Τεχνολογικό Πανεπιστήμιο Κύπρου
Τμήμα Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
Σχολή Μηχανικής

Αυτοματοποιημένη κατανομή υπολογιστικών πόρων για εφαρμογές αρχιτεκτονικής νέφους σε υπολογιστικά κέντρα

Διπλωματική Εργασία

του

ΕΥΑΓΟΡΑ Π. ΜΑΚΡΙΔΗ

Επιβλέπων: Δρ. Κυριάκος Δεληπαράσχος

Λεμεσός, Μαίος 2017

Τεχνολογικό Πανεπιστήμιο Κύπρου
Τμήμα Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
Σχολή Μηχανικής

Αυτοματοποιημένη κατανομή υπολογιστικών πόρων για εφαρμογές αρχιτεκτονικής νέφους σε υπολογιστικά κέντρα

Διπλωματική Εργασία

του

ΕΥΑΓΟΡΑ Π. ΜΑΚΡΙΔΗ

Επιβλέπων: Δρ. Κυριάκος Δεληπαράσχος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Μαίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δρ. Κυριάκος Δεληπαράσχος	*Δρ. Θεμιστοκλής Χαραλάμπους*	*Δρ. Σάββας Λοΐζου*
	Επίκουρος Καθηγητής	*Επίκουρος Καθηγητής*

Λεμεσός, Μαίος 2017

Τεχνολογικό Πανεπιστήμιο Κύπρου
Τμήμα Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
Σχολή Μηχανικής

Υπεύθυνη Δήλωση
Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην διπλωματική εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών της Σχολής Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής.

(Υπογραφή)

.....
Ευαγόρας Μακρίδης

Περίληψη

Η τεχνολογία της εικονικοποίησης (consolidation) είναι το εργαλείο για την καταχώρηση πολλών κατανεμημένων υπηρεσιών σε κέντρα δεδομένων Υπολογιστικής Αρχιτεκτονικής Νέφους (ΥΑΝ). Οι διάφοροι εξυπηρετητές ενός κέντρου δεδομένων πρέπει να παρέχουν τις προδιαγεγραμμένες υπηρεσίες τους στους χρήστες. Εκμεταλλεύοντας την τεχνολογία της εικονικοποίησης, οι εξυπηρετητές μπορούν να εγκατασταθούν σαν εικονικές μηχανές και να συμπεριφέρονται όπως οι ανεξάρτητες φυσικές μηχανές, μοιράζοντας τους υπολογιστικούς πόρους με τις υπόλοιπες εικονικές μηχανές του κέντρου δεδομένων. Στόχος της πτυχιακής εργασίας αυτής είναι να εντρυφήσει στο θέμα της αποδοτικής και αυτοματοποιημένης κατανομής των υπολογιστικών πόρων ενός υπολογιστικού κέντρου δεδομένων για διάφορες διαδικτυακές εφαρμογές. Για την δυναμικά αυτοματοποιημένη κατανομή των πόρων χρησιμοποιήθηκαν διάφορα συστήματα αυτομάτου ελέγχου για τα οποία έγινε και η ανάλογη εξακρίβωση λειτουργίας.

Στόχος του συστήματος της αυτοματοποιημένης κατανομής πόρων είναι να εξασφαλίσουν οι χρήστες τις προδιαγεγραμμένες απαιτήσεις ποιότητας υπηρεσίας χωρίς να επηρεαστεί ο μέσος χρόνος απάντησης στα αιτήματά τους καθώς αλλάζει ο φόρτος εργασίας της εφαρμογής. Απώτερος στόχος είναι (α) η αποδοτική κατανομή των πόρων για την εξυπηρέτηση περισσότερων εφαρμογών με την υπάρχουσα υποδομή και (β) το χαμηλό κόστος λειτουργίας των κέντρων δεδομένων (γ) η εξοικονόμηση ενέργειας με την μείωση εξυπηρετητών στα κέντρα δεδομένων.

Abstract

Server consolidation technology is the key for multiple distributed services in Cloud Computing Architecture data centers. Servers in a data center must provide their Quality of Service (QoS) to their clients. Physical servers can exploit the technology of consolidation so that they transformed into one or more virtual machines that share the resources with other virtual machines in the data center. This dissertation indulgence in effective and automated allocation of resources of a data center for various cloud applications. The dynamic allocation of the resources was done using various control systems in order to evaluate the system performance.

This dissertation aims to build a system of virtualized resource allocation controllers in order to provide the quality of service that clients specified before, without affecting the mean response time while the server's workload changes in a random manner. The ultimate goal is (a) the efficient allocation of the resources in order to install more co-located applications on the same infrastructure. (b) low data center's running cost. (c) energy savings through the reduction of the number of servers in data centers.

