει πολλαπλούς δικτυακούς πόρους και λειτουργίες, όπως εικονικές NIC ή λογικούς μεταγωγείς και δρομολογητές. Τέλος, η "εικονικοποίηση διακομιστών" επιτρέπει την ενοποίηση πολλαπλών απομονωμένων εικονικών διακομιστών σε έναν μόνο φυσικό διακομιστή.

2.1.3 Υπηρεσίες και μοντέλα ανάπτυξης

Σύμφωνα με το NIST υπάρχουν τρία μοντέλα υπηρεσιών νέφους: Η υποδομή ως υπηρεσία (IaaS) που αφορά στην παροχή υπολογιστικών πόρων, η πλατφόρμα ως υπηρεσία (PaaS) που αφορά στην παροχή υπολογιστικών πλατφορμών και το λογισμικό ως υπηρεσία (SaaS) που αφορά στην παροχή

εφαρμογών για τους τελικούς χρήστες [4]. Βασικός παράγοντας για την υλοποίηση εφαρμογών που βασίζονται στο νέφος είναι η επιλογή του σωστού μοντέλου υπηρεσιών. Προκειμένου να επιλέξει κανείς το σωστό μοντέλο υπηρεσιών ή συνδυασμό μοντέλων υπηρεσιών, πρέπει να κατανοήσει σε βάθος τι είναι κάθε μοντέλο υπηρεσιών και ποιες ευθύνες αναλαμβάνουν οι πάροχοι υπηρεσιών νέφους σε σύγκριση με τις ευθύνες που αναλαμβάνει ο καταναλωτής. Κάθε μοντέλο υπηρεσιών νέφους προσφέρει ένα επίπεδο αφαίρεσης και αυτοματοποίησης εργασιών που ελαχιστοποιεί τις ενέργειες που απαιτούνται από κάθε καταναλωτή για τη διαμόρφωση και την ανάπτυξη συστημάτων.



Σχήμα 2.2: Μοντέλα υπηρεσιών υπολογιστικού νέφους

Λογισμικό ως υπηρεσία (Cloud Software as a Service - SaaS): Το SaaS είναι το ανώτερο επίπεδο στη πυραμίδα του υπολογιστικού νέφους που παρέχει ολοκληρωμένες εφαρμογές στους καταναλωτές μέσω διαδικτύου. Ο πάροχος SaaS είναι υπεύθυνος για τη φιλοξενία, τη διαχείριση και τον έλεγχο της εφαρμογής και του περιβάλλοντος λειτουργίας της. Οι λεπτομέρειες σχετικά με την υποκείμενη υποδομή είναι διαφανείς για τους χρήστες, οι οποίοι έχουν απλή πρόσβαση στις λειτουργίες της εφαρμογής χωρίς τη δυνατότητα να ελέγχουν ή να προσαρμόζουν τα χαρακτηριστικά της. Μεταξύ των πιο δημοφιλών εφαρμογών SaaS είναι το GoogleApps [9], που περιλαμβάνει το Gmail, το Google- Docs και το GoogleDrive και το Office 365 της Microsoft.

Πλατφόρμα ως υπηρεσία (Platform as a Service - PaaS): Το εν

λόγω μοντέλο παροχής νέφους είναι σχεδιασμένο για προγραμματιστές λογισμικού και τους παρέχει πλατφόρμες για το σχεδιασμό, την υλοποίηση και την ανάπτυξη εφαρμογών. Οι πλατφόρμες PaaS είναι υψηλού επιπέδου ολοκληρωμένα περιβάλλοντα (λειτουργικό σύστημα, γλώσσες προγραμματισμού, βιβλιοθήκες, βάσεις δεδομένων, διακομιστές ιστού) που υποστηρίζουν τον πλήρη κύκλο ζωής του λογισμικού. Οι χρήστες PaaS έχουν τον πλήρες έλεγχο των εφαρμογών και των ρυθμίσεων διαμόρφωσης του περιβάλλοντος, ωστόσο δεν έχουν τον έλεγχο της υποκείμενης υποδομής που συντηρείται από τον πάροχο του νέφους. Αυτό έχει ως στόχο την επίτευξη απλοποίησης της διαδικασίας ανάπτυξης λογισμικού και επιτρέπει στους προγραμματιστές να εστιάσουν στα βασικά χαρακτηριστικά των εφαρμογών τους χωρίς να τους απασχολούν πολύπλοκες λειτουργίες διαχείρισης χαμηλού επιπέδου. Παραδείγματα γνωστών πλατφορμών PaaS αποτελούν η Google App Engine [10] και η Microsoft Azure Cloud Services [11].

Υποδομή ως υπηρεσία (Infrastructure as a Service - IaaS): Φτάνοντας κάτω στη πυραμίδα, καταλήγουμε στο θεμελιώδες μοντέλο για την παροχή υπηρεσιών νέφους, δηλαδή την υπηρεσία IaaS. Η υπηρεσία αυτή αναφέρεται στην κατά παραγγελία παροχή βασικών πόρων υποδομής (επεξεργαστική ισχύς, μνήμη, αποθήκευση και δίκτυο). Οι χρήστες IaaS μπορούν να αιτηθούν είτε υπολογιστικούς πόρους που παρέχονται με τη μορφή εικονικών μηχανών ή containers (γνωστή ως εικονικοποίηση σε επίπεδο λειτουργικού συστήματος) είτε να νοικιάσουν φυσικούς διακομιστές για λόγους καλύτερης απόδοσης (γνωστή ως υπηρεσία MaaS). Οι χρήστες IaaS έχουν περισσότερο έλεγχο των πόρων τους σε σύγκριση με τα μοντέλα SaaS και PaaS καθώς είναι υπεύθυνοι για τη διαχείριση του λειτουργικού συστήματος, των εφαρμογών και των δεδομένων που έχουν τεθεί σε λειτουργία. Οι χρήστες IaaS είναι σε θέση να κλιμακώνουν δυναμικά τους πόρους που νοικιάζουν ανάλογα με τους φόρτους εργασίας τους, γεγονός που τους επιτρέπει να πληρώνουν μόνο για ό,τι χρησιμοποιούν. Ο επικρατέστερος πάροχος στην αγορά IaaS είναι η Amazon Elastic Compute Cloud (EC2) [12] και άλλοι δημοφιλείς πάροχοι IaaS περιλαμβάνουν το Microsoft Azure [13], το Google Compute Engine (GCE) [14], IBM Cloud [15]. Όλα τα μοντέλα υπηρεσιών μπορούν να εφαρμοστούν σε διαφορετικά σενάρια, τα οποία μπορούν να ταξινομηθούν σε τέσσερα κύρια μοντέλα ανάπτυξης ανάλογα με την ιδιοκτησία της υποδομής νέφους και τα δικαιώματα πρόσβασης στις παρεχόμενες υπηρεσίες.

Ιδιωτικό νέφος (private cloud): προσφέρει ασφαλείς υπηρεσίες που χρησιμοποιούνται μόνο από τον οργανισμό που διαθέτει την υποδομή και έχει τον πλήρη έλεγχό της.

Νέφη κοινότητας (community cloud): η υποδομή χρησιμοποιείται από κοινού για τη συνεργασία μεταξύ μιας ομάδας οργανισμών που έχουν κοινά ζητήματα (π.χ. αποστολή, απαιτήσεις ασφαλείας, θέματα πολιτικής και συμμόρφωσης). Η διαχείρισή της μπορεί να γίνεται από τον ίδιο τον οργανισμό ή από τρίτο μέρος και μπορεί να βρίσκεται εντός ή εκτός των εγκαταστάσεων του οργανισμού.

Δημόσιο νέφος (pulic cloud): περιλαμβάνει μια μεγάλη και ιδιαίτερα αποτελεσματική υποδομή που ανήκει και διαχειρίζεται από έναν εξωτερικό οργανισμό και παρέχει υπηρεσίες κατά παραγγελία στο ευρύ κοινό. Οι υπηρεσίες και τα δεδομένα φιλοξενούνται εκτός των εγκαταστάσεων των χρηστών.

Υβριδικό νέφος (hybrid cloud): αναφέρεται σε μια υποδομή που συνδυάζει δύο ή περισσότερα νέφη (ιδιωτικά, κοινοτικά ή δημόσια), τα οποία παραμένουν ανεξάρτητες οντότητες αλλά διασυνδέονται μεταξύ τους για να επιτρέπουν την ανάπτυξη εσωτερικών και εξωτερικών υπηρεσιών.

2.2 Περίληψη διαθέσιμων προσομοιωτών υπολογιστικού νέφους

Σε αυτό το υποκεφάλαιο παρουσιάζουμε μερικούς από τους πιο δημοφιλείς προσομοιωτές νέφους που είναι διαθέσιμοι, επισημαίνοντας τα σημαντικά χαρακτηριστικά του κάθε προσομοιωτή. Ένας από τους πιο γνωστούς και ευρέως χρησιμοποιούμενους προσομοιωτές είναι η εργαλειοθήκη προσομοίωσης *CloudSim*[16], η οποία είναι μια δημοφιλής λύση για την προσομοίωση περιβαλλόντων νέφους. Το CloudSim είναι ένας προσομοιωτής διακριτών συμβάντων και η αρχιτεκτονική του έχει πέντε επίπεδα (δηλαδή δίκτυο, πόροι νέφους, υπηρεσίες νέφους, υπηρεσίες VM, δομές διεπαφής χρήστη). Οι εικονικές μηχανές στο CloudSim έχουν τρεις καταστάσεις και οι χρήστες μπορούν να διαμορφώσουν μόνο στατικό κόστος χρήσης για τους πόρους (δηλ. χρήση μνήμης, εύρους ζώνης, CPU και αποθήκευσης). Το CloudSim περιέχει επίσης ένα μοντέλο ισχύος, αλλά περιορίζεται μόνο στην κατανάλωση ισχύος της CPU. Στο CloudSim, οι χρήστες καθορίζουν εργασίες δημιουργώντας τα λεγόμενα cloudlets, τα οποία επεξεργάζονται από εικονικές μηχανές που εκτελούνται σε πόρους νέφους. Πρόκειται για μια λύση ανοικτού κώδικα, βασισμένη σε Java, η οποία είναι διαθέσιμη στο GitHub [17]. Το *CloudAnalyst* [18] είναι μια από τις παλαιότερες επεκτάσεις του CloudSim και περιέχει σημαντικές προσθήκες που υλοποιούν εφαρμογές χρηστών, συνδέσεις στο διαδίκτυο, προσομοιώσεις που ορίζονται από χρονικές περιόδους και δι-

αμεσολαβητές υπηρεσιών. Το CloudAnalyst είναι διαθέσιμο στο cloudbus.org [19]. Το *EMUSIM* [20] έχει ως στόχο να συγκεντρώσει την προσομοίωση και την προσομοίωση σε ένα εργαλείο για την πρόβλεψη της συμπεριφοράς των υπηρεσιών όταν αλλάζουν οι πόροι. Στην πρώτη φάση, η φάση της προσομοίωσης μπορεί να χρησιμοποιηθεί για την εξαγωγή πληροφοριών σχετικά με τη συμπεριφορά της εφαρμογής, στη δεύτερη φάση οι πληροφορίες αυτές τροφοδοτούνται σε ένα μοντέλο προσομοίωσης για να καταλήξουμε σε ακριβέστερες μεθόδους προσομοίωσης της εφαρμογής. Είναι διαθέσιμο στη διεύθυνση cloudbus.org [21]. Ο προσομοιωτής κέντρου δεδομένων *DCSim* [22] είναι ένα επεκτάσιμο πλαίσιο προσομοίωσης [33] που χρησιμοποιεί τη γλώσσα προγραμματισμού Java. Αναπτύχθηκε για την προσομοίωση της διαχείρισης εικονικών πόρων. Η βασική ιδιότητα του DCSim είναι να παρέχει μηχανισμό διαμοιρασμού VM για VM που ανήκουν σε μία ενιαία, πολυεπίπεδη εφαρμογή. Η αρχιτεκτονική του αποτελείται από: DataCentre, Host, VMAllocation, και υπάρχουν μοναδικοί διαχειριστές για τη δικτύωση, τις εικονικές μηχανές και την κατανάλωση ενέργειας. Κατά τη διάρκεια της προσομοίωσης μετριούνται οι ακόλουθες τιμές: (i) δυναμική κατανομή VM , (ii) παραβίαση SLA, (iii) ώρες λειτουργίας και χρήση του κεντρικού υπολογιστή, (iv) κατανάλωση ενέργειας και (v) χρόνος εκτέλεσης προσομοίωσης και αλγορίθμου. Το DCSim είναι διαθέσιμο στο GitHub [23]. Το *iCanCloud* [24] είναι ένας προσομοιωτής βασισμένος στο OMNET++ [25] (προγραμματισμένος σε C++), ο οποίος αποσκοπεί στην προσομοίωση υποδομών νέφους. Η πρωτοτυπία του είναι ότι εισάγει ένα στοιχείο hypervisor για τη διαχείριση πολιτικών διαμεσολάβησης νέφους. Τα πειράματά του αφορούν τις ανταλλαγές μεταξύ κόστους και απόδοσης μιας συγκεκριμένης εφαρμογής, η οποία μπορεί να εκτελεστεί με συλλογή από VMs που έχουν οριστεί προηγουμένως από τον χρήστη. Ο προσομοιωτής iCanCloud παρέχει λίγα μοντέλα υφιστάμενων VMs σε γνωστά νέφη όπως το Amazon (EC2). Ο χειρισμός της εικονικής μηχανής είναι αρκετά απλός, μπορεί να εκτελέσει τις εργασίες, αλλά παραμελεί την προσομοίωση δικτύου που σχετίζεται με το σύννεφο. Χρησιμοποιεί το πλαίσιο Inet (που περιλαμβάνεται στο OMNET++) για τα πρωτόκολλα TCP/UDP. Το *GreenCloud* [26] είναι μια επέκταση του προσομοιωτή δικτύου NS-2 [27] και είναι βασισμένο σε σενάρια TCL και C++. Ο βασικός στόχος του προσομοιωτή είναι να παρουσιάσει την ανάλυση των κέντρων δεδομένων του νέφους με ενεργειακή επίγνωση, περιλαμβάνοντας τις μετρήσεις της ενεργειακής χρήσης των στοιχείων του συστήματος (δηλαδή διακομιστές, μεταγωγείς, συνδέσεις). Το μοντέλο ενεργειακής κατανάλωσης περιλαμβάνει το PUE και την αποδοτικότητα της υποδομής DC (και τα δύο αναπαρίστανται σε Watt). Τα στοιχεία της αρχιτεκτονικής μπορούν να είναι διακομιστές και μεταγωγείς και το εργαλείο διαθέτει τη δυνατότητα προσομοίωσης πρωτοκόλλων TCP/IP για τη δικτύωση. Το *NetworkCloudSim* [28] προεκτείνει το CloudSim με

βελτιωμένες επιλογές δικτύου για τη μοντελοποίηση της διασύνδεσης των κέντρων δεδομένων. Το NetworkCloudSim επέκτεινε το CloudSim για να παρέχει ένα επεκτάσιμο δίκτυο και ένα γενικευμένο μοντέλο εφαρμογών. Υποστηρίζει εφαρμογές με στοιχεία επικοινωνίας που περιλαμβάνουν διεπαφή μετάδοσης μηνυμάτων (MPI) και workflows. Σχεδιάζεται ένα μοντέλο δικτύου για κέντρα δεδομένων νέφους και καθίσταται δυνατή η κοινή χρήση εύρους ζώνης. Η επέκταση έχει αναπτυχθεί με τέτοιο τρόπο ώστε οι χρήστες να μπορούν να τροποποιήσουν την τοπολογία του δικτύου απλά τροποποιώντας ένα configuration file. Το *GroudSim* [29] αποσκοπεί στην προσομοίωση συστημάτων Grid και Cloud με κλιμακούμενο τρόπο. Αυτό το λογισμικό που βασίζεται σε Java είναι ένας προσομοιωτής διακριτών γεγονότων με κύριες ικανότητες τον υπολογισμό του κόστους και προσομοιώσεις με χρήση του φορτίου υποβάθρου των πόρων. Η αρχιτεκτονική του αποτελείται από στοιχεία Entity, Job και Cost και περιέχει ένα μοντέλο κόστους που χρησιμοποιεί χρονικές μονάδες χρήσης CPU ή χρήσης δεδομένων. Το *SimIC* [30] είναι ένα διακριτό κιτ εργαλείων προσομοίωσης που αναπτύχθηκε με τη γλώσσα προγραμματισμού Java χρησιμοποιώντας το πακέτο SimJava. Το SimIC στοχεύει στην αναπαραγωγή μιας διευκόλυνσης μεταξύ νέφους όπου πολλαπλά σύννεφα συνεργάζονται μεταξύ τους για τη διανομή αιτημάτων υπηρεσιών σε σχέση με την επιθυμητή ρύθμιση προσομοίωσης. Το *MDCSim* [31] είναι μια εμπορική ολοκληρωμένη και κλιμακούμενη εργαλειοθήκη προσομοίωσης που χρησιμοποιείται για την ανάλυση πολυεπίπεδων κέντρων δεδομένων. Μοντελοποιεί τα υποκείμενα χαρακτηριστικά υλικού των στοιχείων του κέντρου δεδομένων και εκτιμά την κατανάλωση ενέργειας των κέντρων δεδομένων. Η απόδοση και ο χρόνος απόκρισης θεωρούνται μετρικές επιδόσεων και η τοπολογία του κέντρου δεδομένων παρέχεται ως κατευθυνόμενος γράφος από το πακέτο δικτύου MDCSim. Το MDCSim βοηθά τους χρήστες του υπολογιστικού νέφους να εξετάσουν διαφορετικές διαμορφώσεις πόρων για τη βελτίωση της απόδοσης των διαδικτυακών εφαρμογών, διατηρώντας παράλληλα χαμηλή την κατανάλωση ενέργειας. Τέλος το *DsSim* [32] είναι ένας προσομοιωτής κατανεμημένων συστημάτων που αναπτύχθηκε πρόσφατα. Δημιουργήθηκε με τη γλώσσα προγραμματισμού python και αυτό που τον διακρίνει από άλλους υπάρχοντες προσομοιωτές είναι ότι επικεντρώνεται στο σχεδιασμό αλγορίθμων χρονοπρογραμματισμού (Shortest Job First). Διαθέτει GUI που ονομάζεται ds-viz και θεωρείται ελαφρύς προσομοιωτής σε σύγκριση με άλλους προσομοιωτές.
Στα επόμενα κεφάλαια περιγράφονται λεπτομερώς τα εργαλεία CloudSim, GreenCloud και CloudAnalyst. Είναι απαραίτητο να κατανοηθεί η αρχιτεκτονική και οι αρχές σχεδιασμού αυτών των εργαλείων προκειμένου να εκτελεστούν προσομοιώσεις για κάθε προσομοιωτή ξεχωριστά.

Κεφάλαιο 3

CloudSim

Το CloudSim είναι ένας προσομοιωτής νέφους βασισμένος σε Java που αναπτύχθηκε και συντηρείται από το Εργαστήριο Υπολογιστικού Νέφους και Κατανεμημένων Συστημάτων (CLOUDS) στο Πανεπιστήμιο της Μελβούρνης. "Ο πρωταρχικός στόχος αυτού του έργου είναι η παροχή ενός γενικευμένου και κλιμακούμενου πλαισίου προσομοίωσης που επιτρέπει την απρόσκοπτη μοντελοποίηση, προσομοίωση και τον πειραματισμό των αναδυόμενων υποδομών και υπηρεσιών εφαρμογών υπολογιστικού νέφους. Με τη χρήση του CloudSim, οι ερευνητές και οι προγραμματιστές μπορούν να επικεντρωθούν στα συγκεκριμένα ζητήματα σχεδιασμού συστημάτων που θέλουν να διερευνήσουν χωρίς να ανησυχούν για τις λεπτομέρειες χαμηλού επιπέδου που σχετίζονται με τις υποδομές και τις υπηρεσίες που βασίζονται στο νέφος" [17]. Διαθέτει πολλές λειτουργίες, όπως η μοντελοποίηση και η προσομοίωση κέντρων δεδομένων μεγάλης κλίμακας, κεντρικών υπολογιστών, υπολογιστικών πόρων με ενεργειακή επίγνωση και υποστήριξη για πολιτικές που καθορίζονται από τον χρήστη για την κατανομή εικονικών μηχανών στους κεντρικούς υπολογιστές. Στο Σχήμα 3.1 παρουσιάζεται η αρχιτεκτονική του και συμφωνα με το [20] το CloudSim χωρίζεται σε διάφορες ενότητες:

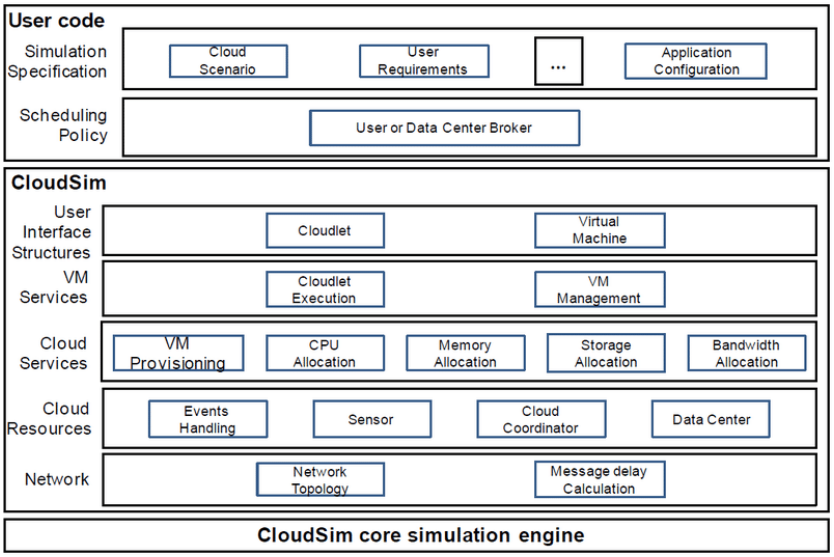
• Κώδικας χρήστη (User Code): κώδικας που υλοποιεί τις απαιτήσεις του χρήστη και τη διαμόρφωση της εφαρμογής.

• Broker: αλγόριθμοι που συνδέουν τις διεπαφές χρήστη με το κέντρο δεδομένων και τις πολιτικές προγραμματισμού.

• Διεπαφή χρήστη (User Interface): αυτό το επίπεδο παρέχει την αλληλεπίδραση μεταξύ του χρήστη και του προσομοιωτή.

• Υπηρεσίες VM (VM Services): αναλαμβάνει την εκτέλεση της εφαρμογής και το διαχείριση των VM.

• Υπηρεσίες νέφους (Cloud Services): αυτό το στρώμα περιλαμβάνει διάφορες υπηρεσίες που παρέχονται στον χρήστη των υπηρεσιών νέφους. Καλύπτει διάφορες δραστηριότητες, όπως η παροχή VM και η κατανομή υποκείμενων φυσικών πόρων (CPU, μνήμη κ.λπ.).
• Πόροι νέφους (Cloud Resources): αυτό το στρώμα περιλαμβάνει διάφορους κύριους πόρους, όπως κέντρα δεδομένων (διασφαλίζει ότι οι διάφοροι πόροι του νέφους μπορούν να λειτουργούν με συνεργατικό τρόπο) στο περιβάλλον νέφους.
• Δίκτυο (Network): διαχείριση των χαρακτηριστικών του δικτύου, όπως ο υπολογισμός της καθυστέρησης μηνυμάτων και τοπολογία δικτύου.



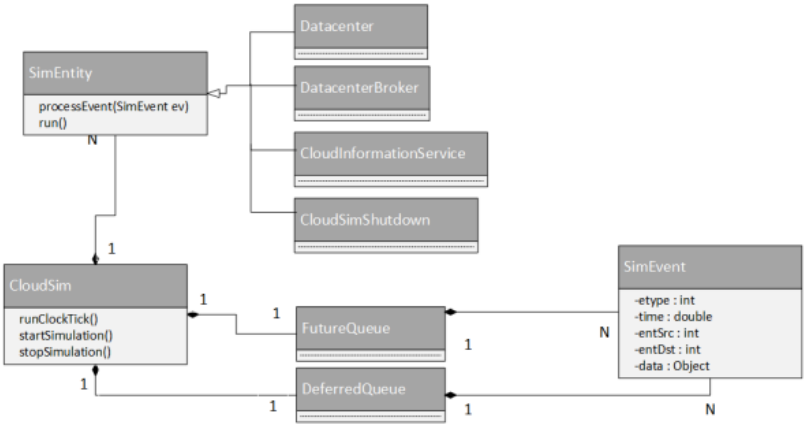
Σχήμα 3.1: Ο σχεδιασμός του CloudSim [20]

3.1 Το μοντέλο συμβάντων του CloudSim

Ο βασικός πυρήνας προσομοίωσης του CloudSim βασίζεται στο GridSim, το οποίο με τη σειρά του βασίζεται στη simjava [33]. Ωστόσο, το GridSim διαχειρίζεται την προσομοίωση εντός του πλαισίου με βάση το μοντέλο προσομοίωσης γεγονότων της simjava, το οποίο είναι μια εργαλειοθήκη για τη δημιουργία μοντέλων που λειτουργούν σε πολύπλοκα συστήματα [33]. Ήδη από την έκδοση 2.0 του CloudSim, η προσομοίωση γεγονότων γίνεται εσωτερικά αντί να χρησιμοποιείται η simjava. Με αυτόν τον τρόπο παρακάμπτονται οι περιορισμοί της simjava, όπως είναι :
i) η αδυναμία επαναφοράς των γεγονότων κατά τη διάρκεια της εκτέλεσης,

ii) η δημιουργία γενικών εξόδων και
iii) η πολυπλοκότητα της multi threading φύσης της simjava.