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον Δρ. Κυριάκο Δεληπαράσχο για την επίβλεψη και την καθοδήγηση αυτής της διπλωματικής εργασίας και για τη δυνατότητα που μου έδωσε να χρησιμοποιήσω την υπολογιστική υποδομή στο δωμάτιο εξυπηρετητών του κτιρίου Ηλεκτρολόγων Μηχανικών του Τεχνολογικού Πανεπιστημίου Κύπρου για τις ανάγκες της παρούσας εργασίας. Επίσης ευχαριστώ ιδιαίτερα τον Δρ. Θεμιστοκλή Χαραλάμπους για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Ευχαριστίες θα ήθελα να δώσω και στην Δρ. Ευαγγελία Καλυβιανάκη για τις σημαντικές λεπτομέρειες που μου υπέδειξε. Σημαντική ήταν η βοήθεια και η υποστήριξη του δικτυακού εξοπλισμού από την κα. Κίκα Χρήστου, Υπηρεσία Συστημάτων Πληροφορικής και Τεχνολογίας, και για αυτό την ευχαριστώ. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η μελέτη του προβλήματος της αυτόματης και δυναμικής κατανομής υπολογιστικών πόρων για εικονικοποιημένες εφαρμογές τεχνολογίας υπολογιστικής νέφους. Η αυτόματη και δυναμική κατανομή των υπολογιστικών πόρων είναι πλέον αναγκαία για σύγχρονα κέντρα δεδομένων που φιλοξενούν διάφορες εφαρμογές, οι οποίες παρέχονται στους χρήστες, έτσι ώστε με την σωστή κατανομή να μην επηρεάζονται οι προδιαγραφές απόδοσης που καθορίστηκαν κατά την συμφωνία επιπέδου υπηρεσίας (Service Level Agreement - SLA). Η δυναμική κατανομή για κάθε εξυπηρετητή είναι πλέον εφικτή με τα διάφορα λογισμικά εικονικοποίησης, όμως οι διάφορες αυξομοιώσεις στην ζήτηση υπολογιστικών πόρων λόγων των φόρτων εργασίας δημιουργεί προκλήσεις στα συστήματα ελέγχου που πρέπει να χαρακτηρίζονται από γρήγορη απόκριση σε συνδυασμό πάντα με την ευστάθεια. Στην παρούσα πτυχιακή εργασία η αυτόματη κατανομή των υπολογιστικών πόρων γίνεται με την χρήση βέλτιστου ελέγχου, εύρωστου ελέγχου αλλά και με αλγορίθμους ευφυών συστημάτων όπως οι προσαρμοστικοί νευρο-ασαφείς ελεγκτές (ANFIS).

1.1 Κίνητρα για την εκπόνηση της διπλωματικής εργασίας

Τα κέντρα δεδομένων αποτελούνται από πολλές εφαρμογές ή πλατφόρμες οι οποίες φιλοξενούνται στις φυσικές μηχανές των κέντρων δεδομένων. Οι εφαρμογές αυτές πρέπει να είναι σε θέση να αντεπεξέλθουν σε κάθε ανάγκη του χρήστη ανά πάσα στιγμή έτσι ώστε να παρέχεται σε αυτόν η καλύτερη δυνατή ποιότητα υπηρεσίας. Με την πάροδο του χρόνου, τα κέντρα δεδομένων μειώνουν τον αριθμό των φυσικών μηχανών μετατρέποντας τις σε πολλές εικονικές

με σκοπό την ανάπτυξη των διαφόρων εφαρμογών στο υπολογιστικό νέφος. Οι λόγοι της μείωσης του αριθμού των φυσικών μηχανών είναι πολλοί. Ένας από αυτούς είναι η επεκτασιμότητα (scalability), δηλαδή να αποφεύγεται η αγορά σε υλικό και να δημιουργούνται σε υφιστάμενο υλικό πολλές εικονικές μηχανές. Ένας ακόμη λόγος για την αύξηση των εικονικών μηχανών είναι η αποδοτική χρήση των υπολογιστικών πόρων. Λόγω του υψηλού κόστους συντήρησης, αναβάθμισης και λειτουργίας των φυσικών μηχανών, υπάρχει η τάση να μειώνεται ο αριθμός τους αφού είναι δεδομένο ότι σε κέντρα δεδομένων υπάρχουν εξυπηρετητές που χρησιμοποιούν λιγότερο από το 30% των φυσικών τους υπολογιστικών πόρων με αποτέλεσμα να χάνονται υπολογιστικοί πόροι που δεν χρησιμοποιούνται [1]. Για τον λόγο αυτό η ανάγκη για ένα αυτοματοποιημένο σύστημα κατανομής υπολογιστικών πόρων επιβάλλεται έτσι ώστε να γίνεται εξοικονόμηση σε υπολογιστικούς πόρους και ταυτόχρονα σε ενέργεια.

Στις μέρες μας παρατηρείται μια αύξηση στον αριθμό των χρηστών που χρησιμοποιούν διαδικτυακές εφαρμογές τεχνολογίας νέφους, λόγω της εύκολης λειτουργίας τους μέσω των περιηγητών ιστού (browsers). Η φύση των διαδικτυακών εφαρμογών, μπορεί να αλλάξει ξαφνικά την ζήτηση σε υπολογιστικούς πόρους, με την αύξηση των χρηστών που χρησιμοποιούν την εφαρμογή σε ώρες αιχμής αλλά και σε στιγμές ξαφνικής φόρτωσης της. Καθώς αυξάνονται οι ανάγκες ενός σύγχρονου κέντρου δεδομένων, αυξάνεται ανάλογα και ο αριθμός των εικονικών μηχανών και εφαρμογών τεχνολογίας νέφους που πρέπει να εγκατασταθούν στην φυσική μηχανή με αποτέλεσμα να απαιτούνται από αυτήν περισσότεροι υπολογιστικοί πόροι. Η τεχνολογία της εικονικοποίησης (virtualization) επιτυγχάνεται με την προσθήκη ενός στρώματος λογισμικού το οποίο παρέχει διάφορες υπηρεσίες εικονικοποίησης. Το στρώμα λογισμικού ονομάζεται και υπερεπόπτης (hypervisor) και έχει τη δυνατότητα να δημιουργεί και να διαχειρίζεται εικονικές μηχανές οι οποίες μοιράζονται τους υπολογιστικούς πόρους μιας φυσικής μηχανής. Μερικά από τα στρώματα αυτά έχουν την δυνατότητα να διαχειρίζονται δυναμικά την κατανομή πόρων σε διάφορες εικονικές μηχανές, ενεργοποιώντας έτσι την δυνατότητα ανάπτυξης συστημάτων ελέγχου για την έξυπνη διαχείριση της κατανομής.