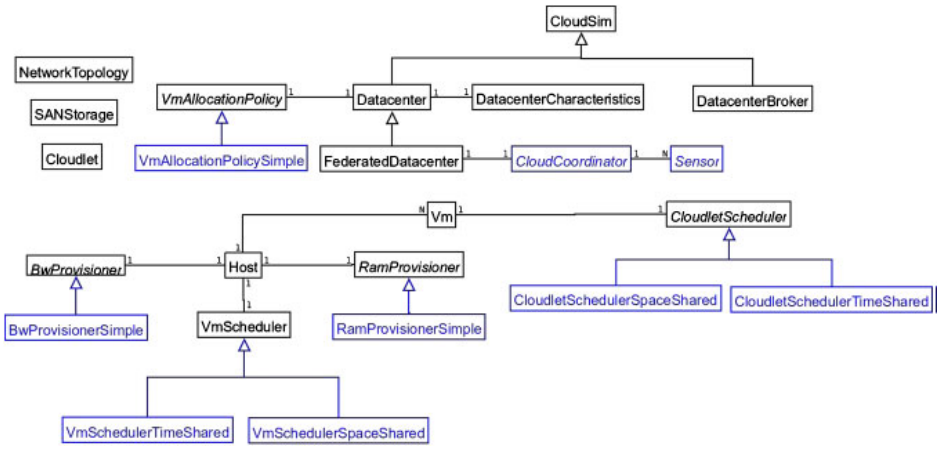
Επομένως, το CloudSim αποθηκεύει τα μελλοντικά συμβάντα σε μια ουρά αναμονής (FutureQueue) μέχρι να έρθει η ώρα να τα εκτελέσει και στη συνέχεια μετακινεί τα ολοκληρωμένα συμβάντα στο DeferredQueue οπώς φαίνεται στο Σχήμα 3.2. Εντός της προσομοίωσης υπάρχει η αφηρημένη κλάση SimEntity. Είναι σε θέση να χειρίζεται συμβάντα και να στέλνει συμβάντα σε άλλες οντότητες. Οι υποκλάσεις της SimEntity είναι οι εξής: Datacenter, DatacenterBroker, CloudInformationService, CloudSimShutdown. Η μέθοδος CloudSim.runClockTick() επαναλαμβάνει όλες τις οντότητες μέσα στην προσομοίωση και εκτελεί τη μέθοδο run() σε αυτά τα αντικείμενα. Μια άλλη κεντρική κλάση στη διαδικασία προσομοίωσης είναι η CloudInformationService. Η υπηρεσία πληροφοριών νέφους (Cloud Information Service - CIS) είναι μια οντότητα που παρέχει υπηρεσίες καταχώρισης, ευρετηρίασης και ανακάλυψης πόρων νέφους. Άλλες οντότητες όπως ο DatacenterBroker, μπορούν να επικοινωνήσουν με αυτή την κλάση για την υπηρεσία εντοπισμού πόρων, η οποία επιστρέφει μια λίστα με καταχωρημένα αναγνωριστικά πόρων.



Σχήμα 3.2: Διάγραμμα κλάσεων του βασικού πλαισίου προσομοίωσης CloudSim

Ο σχεδιασμός του CloudSim παρέχει τέσσερις λειτουργίες για κάθε συμβάν κατά την προσομοίωση: απενεργοποίηση, εναλλαγή οντοτήτων, δημιουργία νέων και επανεκκίνηση της προσομοίωσης κατά την εκτέλεση [20]. Στο Σχήμα 3.3 παρουσιάζεται το διάγραμμα σχεδίασης κλάσεων του CloudSim, το οποίο δείχνει ότι το CloudSim διαθέτει κλάσεις παροχής, κλάσεις πολιτικής κατανομής και κλάσεις χρονοπρογραμματιστή. Οι κλάσεις παροχής είναι υπεύθυνες για την μοντελοποίηση του κέντρου δεδομένων με όλα τα κύρια στοιχεία του, όπως εικονικές μηχανές, μνήμη και εύρος ζώνης,

οι κλάσεις πολιτικής κατανομής διαχειρίζονται πολιτικές κατανομής χρόνου και χώρου και οι κλάσεις χρονοπρογραμματιστή είναι υπεύθυνες για την εφαρμογή των πολιτικών κατανομής χρόνου και χώρου όσον αφορά την κατανομή πυρήνων για τα Vms [20].



Σχήμα 3.3: Διάγραμμα σχεδιασμού του CloudSim [20]

Παρακάτω περιγράφουμε ορισμένες βασικές κλάσεις που αναπαριστούν την προσομοίωση του περιβάλλοντος υπολογιστικού νέφους στο CloudSim:

Cloudlet: για τη μοντελοποίηση υπηρεσιών σε επίπεδο εφαρμογής

Datacenter: για τη μοντελοποίηση υπολογιστικών υπηρεσιών επιπέδου υποδομής.

DataCenterBroker: για την μοντελοποίηση του broker, ο οποίος είναι υπεύθυνος για την μεσολάβηση ανάμεσα στους χρήστες και στον πάροχο υπηρεσιών, ανάλογα με τις απαιτήσεις QoS των χρηστών και αναπτύσσει εργασίες υπηρεσιών στα νέφη.

BwProvisioner: για τη μοντελοποίηση των πολιτικών παροχής εύρους ζώνης στο VM

Host: αυτή η κλάση μοντελοποιεί έναν φυσικό πόρο, όπως έναν κεντρικό υπολογιστή

NetworkTopology: η κλάση αυτή υποδεικνύει τη συμπεριφορά του δικτύου.

RamProvisioner: για την εκχώρηση πρωτογενούς μνήμης (RAM) στα

VMs. Η εκτέλεση και η ανάπτυξη VM σε έναν κεντρικό υπολογιστή είναι εφικτή μόνο εάν το στοιχείο RamProvisioner εγκρίνει ότι ο κεντρικός υπολογιστής διαθέτει την απαιτούμενη ποσότητα ελεύθερης μνήμης. Το RamProvisionerSimple δεν επιβάλλει κανέναν περιορισμό στην ποσότητα μνήμης που μπορεί να ζητήσει ένα VM. Το αίτημα πόρου RAM θα απορριφθεί εάν υπερβαίνει τη διαθέσιμη χωρητικότητα

SanStorage: για τη μοντελοποίηση ενός δικτύου

Sensor: εφαρμόζει μια απλή διεπαφή, που μπορεί να χρησιμοποιεί για την προσομοίωση της αποθήκευσης και την ανάκτηση κάθε τύπου δεδομένων, σε κάθε χρονική στιγμή ανάλογα την διαθεσιμότητα του εύρους ζώνης του δικτύου

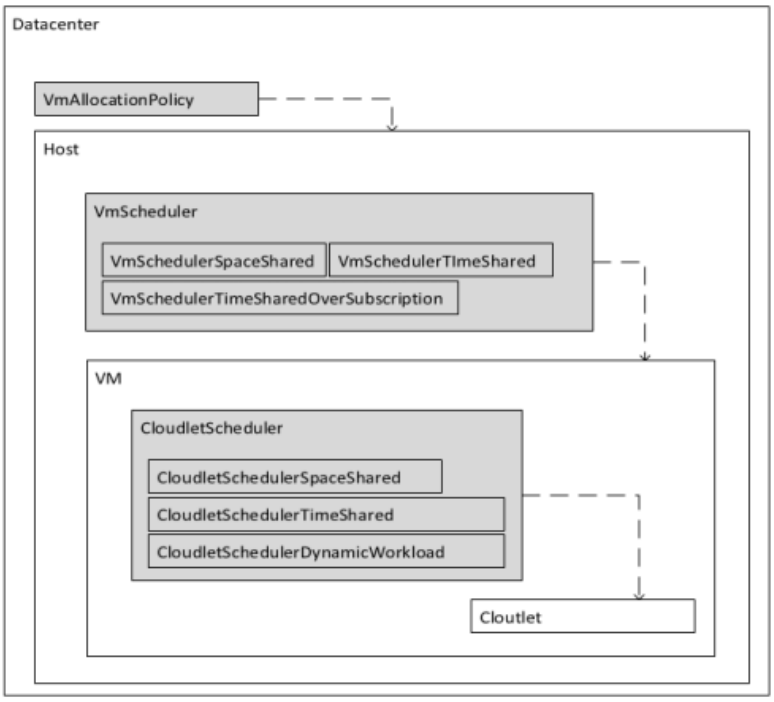
Vm: για τη μοντελοποίηση μιας εικονικής μηχανής που εκτελείται σε έναν συγκεκριμένο κεντρικό υπολογιστή με τις προδιαγραφές της όπως η μνήμη RAM

3.2 Μοντελοποίηση της κατανομής των εικονικών μηχανών

Ο προγραμματισμός των VM και οι φόρτοι εργασίας μπορούν να διαμορφωθούν με διαφορετικούς τύπους πολιτικών προγραμματισμού. Οι πολιτικές αυτές έχουν μεγάλο αντίκτυπο κατά την προσομοίωση ενός νέφους. Για την καλύτερη δυνατή προσομοίωση μιας επιθυμητής συμπεριφοράς θα πρέπει να γνωρίζει κανείς τους διαφορετικούς τύπους αυτών των πολιτικών. Προκειμένου να γίνει κατανοητή η συνολική δομή των πολιτικών χρονοπρογραμματισμού που χρησιμοποιούνται στο CloudSim, το Σχήμα 3.4 απεικονίζει τις διαφορετικές πολιτικές χρονοπρογραμματισμού για ένα κέντρο δεδομένων,σε επίπεδο host και VM.

VmAllocationPolicy: Ένα κέντρο δεδομένων αποτελείται από πολλά VM που πρέπει να διατεθούν σε κεντρικούς υπολογιστές. Για την επίλυση αυτής της πρόκλησης, η πολιτική VmAllocationPolicy συσχετίζει τα VM με τους διαθέσιμους hosts του datacenter. Μια άμεση υλοποίηση είναι η κλάση VmAllocationPolicySimple η οποία απλά βρίσκει τον πρώτο κεντρικό υπολογιστή που διαθέτει τη ζητούμενη χωρητικότητα. Η χωρητικότητα καθορίζεται από τους πόρους που σχετίζονται με το VM και τον host.

VmScheduler: Μόλις ένα VM σταλεί σε έναν host με αρκετούς πόρους,

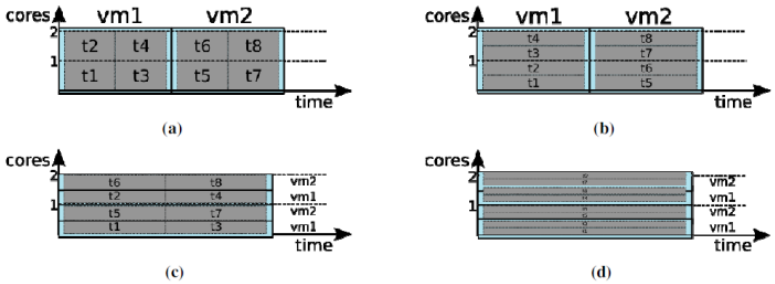


Σχήμα 3.4: Πολιτικές χρονοπρογραμματισμού

υπάρχει δυνατότητα περαιτέρω χρονοπρογραμματισμού στο επίπεδο του ίδιου του host. Υπάρχουν δύο γενικές έννοιες που είναι διαθέσιμες για τον χρονοπρογραμματισμό των VM: Time-shared και space-shared. Η έννοια time-shared σημαίνει ότι η CPU μπορεί να διαμοιραστεί ταυτόχρονα σε πολλαπλά VMs. Σε αντίθεση, το space-shared επιτρέπει μόνο αποκλειστικούς πυρήνες CPU για κάθε VM. Η VmSchedulerTimeShareOversubscription επιτρέπει την υπερ-συνδρομή. Με άλλα λόγια, αυτός ο χρονοπρογραμματιστής εξακολουθεί να επιτρέπει την κατανομή VM στον κεντρικό υπολογιστή που απαιτούν μεγαλύτερη χωρητικότητα CPU από αυτή που είναι πραγματικά διαθέσιμη.

CloudletScheduler: Ένας χρονοπρογραμματιστής εργασιών νέφους (Cloudlet Scheduler) είναι υπεύθυνος για τη διαχείριση του πραγματικού φόρτου εργασίας σε ένα συγκεκριμένο VM. Σε ένα VM μπορεί να εκτελούνται ταυτόχρονα πολλά cloudlets. Ανάλογα με τον VmScheduler,o CloudletScheduler έχει δύο βασικές υλοποιήσεις: τον time-shared και το space-shared. Ο συνδυασμός των πολιτικών time-shared και space-shared μοιράζεται όπως στην εικόνα 3.5. Μόλις η πολιτική VmAllocationPolicy βρει έναν host με τους απαραίτητους πόρους: πυρήνες cpu, μνήμη, εύρος ζώνης δικτύου και απο-

θηκευτικό χώρο, ο προγραμματισμός εντός του VmScheduler και CloudletScheduler γίνεται επί του παρόντος μόνο όσον αφορά την CPU.



Σχήμα 3.5: Διαμοιρασμός χρόνου και χώρου: (α) Space-shared γιαVMs και εργασίες, (β) Space-shared για VMs και time-shared για εργασίες, (γ) Time-shared για VMs,space-shared για εργασίες, και (δ) Time-shared για VMs και εργασίες [20]

3.3 Εκδόσεις του CloudSim

Το CloudSim έχει αναπτυχθεί μέσω έξι επίσημων δημόσιων εκδόσεων και η τελευταία έκδοση είναι η v4.0, η οποία παρέχει υποστήριξη για εικονικοποίηση container. Σε γενικές γραμμές, οι νέες εκδόσεις του CloudSim αποσκοπούν στη βελτίωση των προσομοιώσεων με τη μοντελοποίηση νέων οντοτήτων (π.χ. εσωτερικά δίκτυα) ή την παροχή νέων πολιτικών (π.χ. για την επιλογή VM).

Έκδοση 1.0 : Ήταν η πρώτη έκδοση του CloudSim και εκδόθηκε στις 7 Απριλίου 2009. Αυτή η έκδοση του CloudSim υποστήριζε την μοντελοποίηση και την προσομοίωση μεγάλης κλίμακας υποδομής υπολογιστικής νέφους,συμπεριλαμβανομένων των data centers σε ένα μοναδικό φυσικό υπολογιστικό κόμβο, μια αυτόνομη πλατφόρμα για μοντελοποίηση data center, service brokers, προγραμματισμό και πολιτικές καταμερισμού.

Έκδοση 2.1: Κυκλοφόρησε στις 27 Ιουλίου 2010. Έχει τη δυνατότητα να κάνει μετεγκατάσταση χρησιμοποιώντας το apache Maven. Το Maven παρέχει διάφορα εργαλεία και plugin για την απλοποίηση του έργου, όπως διορθώσεις σφαλμάτων,αναδιοργάνωση και αφαίρεση απαρχαιωμένου κώδικα.

Έκδοση 2.1.1: Κυκλοφόρησε την 1η Φεβρουαρίου 2011. Έχει τη δυνατότητα να διορθώνει ορισμένα σφάλματα που υπάρχουν στην έκδοση 2.1.

Έκδοση 3.0: Κυκλοφόρησε στις 11 Ιουνίου 2012. Αυτή η έκδοση της CloudSim ενημερώνει επίσης για διορθώσεις σφαλμάτων. Επιπλέον, οι ενημερώσεις είναι οι εξής νέος χρονοπρογραμματιστής VM, νέο μοντέλο δικτύου κέντρου δεδομένων, νέες πολιτικές κατανομής και επιλογής VM, νέα μοντέλα ισχύος, νέες διαδρομές φόρτου εργασίας, υποστήριξη για εξωτερικούς φόρτους εργασίας και υποστήριξη για τον ορισμό από τον χρήστη του τέλους της προσομοίωσης. Υποστηρίζει την εξάλειψη μερικών κλάσεων όπως Cloud coordinator, PowerPe, Sensor και Power.PeList. Αυτή η έκδοση έχει τη δυνατότητα διόρθωσης ορισμένων αλλαγών και σφαλμάτων API.

Έκδοση 4.0: CloudSim κυκλοφόρησε τον Ιανουάριο του 2017. Υποστηρίζει container virtualization και πολλές διορθώσεις σφαλμάτων.

3.4 Περιγραφή παραδειγμάτων του CloudSim

Όλα τα παραδείγματα παρέχονται στο φάκελο CloudSim example, στο πακέτο 'org.cloudbus.cloudsim.example', και ακολουθούν ορισμένα τυπικά βήματα για την υλοποίηση μιας συγκεκριμένης διαμόρφωσης για την έναρξη μιας προσομοίωσης. Σε αυτή την ενότητα, η ροή της προσομοίωσης θα γίνει κατανοητή μέσω των παραδειγμάτων του Cloudsim. Υπάρχουν έντεκα βήματα που ακολουθούνται σε κάθε παράδειγμα με ορισμένες παραλλαγές, τα οποία προσδιορίζονται ως εξής:

1. Ορισμός του αριθμού των χρηστών για την τρέχουσα προσομοίωση.

```
int num_user = 1; // number of cloud users Calendar calendar =
Calendar.getInstance();
boolean trace_flag = false;
```

2. Αρχικοποίηση της προσομοίωσης, που περιλαμβάνει την τρέχουσα ώρα, των αριθμό των χρηστών και το trace flag.

```
CloudSim.init(num_user, calendar, trace_flag);
```

3. Δημιουργία κέντρου δεδομένων ,όπου η μέθοδος createDatacenter() αρχικοποιεί τα διάφορα χαρακτηριστικά του κέντρου δεδομένων μαζί με τη λίστα

των hosts. Πρόκειται για την πιο σημαντική οντότητα δίχως αυτήν δεν υφίσταται καμία δυνατότητα εφαρμογής της προσομοίωσης για τη φιλοξενία της εικονικής μηχανής.

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

4. Δημιουργία Datacenter broker. Όπου η μέθοδος createBroker() αρχικοποιεί το αντικείμενο της οντότητας από την κλάση DatacenterBroker.

```
DatacenterBroker broker = createBroker(); int brokerId = broker.getId();
```

5. Δημιουργία ενός ή περισσότερων εικονικών μηχανών.

```
vmlist = new ArrayList<Vm>();
int vmid = 0;
int mips = 1000;
long size = 10000;
int ram = 512;
long bw = 1000;
int pesNumber = 1;
String vmm = "Xen";
Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerTimeShared());
vmlist.add(vm);
```

6. Ανάθεση ενός ή περισσότερων εικονικών μηχανών στον broker.

```
broker.submitVmList(vmlist);
```

7. Δημιουργία των cloudlets και προσδιορισμός των χαρακτηριστικών τους.

```
cloudletList = new ArrayList<Cloudlet>();
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();
Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
```

```
utilizationModel, utilizationModel, utilizationModel);
cloudlet.setUserId(brokerId);
cloudlet.setVmId(vmid);
cloudletList.add(cloudlet);
```

8. Ανάθεση των cloudlets στον broker.

```
broker.submitCloudletList(cloudletList);
```

9. Εντολή για εκκίνηση της προσομοίωσης.

```
CloudSim.startSimulation();
```

10. Εάν δεν υπάρχει κάποιο γεγονός προς εκτέλεση, τότε στέλνεται η εντολή για τερματισμό της προσομοίωσης.

```
CloudSim.stopSimulation();
```

11. Τέλος, εκτυπώνεται στην οθόνη το τελικό αποτέλεσμα της προσομοίωσης.

```
List<Cloudlet> newList = broker.getCloudletReceivedList();
printCloudletList(newList);
```

CloudsimExample1: Παρουσιάζει τη δημιουργία ενός κέντρου δεδομένων με ένα host και την εκτέλεση ενός cloudlet σε αυτό. Μόλις το τρέξουμε, παρατηρούμε ότι ο χρήστης μπορεί να διακρίνει το Cloudlet ID, το οποίο σε αυτή την περίπτωση είναι 0, την κατάσταση εκτέλεσης αυτού του Cloudlet που είναι επιτυχής, το ID του κέντρου δεδομένων και το ID της εικονικής μηχανής, ακόμα τον χρόνο έναρξης (start time), τον χρόνο λήξης (finish time) και τον συνολικό χρόνο εκτέλεσης του Cloudlet (time).

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        400        0.1        400.1
CloudSimExample1 finished!
```

Σχήμα 3.6: Αποτελέσματα πρώτου παραδείγματος

CloudsimExample2: Παρουσιάζει τη δημιουργία ενός κέντρου δεδομένων με έναν κεντρικό υπολογιστή και τη λειτουργία δύο cloudlets σε αυτόν. Τα cloudlets εξακολουθούν να εκτελούνται στα VM με τις ίδιες απαιτήσεις MIPS. Παρατηρούμε ότι ο χρόνος εκτέλεσης κάθε cloudlet είναι περίπου διπλάσιος από τον χρόνο εκτέλεσης του πρώτου παραδείγματος. Αυτό είναι λογικό, καθώς έχουμε έναν κεντρικό υπολογιστή με δύο εικονικές μηχανές εγκατεστημένες σε αυτόν και δύο διεργασίες που εκτελούνται παράλληλα σε αυτές. Τα cloudlets θα χρειαστούν τον ίδιο χρόνο για να ολοκληρώσουν την εκτέλεσή τους.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        1000        0.1        1000.1
    1        SUCCESS        2            1        1000        0.1        1000.1
CloudSimExample2 finished!
```

Σχήμα 3.7: Αποτελέσματα δεύτερου παραδείγματος

CloudSimExample3: Παρουσιάζει την δημιουργία ενός κέντρου δεδομένων με δύο hosts και την εκτέλεση δύο cloudlets σε αυτό. Τα cloudlets εξακολουθούν να εκτελούνται στα VM με διαφορετικές προϋποθέσεις MIPS. Τα cloudlets θα λαμβάνουν μοναδικούς χρόνους για να ολοκληρώσουν την εκτέλεση βασιζόμενη στην απόδοση του VM.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    1        SUCCESS        2            1        80        0.1        80.1
    0        SUCCESS        2            0        160        0.1        160.1
CloudSimExample3 finished!
```

Σχήμα 3.8: Αποτελέσματα τρίτου παραδείγματος

CloudSimExample4: Παρουσιάζει την δημιουργία δύο κέντρων δεδομένων με ένα host ο καθένας και εκτέλεση δύο cloudlets σε αυτά. Τα cloudlets θα λαμβάνουν τον ίδιο χρόνο για να ολοκληρώσουν την εκτέλεση.

CloudSimExample5: Το πέμπτο παράδειγμα, παρουσιάζει τη δημιουργία δύο κέντρων δεδομένων με έναν host το καθένα και την εκτέλεση cloudlets δύο χρηστών σε αυτά.

CloudSimExample6: Παρουσιάζει την δημιουργία κλιμακούμενων προσομοιώσεων. Αυτό σημαίνει ποικίλους αριθμούς cloudlets καθώς και ποικίλους αριθμούς Vms.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        160        0.2        160.2
    1        SUCCESS        3            1        160        0.2        160.2
CloudSimExample4 finished!
```

Figure 3.9: Αποτελέσματα τέταρτου παραδείγματος

```
=============> User 4
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        160        0.1        160.1
=============> User 5
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        3            0        160        0.2        160.2
CloudSimExample5 finished!
```

Figure 3.10: Αποτελέσματα πέμτου παραδείγματος

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    4        SUCCESS        2            4            3        0.2            3.2
    16        SUCCESS        2            4            3        0.2            3.2
    28        SUCCESS        2            4            3        0.2            3.2
    5        SUCCESS        2            5            3        0.2            3.2
    17        SUCCESS        2            5            3        0.2            3.2
    29        SUCCESS        2            5            3        0.2            3.2
    6        SUCCESS        3            6            3        0.2            3.2
    18        SUCCESS        3            6            3        0.2            3.2
```

Figure 3.11: Αποτελέσματα έκτου παραδείγματος

CloudSimExample7: Το έβδομο παράδειγμα, παρουσιάζει τον τρόπο παύσης των προσομοιώσεων. Η προσομοίωση σταματάει προσωρινά για πέντε δευτερόλεπτα και προστίθεται η οντότητα broker όπου με την σειρά του δημιουργεί τις εικονικές μηχανές.

CloudSimExample8: Παρουσιάζει πώς να προσθέτετε οντότητες κατά τη διάρκεια της εκτέλεσης.

```
200.0: The simulation is paused for 5 sec


Adding: Broker_1
Broker_1 is starting...
200.0: Broker_1: Cloud Resource List received with 2 resource(s)
200.0: Broker_1: Trying to Create VM #100 in Datacenter_0
```

Figure 3.12: Αποτελέσματα έβδομου παραδείγματος

```
Adding: GlobalBroker_
GlobalBroker_ is starting...
200.0: GlobalBroker_: Cloud Resource List received with 2 resource(s)
200.0: GlobalBroker_: Trying to Create VM #100 in Datacenter_0
```

Figure 3.13: Αποτελέσματα όγδοου παραδείγματος

3.5 Προσομοιώσεις και αποτελέσματα

Αυτή η ενότητα δείχνει πώς λειτουργεί η προσομοίωση με διαφορετικές πολιτικές χρονοπρογραμματισμού για ένα δεδομένο σύνολο τυχαίων ροών εργασίας και ποια πολιτική χρονοπρογραμματισμού παρέχει καλύτερα αποτελέσματα όσον αφορά τον χρόνο αναμονής, τον χρόνο εκτέλεσης κάθε εργασίας, τον συνολικό χρόνο που χρειάζεται κάθε εικονική μηχανή για να ολοκληρώσει την εκτέλεση όλων των εργασιών, τον μέσο χρόνο αναμονής και τον μέσο χρόνο απόκρισης κάθε εικονικής μηχανής. Το ακόλουθο σενάριο βασίζεται σε αυτό το έγγραφο [34] που παρέχει λεπτομερείς πληροφορίες για να βοηθήσει τους αρχάριους να επιλέξουν την πλατφόρμα προσομοίωσης CloudSim και τις κατάλληλες προσεγγίσεις για την υπολογιστική νέφους, στις [35] και [36]. Κατά τη διεξαγωγή των προσομοιώσεων, έγιναν αρκετές αλλαγές σε σχέση με το αρχικό σενάριο για την επίτευξη των επιθυμητών αποτελεσμάτων. Για την αξιολόγηση των επιδόσεων του νέφους, οι προσομοιώσεις πραγματοποιήθηκαν σε Ubuntu 22.04 LTS (64-bit) με επεξεργαστή AMD® Ryzen 7 5700u, 4.3GHz με μνήμη 16GB και JDK 1.8.

Σενάριο προσομοίωσης

Για το σενάριο αυτό, διαμορφώσαμε το πειραματικό περιβάλλον CloudSim Example6 που παρέχει δυνατότητα κλιμακούμενης προσομοίωσης. Αρχικά, παρουσιάζουμε πώς μπορούμε να δημιουργήσουμε τυχαίους φόρτους εργασίας (cloudlets) που υπόκεινται σε προσομοίωση. Στη συνέχεια, μελετάμε τον αντίκτυπο των πολιτικών χρονοπρογραμματισμού των cloudlets, όπως Time-

Shared και SpaceShared, και εφαρμόζουμε τον αλγόριθμο Shortest Job First (SJF) για να διασφαλίσουμε ότι τα cloudlets που έχουν μικρότερο αριθμό εντολών εκτελούνται πρώτα.

Πίνακας 3.1: Ιδιότητες των VM

VM Id	**Image Size**	**Ram**	**MIPS**	**Bw**	**No. of CPUs**
0	10000	512	250	1000	1
1	10000	512	250	1000	1
2	10000	512	250	1000	1

Εντός της μεθόδου vmlist = createVM(brokerId, 3); cloudletList =createCloudlet(brokerId, 15); βλέπουμε ότι η λίστα vm θα δημιουργήσει 3 vms για το σκοπό αυτής της προσομοίωσης με τις ιδιότητες που καταγράφονται στον πίνακα 3.1. Επίσης, βλέπουμε τον αριθμό των cloudlets, τα οποία θα δημιουργηθούν. Επομένως, αυτό σημαίνει ότι θα δημιουργηθούν 15 cloudlets και αναμένεται να κατανεμηθούν σε 3 διαφορετικά vms.

Α. Προγραμματισμός φόρτου εργασίας για ένα σύνολο από cloudlets

Στο πρόγραμμά μας (CloudSimExaple6) αλλάζουμε το μήκος των εργασιών με τη βοήθεια ενός κώδικα που παράγει έναν τυχαίο αριθμό. Κάθε επιμέρους cloudlet θα έχει διαφορετικό μήκος. Προκειμένου να υπολογιστεί και η χρήση της CPU, το μοντέλο χρήσης του cloudlet έχει αλλάξει από UtilizationModelFull() σε UtilizationModelStochastic(). Η διαφορά για αυτό το μοντέλο χρήσης είναι ο τρόπος με τον οποίο λειτουργεί με το cloudlet. Η UtilizationModelFull() θα χρησιμοποιεί πάντα την CPU στο 100%, αλλά η UtilizationModelStochastic() θα δίνει την αναμενόμενη χρήση της CPU για κάθε φορά που καλείται αυτή η μέθοδος.

Ωστόσο, η πρόκληση που αντιμετωπίζουμε κατά την εφαρμογή των πόρων είναι ότι δεν μπορούμε να επαναλάβουμε το ίδιο συγκεκριμένο σύνολο τυχαίων ροών εργασίας. Επομένως, αυτό που μπορούμε να κάνουμε είναι να δημιουργήσουμε πρώτα τυχαίους αριθμούς για ένα σύνολο από cloudlets σε ένα αρχείο και αφού επιχειρούμε να εκτελέσουμε μια συγκεκριμένη ροή εργασίας, να συλλέξουμε τους τυχαίους αριθμούς από αυτό το αρχείο για να

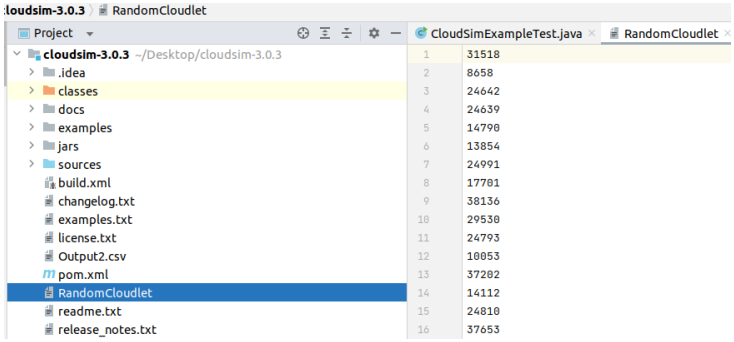
```
    UtilizationModel utilizationModel = new UtilizationModelStochastic();

    Cloudlet[] cloudlet = new Cloudlet[cloudlets];
    Random rnd = new Random();

    for(int i=0;i<cloudlets;i++){
        long newLen=rnd.nextInt( i: 4000);
        Log.printLine("The new length is: "+  (length+newLen));
        cloudlet[i] = new Cloudlet(i, (length+newLen), pesNumber, fileSize, outputSize,
                new UtilizationModelStochastic(), utilizationModel, utilizationModel);
        // setting the owner of these Cloudlets
        cloudlet[i].setUserId(userId);
        list.add(cloudlet[i]);
    }

    return list;
}
```

```
Starting CloudSimExample6...
Initialising...
The new length is: 3220
The new length is: 2797
The new length is: 3816
The new length is: 2806
The new length is: 2243
The new length is: 4305
The new length is: 1345
The new length is: 3880
The new length is: 2910
The new length is: 1820
The new length is: 3474
The new length is: 2711
The new length is: 3734
The new length is: 2948
The new length is: 2254
```

Σχήμα 3.14: Αποτελέσματα έκτου παραδείγματος με νέα μήκη cloudlets

υλοποιήσουμε την προσομοίωση χρησιμοποιώντας τους. Πρώτα δημιουργούμε ένα σύνολο τυχαίων αριθμών μεταξύ του εύρους 2000-40000 και τους μεταφέρουμε σε ένα αρχείο .txt με το όνομα αρχείου RandomCloudlet.

Δημιουργούμε μια μέθοδο που θα επιστρέφει την ίδια τιμή πίσω στο clouldlet. Στη συνέχεια δημιουργούμε μια συνάρτηση για να προσπελάσουμε τους αριθμούς από το αρχείο και να τους χρησιμοποιήσουμε για να βρούμε το νέο μήκος του clouldlet. Το cloudletcount μειώνεται κάθε φορά, επειδή έχουμε προσθέσει 400 αριθμούς στο RandomCloudlet και θέλουμε να εκτελέσουμε μόνο 15. Στο παράδειγμά μας προσθέτουμε τις ακόλουθες γραμμές κώδικα.




```java
//create a method because we want to return the same value back to the clouldlet
1 usage
private static ArrayList<Integer> getNumberValue(int cloudletcount){ //number is 400
    ArrayList<Integer> rnd = new ArrayList<Integer>(); //Create a list of integers.
   //to see which path is created
    Log.printLine(System.getProperty("user.dir")+ "/RandomCloudlet"); //will get the directory
    // and then be added to the path
    // specify which file will be used
    File myfile= new File( s: System.getProperty("user.dir")+"/RandomCloudlet");
    try {
        // read the file
        Scanner readFile= new Scanner(myfile);
        while (readFile.hasNextLine()&& cloudletcount>0){ //checks if the given file has a next line ,
            // so each time the next value
            // must be taken as an integer and placed in the list and the cloudlet number must be greater than zero
            //reading the value the txt
            rnd.add(readFile.nextInt());
            cloudletcount--; //counter reduction,the number of cloudlet must be reduced
            // each time because we know that there
            // are 400 numbers and because we want to apply it to 15, we have to run only 15 numbers out of 400
        }
        readFile.close(); //close the scanner
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
    return rnd; //return the array list
}
```

Σε αυτό το σημείο πρέπει να αντιληφθούμε το γεγονός ότι αυτός ο φόρτος εργασίας μπορεί να επαναλαμβάνεται ξανά και ξανά. Έτσι, εάν επιθυμούμε να δοκιμάσουμε ένα συγκεκριμένο σενάριο όπου θέλουμε να δημιουργήσουμε ένα σύνολο τυχαίων τιμών για τα cloudlets, ώστε να μπορέσουμε να δούμε τη διακύμανση στην έξοδο, πρέπει να εξετάσουμε αυτό το είδος σεναρίου.

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Starting CloudSimExample6...
Initialising...
/home/meggie/Desktop/cloudsim-3.0.3/RandomCloudlet
The new final length of the cloudlet is :32518
The new final length of the cloudlet is :9658
The new final length of the cloudlet is :25642
The new final length of the cloudlet is :25639
The new final length of the cloudlet is :15790
The new final length of the cloudlet is :14854
The new final length of the cloudlet is :25991
The new final length of the cloudlet is :18701
The new final length of the cloudlet is :39136
The new final length of the cloudlet is :30530
The new final length of the cloudlet is :25793
```

Σχήμα 3.15: Αποτελέσματα έκτου παραδείγματος με τα τελικά μήκη των cloudlets

Β. Εφαρμογή πολιτικών Timeshared, Spaceshared και Shortest Job First

Αφού εκτελέσαμε το πρόγραμμά μας και λάβαμε τα νέα μεγέθη των cloudlets, παρατηρούμε ότι τα Vms ξεκινούν την ίδια στιγμή (start time) στο 0,1, παρόλο που τα cloudlets είναι διαφορετικά. Επομένως, από αυτό το αποτέλεσμα μπορούμε να θεωρήσουμε ότι αυτό το μοντέλο ενδεχομένως λειτουργεί σε timeshared (Σχήμα 3.16). Στην πολιτική χρονοπρογραμματισμού timeshared, επειδή όλες οι εργασίες προγραμματίζονται ταυτόχρονα στην εικονική μηχανή, ο χρόνος αναμονής, δηλαδή ο χρόνος που έπρεπε να περιμένει κάθε cloudlet πριν ξεκινήσει, είναι 0.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time    Cloudlet length    Waiting time    CPU utilization
    1        SUCCESS        2             1          193.14        0.1          193.24           9658            0                92.34 %
    11        SUCCESS        2             2          221.04        0.1          221.14           11053           0                29.08 %
    13        SUCCESS        2             1          280.41        0.1          280.51           15112           0                80.70 %
    5        SUCCESS        2             2          281.87        0.1          281.97           14854           0                90.12 %
    4        SUCCESS        2             1          288.55        0.1          288.65           15790           0                88.57 %
    7        SUCCESS        2             1          311.84        0.1          311.94           18701           0                80.30 %
    10        SUCCESS        2             1          340.21        0.1          340.31           25793           0                58.33 %
    2        SUCCESS        2             2          411.32        0.1          411.42           25642           0                34.44 %
    14        SUCCESS        2             2          412.66        0.1          412.76           25810           0                83.11 %
    8        SUCCESS        2             2          465.97        0.1          466.07           39136           0                21.72 %
    3        SUCCESS        2             0          512.77        0.1          512.87           25639           0                56.65 %
    6        SUCCESS        2             0          518.4         0.1          518.5            25991           0                4.99 %
    9        SUCCESS        2             0          572.87        0.1          572.97           30530           0                55.16 %
    0        SUCCESS        2             0          588.77        0.1          588.87           32518           0                63.22 %
    12        SUCCESS        2             0          611.51        0.1          611.61           38202           0                44.68 %
CloudSimExample6 finished!
total time required = 9497213
```

Σχήμα 3.16: Αποτελέσματα έκτου παραδείγματος RR

Στη συνέχεια, στο πρόγραμμά μας αλλάζουμε την πολιτική χρονοπρογραμματισμού από timeshared σε spaceshared για να δούμε τη βασική διαφορά vm[i] = new Vm(i,userId,mips,pesNumber,ram,bw, size, vmm,new

CloudletSchedulerSpaceShared()); Επομένως, αφού χρησιμοποιήσουμε τον αλγόριθμο first come first serve παρατηρούμε την κατανομή των cloudlets, το cloudlet με ID=0 είναι το πρώτο που εξυπηρετείται. Αν επανεξετάσουμε την έξοδο για το Vm με ID=0, το cloudlet με ID=0 ξεκίνησε στο χρονικό διάστημα 0,1 και τελείωσε στο 130,17 (Σχήμα 3.17). Αυτή η χρονική κλίμακα, από τη στιγμή που ξεκίνησε έως τη στιγμή που τελείωσε, εξαρτάται από τη διάρκεια κάθε εργασίας και τα mips της εικονικής μηχανής. Επομένως, αν παρατηρήσουμε την κατανομή των cloudlets στα vms, διαπιστώνουμε ότι μια εργασία προγραμματίζεται κάθε φορά σε μια εικονική μηχανή, όταν αυτή η εργασία ολοκληρωθεί, μια άλλη εργασία προγραμματίζεται στην εικονική μηχανή και οι υπόλοιπες εργασίες μπαίνουν στην ουρά (Σχήμα 3.17).

```
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
0.2: Broker: Sending cloudlet 2 to VM #2
0.2: Broker: Sending cloudlet 3 to VM #3
0.2: Broker: Sending cloudlet 4 to VM #4
0.2: Broker: Sending cloudlet 5 to VM #5
0.2: Broker: Sending cloudlet 6 to VM #6
0.2: Broker: Sending cloudlet 7 to VM #7
```

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time    Cloudlet length    Waiting time    CPU utilization    user id
    1        SUCCESS        2            1        38.63       0.1           38.73           9658            0                51.13 %            4
    4        SUCCESS        2            1        63.16       38.73          101.89          15790           38.63               39.58 %
    2        SUCCESS        2            2        102.57      0.1           102.67          25642           0                58.60 %            4
    0        SUCCESS        2            0        130.07      0.1           130.17          32518           0                43.58 %            4
    5        SUCCESS        2            2        59.42       102.67         162.08          14854           102.57              36.53 %
    7        SUCCESS        2            1        74.8        101.89         176.7           18701           101.79              70.82 %
    3        SUCCESS        2            0        102.56      130.17         232.73          25639           130.07               2.69 %
    10        SUCCESS        2            1        103.17      176.7          279.87          25793           176.6               88.88 %
    8        SUCCESS        2            2        156.54      162.08         318.63          39136           161.98               71.47 %
    6        SUCCESS        2            0        103.96      232.73         336.69          25991           232.63               20.53 %
    13        SUCCESS        2            1        60.45       279.87         340.32          15112           279.77               70.96 %
    11        SUCCESS        2            2        44.21       318.63         362.84          11053           318.53               81.67 %
    9        SUCCESS        2            0        122.12      336.69         458.81          30530           336.59               9.41 %
    14        SUCCESS        2            2        103.24      362.84         466.08          25810           362.74               56.61 %
    12        SUCCESS        2            0        152.81      458.81         611.62          38202           458.71               54.02 %
CloudSimExample6 finished!
total time required = 8464624
```

Σχήμα 3.17: Αποτελέσματα έκτου παραδείγματος FCFS

Για να υλοποιήσουμε τον αλγόριθμο Shortest Job First [36] τροποποιούμε το CloudSimExample6 ως εξής:
Αρχικά, προσθέτουμε τη νέα λίστα των cloudlets, στη συνέχεια δημιουργούμε μια μέθοδο όπου τα cloudlets ταξινομούνται με βάση το μήκος και στη συνέχεια υποβάλλουμε τη λίστα των δημιουργημένων cloudlets στον broker.

```java
private static void getCloudletListSJF(List<Cloudlet> cloudletList1)
{
    int min=0;
    for (int i=0; i<cloudletList1.size();i++)
        if (cloudletList1.get(i).getCloudletLength() < cloudletList1.get(min).getCloudletLength())
            min=i;
    cloudletListSJF.add(cloudletList1.get(min));
    cloudletList1.remove(min);
    if (cloudletList1.size()!=0)
        getCloudletListSJF(cloudletList1);
}
```

Στο Σχήμα 3.18 παρουσιάζεται η έξοδος της προσομοίωσης SJF. Όπως βλέπουμε ο αλγόριθμος λειτουργεί με το μήκος της εργασίας. Όταν ένα VM είναι διαθέσιμο, ο SJF υποβάλλει στο VM την εργασία με το μικρότερο μήκος στην ουρά. Εάν δύο ή περισσότερες εργασίες με το ίδιο μήκος βρίσκονται στην ουρά, τότε εφαρμόζεται το FCFS. Τεχνικά, ο SJF δίνει προτεραιότητα στις εργασίες με βάση το μέγεθός τους, καθιστώντας τον έναν αλγόριθμο δυναμικής εξισορρόπησης φορτίου.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time    Cloudlet length    Waiting time    CPU util
    1        SUCCESS        2            0          38.63        0.1           38.73            9658            0                48.
    11        SUCCESS        2            1          44.21        0.1           44.31            11053           0                 8
    5        SUCCESS        2            2          59.42        0.1           59.52            14854           0                48
    13        SUCCESS        2            0          60.45        38.73         99.18            15112           38.63
    4        SUCCESS        2            1          63.16        44.31         107.47           15790           44.21
    7        SUCCESS        2            2          74.8         59.52         134.32           18701           59.42
    3        SUCCESS        2            0          102.56       99.18         201.74           25639           99.08
    2        SUCCESS        2            1          102.57       107.47        210.04           25642           107.37
    10        SUCCESS        2            2          103.17       134.32        237.49           25793           134.22
    14        SUCCESS        2            0          103.24       201.74        304.98           25810           201.64
    6        SUCCESS        2            1          103.96       210.04        314              25991           209.94
    9        SUCCESS        2            2          122.12       237.49        359.61           30530           237.39
    0        SUCCESS        2            0          130.07       304.98        435.05           32518           304.88
    12        SUCCESS        2            1          152.81       314           466.81           38202           313.9
    8        SUCCESS        2            2          156.54       359.61        516.16           39136           359.51
CloudSimExample6 finished!
```

Σχήμα 3.18: Αποτελέσματα έκτου παραδείγματος SJF

Γ. Makespane, μέσος χρόνος απόκρισης και μέσος χρόνος αναμονής των VMs

Για να λάβουμε το συνολικό χρόνο που απαιτείται για την ολοκλήρωση όλων των εργασιών (makespane), το μέσο χρόνο αναμονής και το μέσο χρόνο απόκρισης των VM προσθέτουμε τις ακόλουθες γραμμές κώδικα [36] στο CloudsimExample6.

```
private static double VmAvergeResponseTime (List<Cloudlet> list, int VmId)
{
    int c = 0;
    double avgt = 0;
    for(int i=0;i<list.size();i++)
        if (list.get(i).getVmId() == VmId)
        {
            avgt = avgt + list.get(i).getExecStartTime();    c++;
        }
    avgt =  avgt / c;
    return avgt;
}
1 usage
private static double VmAvergeWatingTime (List<Cloudlet> list, int VmId)
{
    int c = 0;
    double awt = 0;
    for(int i=0;i<list.size();i++)
        if (list.get(i).getVmId() == VmId)
        {
            awt = awt + list.get(i).getWaitingTime();    c++;
        }
    awt =  awt / c;
    return awt;
}


private static double VmMakespane(List<Cloudlet> list, int VmId)
{
    double makespane = 0;
    for(int i=0;i<list.size();i++)
        if (list.get(i).getVmId() == VmId)
            if (list.get(i).getFinishTime() > makespane)
                makespane=  list.get(i).getFinishTime();
    return makespane;
}
```