Η απόδοση των εξυπηρετητών και παράλληλα των διαδικτυακών εφαρμογών αξιολογείται με βάση την δυνατότητα τους να «απαντούν» όσο πιο γρήγορα γίνεται στα αιτήματα (requests) των χρηστών. Ο χρόνος απόκρισης των αιτημάτων έχει άμεση σχέση με την χρήση της κεντρικής μονάδας επεξεργασίας των μηχανών και ταυτόχρονα με την ζήτηση υπολογιστικών πόρων. Η κατανομή πρέπει να αλλάζει δυναμικά έτσι ώστε οι πόροι που δεν χρησιμοποιούνται από κάποια εφαρμογή να κατανέμονται ανά πάσα στιγμή σε άλλες εφαρμογές οι

οποίες έχουν ανάγκη για υπολογιστικούς πόρους, και έτσι περισσότερες εφαρμογές να μπορούν να φιλοξενούνται στην ίδια φυσική μηχανή, τηρώντας όμως την ποιότητα υπηρεσίας (QoS).

1.2 Αντικείμενο της διπλωματικής εργασίας

Η πτυχιακή εργασία αυτή, εντρύφησε στην εξομοίωση ενός σύγχρονου κέντρου δεδομένων στο οποίο αναπτύχθηκε μια εφαρμογή τεχνολογίας νέφους, διαδικτυακών δημοπρασιών RUBiS, με σκοπό την αξιολόγηση των διαφόρων ελεγκτών καθώς αυτά κατανέμουν τους υπολογιστικούς πόρους στις ανάλογες εικονοποιημένες μηχανές. Το βασικό ζήτημα που προκύπτει είναι το πως η διαδικασία της κατανομής των πόρων θα γίνεται αυτόματα και δυναμικά ενώ ο φόρτος εργασίας της εφαρμογής αλλάζει ξαφνικά. Το σύστημα που υλοποιήθηκε θα έπρεπε να προβλέπει τους διάφορους φόρτους εργασίας της εφαρμογής και να αλλάζει την κατανομή των υπολογιστικών πόρων χωρίς να επηρεάζεται η απόδοση της η οποία καθορίστηκε κατά το SLA, ως ποιότητα υπηρεσίας QoS. Οι μελλοντικές αυξομοιώσεις στην χρήση υπολογιστικών πόρων θα πρέπει να προβλέπονται από τον κάθε στοιχείο που αποτελεί το σύμπλεγμα της εφαρμογής. Στο σημείο αυτό απαιτείται η χρήση συστημάτων αυτομάτου ελέγχου. Ο έλεγχος αυτός μπορεί να γίνεται με δύο τρόπους:

1. Ο πρώτος τρόπος είναι η χρήση αναδραστικών συστημάτων αυτομάτου ελέγχου τα οποία θα παίρνουν σαν είσοδο στατιστικά της χρήσης αλλά και προηγούμενων κατανομών σε CPU, έτσι ώστε να προσαρμόζουν ανάλογα την αντίστοιχη κατανομή για να επιτυγχάνεται η μέγιστη χρήση υπολογιστικών πόρων χωρίς να μειώνεται η απόδοση της εφαρμογής.

2. Ο δεύτερος τρόπος είναι η χρήση συστημάτων ασαφής λογικής τα οποία θα εκπαιδεύονται με τα αποθηκευμένα στατιστικά των συστημάτων που αναφέρθηκαν παραπάνω, και με βάση αυτά θα υπολογίζουν και θα προσαρμόζουν την βέλτιστη κατανομή υπολογιστικών πόρων.

1.3 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε επτά κεφάλαια:

Στο Κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο των βασικών τεχνολογιών που σχετίζονται με την πτυχιακή εργασία αυτή. Αρχικά περιγράφεται η τεχνολογία υπολογιστικής νέφους, τα κέντρα δεδομένων, στη συνέχεια η απόδοση των

εφαρμογών υπολογιστικής νέφους, η διαχείριση των υπολογιστικών πόρων και τέλος η τεχνολογία της εικονικοποίησης.

Στο Κεφάλαιο 3 περιγράφεται η αρχιτεκτονική του συστήματος που υλοποιήθηκε για την εκπόνηση της πτυχιακής εργασίας. Περιγράφεται αρχικά η εξομοίωση ενός κέντρου δεδομένων και γενικά όλη η υποδομή που χρησιμοποιήθηκε, ο πυρήνας στρώματος εικονικοποίησης Xen Hypervisor, και τέλος η εφαρμογή μοντελοποίησης διαδικτυακών δημοπρασιών RUBiS.

Το Κεφάλαιο 4 αναλύει τον σχεδιασμό του συστήματος που υλοποιήθηκε, την λογική του πηγαίου κώδικα που αναπτύχθηκε αλλά και τον σχεδιασμό των διαφόρων συστημάτων ελέγχου που χρησιμοποιήθηκαν στην εργασία αυτή. Περιγράφεται επίσης ο τρόπος σύνδεσης των υποσυστημάτων και οι διάφορες τροποποιήσεις που υπέστησαν.