```
Log.printLine("CloudSimExample6 finished!");
System.out.println("Total time required = "+(end-start));

for (int a=0; a<vmlist.size();a++)
    Log.printLine("Average Response Time of Vm :" + vmlist.get(a).getId() +
            "   =  " + VmAvergeResponseTime( newList, vmlist.get(a).getId()));

for (int a=0; a<vmlist.size();a++)
    Log.printLine("Makespane of Vm :" + vmlist.get(a).getId() +
            "   =  " + VmMakespane( newList, vmlist.get(a).getId()));

for (int a=0; a<vmlist.size();a++)
    Log.printLine("Average Waiting Time of Vm :" + vmlist.get(a).getId() +
            "   =  " + VmAvergeWatingTime( newList, vmlist.get(a).getId()));
```

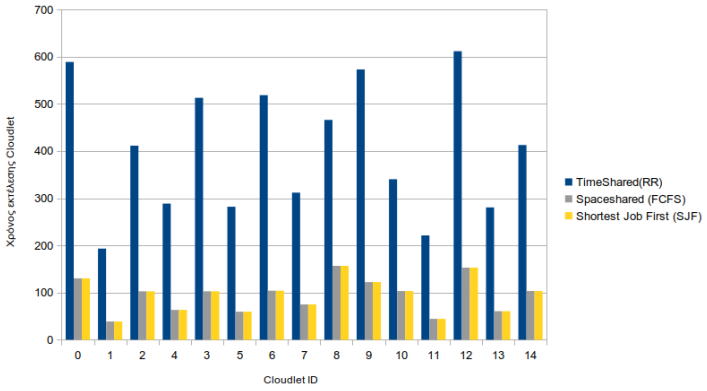
```
Average execution Time = 94.51440000000001
Average waiting Time = 140.67946666666666
CloudSimExample6 finished!
total time required = 9407019
Average Response Time of Vm :0   =  128.9448
Average Response Time of Vm :1   =  135.1856
Average Response Time of Vm :2   =  158.208
Makespane of Vm :0   =  435.048
Makespane of Vm :1   =  466.812
Makespane of Vm :2   =  516.1560000000001
Average Waiting Time of Vm :0   =  128.8448
Average Waiting Time of Vm :1   =  135.0856
Average Waiting Time of Vm :2   =  158.108
```

Σχήμα 3.19: Makespane, μέσος χρόνος απόκρισης και μέσος χρόνος αναμονής των VM

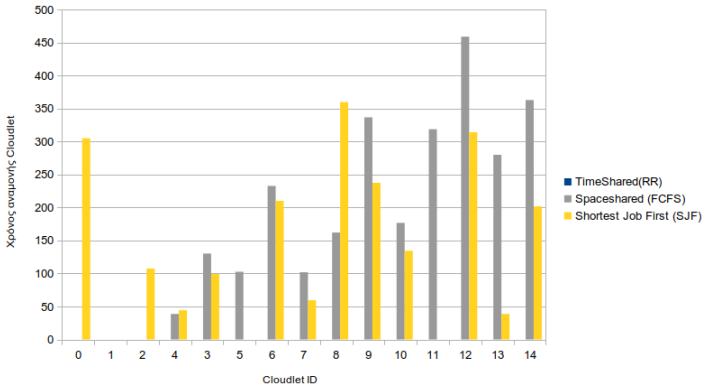
Αποτελέσματα και ανάλυση

Στο Σχήμα 3.20 παρουσιάζεται το γράφημα που συγκρίνει το συνολικό χρόνο εκτέλεσης κάθε cloudlet μεταξύ των αλγορίθμων TimeShared (RR), Space-Shared (FCFS) και JSF. Αυτό το γράφημα δείχνει ότι ο SJF έχει, με μικρή διαφορά από την πολιτική spaceshared, τον χαμηλότερο χρόνο εκτέλεσης. Ο υψηλότερος χρόνος εκτέλεσης είναι για την πολιτική time-shared (RR). Στο Σχήμα 3.21 παρουσιάζεται το γράφημα ως προς τον χρόνο αναμονής. Η πολιτική time-shared (RR) έχει τον μικρότερο χρόνο αναμονής επειδή όλες οι εργασίες εκτελούνται ταυτόχρονα. Η πολιτική spaceshared (FCFS) έχει τον μεγαλύτερο χρόνο αναμονής επειδή οι εργασίες εκτελούνται με βάση το cloudlet που μπήκε πρώτο στην ουρά. Όσο μεγαλύτερος είναι ο χρόνος αναμονής, τόσο μεγαλύτερος είναι ο χρόνος αναμονής που πρέπει να περιμένει ο χρήστης για την εκτέλεση του cloudlet, κάτι που δεν είναι αποδοτικό αν υπ-

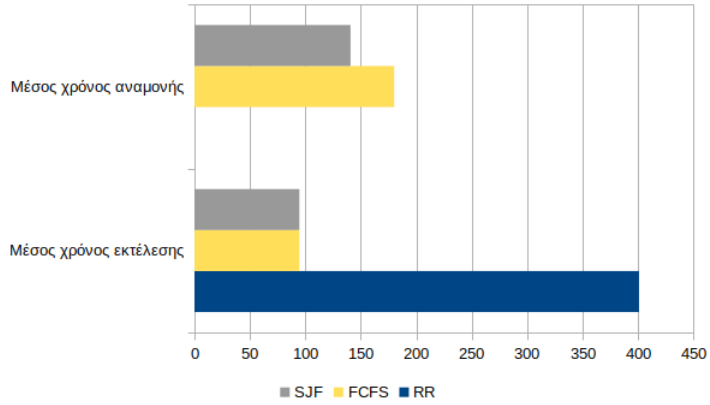
άρχουν πολλά cloudlets. Στο Σχήμα 3.22 παρουσιάζεται ο μέσος χρόνος αναμονής και ο μέσος χρόνος εκτέλεσης των cloudlets. Παρόλο που το RR έχει τον χαμηλότερο μέσο χρόνο αναμονής, έχει τον υψηλότερο χρόνο εκτέλεσης. Το SJF έχει τον δεύτερο χαμηλότερο μέσο χρόνο αναμονής και παρόμοιο μέσο χρόνο εκτέλεσης με το FCFS. Στο Σχήμα 3.23 παρουσιάζεται ο μέσος χρόνος απόκρισης που προέκυψε κατά τη διάρκεια της προσομοίωσης από κάθε Vm. Από τα αποτελέσματα μπορεί να παρατηρηθεί ότι, η SJF παρέχει καλύτερο χρόνο σε σύγκριση με την FCFS στα Vm 0 και Vm 2. Αυτό οφείλεται στο γεγονός ότι έχουμε περισσότερο χρόνο αναμονής στην πολιτική διαμοιρασμού χώρου. Στο Σχήμα 3.24 παρουσιάζεται ο μέσος χρόνος αναμονής των VM, δηλαδή ο χρόνος που κάθε VM θα πρέπει να περιμένει πριν εκτελέσει μια εργασία. Από τα αποτελέσματα παρατηρούμε ότι υπάρχει λιγότερη αναμονή με το JFS σε σύγκριση με την πολιτική spaceshared. Στην πολιτική με διαμοιρασμό χρόνου δεν υπάρχει καθυστέρηση. Στο Σχήμα 3.25 παρουσιάζεται ο χρόνος που χρειάζεται κάθε VM για να ολοκληρώσει τις εργασίες που του έχουν ανατεθεί. Από τα αποτελέσματα παρατηρείται ότι οι πολιτικές spaceshared και timeshared χρειάζονται περίπου τον ίδιο χρόνο για την εκτέλεση των εργασιών στα VM 0 και 1.



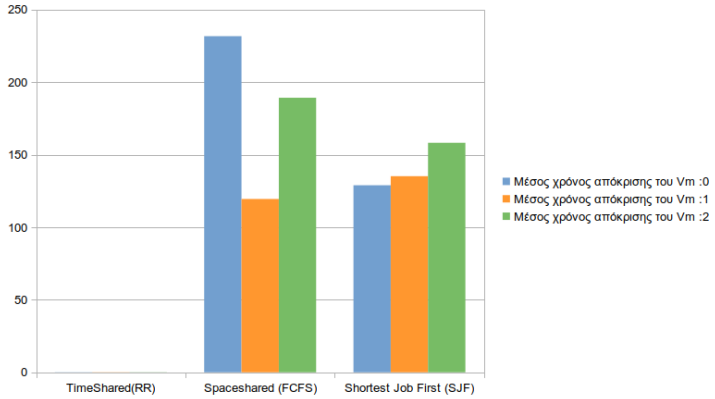
Σχήμα 3.20: Σύγκριση ως προς χρόνο εκτέλεσης



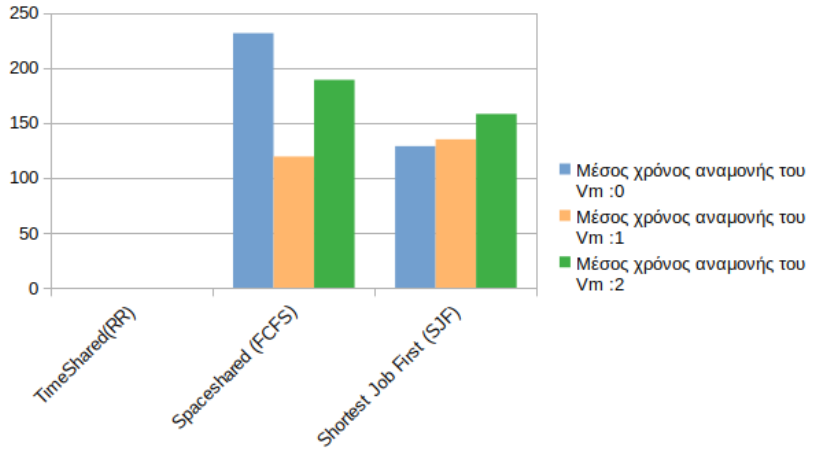
Σχήμα 3.21: Σύγκριση ως προς χρόνο αναμονής



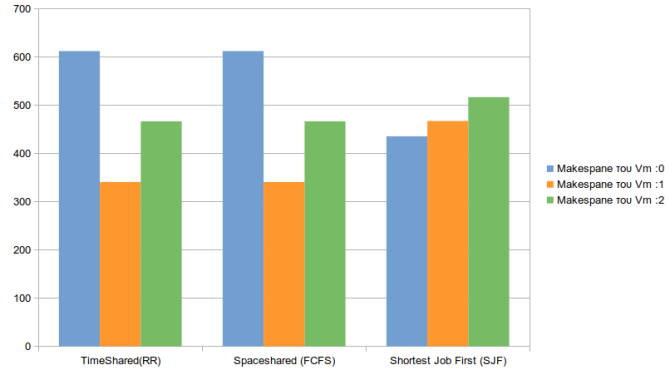
Σχήμα 3.22: Σύγκριση ως προς τον μέσο χρόνο αναμονής και τον μέσο χρόνο εκτέλεσης



Σχήμα 3.23: Σύγκριση ως προς τον χρόνο απόκρισης των VM



Σχήμα 3.24: Σύγκριση ως προς τον χρόνο αναμονής των VM



Σχήμα 3.25: Σύγκριση ως προς makespan των VM

Κεφάλαιο 4

GreenCloud

Το GreenCloud σχετίζεται με τον τομέα των κέντρων δεδομένων με ενεργειακή επίγνωση. Τα κέντρα δεδομένων στα οποία απευθύνεται το GreenCloud σχετίζονται κυρίως με υπηρεσίες υπολογιστικού νέφους. Τα πλεονεκτήματα του προσομοιωτή είναι οι δυνατότητες μοντελοποίησης σε επίπεδα κατανάλωσης ενέργειας που σχετίζονται με τον τεχνολογικό εξοπλισμό ενός κέντρου δεδομένων. Ο εξοπλισμός που μοντελοποιεί το GreenCloud περιλαμβάνει διακομιστές, συνδέσεις επικοινωνίας και μεταγωγείς δικτύου.

4.1 Αρχιτεκτονικές κέντρων δεδομένων

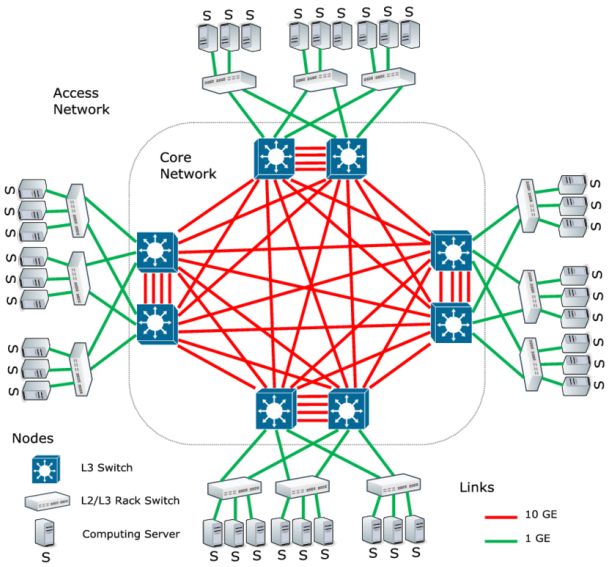
Τα κέντρα δεδομένων είναι εγκαταστάσεις που περιέχουν κυρίως ηλεκτρονικό εξοπλισμό που χρησιμοποιείται για την επεξεργασία δεδομένων (διακομιστές), την αποθήκευση δεδομένων (εξοπλισμός αποθήκευσης) και την επικοινωνία (εξοπλισμός δικτύου). Τα διάφορα στοιχεία του κέντρου δεδομένων παράγουν σημαντική ποσότητα θερμότητας. Ως εκ τούτου, τα κέντρα δεδομένων είναι εξοπλισμένα με μονάδες κλιματισμού και ψύξης, καθώς είναι απαραίτητο να παραμένουν διαθέσιμα και αξιόπιστα. Από τότε που αυξήθηκε το κόστος τροφοδοσίας και ψύξης, η βελτίωση της ενεργειακής απόδοσης αποτελεί μείζον ζήτημα στο υπολογιστικό νέφος [37]. Το ποσοστό της ενέργειας που καταναλώνεται αυξάνεται κάθε χρόνο και για αυτόν τον λόγο οι πάροχοι υποδομών δέχονται τεράστια πίεση για τη μείωση της κατανάλωσης ενέργειας. Αυτό μπορεί να αντιμετωπιστεί με διάφορες προσεγγίσεις. Μια προσέγγιση εί-

ναι να προταθούν μηχανισμοί χρονοπρογραμματισμού με επίγνωση της ενέργειας που επιτρέπουν την απενεργοποίηση των αδρανών πόρων. Ωστόσο, η μεγάλη πρόκληση είναι ο τρόπος επίτευξης ενός καλού συμβιβασμού μεταξύ της κατανάλωσης ενέργειας και της απόδοσης. Υπάρχει ανάγκη για τεχνικές που μπορούν να διαχειριστούν τον συνεχώς αυξανόμενο όγκο εργασίας διατηρώντας παράλληλα ένα αποδεκτό επίπεδο επιδόσεων,το οποίο αναφέρεται ως επεκτασιμότητα [37].

4.1.1 Αρχιτεκτονική κέντρου δεδομένων δύο και τριών επιπέδων

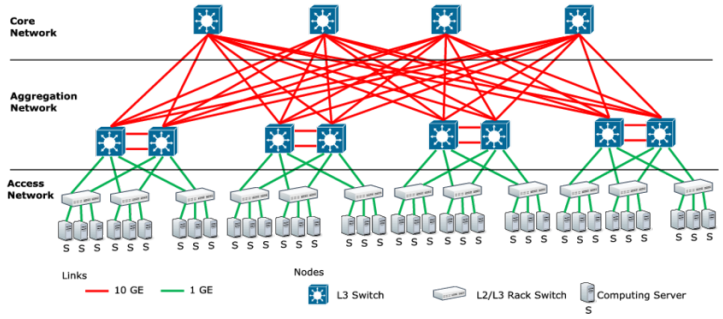
Το Σχήμα 4.1 παρουσιάζεται η δομή της αρχιτεκτονικής του κέντρου δεδομένων 2 επιπέδων [37]. Οι εξυπηρετητές/servers (S) κατανέμονται σε racks στο δίκτυο πρόσβασης. Τα switches επιπέδου 3 (L3 Switch) στο δίκτυο πυρήνα προσφέρουν πλήρη διασύνδεση πλέγματος χρησιμοποιώντας συνδέσεις 10 GE. Η αρχιτεκτονική δυο επιπέδων έχει λειτουργήσει ικανοποιητικά σε παλαιότερα κέντρα δεδομένων στα οποία οι διακομιστές ήταν περιορισμένου αριθμού. Τα κέντρα δεδομένων δυο επιπέδων μπορούν να υποστηρίξουν έως και 5500 κόμβους ανάλογα με το είδος των switches στο δίκτυο πρόσβασης. Η πιο συνηθισμένη αρχιτεκτονική κέντρου δεδομένων είναι η αρχιτεκτονική τριών επιπέδων που παρουσιάζεται στο Σχήμα 4.2. Περιλαμβάνει τρία επίπεδα: το επίπεδο πρόσβασης (access networrk), το επίπεδο συνάθροισης (aggregation network) και το επίπεδο πυρήνα (core network). Η παρουσία του μεσαίου επιπέδου επιτρέπει την αύξηση του αριθμού των κόμβων των διακομιστών έως και πάνω από 10.000 διακομιστές, ενώ ταυτόχρονα διατηρούνται οι μεταγωγείς στο δίκτυο πρόσβασης, και αυτό προσφέρει μια τοπολογία χωρίς βρόχους. Ένα τυπικό μοντέλο αρχιτεκτονικής τριών επιπέδων περιλαμβάνει 8 μεταγωγείς στο δίκτυο πυρήνα. Αυτή η αρχιτεκτονική διαθέτει οκτώ δρομολογήσεις πολλαπλών διαδρομών ίσου κόστους (ECMP) που περιέχουν 10 GE Line Aggregation Groups (LAGs) [37].

Αυτό το μοντέλο επιτρέπει σε έναν πελάτη να έχει πρόσβαση σε διαφορετικές συνδέσεις χρησιμοποιώντας την ίδια διεύθυνση MAC. Το LAG είναι μια τεχνολογία για την αύξηση της χωρητικότητας των συνδέσεων επικοινωνίας, αλλά η απόδοση και η ευελιξία του δικτύου είναι το κύριο μειονέκτημα, επιπλέον, στο κεντρικό δίκτυο εφαρμόζεται μια πλήρης συνδεσιμότητα πλέγματος, στην οποία απαιτείται μεγάλη ποσότητα καλωδίωσης. Οι αρχιτεκτονικές υψηλής ταχύτητας τριών επιπέδων προτείνονται για τη βελτιστοποίηση των υπολογιστικών κόμβων ωστόσο, οι χωρητικότητες των κόμβων επικοινωνίας (



Σχήμα 4.1: Η αρχιτεκτονική δύο επιπέδων (two-tier) [26]

δίκτυο πυρήνα και δίκτυο συνάθροισης) αποτελούν σημείο συμφόρησης, επειδή περιορίζουν τον αριθμό των κόμβων στο κέντρο δεδομένων. Η παρουσία συνδέσμων διασύνδεσης 100 GE μεταξύ των δικτύων πυρήνα και συνάθροισης, μειώνει τον αριθμό των μεταγωγέων πυρήνα, μειώνει τις καλωδιώσεις και αυξάνει εξαιρετικά το μέγιστο μέγεθος του κέντρου δεδομένων λόγω φυσικών περιορισμών. Ένας μικρός αριθμός δρομολόγησης πολλαπλών διαδρομών ίσου κόστους (ECMP) θα ενισχύσει την απόδοση και την ευελιξία. Ωστόσο, υπάρχουν και άλλες αρχιτεκτονικές που είναι πιο προηγμένες, όπως η Bcube και η Dcell στις οποίες εφαρμόζεται μια δομή δικτύου με επίκεντρο τον διακομιστή [38]. Οι διακομιστές απαιτούν ένα συγκεκριμένο πρωτόκολλο δρομολόγησης για να εξασφαλίσουν την ανοχή σε σφάλματα, επειδή αυτοί εκτελούν τη δρομολόγηση. Σε καμία από αυτές τις δύο αρχιτεκτονικές δεν εφαρμόζεται το επίπεδο πυρήνα ή το επίπεδο συνάθροισης και προσφέρεται επεκτασιμότητα σε τεράστιο αριθμό διακομιστών.



Σχήμα 4.2: Η αρχιτεκτονική τριών επιπέδων (three-tier) [26]

4.2 Προσομοίωση κέντρου δεδομένων

Όπως είναι γνωστό, ένα μεγάλο μέρος της συνολικής ποσότητας ενέργειας που καταναλώνεται από ένα τυπικό κέντρο δεδομένων (δύο ή τριών επιπέδων) χρησιμοποιείται για: 1) τη συντήρηση των συνδέσεων διασύνδεσης και των λειτουργιών του εξοπλισμού δικτύου (δρομολογητές και μεταγωγείς) και 2) τη συντήρηση των στοιχείων επεξεργασίας (επεξεργαστές, μνήμες και δίσκοι αποθήκευσης). Ωστόσο, η υπόλοιπη ενέργεια χάνεται (για την ακρίβεια, σπαταλιέται) στο σύστημα διανομής ενέργειας (ιδίως στα καλώδια). Αυτό το μέρος της ενέργειας: 1) διαχέεται ως θερμότητα ή 2) καταναλώνεται από τις μονάδες κλιματισμού (AC) [26].
Το GreenCloud διακρίνει τρεις συνιστώσες κατανάλωσης ενέργειας:
α) κατανάλωση υπολογιστικής ενέργειας
β) κατανάλωση επικοινωνιακής ενέργειας και
γ) ενέργεια υποδομής του κέντρου δεδομένων.
Ως επεκτασιμότητα νοείται η ικανότητα ενός συστήματος να ανταποκρίνεται και να αποδίδει όταν ο φόρτος εργασίας αυξάνεται ή επεκτείνεται, όπως αναφέρεται στην ενότητα 4.1. Το σύστημα στη συγκεκριμένη περίπτωση είναι το κέντρο δεδομένων με όλα τα στοιχεία του (καλώδια, στοιχεία επεξεργασίας, συσκευές αποθήκευσης, συσκευές δικτύου και άλλα) [37]. Η "ποσότητα εργασίας" αποτελείται από:
1) Φορτίο υπολογισμών: αντιπροσωπεύεται από το ποσό των υπολογισμών που απαιτούνται από τις εισερχόμενες εργασίες.
2) Φορτίο επικοινωνιών: αντιπροσωπεύεται από το ποσό των επικοινωνιών που απαιτούνται από τις εισερχόμενες εργασίες, τον αριθμό των μηνυμάτων που πρέπει να αποσταλούν από κάθε εργασία καθώς και το μέγεθος αυτών των μηνυμάτων.
3) Φορτίο εργασιών: αντιπροσωπεύεται από τον αριθμό των εισερχόμενων ερ-

γασιών.
4) Φορτίο δεδομένων: Αντιπροσωπεύεται από το μέγεθος των δεδομένων που μεταφέρονται από και προς ένα στοιχείο επεξεργασίας από την εργασία που εκτελείται στο συγκεκριμένο στοιχείο επεξεργασίας.

Μοντέλο υπολογιστικά εντατικών φόρτων εργασίας (CIWs): παράγει μεγάλο υπολογιστικό φόρτο στους διακομιστές, αλλά δεν απαιτεί σχεδόν καθόλου μεταφορά δεδομένων μέσω του δικτύου. Η σημασία του μοντέλου είναι η επίλυση υπολογιστικού φόρτου εργασίας και μια από τις κύριες ανησυχίες του είναι η κατανάλωση ενέργειας του κέντρου δεδομένων. Αυτό το μοντέλο CIW έχει τα ακόλουθα χαρακτηριστικά:

- Φορτώνει δύσκολους υπολογισμούς στους διακομιστές.
- Δεν απαιτεί σχεδόν καθόλου μεταφορές δεδομένων στο δίκτυο διασύνδεσης του κέντρου δεδομένων.
- Η ενεργειακά αποδοτική διαδικασία χρονοπρογραμματισμού CIW εστιάζει στη κατανάλωση ενέργειας του διακομιστή. Αυτό επιτυγχάνεται με την ομαδοποίηση των φόρτων εργασίας στο ελάχιστο σύνολο υπολογιστικών διακομιστών και τη δρομολόγηση της παραγόμενης κίνησης χρησιμοποιώντας ένα ελάχιστο σύνολο συσκευών δικτύου.
- Η θέση των περισσότερων switches σε κατάσταση αναστολής λειτουργίας θα εξασφαλίσει τη χαμηλότερη ισχύ του κέντρου δεδομένων.

Φορτία εργασίας έντασης δεδομένων (DIWs): δεν παράγουν σχεδόν καθόλου υπολογιστικό φορτίο στους διακομιστές, αλλά απαιτούν βαριές μεταφορές δεδομένων (μέσω της δικτυακής υποδομής του κέντρου δεδομένων). Ένα παράδειγμα DIWs είναι οι εφαρμογές διαμοιρασμού αρχείων βίντεο, όπου κάθε απλό αίτημα χρήστη μετατρέπεται σε διαδικασία ροής βίντεο. Αυτό το μοντέλο έχει τα ακόλουθα χαρακτηριστικά:

- Το δίκτυο διασύνδεσης και όχι η υπολογιστική χωρητικότητα γίνεται το σημείο συμφόρησης του κέντρου δεδομένων για τις DIWs.
- Ιδανικά, θα πρέπει να εφαρμόζεται μια συνεχής ανατροφοδότηση μεταξύ των switches και του κεντρικού χρονοπρογραμματιστή δικτύου φόρτου εργασίας.

- Με βάση αυτή την ανατροφοδότηση, ο χρονοπρογραμματιστής φόρτου δικτύου θα πρέπει να κατανέμει τους φόρτους εργασίας στους διαθέσιμους μεταγωγείς λαμβάνοντας υπόψη τα τρέχοντα επίπεδα συμφόρησης των συνδέσεων επικοινωνίας. Δηλαδή, θα αποφεύγει την αποστολή φορτίων εργασίας μέσω υπερφορτωμένων συνδέσμων, ακόμη και αν η υπολογιστική ικανότητα ορισμένου διακομιστή επιτρέπει την υποδοχή του φορτίου εργασίας. Η προσέγγιση αυτή υπόσχεται εξισορρόπηση της κυκλοφορίας στο δίκτυο του κέντρου δεδομένων και, ως εκ τούτου, μείωση του μέσου χρόνου που απαιτείται για την παράδοση μιας εργασίας από τους κεντρικούς μεταγωγείς στους υπολογιστικούς διακομιστές.

Ισορροπημένα φορτία εργασίας (BWs): Σε αυτό το μοντέλο, οι εφαρμογές έχουν τόσο απαιτήσεις υπολογισμού όσο και μεταφοράς δεδομένων. Τα BWs φορτώνουν αναλογικά τους διακομιστές υπολογιστών καθώς και τις συνδέσεις επικοινωνίας. Παράδειγμα εφαρμογών BWs είναι τα γεωγραφικά συστήματα πληροφοριών που απαιτούν τόσο μεγάλες μεταφορές γραφικών δεδομένων όσο και βαριά επεξεργασία χαρτών. Κατά συνέπεια, ο προγραμματισμός των BWs πρέπει να λαμβάνει υπόψη τόσο το φορτίο των εξυπηρετητών όσο και το φορτίο του δικτύου διασύνδεσης.

Η αποδοτικότητα ενός κέντρου δεδομένων μπορεί να οριστεί από την πλευρά της απόδοσης που παρέχεται ανά watt, αυτό μπορεί να ποσοτικοποιηθεί και να μετρηθεί με τις ακόλουθες δύο μετρήσεις:
(I) Αποτελεσματικότητα χρήσης ισχύος (PUE) και
(II) Αποδοτικότητα υποδομής κέντρου δεδομένων (DciE).
Τόσο η PUE όσο και η DCiE περιγράφουν ποιο μέρος της συνολικά καταναλισκόμενης ενέργειας αποδίδεται στους διακομιστές υπολογιστών [26].

4.2.1 Η αρχιτεκτονική του GreenCloud

Κατά βάση, ο αρχιτεκτονικός σχεδιασμός του GreenCloud βασίζεται στην αρχιτεκτονική των κέντρων δεδομένων τριών επιπέδων. Τα στοιχεία του διακομιστή στο GreenCloud υλοποιούν κόμβους, οι οποίοι έχουν 1) περιορισμένη ταχύτητα επεξεργασίας (μετρούμενη σε MIPS εκατομμύρια εντολές ανά δευτερόλεπτο), 2) περιορισμένο μέγεθος της κοινής μνήμης και του αποθηκευτικού χώρου και 3) διαθέτουν διαφορετικούς μηχανισμούς χρονοπρογραμματισμού εργασιών που κυμαίνονται από τον απλό round-robin έως τους εξελιγμένους μηχανισμούς DVFS και DNS [26]. Οι διακομιστές είναι τοπο-

θετημένοι σε racks με ένα Top-of-Rack (ToR) μεταγωγέα που τον συνδέει με το τμήμα πρόσβασης του δικτύου. Το μοντέλο ενέργειας των στοιχείων του διακομιστή εξαρτάται από την κατάσταση του διακομιστή και τη χρήση της CPU. Ένας αδρανής διακομιστής καταναλώνει περίπου το 66% της ενέργειας σε σύγκριση με έναν πλήρως χρησιμοποιημένο διακομιστή. Αυτό οφείλεται στο γεγονός ότι οι διακομιστές πρέπει να διαχειρίζονται μονάδες μνήμης, δίσκους, πόρους εισόδου/εξόδου και άλλα σε μια αποδεκτή κατάσταση. Τότε η κατανάλωση ενέργειας αυξάνεται γραμμικά μαζί με τα επίπεδα φόρτου της CPU. Ως αποτέλεσμα, το προαναφερθέν μοντέλο επιτρέπει την υλοποίηση της εξοικονόμησης ενέργειας σε έναν κεντρικό χρονοπρογραμματιστή που μπορεί να παρέχει την ενοποίηση των φορτίων εργασίας σε ένα ελάχιστο δυνατό αριθμό υπολογιστικών διακομιστών [26]. Μια άλλη λύση για τη διαχείριση ενέργειας είναι η δυναμική κλιμάκωση τάσης/συχνότητας (Dynamic Voltage/Frequency Scaling – DVFS). Το DVFS χρησιμοποιείται επίσης για τη μείωση της ενέργειας στα κέντρα δεδομένων. Αποτελεί μια τεχνική διαχείρισης ισχύος σε συστήματα υπολογιστών, μια τεχνική για την προσαρμογή της τάσης και της συχνότητας μιας υπολογιστικής μονάδας με σκοπό τον έλεγχο της κατανάλωσης ενέργειας. Η λειτουργία εξοικονόμησης ενέργειας στο GreenCloud βασίζεται στο πρότυπο DVFS. Το DVFS επιτρέπει έναν συμβιβασμό μεταξύ της υπολογιστικής απόδοσης και της ενέργειας που καταναλώνει ο διακομιστής. Το DVFS βασίζεται στο γεγονός ότι η ισχύς μεταγωγής σε ένα τσιπ μειώνεται αναλογικά, όπως ορίζεται στην εξίσωση (1):

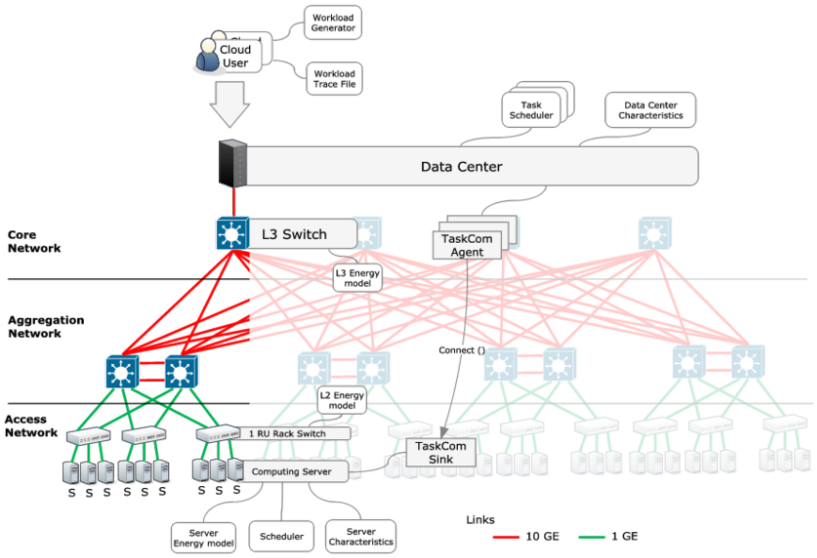
$$P = P_f + f^3.P_{fixed}$$

όπου P_{fixed} είναι το ποσοστό της κατανάλωσης ενέργειας που δεν μεταβάλλεται με τη συχνότητα λειτουργίας f , ενώ P_f είναι η καταναλισκόμενη από τη συχνότητα ισχύς της CPU. Ο αριθμός των εγκατεστημένων μεταγωγέων εξαρτάται από την αρχιτεκτονική του κέντρου δεδομένων. Παρόλα αυτά, καθώς οι υπολογιστικοί διακομιστές είναι συνήθως τοποθετημένοι σε racks, ο πιο συνηθισμένος διακόπτης σε ένα κέντρο δεδομένων είναι ο Top-of-Rack (ToR) διακόπτης. Ο διακόπτης ToR τοποθετείται συνήθως στην επάνω μονάδα της rack unit (1RU) για να μειωθεί η ποσότητα των καλωδίων και η παραγόμενη θερμότητα. Όπως και στους υπολογιστικούς διακομιστές, οι αρχικές προτάσεις βελτιστοποίησης της ενέργειας για το δίκτυο διασύνδεσης βασίστηκαν σε συνδέσεις DVS. Το DVS εισάγει ένα στοιχείο ελέγχου σε κάθε θύρα του switch, το οποίο ανάλογα με το ρυθμό κυκλοφορίας και τα τρέχοντα επίπεδα χρήσης της ζεύξης μπορεί να μειώσει το ρυθμό μετάδοσης. Ωστόσο,η αποδοτικότητα της ισχύος των συνδέσεων με δυνατότητα DVS

είναι ελεγχόμενη, καθώς μόνο το 3-15% της καταναλισκόμενης ενέργειας κλιμακώνεται γραμμικά με το ρυθμό σύνδεσης. Η ενέργεια που καταναλώνεται από έναν διακόπτη και όλους τους πομποδέκτες του ορίζεται στην (2):

$$P_{switch} = P_{chassis} + n_{linecards} + p_{linecard} + \sum_{i=0}^{R} n_{potrs,r} + P_r$$

όπου $P_{chassis}$ σχετίζεται με την κατανάλωση ενέργειας από το υλικό του switch,$P_{linecard}$ είναι η ενέργεια που καταναλώνεται από οποιαδήποτε ενεργή κάρτα γραμμής δικτύου και Pr αντιστοιχεί στην ενέργεια που καταναλώνεται από μια θύρα που λειτουργεί με ρυθμό r.



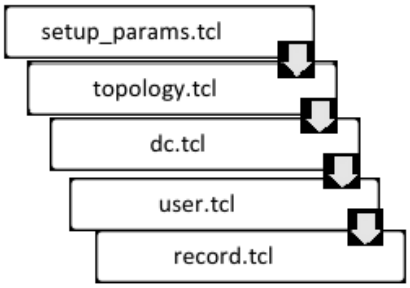
Σχήμα 4.3: Η αρχιτεκτονική του GreenCloud [26]

Ο πηγαίος κώδικας του προσομοιωτή GreenCloud [39] περιλαμβάνει τις ακόλουθες κύριες κατηγορίες:
(i) TskObject ορίζει εργασίες που αντιπροσωπεύουν έναν ενιαίο φόρτο εργασίας ο οποίος εκτελείται στο DC,
(ii) οι κλάσεις TskComAgent και TskComSink είναι κλάσεις agent πρωτοκόλλου που τμηματοποιούν μια εργασία σε πακέτα IP και εκτελούν την παράδοσή τους,
(iii) η κλάση CloudUser είναι η γονική κλάση όλων των γεννητριών εργασιών φόρτου εργασίας και υποστηρίζει τη δημιουργία εργασιών,

(iv) η ExpCloudUser υλοποιεί τη δημιουργία αντικείμενων φόρτου εργασίας με εκθετικά κατανεμημένο χρόνο μεταξύ των αφίξεων,
(v) η κλάση CBRCloudUser υλοποιεί τη δημιουργία αντικειμένων φόρτου εργασίας με σταθερό ρυθμό μετάδοσης
(vi) η κλάση DcHost, στο DC, αυτή η κλάση είναι η κύρια κλάση που συντονίζει την εκτέλεση και τη διανομή του φόρτου εργασίας,
(vii) Οι υπολογιστικοί διακομιστές DcHost αντιπροσωπεύονται από αυτή την κλάση και λαμβάνουν έργο φόρτου εργασίας μέσω της
TxkComSink, προγραμματίζοντας την εκτέλεση του τοπικά. Διατηρεί επίσης συνεχώς έναν κατάλογο ενεργών εργασιών, (viii) SwitchEnergyModel παρακολουθεί την κατανάλωση ενέργειας των μεταγωγέων δικτύου ανάλογα με το φορτίο και την ενεργοποιημένη λειτουργία εξοικονόμησης ενέργειας.

4.3 Προσομοιώσεις και αποτελέσματα

Το GreenCloud περιλαμβάνει μια προεπιλεγμένη προσομοίωση δοκιμής με 144 διακομιστές και έναν χρήστη Cloud. Όλες οι παράμετροι μπορούν να αλλάξουν και να δοκιμαστούν με βάση τις εισόδους που δίνονται στο αρχείο Tcl. Τα αρχεία Tcl βρίσκονται στον κατάλογο greencloud/src/scripts/.

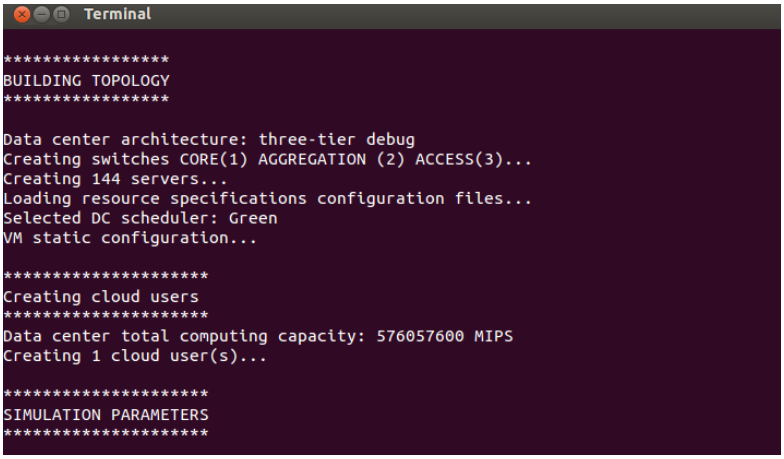


Σχήμα 4.4: Λειτουργία προσομοιωτή GreenCloud [40]

- setup-params.tcl - Για την τροποποίηση της γενικής διαμόρφωσης εργασιών, διακομιστών, διακοπτών και παρακολούθησης και μετάβασης διακοπτών .
- topology.tcl - Για τη δημιουργία της τοπολογίας του DC.
- dc.tcl - Για τη δημιουργία διακομιστών και VM του DC.

- user.tcl - Καθορισμός της συμπεριφοράς των χρηστών που χρησιμοποιούν το νέφος.
- record.tcl - Καταγραφή και δημιουργία διαδικασιών που θα χρησιμοποιηθούν για την αναπαράσταση των αποτελεσμάτων.
- finish.tcl: Υπολογίζει και παρουσιάζει τα στατιστικά της προσομοίωσης.

Η "run" είναι η προκαθορισμένη στο πακέτο εντολή, η οποία εκτελεί την προσομοίωση στο τερματικό σύμφωνα με τις δεδομένες παραμέτρους. Χρησιμοποιούμε την "show-dashboard.html" που είναι διαθέσιμη, η οποία είναι η ιστοσελίδα που εμφανίζει όλες τις εξόδους της προσομοίωσης. Τα δεδομένα εξόδου είναι προσβάσιμα με τη χρήση των αρχείων καταγραφής(./traces/)



Σχήμα 4.5: Στιγμιότυπο οθόνης μιας τυπικής προσομοίωσης στο GreenCloud

Παράμετροι προσομοίωσης

Κατά την πρώτη προσομοίωση, τροποποιούνται δύο αρχεία main.tcl (όπου ορίζονται οι πληροφορίες της προσομοίωσης) και topology.tcl (όπου ορίζεται η τοπολογία του δικτύου), τα οποία εξυπηρετούν 160 διακομιστές και ένα κέντρο δεδομένων νέφους ενός χρήστη με μέση χωρητικότητα φορτίου/εξυπηρετητή 0,3 (όπως φαίνεται στον πίνακα 3.2).

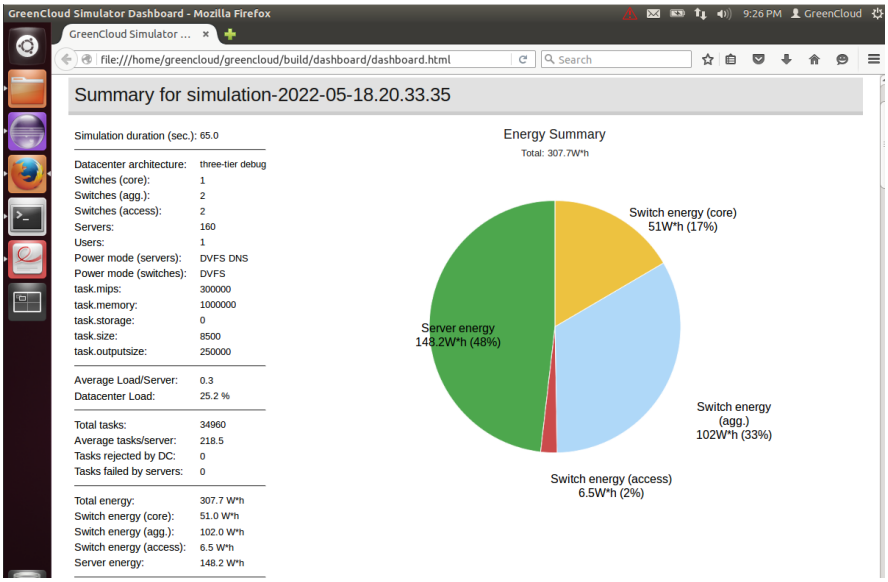
```
# ------------------------------------------------------------
# ------------- Main GreenCloud simulation script ------------
# ------------------------------------------------------------