Στο Κεφάλαιο 5 παρουσιάζεται ο έλεγχος σωστής λειτουργίας και τα πειραματικά αποτελέσματα με διάφορα σενάρια φόρτων εργασίας. Τέλος γίνεται μια σύγκριση μεταξύ τους και εξάγονται τα διάφορα συμπεράσματα.

Το Κεφάλαιο 6 δίνει συνοπτικά μια σύντομη αναφορά σε προηγούμενες σχετικές εργασίες με την παρούσα, αλλά και η συνεισφορά αυτής της πτυχιακής εργασίας.

Τέλος στο Κεφάλαιο 7 δίνεται μια συνοπτική ανακεφαλαίωση στο σύστημα αυτόματης κατανομής πόρων και κάποιες γενικές σκέψεις και απόψεις για μελλοντικές εργασίες.

1.4 Δημοσιεύσεις

Η παρακάτω δημοσίευση, βασίστηκε στα αποτελέσματα που προέκυψαν από την παρούσα πτυχιακή εργασία.

1. *Evagoras Makridis, Kyriakos Deliparaschos, Evangelia Kalyvianaki and Themistoklis Charalambous. Robust Dynamic CPU Resource Provisioning in Virtualized Servers. In Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation, (ETFA), 2017.*

Στη δημοσίευση υλοποιήθηκε το φίλτρο MCC-KF και έγινε σύγκριση με άλλα φίλτρα που χρησιμοποιήθηκαν στην παρούσα πτυχιακή εργασία, όπως το φίλτρο Kalman και το φίλτρο $\mathcal{H}_\infty$.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Οι εφαρμογές τεχνολογίας νέφους εμφανίζονται όλο και περισσότερο στη ζωή μας με την πάροδο των χρόνων. Οι βασικοί λόγοι για αυτή την ανάπτυξη των εφαρμογών τεχνολογίας νέφους είναι η υψηλή διαθεσιμότητα σε υπολογιστικούς πόρους. πράγμα που επηρεάζει άμεσα την επίδοση των εφαρμογών, αλλά και η εύκολη επεκτασιμότητα τους με λιγότερο κόστος. Για να καταστεί δυνατή η ανάπτυξη εφαρμογών τεχνολογίας νέφους απαιτούνται σύγχρονα κέντρα δεδομένων αλλά και η συνεισφορά της τεχνολογίας της εικονικοποίησης με σκοπό τη σωστή διαχείριση των υπολογιστικών πόρων. Στο κεφάλαιο αυτό παρουσιάζεται το θεωρητικό υπόβαθρο για τις εφαρμογές τεχνολογίας νέφους καθώς και οι διάφορες τεχνολογίες εικονικοποίησης μηχανών οι οποίες μπορούν να παίξουν τον βασικό ρόλο στην διαχείριση των υπολογιστικών πόρων. Επίσης σε αυτό το κεφάλαιο αναλύονται και διάφοροι όροι απόδοσης εφαρμογών υπολογιστικής νέφους.

2.1 Τεχνολογία υπολογιστικής νέφους

Με την πάροδο των χρόνων παρατηρείται όλο και περισσότερο μια αύξηση στους χρήστες που χρησιμοποιούν εφαρμογές τεχνολογίας νέφους. Η τεχνολογία αυτή βασίζεται στην διαδικτυακή υπολογιστική η οποία παρέχει υπολογιστικούς πόρους προς κοινή χρήση σε όλους τους χρήστες που ζητούν δεδομένα ή υπηρεσίες μια δεδομένη στιγμή. Σχεδόν όλες οι μεγάλες επιχειρήσεις έχουν την δυνατότητα να παρέχουν στους χρήστες διάφορες υπηρεσίες, πλατφόρμες ή ακόμη και υποδομές, χρησιμοποιώντας αποκλειστικά δικά τους κέντρα δεδομένων ή ακόμη και κέντρα δεδομένων από τα οποία ενοικιάζουν κατά κάποιο τρόπο υπολογιστικούς αλλά και διάφορων άλλων ειδών πόρους όπως δικτύου, μνήμης κλπ. Με την δυνατότητα αυτή, οι μικρές επιχειρήσεις μπορούν να ενοι-

κιάζουν υπολογιστικούς πόρους από κέντρα δεδομένων τρίτων, χωρίς να βρίσκονται στους δικούς τους χώρους και χωρίς να έχουν οποιαδήποτε υποχρέωση για αναβάθμιση ή συντήρηση εξοπλισμού των κέντρων δεδομένων. Με αυτό τον τρόπο οι επιχειρήσεις μπορούν να αποφύγουν το μεγάλο κόστος της υποδομής ενός κέντρου δεδομένων.

2.1.1 Εφαρμογές τεχνολογίας υπολογιστικής νέφους

Οι εφαρμογές τεχνολογίας υπολογιστικής νέφους είναι ένας τύπος εφαρμογών που αναπτύσσεται σε δίκτυα εξυπηρετητών απομακρυσμένης πρόσβασης οι οποίες μπορούν να παρέχουν την δυνατότητα κοινής χρήσης των υπολογιστικών τους πόρων μέσω του δικτύου αυτού. Μερικές εφαρμογές υπολογιστικής νέφους μπορεί να είναι για λόγους επικοινωνίας, παροχής πληροφοριών, υπηρεσίες οικονομικών, ψυχαγωγίας κ.α. Οι εφαρμογές αυτές δεν διαφέρουν στην λογική ανάπτυξης της εφαρμογής αλλά διαφέρουν στον τρόπο αλληλεπίδρασης και επικοινωνίας με τους χρήστες. Για παράδειγμα, ένας χρήστης μπορεί να εκμεταλλεύεται τις υπηρεσίες μιας εφαρμογής υπολογιστικής νέφους χωρίς να καταναλώνει υπολογιστικούς πόρους από τον δικό του προσωπικό υπολογιστή αλλά καταναλώνοντας τους πόρους που κατανεμήθηκαν από το κέντρο δεδομένων προς την διάθεση της εφαρμογής.

Η τεχνολογία υπολογιστικής νέφους χωρίζεται σε τρεις βασικές υπηρεσίες. Το Σχήμα 2.1 παρουσιάζει τα τρία μοντέλα ή αλλιώς τις τρεις βασικές υπηρεσίες των εφαρμογών τεχνολογίας νέφους.

1. Λογισμικό ως υπηρεσία νέφους (SaaS - Software as a Service)

   Οι εφαρμογές αυτές δεν απαιτούν οποιαδήποτε εγκατάσταση λογισμικού, αλλά είναι διαθέσιμες στους χρήστες μέσω των προγραμμάτων περιήγησης (browsers) στο διαδίκτυο σε υπολογιστές, έξυπνα τηλέφωνα ταμπλέτες κ.α. Οι χρήστες που χρησιμοποιούν αυτόν τον τύπο εφαρμογών χρεώνονται με βάση των αριθμό των τελικών λογαριασμών, τον όγκο δεδομένων και τον χρόνο χρήσης της εφαρμογής. Στο μοντέλο αυτό ο χρήστης έχει περιορισμένο έλεγχο στο υλικό αλλά και στο λογισμικό στο οποίο είναι αναπτυγμένη η εφαρμογή. Το πιο ευρέως γνωστό παράδειγμα Λογισμικού ως Υπηρεσία είναι τα Google Apps.

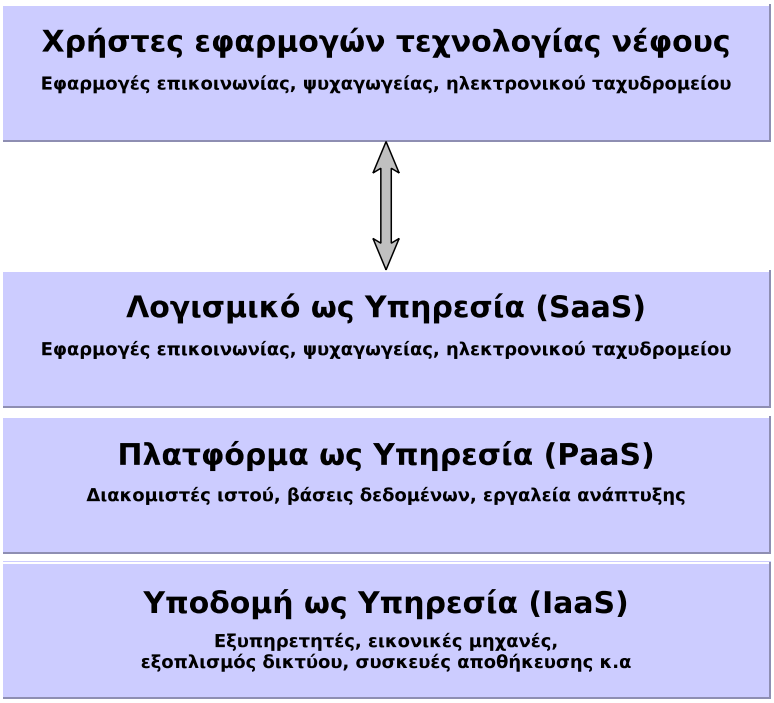
2. Πλατφόρμα ως υπηρεσία νέφους (PaaS - Platform as a Service)

   Το μοντέλο της Πλατφόρμας ως υπηρεσία νέφους παρέχει ένα περιβάλλον ανάπτυξης εφαρμογών, αποκρύπτοντας το υλικό αλλά και το λειτουργικό σύστημα στο οποίο λειτουργεί η πλατφόρμα αυτή. Απευθύνεται σε

προγραμματιστές και χρησιμοποιείται για την γρήγορη ανάπτυξη εφαρμογών διαδικτύου πράγμα που ευκολύνει τους χρήστες στη διάθεση των εφαρμογών αλλά και στην αποφυγή αγοράς εξειδικευμένου εξοπλισμού ή και λογισμικού. Η Πλατφόρμα ως υπηρεσία νέφους απλοποιεί την ανάπτυξη των εφαρμογών αφού υπάρχει η αφαιρετικότητα μεταξύ της υποδομής στην οποία στηρίζεται η πλατφόρμα με την ίδια την πλατφόρμα. Γνωστή Πλατφόρμα ως υπηρεσία νέφους είναι το Google App Engine (GAE).

3. Υποδομή ως υπηρεσία νέφους (IaaS - Infrastructure as a Service)

   Απευθύνεται κυρίως σε διαχειριστές δικτύων και υπολογιστικών συστημάτων οι οποίοι μπορούν να διαχειρίζονται εικονικές μηχανές, λειτουργικά συστήματα, υπολογιστικούς πόρους κ.α. Η υπηρεσία αυτή χρησιμοποιείται με σκοπό την παροχή των διάφορων υπηρεσιών που μπορεί να προσφέρει μια υποδομή υπολογιστικής νέφους, δηλαδή την δημιουργία και συντήρηση των εικονικών μηχανών που εικονικοποιούν τις διάφορες εφαρμογές νέφους. Ο χρήστης με τον τρόπο αυτό δεν χρειάζεται να αγοράσει εξυπηρετητές, διαδικτυακό εξοπλισμό και λογισμικά αλλά τα ενοικιάζει ανάλογα με τις προδιαγραφές που θέτει ο ίδιος. Παράδειγμα για το συγκεκριμένο μοντέλο είναι η Amazon Web Services (AWS).