# Type of DC architecture
set sim(dc_type) "three-tier debug";                    ;# can be "three-tier", "three-tier high-speed",
                                    # in case of heterogenous topologies make sure that VMs are not

# Set the time of simulation end
set sim(post_time) 0.6
set sim(end_time) [ expr 60.1 + [lindex $argv 1] ] ;# simualtion length set to 60 s + deadline of tasks

# Start collecting statistics
set sim(start_time) 0.1

set sim(tot_time) [expr $sim(end_time) - $sim(start_time)]

set sim(linkload_stats) "enabled"

# Set the interval time (in seconds) to make graphs and to create flowmonitor file
set sim(interval) 0.1

# Setting up main simulation parameters
source "setup_params.tcl"
```

```
# ------------------------------------------------------------
# ------------- Creating data center topology -------------
# ------------------------------------------------------------

# SWITCHES

switch $sim(dc_type) {
  "three-tier high-speed" {
    set top(NCore)              2                       ;# Number of L3 Switches in the CORE network
    set top(NAggr)              [expr 2*$top(NCore)]    ;# Number of Switches in AGGREGATION network
    set top(NAccess)            256                     ;# Number switches in ACCESS network per pod
    set top(NRackHosts)         3                       ;# Number of Hosts on a rack
  }
  "three-tier debug" {
    set top(NCore)              1                       ;# Number of L3 Switches in the CORE network
    set top(NAggr)              [expr 2*$top(NCore)]    ;# Number of Switches in AGGREGATION network
    set top(NAccess)            2                       ;# Number switches in ACCESS network per pod
    set top(NRackHosts)         80                      ;# Number of Hosts on a rack
  }
  "three-tier heterogenous debug" {
    set top(NCore)              1                       ;# Number of L3 Switches in the CORE network
    set top(NAggr)              [expr 2*$top(NCore)]    ;# Number of Switches in AGGREGATION network
    set top(NAccess)            3                       ;# Number switches in ACCESS network per pod
    set top(NRackHosts)         40                      ;# Number of Hosts on a rack
  }
  # three-tier
  default {
    set top(NCore)              8                       ;# Number of L3 Switches in the CORE network
    set top(NAggr)              [expr 2*$top(NCore)]    ;# Number of Switches in AGGREGATION network
    set top(NAccess)            64                      ;# Number switches in ACCESS network per pod
    set top(NRackHosts)         3                       ;# Number of Hosts on a rack
  }
}