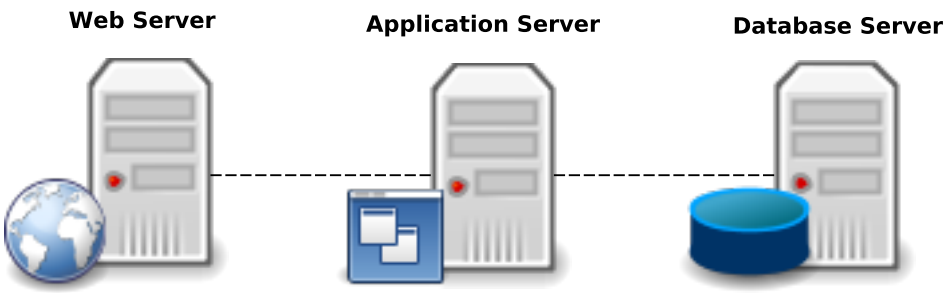


Σχήμα 2.1: Μοντέλα εφαρμογών τεχνολογίας νέφους

2.2 Κέντρα δεδομένων

Η εκθετική αύξηση στις διαδικτυακές εφαρμογές έχει ως αποτέλεσμα την ανάγκη για δημιουργία σύγχρονων κέντρων δεδομένων έτσι ώστε να παρέχουν στους χρήστες την δυνατότητα αναζήτησης επεξεργασίας και αποθήκευσης δεδομένων μέσω του διαδικτύου. Η αύξηση αυτή στις δικτυακές εφαρμογές οφείλεται και στις δυνατότητες των σημερινών έξυπνων συσκευών να επικοινωνούν διαδικτυακά σε αυτό το δίκτυο που ονομάζεται υπολογιστικό νέφος. Οι παραπάνω ανάγκες δημιουργούν τυχαίες κατανομές ζήτησης σε υπολογιστικούς πόρους πράγμα το οποίο δημιουργεί απότομες αυξομοιώσεις ζήτησης υπολογιστικών πόρων στα κέντρα δεδομένων. Τα σύγχρονα κέντρα δεδομένων έχουν την δυνατότητα να φιλοξενούν μια πληθώρα από υπολογιστικά συστήματα και συστήματα που έχουν άμεση σχέση με την ανάπτυξη των διαδικτυακών εφαρμογών όπως είναι τα συστήματα τηλεπικοινωνιών και αποθήκευσης. Ο βασικός στόχος των κέντρων δεδομένων είναι να παράγουν ή και να επεξεργάζονται δεδομένα με το λιγότερο δυνατό κόστος στον λιγότερο δυνατό χρόνο. Τα κέντρα δεδομένων από πλευράς υπολογιστικού υλικού συνθέτονται από εξυπηρετητές, εξοπλισμό δικτύου και αποθήκευσης. Αποτελούνται από πολλές βαθμίδες δηλαδή από πολλά είδη εξυπηρετητών βάση των λειτουργιών που τους ανατίθεται να κάνουν και έτσι ονομάζονται κέντρα δεδομένων πολλών βαθμίδων (multi-tier). Η κάθε βαθμίδα αποτελείται από πολλούς εξυπηρετητές του ίδιου τύπου. Στο Σχήμα 2.2 παρουσιάζεται το μοντέλο τριών βαθμίδων το οποίο αποτελείται από:

1. *Διακομιστές ιστού (Web Servers)*
2. *Εξυπηρετητές εφαρμογών (Application Servers)*
3. *Βάσεις δεδομένων (Databases)*



Σχήμα 2.2: Κέντρο δεδομένων τριών βαθμίδων

2.2.1 Διακομιστής Ιστού

Ο διακομιστής ιστού είναι ένα υπολογιστικό σύστημα το οποίο επεξεργάζεται τα αιτήματα των χρηστών μέσω του πρωτοκόλλου HTTP (Hypertext Transfer Protocol) το οποίο είναι το βασικό δικτυακό πρωτόκολλο που χρησιμοποιείται για την διάδοση της πληροφορίας στον παγκόσμιο ιστό [2]. Οι διακομιστές ιστού έχουν την δυνατότητα να δέχονται πολλαπλά αιτήματα από διαφορετικούς χρήστες ταυτόχρονα, χωρίς αυτό να σημαίνει πως δεν επηρεάζονται και οι επιδόσεις τους. Οι χρήστες έχουν μια έμμεση επικοινωνία με τους διακομιστές ιστού, οι οποίοι φιλοξενούν τις διάφορες ιστοσελίδες, μέσω των προγραμμάτων περιήγησης όπως π.χ., Mozilla Firefox, Internet Explorer, Google Chrome κ.α. Ο βασικός στόχος του διακομιστή ιστού είναι να λαμβάνει τα αιτήματα των χρηστών και στη συνέχεια με βάση τα χαρακτηριστικά του αιτήματος να καθορίζει στον εξυπηρετητή εφαρμογών το είδος της απάντησης που πρέπει να επιστρέψει στον διακομιστή ιστού και αυτός με τη σειρά του στον χρήστη. Η απάντηση του διακομιστή ιστού μπορεί να περιέχει δεδομένα τα οποία εξάχθηκαν από την βάση δεδομένων ή ακόμα και στατικά δεδομένα που υπάρχουν στην σελίδα του διακομιστή ιστού. Αυτά τα δεδομένα μπορεί να είναι αρχεία HTML, εικόνες πολυμέσα κ.α. Ο αριθμός των χρηστών που είναι συνδεδεμένοι την ίδια στιγμή στην σελίδα της εφαρμογής και βασικά στον διακομιστή ιστού, καθώς και το είδος των αιτημάτων που «στέλνει» στον διακομιστή, είναι ανάλογος με τον χρόνο που χρειάζεται ο διακομιστής ιστού να απαντήσει στα αιτήματα.