# Number of racks
set top(NRacks) [expr $top(NAccess)*$top(NCore)]

# Number of servers is set to 2 * 80 (160 servers)
set top(NServers) [expr $top(NRacks)*$top(NRackHosts)]
```

Στην έξοδο του προσομοιωτή GreenCloud περιλαμβάνονται το φορτίο του κέντρου δεδομένων (ποσοστό), ο συνολικός αριθμός των υποβληθεισών εργασιών, ο μέσος όρος εργασιών ανά διακομιστή, ο αριθμός των εργασιών που απορρίφθηκαν από τον χρονοπρογραμματιστή του κέντρου δεδομένων, ο συνολικός αριθμός των αποτυχημένων εργασιών από τους διακομιστές και η κατανάλωση ενέργειας των διακομιστών.

Πίνακας 4.1: Στοιχεία προσομοίωσης

Αρχιτεκτονική κέντρου δεδομένων	Three tier debug
Core Switches	1
Aggregation Switches	2
Access Switches	2
Αριθμός εξυπηρετητών	160
Χρήστες	1
Μέσος φόρτος/εξυπηρετητή	0.3
Συνολικές εργασίες	34960
Μέσος όρος εργασιών/εξυπηρετητή	218.5
Συνολική υπολογιζόμενη ενέργεια	307.7 W*h
Ενέργεια εξυπηρετητή	148.2 W*h
Συνολική Ενέργεια Διακόπτη ("Switch")	159.5 W*h
Χρόνος προσομοίωσης	65



Σχήμα 4.6: Σύνοψη προσομοίωσης

Έπειτα, οι εργασίες που φτάνουν στο κέντρο δεδομένων κατανέμονται στους διακομιστές χρησιμοποιώντας έναν από τους δύο χρονοπρογραμματιστές: τον Green και τον Round Robin. Ο Round Robin είναι ένας απλός,

εύκολος στην υλοποίηση και ευρέως γνωστός χρονοπρογραμματιστής. Πρόκειται για έναν στατικό αλγόριθμο που αναθέτει εργασίες σε διακομιστές με κυκλικό τρόπο και κατανέμει τις εργασίες ισομερώς μεταξύ των διακομιστών, έτσι ώστε να λειτουργούν ταυτόχρονα όσοι διακομιστές χρειάζονται. Ο Green αναθέτει εργασίες στον μικρότερο δυνατό αριθμό διακομιστών, γεγονός που θα επιτρέψει την απενεργοποίηση των αδρανών διακομιστών για τη μείωση της ενέργειας που καταναλώνεται από αυτούς τους διακομιστές. Η συνολική υπολογιστική ικανότητα του κέντρου δεδομένων είναι 640064000 MIPS. Με βάση την εργασία της Dhillon [41] σε αυτό το σημείο θα τροποποιήσω τις συνολικές εργασίες σε σχέση με το αρχικό σενάριο προκειμένου να ελέγξω την απόδοση των δύο αλγορίθμων όσον αφορά τη συνολική ενέργεια και την ενέργεια που χρησιμοποιείται από τον διακομιστή.

Πίνακας 4.2: Τροποποίηση παραμέτρων

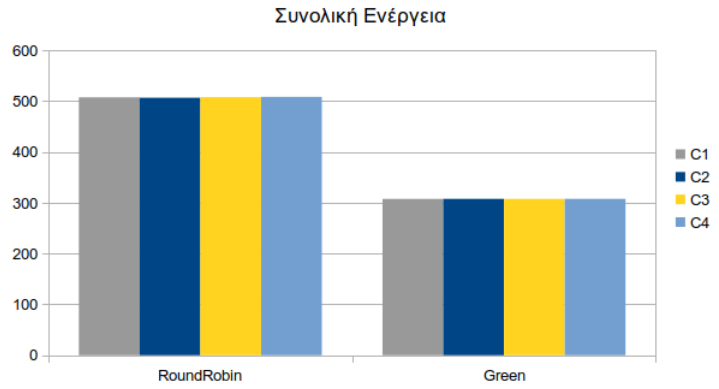
Αρίθμηση	**Αριθμός εργασιών**	**Αριθμός εντολών ανά εργασία (MIPS)**
C1	35258	600000
C2	48658	500000
C3	68050	400000
C4	99458	300000
C5	152687	200000

Οι παράμετροι που αναλύονται είναι ο συνολικός αριθμός των εργασιών, ο οποίος θα κυμαίνεται από 35258 έως 152687, και ο αριθμός εντολών ανά εργασία MIPS (που εκφράζεται σε εκατομμύρια εντολές το δευτερόλεπτο), οι οποίες κυμαίνονται από 600000 έως 200000. Το αρχείο που τροποποιείται είναι το setup_params.tcl.

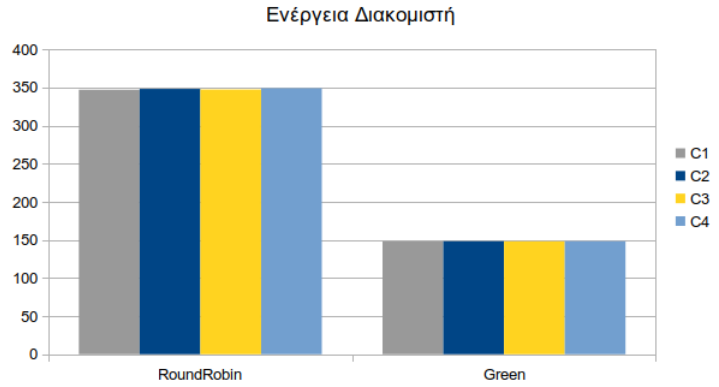
Αποτελέσματα και ανάλυση

Παρακάτω παρουσιάζεται γραφικά η διακύμανση της συνολικής ενέργειας και της ενέργειας του διακομιστή, συγκρίνοντας δύο αλγορίθμους, τον RoundRobin και τον Green, μεταξύ τους σύμφωνα με τις παραμέτρους του αριθμού των εργασιών και του αριθμού των εντολών ανά εργασία (MIPS). Από τα γραφήματα σύγκρισης παρατηρείται ότι η συνολική ενέργεια και η ενέργεια που καταναλώνεται από τους διακομιστές είναι λιγότερη με τη χρήση του Green. Ο Green αναθέτει εργασίες στον ελάχιστο αριθμό διακομιστών, επιτρέποντας έτσι τον τερματισμό των αδρανών διακομιστών. Η εφαρμογή του

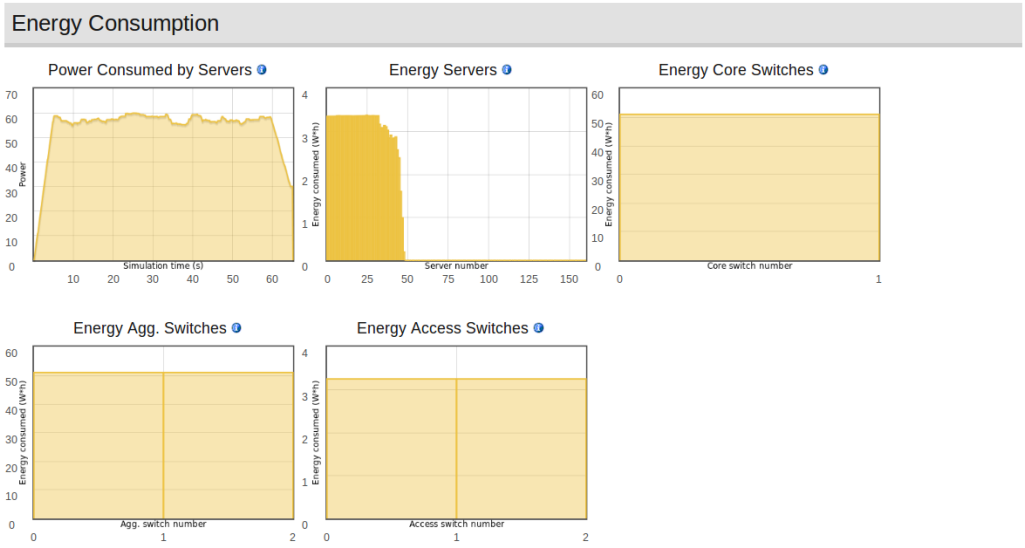
Round Robin έχει γενικά σχετικά πολύ χαμηλό αντίκτυπο στην κατανάλωση ενέργειας του DC. Αυτό οφείλεται στο γεγονός ότι ο RR κατανέμει τις εργασίες ομοιόμορφα στους διακομιστές, με αποτέλεσμα να μην υπάρχουν σχεδόν καθόλου αδρανείς διακομιστές.



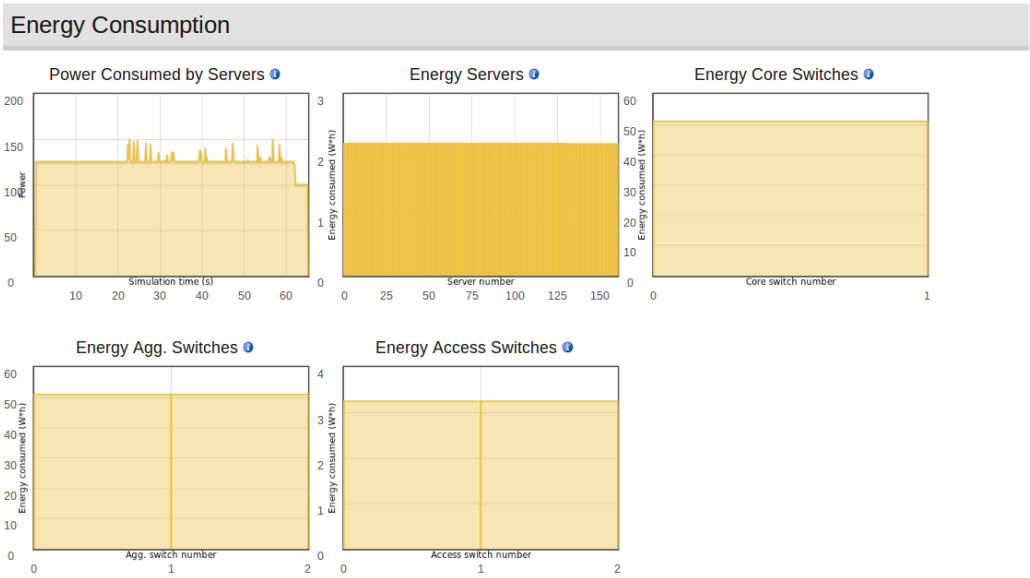
Σχήμα 4.7: Συγκριτική ανάλυση της ενέργειας που καταναλώνεται από τους διακομιστές στο DCN



Σχήμα 4.8: Συγκριτική ανάλυση της συνολικής ενέργειας που χρησιμοποιείται από το DCN



Σχήμα 4.9: Λεπτομέρειες κατανάλωσης ενέργειας με Green



Σχήμα 4.10: Λεπτομέρειες κατανάλωσης ενέργειας με RR

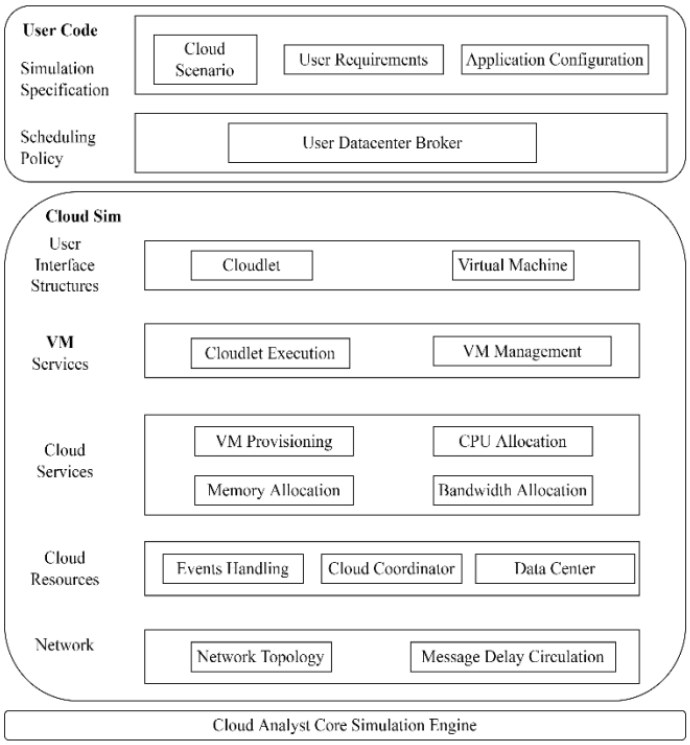
Κεφάλαιο 5

CloudAnalyst

Το CloudAnalyst σχεδιάστηκε από τον Wickremasinghe [18] για να βοηθήσει στην κατανόηση της λειτουργίας μιας μεγάλης διαδικτυακής εφαρμογής στο πλαίσιο ενός περιβάλλοντος νέφους. Βασίζεται στο CloudSim επεκτείνοντας τη λειτουργικότητά του με την εισαγωγή εννοιών που μοντελοποιούν τη συμπεριφορά του διαδικτύου και των διαδικτυακών εφαρμογών. Οι κλάσεις και οι νέες λειτουργίες που περιλαμβάνονται στο CloudAnalyst διευκολύνουν μια σειρά από εργασίες και διαδικασίες. Σε αυτές περιλαμβάνονται η προσομοίωση ενός ρεαλιστικού μοντέλου συμπεριφοράς, η καθυστέρηση δικτύου όσον αφορά τα δεδομένα που μεταδίδονται με τη διαχείριση των αιτημάτων των χρηστών μεταξύ περισσότερων από ένα κέντρων δεδομένων και η παροχή γραφικής διεπαφής χρήστη (GUI). Τα βασικά στοιχεία που συνθέτουν το CloudAnalyst περιγράφονται στα εξής: Region, Internet, Cloud Application Service Broker, User base, Internet cloudlet, Data center controller, VM load balancer και GUI. Στο Σχήμα 5.1 παρουσιάζεται η αρχιτεκτονική του προσομοιωτή που έχει αναπτυχθεί πάνω στον προσομοιωτή CloudSim.

5.1 Βασικά συστατικά στοιχεία

Region: Στο CloudAnalyst ο κόσμος έχει χωριστεί σε έξι περιοχές, οι οποίες καλούνται "Regions" και ταυτίζονται με τις έξι ηπείρους του πλανήτη. Οι κύριες οντότητες όπως οι User Bases και τα Data Centers, ανήκουν σε μια από τις προαναφερθείσες περιοχές. Αυτή η γεωγραφική ομαδοποίηση χρησιμοποιείται για να διατηρηθεί ένα επίπεδο ρεαλιστικής απλότητας για την προσομοίωση



Σχήμα 5.1: Αρχιτεκονική του CloudAnalyst [42]

μεγάλης κλίμακας που επιχειρείται στο CloudAnalyst.

User Base: Υποδεικνύει την ομάδα χρηστών που λαμβάνονται υπόψη ως μοναδικό στοιχείο στην προσομοίωση και δημιουργούν κίνηση, αυτοί μπορεί να είναι ένας χρήστης ή ένας μεγάλος αριθμός χρηστών.

Data Center Controller: Ο Data Center Controller είναι η πιο σημαντική οντότητα του CloudAnalyst. Ένας ελεγκτής κέντρου δεδομένων αντιστοιχεί σε ένα μόνο αντικείμενο cloudsim.DataCenter και χειρίζεται τις δραστηριότητες διαχείρισης του κέντρου δεδομένων, όπως η δημιουργία και η καταστροφή εικονικών μηχανών και η δρομολόγηση αιτημάτων που λαμβάνονται από βάσεις χρηστών, μέσω του Διαδικτύου στις εικονικές μηχανές.

Internet: Το διαδίκτυο στο CloudAnalyst είναι μια αφηρημένη έννοια του πραγματικού διαδικτύου και έχει υλοποιήσει μόνο εκείνα τα χαρακτηριστικά που είναι σημαντικά για την προσομοίωση. Μοντελοποιεί τη δρομολόγηση της κυκλοφορίας σε όλο τον κόσμο κάνοντας χρήση της κατάλληλης κα-

θυστέρησης μετάδοσης και των καθυστερήσεων μεταφοράς δεδομένων. Η καθυστέρηση μετάδοσης και το διαθέσιμο εύρος ζώνης μεταξύ των έξι περιοχών είναι παραμετροποιήσιμες τιμές.

VM Load Balancer: Χρησιμοποιείται στον ελεγκτή κέντρου δεδομένων για να επιλέξει το εικονικό μηχάνημα για να αναθέσει το επόμενο cloudlet προς επεξεργασία. Υπάρχουν τρεις διαφορετικοί ισσοροπιστές φόρτου(load balancers) που χρησιμοποιούνται: round robin, equally spread current execution load και throttled load balancer.

Internet Cloudlet: Ομαδοποίηση αιτημάτων από χρήστες. Τα αιτήματα μπορούν να ομαδοποιηθούν ως ένα ενιαίο internet cloudlet, με πληροφορίες όπως το μέγεθος της σειράς εκτέλεσης του αιτήματος και τα αρχεία εισόδου και εξόδου.

Service Broker: Ο ServiceBroker αναλαμβάνει την επιλογή της πολιτικής της κίνησης μεταξύ των κέντρων δεδομένων και των χρηστών. Το εργαλείο CloudAnalyst διαθέτει τρεις πολιτικές αναλύονται στην υποενότητα 5.4.

5.2 Δρομολόγηση αιτημάτων των χρηστών

Η δρομολόγηση αιτημάτων από τους χρήστες είναι μια σημαντική παράμετρος στο νέφος. Στο CloudAnalyst μοντελοποιείται αυτή η συμπεριφορά μέσω της χρήσης αναγνωριστικών εφαρμογών και της δρομολόγησης με χρήση του στοιχείου Internet, όπως περιγράφεται παρακάτω:
1. Η βάση των χρηστών δημιουργεί ένα InternetCloudlet, που περιέχει το Application ID για την εφαρμογή και το όνομα της βάσης του χρήστη για να μπορέσει να δρομολογήσει απάντηση όταν του ζητηθεί (RESPONCE).
2. REQUEST στέλνετε στο Internet με μηδενική καθυστέρηση.
3. Η βαθμίδα Internet συμβουλεύεται το ServiceBroker για την επιλογή των DataCenter. Ο ServiceBroker χρησιμοποιεί μια από τις διαθέσιμες πολιτικές βάση της πληροφορίας του αιτήματος(REQUEST).
4. Ο Service Broker στέλνει τις πληροφορίες για τα επιλεγμένα Data Center στο Internet.
5. Το Internet προσθέτει τα ανάλογα χαρακτηριστικά της γραμμής (καθυστέρηση δικτύου) στο αίτημα REQUEST και το στέλνει στο DataCenterController.
6. Ο DataCenter Controller χρησιμοποιεί κάποια από τις διαθέσιμες πολιτικές για την ανάθεση των αιτημάτων προς τις εικονικές μηχανές.

7. Ο VMLoadBalancer αναθέτει τα αιτήματα των χρηστών στα VM.
8. Το επιλεγμένο DC στέλνει απάντηση (RESPONCE) στο Internet μετά την επεξεργασία του αιτήματος REQUEST.
9. Το Internet χρησιμοποιεί τις πληροφορίες από το InternetCloudlet και προσθέτει την αντίστοιχη καθυστέρηση για το RESPONSE, ενημερώνοντας την UserBase.

5.3 Υπολογισμός καθυστέρησης μετάδοσης των δεδομένων

Ο υπολογισμός της συνολικής καθυστέρησης μετάδοσης υπολογίζεται με τον ακόλουθο τύπο:

$$T_{total} = T_{latency} + T_{transfer}$$

όπου $T_{latency}$είναι η καθυστέρηση του δικτύου και $T_{transferr}$είναι ο χρόνος που χρειάζεται για να μεταφερθεί η πληροφορία ενός αιτήματος του χρήστη από την αφετηρία μέχρι τον προορισμό. Η πληροφορία για την καθυστέρηση του δικτύου $T_{latency}$ υπάρχει στον πίνακα με τα χαρακτηριστικά του δικτύου (Internet characteristics), ενώ ο υπολογισμός του χρόνου μετάδοσης.

Υπολογίζετε το εύρος ζώνης για κάθε χρήστη $BW_{peruser}$

$$BW_{peruser} \frac{BW_{total}}{Nr}$$

Υπολογίζετε η καθυστέρηση μετάδοσης για κάθε αίτημα του χρήστη $T_{transfer}$.

$$T_{transfer} \frac{D}{BW_{peruser}}$$

Όπου BW_{total}: το διαθέσιμο εύρος ζώνης και δίνεται από τον πίνακα Internet Characteristics και Nr είναι ο αριθμός των αιτημάτων των χρηστών που μεταδίδονται.

5.4 Αλγόριθμοι επιλογής DataCenter

Αναφορικά με τα κριτήρια που χρησιμοποιούνται για την επιλογή του κέντρου δεδομένων που θα εξυπηρετεί τον χρήστη κατά τη διάρκεια της προσομοίωσης, το CloudAnalyst διαθέτει τρεις ενσωματωμένους αλγορίθμους για τον καθορισμό αυτής της πολιτικής (Service Broker Policies).

Δρομολόγηση με βάση το Service Proximity: Αυτός ο broker διατηρεί ένα αρχείο όλων των Datacenters ταξινομημένα με βάση την περιοχή τους. Όταν ο χρήστης που ανήκει στη βάση χρηστών στέλνει ένα μήνυμα, αναζητείται το πλησιέστερο Datacenter του Service Broker για τη διαδρομή Datacenter Controller. Το πλησιέστερο Datacenter του Service Broker απευθύνεται στην περιοχή του αιτούντος και ζητά για την περιοχή ένα πρόγραμμα κοντινής προσέγγισης για την εν λόγω περιοχή από τα χαρακτηριστικά του Διαδικτύου. Το πλησιέστερο κέντρο δεδομένων του Service Broker βρίσκει το πρώτο κέντρο δεδομένων που ταιριάζει με την πλησιέστερη περιοχή στο πρόγραμμα γειτνίασης [43].

Δρομολόγηση Performance Optimized: Αυτή η πολιτική υλοποιείται από την κλάση BestResponseTimeServiceBroker, που επεκτείνει την κλάση ServiceProximityServiceBroker. Ο BestResponseTimeServiceBroker διατηρεί έναν πίνακα περιεχομένων με όλα τα διαθέσιμα Data Centers. Όταν το στοιχείο Internet λάβει ένα μήνυμα από μια user base, τότε υποβάλλει ένα ερώτημα στον BestResponseTimeServiceBroker ζητώντας να μάθει τον παραλήπτη DataCenterController. Ο BestResponseTimeServiceBroker αναγνωρίζει το κοντινότερο (από πλευράς καθυστέρησης) data center χρησιμοποιώντας τον αλγόριθμο ServiceProximityServiceBroker. Τότε ο BestResponseTimeServiceBroker επαναλαμβάνει τη λίστα με όλα τα data centers και υπολογίζει τον τρέχοντα χρόνο απόκρισης σε κάθε data center. Εάν ο ελάχιστος εκτιμώμενος χρόνος είναι για το κοντινότερο data center, τότε ο BestResponseTimeServiceBroker το επιλέγει. Αλλιώς, ο BestResponseTimeServiceBroker επιλέγει είτε το κοντινότερο data center ή εκείνο με τον ελάχιστο χρόνο απόκρισης με πιθανότητα 50:50.

Δυναμικός αλγόριθμος Service Broker: Ο αλγόριθμος DynamicServiceBroker αποτελεί επέκταση των δύο αλγορίθμων που αναφέρθηκαν παραπάνω , του πλησιέστερου κέντρου δεδομένων και της δρομολόγησης βελτιστοποιημένης απόδοσης. Ο DynamicServiceBroker διατηρεί ένα χρονοδιάγραμμα των DCs και ένα άλλο χρονοδιάγραμμα για τον καλύτερο χρόνο που έχει καταχωρηθεί για κάθε DC.

5.5 Πολιτική εξισορρόπησης φορτίου στο CloudAnalyst

Η εξισορρόπηση φορτίου εφαρμόζεται για την επιτυχή αξιοποίηση των πόρων, τη βελτίωση του χρόνου απόκρισης και την αποφυγή καταστάσεων στις οποίες ορισμένα VM είναι πολύ φορτωμένα, ενώ άλλα είναι μόνο οριακά φορτωμένα. Στο CloudAnalyst τα κέντρα δεδομένων χρησιμοποιούν το VM-LoadBalancer και με αυτόν τον τρόπο μια οντότητα DC εξισορροπεί το φορτίο των αιτήσεων μεταξύ όλων των διαθέσιμων εικονικών μηχανών. Ο πρώτος αλγόριθμος round-robin είναι καλά κατανοητός από τις προσομοιώσεις με τα εργαλεία Cloudsim και GreenCloud και επομένως δεν αναλύεται. Οι επόμενοι δύο αλγόριθμοι Throttle και Active Monitoring εξηγούνται παρακάτω.

Αλγόριθμος Throttled: Ο αλγόριθμος throttled, ξεκινά με την κατανομή των κατάλληλων VMs όταν ο χρήστης στέλνει ένα αίτημα στον κατανεμητή φορτίου. Ο εξισορροπητής φορτίου (ThrottledVmLoadBalancer) διατηρεί έναν πίνακα αρχείων με όλα τα VM μαζί με τις καταστάσεις τους, κατάσταση κατειλημμένης ή ανοικτής λειτουργίας (BUSY/AVAILABLE). Στην αρχή, κάθε VM τίθεται σε διαθέσιμη κατάσταση. Κατόπιν, ο ελεγκτής διακομιστή καθοδηγεί τον εξισορροπητή για την επόμενη κατανομή VM, πριν λάβει άλλη αίτηση. Ο εξισορροπητής ελέγχει εξ ολοκλήρου τον πίνακα μέχρι να βρεθεί μια ουσιαστική αντιστοίχιση με ένα VM. Εφόσον βρεθεί το κατάλληλο VM, τότε ο εξισορροπητής επιστρέφει το ID του συγκεκριμένου VM στον ελεγκτή διακομιστή (DataCenterController). Αμέσως, ο ελεγκτής διακομιστή στέλνει μια αίτηση προς το VM με το συγκεκριμένο ID. Έκτοτε, ο ελεγκτής διακομιστή στέλνει μια προειδοποίηση για την εξισορρόπηση της εκ νέου κατανομής με στόχο την ανανέωση του πίνακα. Εάν υπάρχει κάποια περίπτωση, κατά την οποία το VM είναι ήδη διαθέσιμο, ο εξισορροπητής επιστρέφει με τις εν λόγω πληροφορίες του διακομιστή. Όταν το VM ολοκληρώσει την επεξεργασία της αίτησης, ο ελεγκτής διακομιστή λαμβάνει ένα cloudlet απάντησης και στέλνει ένα μήνυμα στον εξισορροπητή φορτίου για την αποδέσμευση του VM.

Αλγόριθμος Active Monitoring: Ο αλγόριθμος αυτός ονομάζεται επίσης εξισορρόπηση φορτίου τρέχουσας εκτέλεσης με ίση κατανομή. Χρησιμοποιεί ενεργό εξισορροπητή φορτίου παρακολούθησης για την ισόρροπη κατανομή της εκτέλεσης των φορτίων σε διαφορετικές εικονικές μηχανές. Τα βήματα αυτού του αλγορίθμου περιγράφονται ως εξής: Ο ενεργός εξισορροπητής φορτίου παρακολούθησης (ActiveVmLoadBalancer) διατηρεί έναν πίνακα ευρετηρίου των εικονικών μηχανών και τον αριθμό των κατανομών που αντισ-

τοιχούν σε κάθε εικονική μηχανή. Ο ελεγκτής κέντρου δεδομένων λαμβάνει (DataCenterController) ένα νέο αίτημα από έναν χρήστη-πελάτη. Οταν φτάνει στο ActiveVmLoadBalancer ένα αίτημα για κατανομή νέου VM από τον ελεγκτή κέντρου δεδομένων, αναλύει τον πίνακα ευρετηρίου από την αρχή μέχρι να βρει το λιγότερο φορτωμένο VM. Εφόσον το βρει, επιστρέφει το αναγνωριστικό VM στον ελεγκτή κέντρου δεδομένων. Εάν βρεθούν περισσότερα από ένα, ο ActiveVmLoadBalancer χρησιμοποιεί τη βάση FCFS (first come first serve) για να επιλέξει το λιγότερο φορτωμένο. Συγχρόνως, επιστέφει επίσης το αναγνωριστικό VM στον ελεγκτή κέντρου δεδομένων. Στη συνέχεια, ο DataCenterController ενημερώνει το VM που αναγνωρίζεται από αυτό το id. Ο ελεγκτής κέντρου δεδομένων ενημερώνει τον ActiveVmLoadBalancer για τη νέα κατανομή. Μετά από αυτό ο ActiveVmLoadBalancer ενημερώνει τον πίνακα κατανομής αυξάνοντας τον αριθμό κατανομής κατά 1 για το συγκεκριμένο VM. Όταν ένα VM ολοκληρώσει κατάλληλα την επεξεργασία του εκχωρημένου αιτήματος, προωθεί μια απάντηση στον ελεγκτή κέντρου δεδομένων. Με τη λήψη της απάντησης ειδοποιεί τον ActiveVmLoadBalancer για την αποδέσμευση του VM. Ο ActiveVmLoadBalancer ενημερώνει τον πίνακα κατανομής μειώνοντας τον αριθμό κατανομής για το συγκεκριμένο VM κατά 1.

5.6 Προσομοιώσεις και αποτελέσματα

Για αυτό το σενάριο, θα συγκρίνουμε τις πολιτικές Round Robin, Equally Spread Current Execution και Throttle που χρησιμοποιούνται για την εξισορρόπηση φορτίου στο CloudAnalyst και θα εφαρμόσουμε έναν άλλο αλγόριθμο, τον Threshold , όπου θα λάβουμε τα αποτελέσματα για τους χρόνους απόκρισης, τους χρόνους εξυπηρέτησης αιτημάτων του κέντρου δεδομένων και το κόστος της διαδικασίας. Το ακόλουθο σενάριο βασίζεται στο έγγραφο [44] με σημαντικές αλλαγές και ο κώδικας του αλγορίθμου Threshold που τοποθετείται στο CloudAnalyst αναπαράγεται από το εν λόγω πρότυπο [45] χωρίς οποιαδήποτε αλλαγή. Σύμφωνα με τον αλγόριθμο Threshold, η χρήση πόρων των VM παρακολουθείται και ενημερώνει αυτόματα τα κατώτατα όρια φόρτου εργασίας σε κάθε σημαντική αλλαγή. Εάν υπερβαίνει τις προκαθορισμένες τιμές κατωφλίου, τότε η χωρητικότητα των VM αυξάνεται ή μειώνεται δυναμικά ανάλογα με τις ανάγκες χωρίς να τερματίζονται τα VM, γεγονός που ελαχιστοποιεί τη σπατάλη πόρων. Για την εφαρμογή της νέας πολιτικής εξισορρόπησης φορτίου, τροποποιήθηκαν οι ακόλουθες κλάσεις για την προσομοίωση του προαναφερθέντος σεναρίου.

cloudsim.ext.gui.screens - ConfigureSimulationPanel - createAdvancedTab
cloudsim.ext - Constants
cloudsim.ext,datacenter - Datacentercontroller
cloudsim.ext.datacenter - Προσθήκη νέας κλάσης (NewVmloadBalancer). Αυτή η κλάση υλοποιεί τον VmLoadBalancer ως εξισορροπητή φορτίου Threshold. Αποτελείται από σταθερές τιμές tUnder και tUpper, ώστε να μπορεί να διακρίνει μεταξύ υποφορτωμένων και υπερφορτωμένων κόμβων. Αρχικά, όλοι οι επεξεργαστές θεωρούνται υποφορτωμένοι. Εάν ο κόμβος είναι υποφορτωμένος, τότε η διεργασία κατανέμεται τοπικά. Διαφορετικά, επιλέγεται ένας απομακρυσμένος υποφορτωμένος επεξεργαστής και αν δεν υπάρχει τέτοιος υποδοχέας, η διεργασία κατανέμεται επίσης τοπικά. Ένας κόμβος είναι υποφορτωμένος: αν (load $<$ tUnder), μέτρια φορτωμένος αν (tUnder: $\leq$ load $\leq$ tUpper) και υπερφορτωμένος: αν (load $>$ tUpper). Ο αλγόριθμος Threshold υλοποιεί τον CloudSimEventListener για να ενημερώνεται για τα VM που διατίθενται και απελευθερώνονται από τον DatacenterController.

Για την πρώτη δοκιμή προσομοιώσαμε το περιβάλλον λαμβάνοντας 5 βάσεις χρηστών και κάθε σύνολο αιτημάτων χρηστών αποτελείται από περίπου 400 υποεργασίες, τρία κέντρα δεδομένων, με 15 VM σε κάθε κέντρο δεδομένων. Υποθέσαμε ότι ο μέσος όρος των χρηστών αιχμής και ο μέσος όρος των χρηστών εκτός αιχμής αλλάζουν σε κάθε βάση χρηστών που ανήκει σε μια χρονική ζώνη. Κάθε προσομοίωση εκτελείται για 60 λεπτά και η πολιτική διαμεσολαβητή υπηρεσιών που χρησιμοποιείται είναι η πολιτική Closest Data Center (πλησιέστερο κέντρο δεδομένων). Για τη διεξαγωγή της δεύτερης δοκιμής, προσομοιώσαμε το περιβάλλον λαμβάνοντας 10 βάσεις χρηστών και κάθε σύνολο αιτημάτων χρηστών αποτελείται από 600 υποεργασίες. Τα DataCenters αυξήθηκαν από 3 σε 5 έχοντας 25 VMs σε κάθε DataCenter. Οι προαναφερθέντες αλγόριθμοι προσομοιώθηκαν για δύο περιβάλλοντα δοκιμών υπό διαφορετικές περιπτώσεις φόρτου εργασίας αιτημάτων χρηστών. Οι πίνακες 5.1 και 5.2 δείχνουν πώς διαμορφώθηκε το CloudAnalyst.

Πίνακας 5.1: Προδιαγραφές περιβάλλοντος νέφους κατά την πρώτη προσομοίωση

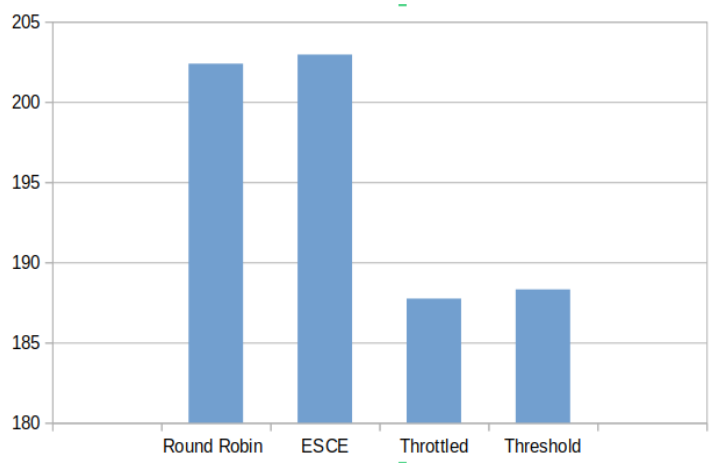
Παράμετροι	**Εύρος τιμών / Προδιαγραφές**
Αριθμός VMs	15
Αριθμός χρήσης περιοχών (Number of region use)	5
Αιτήματα ανά χρήστη / ώρα (Per User Request)	400
Virtual Machine Manager (VMM)	Xen
Λειτουργικό σύστημα (OS)	Linux
Αριθμός επεξεργαστών ανά κέντρο δεδομένων (Number of Processors per Data Centre)	8
Μήκος εντολών (Executable Instructions Length)	1000
Πολιτική διαμεσολαβητή (Service broker policy)	Closest Data Center

Πίνακας 5.2: Προδιαγραφές περιβάλλοντος νέφους κατά την δεύτερη προσομοίωση

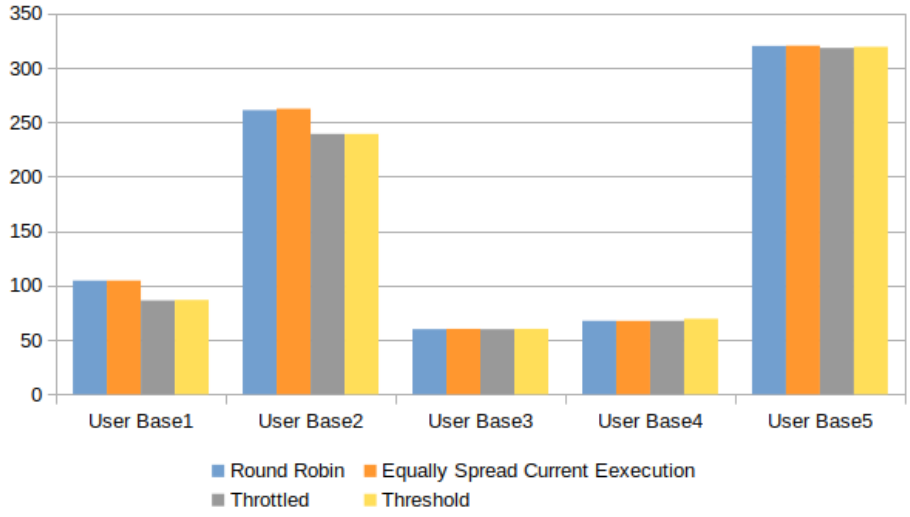
Παράμετροι	**Εύρος τιμών / Προδιαγραφές**
Αριθμός VMs	25
Αριθμός χρήσης περιοχών (Number of region use)	10
Αιτήματα ανά χρήστη / ώρα (Per User Request)	600
Virtual Machine Manager (VMM)	Xen
Λειτουργικό σύστημα (OS)	Linux
Αριθμός επεξεργαστών ανά κέντρο δεδομένων (Number of Processors per Data Centre)	8
Μήκος εντολών (Executable Instructions Length)	1000
Πολιτική διαμεσολαβητή (Service broker policy)	Closest Data Center

Αποτελέσματα και ανάλυση

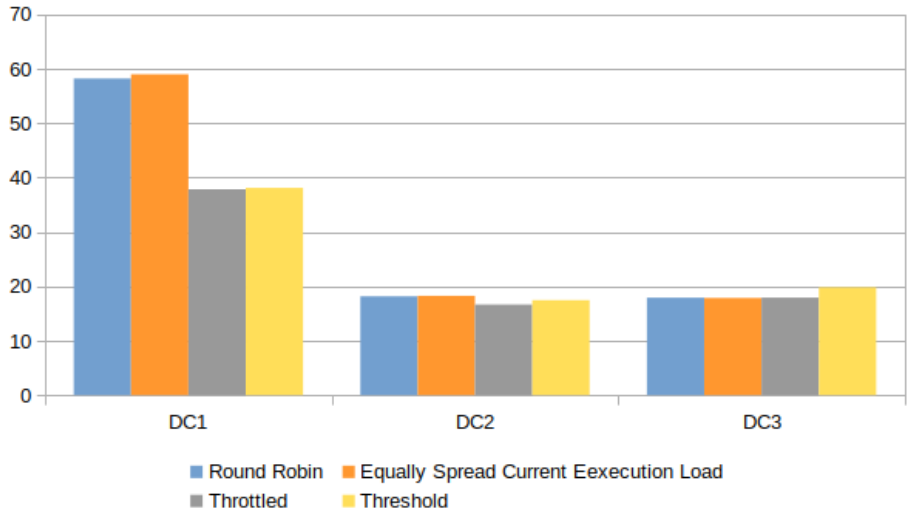
Τα αποτελέσματα για το συνολικό χρόνο απόκρισης για την πρώτη δοκιμή παρουσιάζονται στο Σχήμα 5.2 για τον αλγόριθμο Round robin, τον αλγόριθμο Equally Spread Current Execution, τον αλγόριθμο Throttle,και τον αλγόριθμο Thresshold. Με βάση τα αποτελέσματα της προσομοίωσης, διαπιστώθηκε ότι οι RR και ESCE έχουν σχεδόν τον ίδιο συνολικό χρόνο απόκρισης, ενώ στην περίπτωση της πολιτικής Throttled και Threshold υπάρχει σημαντική διαφορά σε σύγκριση με τις άλλες δύο πολιτικές.
Στα σχήματα 5.3 και 5.4 παρουσιάζεται ο μέσος χρόνος απόκρισης ανά γεωγραφική περιοχή και ο μέσος χρόνος επεξεργασίας των αιτημάτων στα κέντρα δεδομένων σε διάφορετικές καταστάσεις φόρτου εργασίας. Οι αλγόριθμοι Throttled και Threshold παρέχουν καλύτερο χρόνο σε σχέση με τους RR και ESCE και χρειάστηκαν λιγότερο χρόνο για την επεξεργασία των αιτημάτων που υπέβαλαν οι χρήστες στα DC1 και DC2. Στα Σχήματα 5.5 και 5.6 παρουσιάζονται τα αποτελέσματα για την επεξεργασία των αιτημάτων και τον μέσο χρόνο απόκρισης σε καταστάσεις μεγάλου φόρτου εργασίας. Ο Throttled δίνει καλύτερα αποτελέσματα με μικρή διαφορά από το Threshold σε σύγκριση με τους RR και ESCE, παρουσιάζει ιδιαίτερη προσαρμοστικότητα και χρειάζεται λιγότερο χρόνο απόκρισης για την επεξεργασία των αιτημάτων που υποβλήθηκαν από τους χρήστες σε πέντε κέντρα δεδομένων για τη δεύτερη δοκιμή. Το συνολικό κόστος επεξεργασίας και για τις τέσσερις πολιτικές είναι ακριβώς το ίδιο, επομένως η τεχνική εξισορρόπησης φορτίου δεν επηρεάζει το κόστος επεξεργασίας.



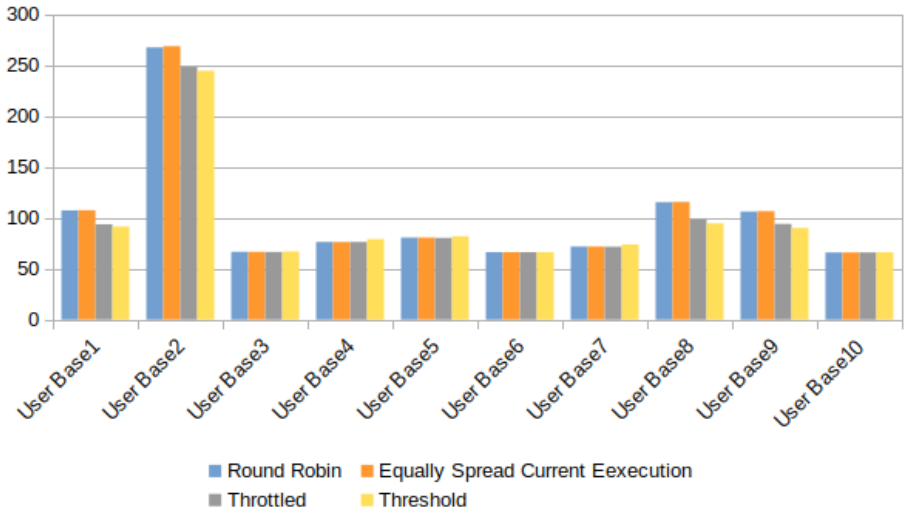
Σχήμα 5.2: Συνολικός χρόνος απόκρισης για διαφορετικούς αλγόριθμους



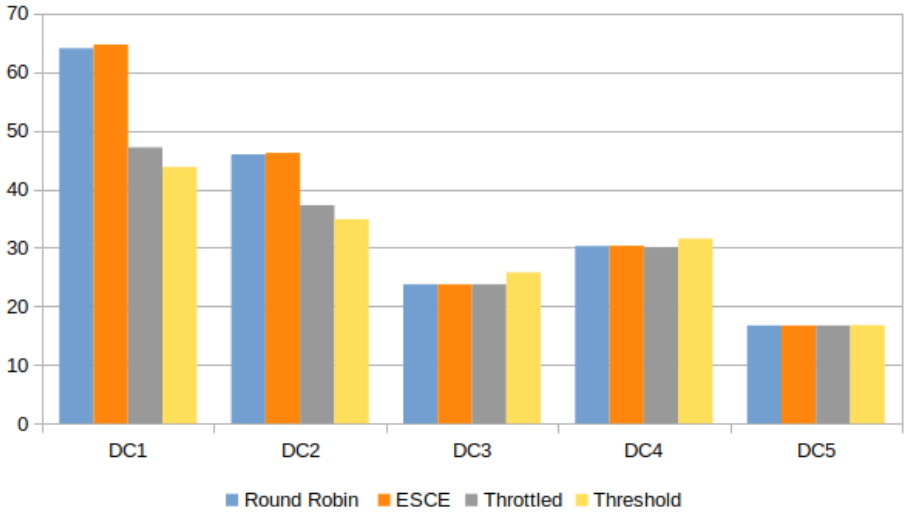
Σχήμα 5.3: Μέσος χρόνος απόκρισης ανά γεωγραφική περιοχή για διαφορετικούς αλγόριθμους



Σχήμα 5.4: Μέσος χρόνος επεξεργασίας αιτημάτων στο κέντρο δεδομένων για διαφορετικούς αλγορίθμους



Σχήμα 5.5: Μέσος χρόνος απόκρισης για διαφορετικούς αλγόριθμους



Σχήμα 5.6: Μέσος χρόνος επεξεργασίας αιτημάτων στο κέντρο δεδομένων για διαφορετικούς αλγορίθμους

Κεφάλαιο 6

Συμπεράσματα

Μετά τη μελέτη και την εκτέλεση των προηγούμενων προσομοιώσεων, καταλήξαμε στα ακόλουθα γενικά συμπεράσματα, τα οποία παρουσιάζονται παρακάτω. Και με τα τρία εργαλεία, οι προσομοιώσεις που πραγματοποιήθηκαν σχετίζονται με την αξιολόγηση γνωστών αλγορίθμων εξισορρόπησης φορτίου στο υπολογιστικό νέφος.

Το CloudSim δεν έχει εξαρτήσεις από άλλα πλαίσια προσομοίωσης, επομένως είναι ένα αυτόνομο πλαίσιο προσομοίωσης με όλα τα απαιτούμενα στοιχεία. Μελετώντας τη βιβλιοθήκη java του CloudSim, μπορεί κανείς να βρει μια δυνατότητα προσομοίωσης και ένα μοντέλο νέφους όπως αναμενόταν. Στοιχεία όπως το κέντρο δεδομένων, ο διακομιστής, τα VM, οι πολιτικές και οι φόρτοι εργασίας είναι διαθέσιμα στα χέρια του προγραμματιστή για να μοντελοποιήσει εύκολα ένα επιθυμητό περιβάλλον. Με το ενσωματωμένο μοντέλο ισχύος, προσφέρει ακόμη και ένα μοντέλο με ενεργειακή επίγνωση που επιτρέπει σε όσους ερευνητές δραστηριοποιούνται στον τομέα της ενεργειακής απόδοσης να μοντελοποιήσουν τα επιθυμητά σενάρια. Εκτός αυτού, επιτρέπει την αξιολόγηση του οικονομικού αντίκτυπου διαφορετικών προτύπων κατανάλωσης. Βασίζεται σε ένα ρολόι που δεν καθορίζεται από την πραγματική ώρα της ημέρας και απλά ξεκινά από το μηδέν. Τα συμβάντα εκτελούνται με διαδικαστικό τρόπο και όχι σε πραγματικό χρόνο, όπως θα αναμενόταν από ένα πλαίσιο προσομοίωσης, ωστόσο η προσομοίωση εκτελείται γρήγορα και το αρχείο καταγραφής που δημιουργείται περιέχει τους σωστούς χρόνους, όπως θα είχε εκτελεστεί σε πραγματικό χρόνο. Το CloudSim είναι σίγουρα μια βιώσιμη λύση όταν πρόκειται για την επιλογή ενός πλαισίου για την προσομοίωση υπολογιστικού νέφους. Επιπλέον, δεν πρέπει

να παραβλέψουμε ότι διαθέτει κάποιες επεκτάσεις. Για παράδειγμα, επεκτάσεις όπως το Cloud Reports προσφέρει GUI για προσομοιωτές Cloudsim. Ακόμα το CloudsimEx επεκτείνει το CloudSim προσθέτοντας δυνατότητες προσομοίωσης map reduce καθώς και παράλληλες προσομοιώσεις. Στον δικτυακό τόπο του CloudSim υπάρχει λεπτομερής κατάλογος των διαθέσιμων επεκτάσεων [17].

Για σκοπούς ενεργειακής επίγνωσης, η προσομοίωση Green Cloud είναι η καταλληλότερη, καθώς βασίζεται στο πλαίσιο προσομοίωσης NS2, το οποίο επικεντρώνεται στην επικοινωνία μεταξύ των υπηρεσιών Cloud σε επίπεδο πακέτων, γεγονός που της δίνει το πλεονέκτημα του ακριβούς υπολογισμού της κατανάλωσης ενέργειας. Διαφέρει αρκετά από το CloudSim και έχει κατασκευαστεί ειδικά για περιβάλλον με γνώμονα την ενεργειακή επίγνωση. Ένα από τα μειονεκτήματα του GreenCloud είναι ότι χρειάζεται μερικά λεπτά για την προσομοίωση ενός μοντέλου καθώς και μεγάλη μνήμη. Παρόλο που παρέχει τη δυνατότητα προσθήκης ή τροποποίησης του υπάρχοντος πηγαίου κώδικα, είναι γραμμένο σε C++ και OTcl, γεγονός που αποτελεί άλλο ένα μειονέκτημα αυτού του προσομοιωτή, καθώς πρέπει να χρησιμοποιηθούν δύο διαφορετικές γλώσσες για την υλοποίηση ενός πειράματος.

Το CloudAnalyst περιλαμβάνει νέα ισχυρά χαρακτηριστικά, όπως ένα εύχρηστο γραφικό περιβάλλον, ευελιξία στη διαμόρφωση οποιουδήποτε συστήματος γεωγραφικής κατανομής, όπως ο καθορισμός παραμέτρων υλικού (μνήμη, όριο εύρους ζώνης, καθυστέρηση δικτύου κ.λπ.) μιας εικονικής μηχανής ή ενός κέντρου δεδομένων, δυνατότητα διαχωρισμού μιας προσομοίωσης από τον κώδικα προγραμματισμού, γρήγορη ρύθμιση της προσομοίωσης και βελτιωμένη γραφική απεικόνιση των αποτελεσμάτων, με χρήσιμες μορφές, όπως πίνακες και διαγράμματα. Το ιδιαίτερο σε αυτόν τον προσομοιωτή είναι η γραφική παρουσίαση των αποτελεσμάτων και η επανάληψη των προσομοιώσεων σύμφωνα με τις απαιτήσεις του χρήστη. Επίσης, μπορούν εύκολα να προστεθούν νέες πολιτικές εξισορρόπησης φορτίου και διαμεσολάβησης υπηρεσιών. Το εργαλείο CloudAnalyst είναι χρήσιμο σε περιπτώσεις όπου ο κύριος στόχος είναι η προσομοίωση εφαρμογών μεταξύ πολλαπλών κέντρων δεδομένων και ομάδων χρηστών. Κατά τη γνώμη μας, οι χρήστες διαδικτυακών εφαρμογών που ενδιαφέρονται περισσότερο για το χρόνο που απαιτείται για την επεξεργασία των αιτημάτων τους, ανεξάρτητα από τη θέση του διακομιστή, θα πρέπει να επιλέξουν το πακέτο προσομοίωσης Cloud Analyst.