2.2.2 Εξυπηρετητής Εφαρμογών

Ο εξυπηρετητής εφαρμογών δίνει την δυνατότητα στον χρήστη να εκτελεί διάφορες υπηρεσίες που προσφέρει, μέσω της σελίδας του διακομιστή ιστού. Ο στόχος του εξυπηρετητή εφαρμογών είναι να λάβει τις ανάλογες εντολές από τον διακομιστή ιστού και να προχωρήσει σε κάποια επεξεργασία ή δημιουργία δεδομένων και στη συνέχεια να τα επιστρέψει πίσω στη σελίδα του διακομιστή ιστού, να τα αποθηκεύσει ή ακόμη και να τα ανανεώσει στη βάση δεδομένων. Εκτός από την βασική του λειτουργία αυτή, είναι υπεύθυνος για την ισορροπία του φόρτου εργασίας καθώς και για την ασφάλεια των δεδομένων αλλά και της ίδιας της εφαρμογής.

2.2.3 Βάση Δεδομένων

Ένας από τους βασικούς λόγους για τους οποίους ένας χρήστης επισκέπτεται μια σελίδα είναι η αναζήτηση ή αποθήκευση πληροφοριών, που θα ανακτη-

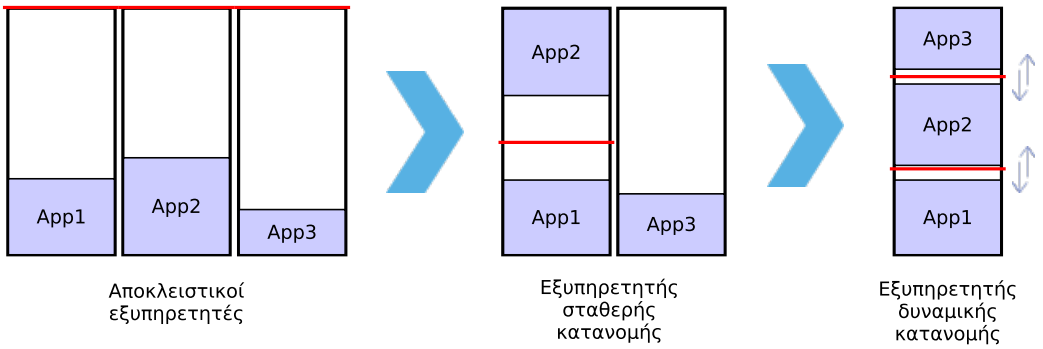
θούν ή θα καταχωρηθούν αντίστοιχα, από μια βάση δεδομένων. Αυτές οι πληροφορίες αποθηκεύονται στις βάσεις δεδομένων έτσι ώστε να χρησιμοποιηθούν όταν ζητηθούν από τους χρήστες σε μελλοντικούς χρόνους. Κάθε χρήστης μπορεί έμμεσα να λάβει τις πληροφορίες που αναζητά, να τις επεξεργαστεί αλλά ακόμη και να τις διαγράψει εάν δεν του είναι χρήσιμες πλέον. Η βάση δεδομένων λαμβάνει τα αιτήματα του διακομιστή ιστού απευθείας ή δια μέσου του εξυπηρετητή εφαρμογών και αναζητά ή γράφει νέα δεδομένα σε αυτήν.