Βιβλιογραφία

[1] B. Mohammad Hasani Zadea N. Mansouria R. Ghafaria. *Cloud computing simulators: A comprehensive review*. 2020.

[2] M.D. Dikaiakos et al. *Cloud computing:Distributed internet computing for it and scientific research. Internet Computing,IEEE, 13(5):10–13*. 2009.

[3] Salma Rebai. *Resource allocation in Cloud federation*. 2017.

[4] Grance Timothy Mell Peter. *The NIST definition of Cloud Computing: Recommedations of the National Institute of Stamdars and Technology" NIST Special Publication 800-145*. 2011.

[5] Raouf Boutaba Qi Zhang Lu Cheng. *Cloud computing: state-of-the-art and research challenges" J Internet Serv Appl (2010) 1: 7–18*. 2010.

[6] M. Cafaro and G. Aloisio. *Grids, clouds, and virtualization. In Grids, Clouds and Virtualization, pages 1–21*. 2011.

[7] J.E. Smith and R. Nair. *The architecture of virtual machines. Computer, 38(5):32–38*. 2005.

[8] Qian Chen et al. *On State of The Art in Virtual Machine Security*. 2012.

[9] *Google apps*. `https://workspace.google.com/`.

[10] *Google app engine*. `https://cloud.google.com/appengine`.

[11] *Microsoft azure cloud services*. `https://azure.microsoft.com/en-us/services/cloud-services/`.

[12] *Amazon Elastic Compute Cloud (EC2)*. `http://aws.amazon.com/ec2/`.

[13] *Microsoft Azure IaaS*. `http://azure.microsoft.com/en-us/services/virtual-machines/`.

[14] *Google compute engine.* `https://cloud.google.com/compute/`.

[15] *IBM Cloud.* `https://www.ibm.com/cloud/infrastructure`.

[16] Rodrigo N. Calheiros et al. *Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exper. 41:23–50.* 2011.

[17] *CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services:* `https://github.com/cloudslab/cloudsim`.

[18] R. N. Calheiros B. Wickremasinghe. *CloudAnalyst: A CloudSim-based Visual Modeller for analysing Cloud Computing Environments and Applications, 24th IEEE International Conference on Advanced Information Networking and Applications.* 2010.

[19] *CloudAnalyst:* `http://www.cloudbus.org/cloudsim/`.

[20] Rodrigo N. Calheiros et al. *EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications, Software: Practice and Experience. DOI:10.1002/spe.2124.* 2012.

[21] *EMUSIM:* `www.cloudbus.org/cloudsim/emusim`.

[22] M. Tighe et al. *DCSim: A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management, 8th international conference and 2012 workshop on systems virtualization management (svm) Network and service management (cnsm).* 2012.

[23] *A Data Centre Simulation Tool for Evaluating Dynamic Virtualized Resource Management:* `https://github.com/digs-uwo/dcsim`.

[24] Jesús Carretero Pérez Gabriel González and Castañé Javier Prieto Cepeda. *Grupo de Arquitectura de Computadores Universidad Carlos III de Madrid ,iCanCloud Quick Installation Guide.*

[25] *OMNeT++:* `https://omnetpp.org`.

[26] S. U. Khan D. Kliazovich P. Bouvry. *GreenCloud: A Packetlevel Simulator of Energy-aware Cloud Computing Data Centers,Journal of Supercomputing, vol. 62, no. 3, pp. 1263-1283.* 2012.

[27] *NS-2:* `www.isi.edu/nsnam/ns`.

[28] Saurabh Kumar Garg and Rajkumar Buyya. *NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations.* 2021.

[29] S. Osterman et al. *GroudSim: An Event based Simulation Framework for Computational Grids and Clouds,in Euro Par 2010 Parallel Process. Workshops,vol. 6586, pp. 305–313*. 2011.

[30] S. Sotiriadis et al. *SimIC: Designing a new Inter Cloud simulation platform for integrating large scale resource management, Proc. Int. Conf. Adv. Inf. Netw. Appl. AINA, pp. 90–97*. 2013.

[31] S. Lim et al. *MDCSim: A Multi-tier Data Center Simulation Platform, Cluster Computing and Workshops*. 2009.

[32] Oladosu Oyebisi Oladimeji et al. *A Comprehensive Survey on Cloud Computing Simulators*. 2021.

[33] Rajkumar Buyya and Manzur Murshed. *A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing*. 2002.

[34] Mohammad Oqail Ahmad and Rafiqul Zaman Khan. *Cloud Computing Modeling and Simulationusing CloudSim Environmen*. 2019.

[35] Anupinder Singh. *Cloudsim Tutorial: Random Cloudlets Workload*. 2020.

[36] Altaf Hussain et al. *Load Balancing in Cloud Computing*. 2020.

[37] Saleh Sa adeh Sulieman Bani-Ahmad. *Scalability of the DVFS Power Management Technique as Applied to 3-Tier Data Center Architecture in Cloud Computing*. 2017.

[38] Rastin Pries et al. *Power Consumption Analysis of Data Center Architectures*. 2012.

[39] Khadijah Bahwaireth1 et al. *Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications*. 2012.

[40] Mridul Wadhwa, Approv Goel, and Tanupriya Choudhury. *Green Cloud Computing - A Greener Approach To IT*, 2019.

[41] Manpreet Dhillon. *Performance (Energy) Comparison of Task Scheduling Algorithms in Cloud Computing using GreenCloud*. 2018.

[42] Pericherla S Suryateja. *A Comparative Analysis of Cloud Simulators*. 2016.

[43] Anju Sharma Simar Preet Singh and Rajesh Kumar. *Analysis of Load Balancing Algorithms using Cloud Analyst. In: International Journal of Grid and Distributed Computing 9.9, pp. 11–24*. 2016.

[44] Sandip Patel et al. *CloudAnalyst : A Survey of Load Balancing Policies*. 2015.

[45] Jagmeet Singh. *Implementaion of Threshold Algorithm, (2016), GitHub repository,* `https://github.com/jagmeet787/CloudAnalyst`.

Παράρτημα Α

Οδηγίες εγκατάστασης προσομοιωτών

Α.1 Συνοπτική παρουσίαση εγκατάστασης του εργαλείου Cloudsim

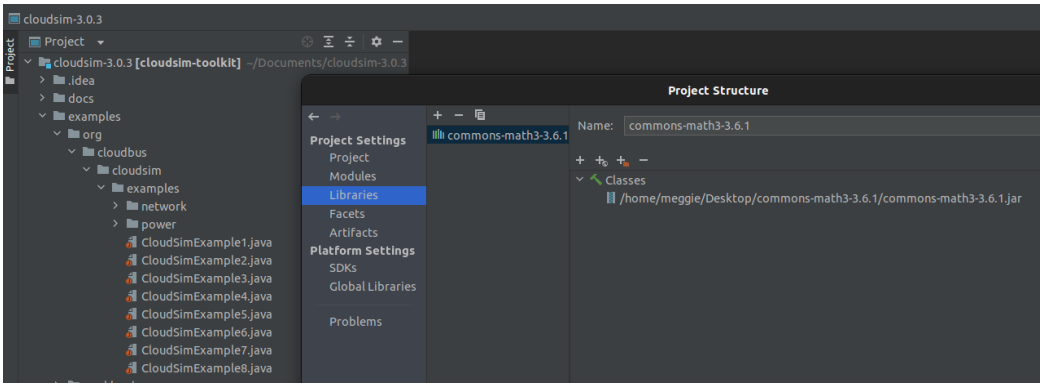
Η βασική απαίτηση για τη διαμόρφωση του CloudSim έχει ως εξής:
1. Εγκατάσταση του CloudSim (3.0.3 ή 4.0) από τον ιστότοπο
https://github.com/Cloudslab/cloudsim/releases
2. Εγκατάσταση του Intellij IDE από
https://www.jetbrains.com/idea/download/
3. Java Development Kit (1.7 ή 1.8)

Διαμόρφωση και εκτέλεση με το IntelliJ IDE

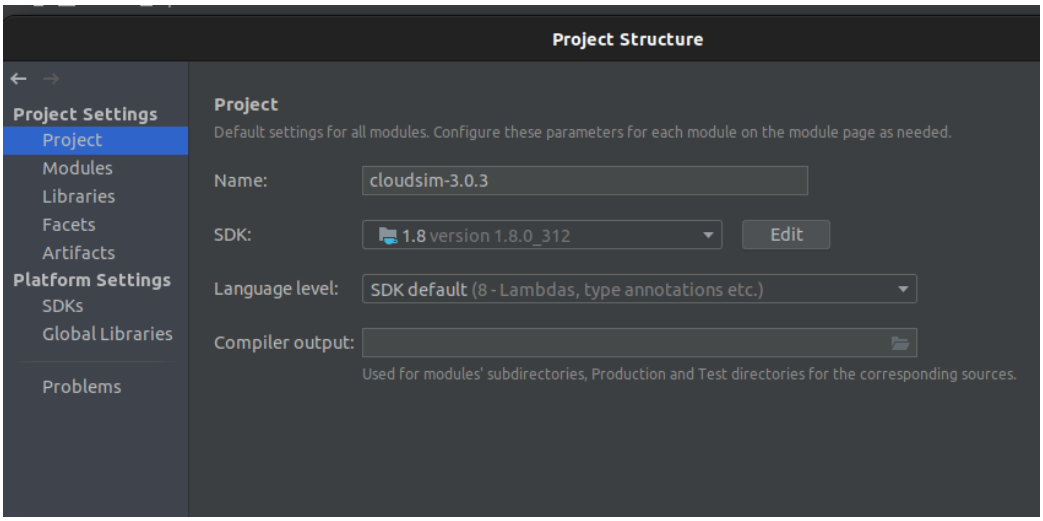
Αποσυμπιέστε το CloudSim (είναι ένα αρχείο zip για Windows και ένα αρχείο tar για Mac και Linux) και ανοίξτε το φάκελο IntelliJ.

Θα χρειαστεί να προσθέσουμε μια βιβλιοθήκη που ονομάζεται apache commons στο έργο. Για να το κάνετε αυτό, μεταβείτε στο File -> Project Structure και ανοίξτε την καρτέλα Libraries. Κάντε κλικ στο εικονίδιο Plus(+) και προσθέστε το αρχείο jar apache commons. Κάντε κλικ στο Ok για να εφαρμόσετε και κλείστε το παράθυρο.

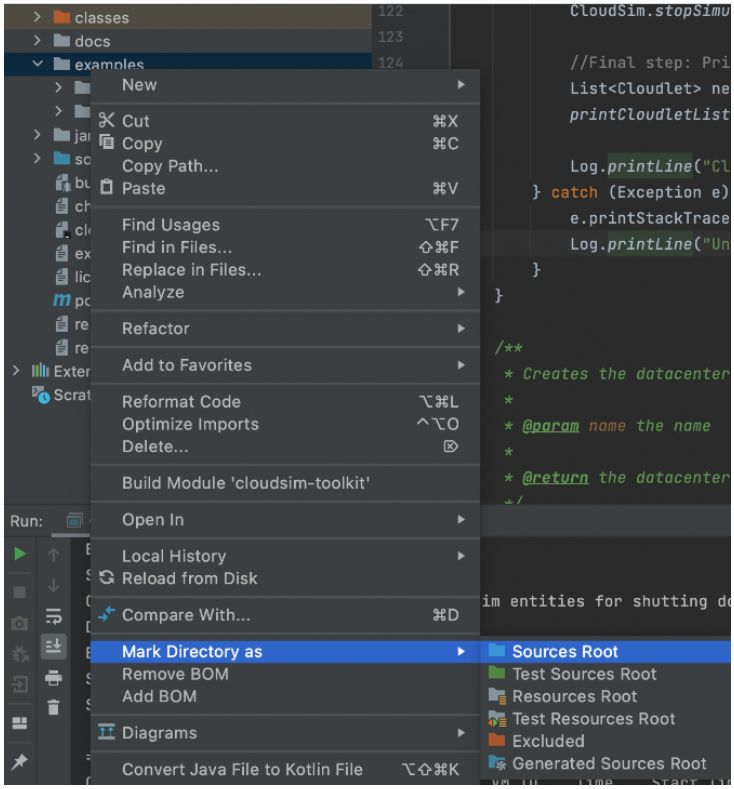
Το CloudSim θα έπρεπε να τρέχει με Java 7, αλλά στο Ubuntu 16.04 και



άνω, η Java 7 δεν είναι πλέον διαθέσιμη. Ως εκ τούτου, είναι καλύτερο να εγκαταστήσετε τη Java 8 (ή 9). Συνεπώς, ας ρυθμίσουμε το έργο ώστε να χρησιμοποιεί την JRE 8. Αρχικά, ρυθμίστε το JRE 8 στον υπολογιστή σας. Συνήθως μπορούμε να εγκαταστήσουμε την Java 8 στο Ubuntu 22.04/20.04/18.04/18.04 από τα διαθέσιμα αρχεία OpenJDK και να εκτελέσουμε την εντολή: sudo apt install openjdk-8-jdk και μπορούμε να επιλέξουμε την προεπιλεγμένη έκδοση της Java χρησιμοποιώντας την εντολή update-alternatives –config java. Για το σκοπό αυτό, ανοίξτε το File -> Project Structure και ανοίξτε τον πίνακα Project,στην ενότητα Project SDK, βεβαιωθείτε ότι το JRE είναι επισημασμένο.



Σε αυτό το σημείο, η εργαλειοθήκη CloudSim είναι έτοιμη για εκτέλεση. Αλλά πρώτα πρέπει να ενημερώσουμε το IntelliJ ότι πρόκειται για αρχεία πηγαίου κώδικα Java που μπορούν να εκτελεστούν ως εφαρμογές Java. Έτσι, κάντε δεξί κλικ στο φάκελο examples και πλοηγηθείτε -> Mark Directory as -> Sources Root.



A.2 Συνοπτική παρουσίαση εγκατάστασης του εργαλείου GreenCloud

Η λήψη και εγκατάσταση του GreenCloud από το μηδέν μπορεί να γίνει με τα ακόλουθα βήματα:
Η λήψη του GreenCloud μπορεί να πραγματοποιηθεί από:
"https://download.uni.lu/GreenCloud/greencloud-v2.1.2.tar.gz".

Για να το κάνετε αυτό με επιτυχία, είναι σημαντικό να χρησιμοποιήσετε ένα μηχάνημα (ή VM) που τρέχει Ubuntu. Το πακέτο έρχεται ενσωματωμένο με τον πηγαίο κώδικα του NS2, ο οποίος αφού αποσυμπιεστεί μπορεί να εγκατασταθεί. Ανοίξτε τον κατάλογο με το πακέτο. Εκτελέστε το ./install.sh το οποίο θα ξεκινήσει την εγκατάσταση (απαιτεί Ubuntu 12.x ή νεότερη έκδοση). Τώρα το GreenCloud είναι έτοιμο για χρήση και εκτέλεση προσομοιώσεων. Το GreenCloud διαθέτει επίσης την εικονική μηχανή (UbuntuOS) που χρησιμοποιείται για τις προσομοιώσεις στη διεύθυνση:http://greencloud.gforge.uni.lu/ftp/GreenCloud-VM-

2.1.2.tar.gz". Αυτό είναι ένα αρχειοθετημένο δέντρο πηγής και είναι προ-ρυθμισμένο για να χρησιμοποιηθεί ως VM, το οποίο θα είναι προσβάσιμο μόνο με τη χρήση VMWare Workstation ή Virtual Box ή οποιοδήποτε λογισμικό που μπορεί να εκτελέσει VMs. Το προ-ρυθμισμένο VM παρέχει το Eclipse IDE (προεγκατεστημένο).

Α.3 Εγκατάσταση και περιβάλλον CloudAnalyst

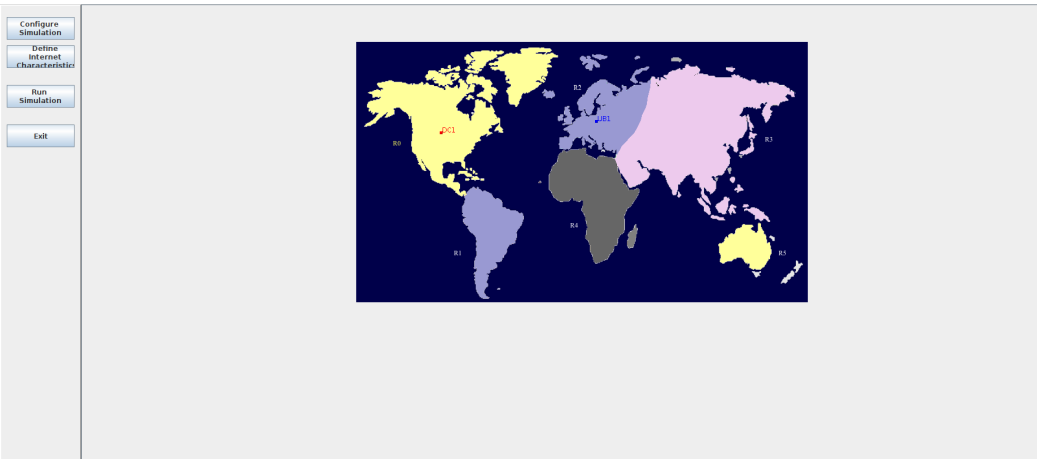
Η διαδικασία της εγκατάστασης του CloudAnalyst είναι αρκετά απλή και εύκολη. Το πρόγραμμα είναι γραμμένο σε κώδικα java. Η διαδικασία για να εγκαταστήσουμε και να εκτελέσουμε τη πλατφόρμα προσομοίωσης είναι η εξής:

Το αρχείο του CloudAnalyst είναι σε συμπιεσμένη μορφή και βρίσκεται στη παρακάτω ιστοσελίδα:
http://www.cloudbus.org/cloudsim/CloudAnalyst.zip.
Στη συνέχεια, αποσυμπιέζουμε το αρχείο CloudAnalyst και το ανοίγουμε με το Eclipse ή το Netbeans IDE. Ανοίγουμε το φάκελο του πακέτου πηγής μέσα στον οποίο βρίσκεται το cloudsim.ext.gui και κάνουμε δεξί κλικ στο gui.main.java και επιλέγουμε run. Μόλις εκτελέσουμε τις παραπάνω διαδικασίες, θα εμφανιστεί στην οθόνη η αρχική σελίδα της πλατφόρμας προσομοίωσης όπως φαίνεται στην Εικόνα Α.1. Στην αριστερή πλευρά υπάρχουν οι επιλογές που οδηγούν στις οθόνες για την εισαγωγή των παραμέτρων.

Για να ορίσουμε τις ρυθμίσεις της προσομοίωσης κάνουμε κλικ στην επιλογή configure simulation (Διαμόρφωση προσομοίωσης), η οποία αποτελείται από τρεις καρτέλες:

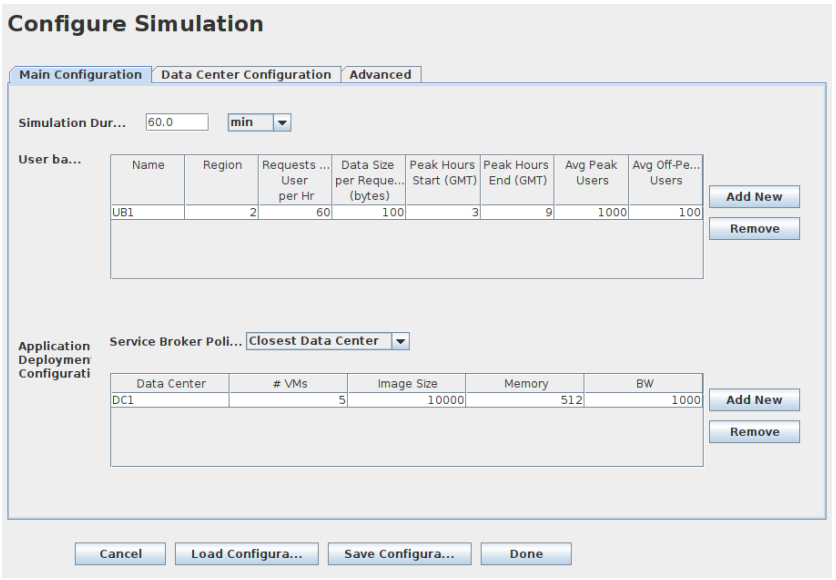
- Main configuration
- Data center configuration
- Advanced

Main configuration: Σε αυτό το στοιχείο ο χρήστης επιλέγει και εισάγει δεδομένα που αντιπροσωπεύουν τους χρήστες και τα αιτήματα που στέλνουν. Μπορούμε να παρατηρήσουμε τα πεδία που μπορεί να βάλει ο χρήστης όπως:



Σχήμα Α΄.1: Η κύρια οθόνη του CloudAnalyst

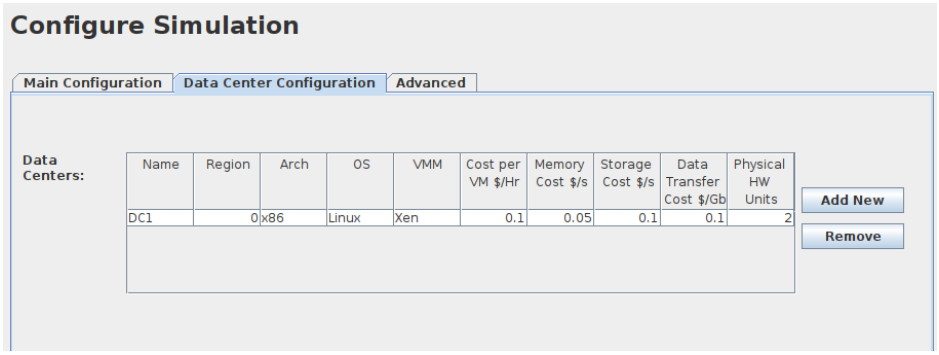
Περιοχή χρηστών, αιτήματα χρήστη ανά ώρα, μέγεθος δεδομένου ανά αίτηση του χρήστη, ώρες αιχμής και πλήθος χρηστών σε εκείνες τις ώρες αλλά και στις υπόλοιπες ώρες της ημέρας.



Σχήμα Α΄.2: Ρύθμιση βάσης χρηστών στον CloudAnalyst

Data center configuration: Στον τομέα αυτό ελέγχονται οι δραστηριότητες των κέντρων δεδομένων. Ο χρήστης μπορεί να προσθέσει κέντρα δεδομένων, να ορίσει την περιοχή όπου βρίσκονται, το κόστος καθώς και το πλήθος των εικονικών μηχανημάτων που διαθέτουν. Στην εικόνα που ακολουθεί μπορούμε να παρατηρήσουμε τα πεδία που μπορεί να βάλει ο

χρήστης όπως: Περιοχή Data Center, αρχιτεκτονική, λειτουργικό σύστημα, κόστος ανά εικονικό μηχάνημα, κόστος μνήμης, κόστος μεταφοράς δεδομένων καθώς και πλήθος φυσικών υπολογιστικών μονάδων.

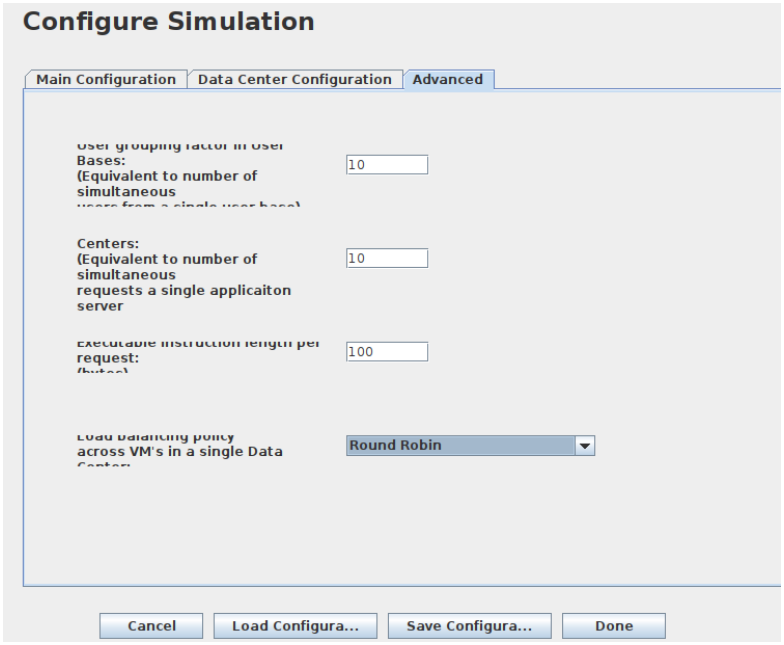


Σχήμα Α΄.3: Τα πεδία για την ρύθμιση των παραμέτρων για τα Danta Center στον CloudAnalyst

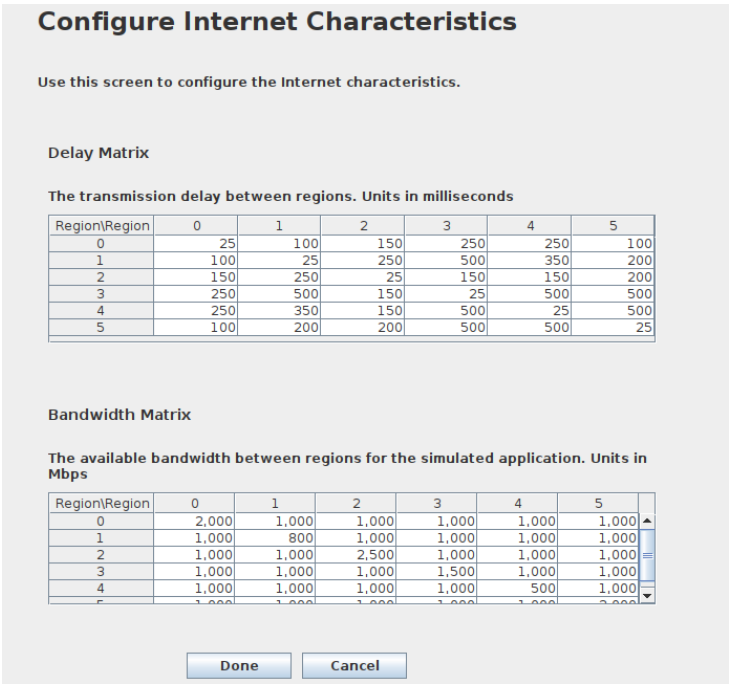
Advanced: Στη καρτέλα advanced γίνεται διαμόρφωση προηγμένων επιλογών σύμφωνα με τις ανάγκες του χρήστη. Οι επιλογές που μπορούμε να διαμορφώσουμε είναι η εξής:

- User grouping factor in user bases : Εδώ ορίζεται τον αριθμό των παράλληλων χρηστών από μια μοναδική βάση χρήστη.
- Request grouping factor in data centers: Εδώ ορίζεται τον αριθμό των παράλληλων αιτημάτων όπου ένας μοναδικός διακομιστής εφαρμογών μπορεί να υποστηρίξει.
- Executable instruction length per request(bytes): Εδώ ορίζεται το μέγεθος σε bytes των εκτελέσιμων εντολών ανά αίτημα.
- Load balancing policy across VM's in a single data center: Εδώ ορίζεται τη πολιτική εξισορρόπησης φορτίου ενός εικονικού μηχανήματος σε ένα μοναδικό κέντρο δεδομένων. Οι αλγόριθμοί που ορίζουν αυτή την πολιτική είναι οι εξής: Round Robin, Equally spread current execution load,Throttled.

Internet Characteristics: Εδώ ρυθμίζονται οι παράμετροι του Internet όπως η ταχύτητα και η καθυστέρηση μεταξύ των περιοχών μοντελοποιώντας την κίνηση της δρομολόγησης ανά τον κόσμο.

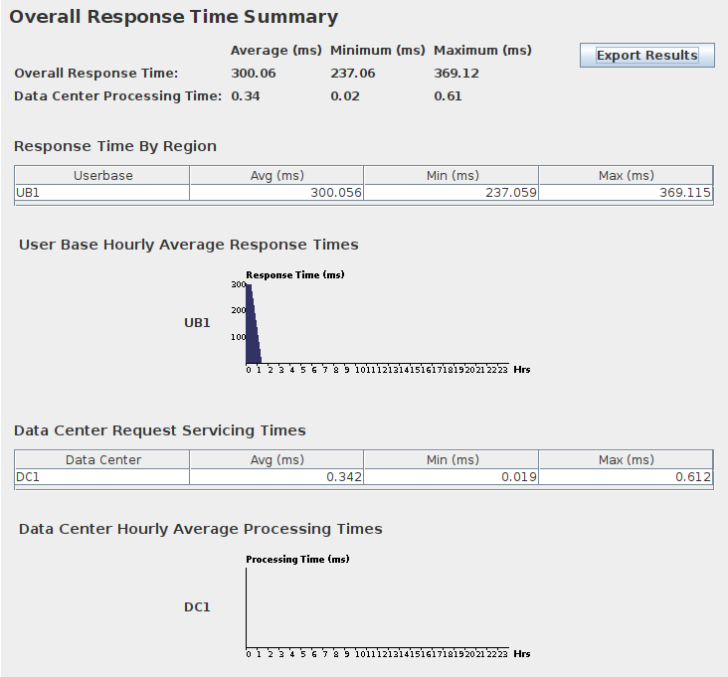


Σχήμα Α΄.4: Οι ρυθμίσεις στην καρτέλα Advanced



Σχήμα Α΄.5: Οι πίνακες παραμετροποίησης των Delay και Bandwidth ανά περιοχή

Ως αποτέλεσμα του CloudAnalyst, μπορούν να παραχθούν διάφορα δεδομένα, όπως ο χρόνος απόκρισης της προσομοιωμένης εφαρμογής, τα πρότυπα χρήσης της εφαρμογής, ο χρόνος που απαιτείται από τα κέντρα δεδομένων για την εξυπηρέτηση ενός αιτήματος χρήστη και το κόστος λειτουργίας.



Σχήμα Α΄.6: Αποτελέσματα προσομοίωσης