2.3 Απόδοση εφαρμογών

Μια εφαρμογή περιγράφεται από το μέγεθος του φόρτου εργασίας που έχει να εξυπηρετήσει μια δεδομένη στιγμή. Ο φόρτος εργασίας μιας εφαρμογής εξαρτάται από πολλές παραμέτρους οι οποίες μπορεί να είναι ο αριθμός των χρηστών που στέλνουν αιτήματα μια δεδομένη στιγμή και ο αριθμός των αιτημάτων ίδιου τύπου της εφαρμογής. Ο φόρτος εργασίας μπορεί να χαρακτηριστεί επίσης και από την συχνότητα των αιτημάτων που στέλνουν οι χρήστες στην εφαρμογή. Εάν η συχνότητα αυτή μεταβάλλεται με τον χρόνο, τότε ο φόρτος εργασίας θεωρείται δυναμικός. Σε περιπτώσεις μεγάλων παγκόσμιων συμβάντων όπως για παράδειγμα το Παγκόσμιο Κύπελλο ποδοσφαίρου [3] παρατηρούνται μεγάλες αλλαγές στην συχνότητα των αιτημάτων αλλά και στον αριθμό των χρηστών που χρησιμοποιούν τις διάφορες εφαρμογές, δημιουργώντας έτσι μεγάλες και απότομες αλλαγές στον φόρτο εργασίας των εφαρμογών. Σε αντίθετη περίπτωση όταν η συχνότητα αιτημάτων είναι σχετικά σταθερή, τότε ο φόρτος εργασίας θεωρείται εύκολα προβλέψιμος και έτσι ο χρόνος απάντησης στα αιτήματα των χρηστών παραμένει χαμηλός. Για τον λόγο αυτό πρέπει να εξαχθούν τα αναγκαία στατιστικά για την αξιολόγηση των εφαρμογών όπως για παράδειγμα ο μέσος χρόνος απάντησης στα αιτήματα των χρηστών mRT, ο αριθμός των ολοκληρωμένων αιτημάτων ανά δευτερόλεπτο κ.α. Στην εργασία αυτή η εφαρμογή μας αξιολογήθηκε αποκλειστικά με βάση τον μέσο χρόνο απάντησης στα αιτήματα των χρηστών (mRT). Στη παρούσα πτυχιακή εργασία χρησιμοποιήθηκε το mRT για την εξακρίβωση της απόδοσης των εφαρμογών αφού αποτελεί την πιο αντιπροσωπευτική παράμετρο για εφαρμογές τεχνολογίας νέφους. Ο χρόνος αυτός έχει άμεση σχέση με το πλήθος των χρηστών που μπορεί να εξυπηρετήσει η εφαρμογή. Η αξιοπιστία και η απόδοση της εφαρμογής μπορεί να κρατηθεί σταθερή με το να ελέγχεται ο μέσος χρόνος απάντησης αιτημάτων mRT έτσι ώστε να παραμένει στο επιτρεπτό όριο που καθορίζεται από την ποιότητα της υπηρεσίας (QoS) η οποία δηλώθηκε κατά την συμφωνία της υπηρεσίας (SLA) μεταξύ του παροχέα της υπηρεσίας και του πελάτη.

2.4 Κατανομή υπολογιστικών πόρων

Υπολογιστικοί πόροι ονομάζονται όλα τα στοιχεία ενός υπολογιστικού συστήματος που λαμβάνουν μέρος σε κάποιο υπολογισμό ή εργασία. Μερικά από αυτά τα στοιχεία είναι η κεντρική μονάδα επεξεργασίας (CPU), η μνήμη τυχαίας προσπέλασης (RAM), ο αποθηκευτικός χώρος και η διακίνηση δικτύου (Network Bandwidth). Η κατανομή των υπολογιστικών πόρων μπορεί να παίξει μεγάλο ρόλο στην ανάπτυξη και διαχείριση των εφαρμογών τεχνολογίας νέφους, στην εξοικονόμηση ενέργειας αλλά και κόστους εγκατάστασης στα κέντρα δεδομένων. Αφού ο στόχος των εφαρμογών αυτών είναι να κρατήσει την ποιότητα υπηρεσίας που παρέχει στα επιτρεπτά όρια του QoS στους χρήστες, οι υπολογιστικοί πόροι πρέπει να είναι σε θέση να κατανέμονται αυτόματα στις εφαρμογές που έχουν ανάγκη λόγω μεγάλης ζήτησης. Παρακάτω στο Σχήμα 2.3 παρουσιάζεται ένα παράδειγμα των τριών τρόπων ανάπτυξης εφαρμογών.



Σχήμα 2.3: Παράδειγμα ανάπτυξης τριων εφαρμογών σε μια φυσική μηχανή με την μέθοδο δυναμικής κατανομής υπολογιστικών πόρων.

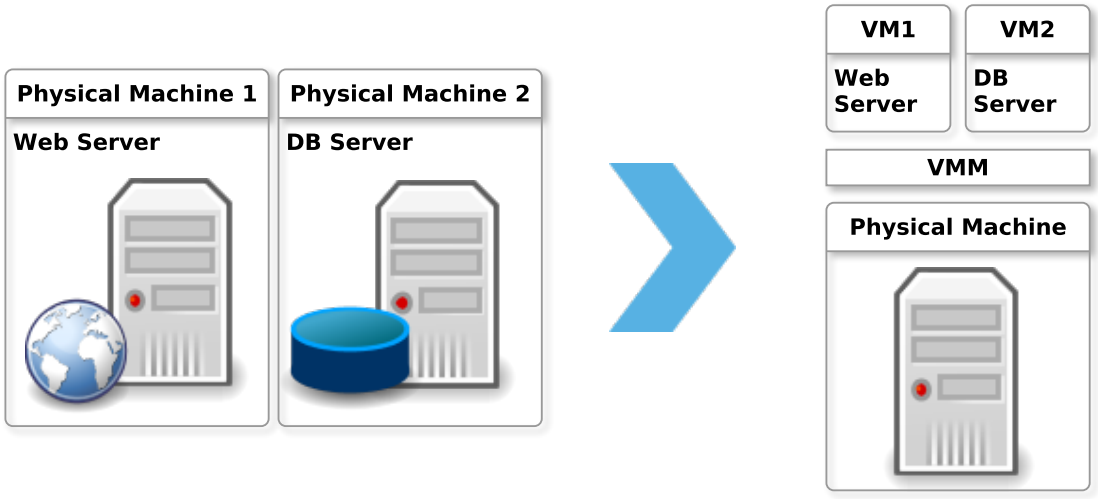
Στα αριστερά του σχήματος, απεικονίζεται ο κλασικός τρόπος, όπου κάθε εφαρμογή βρίσκεται αποκλειστικά σε μια φυσική μηχανή με αποτέλεσμα οι υπολογιστικοί πόροι που δεν χρησιμοποιούνται από την εφαρμογή να χάνονται. Πολλές αναφορές δείχνουν ότι οι περισσότεροι εξυπηρετητές, χρησιμοποιούν μόνο το 15% - 20% των δυνατοτήτων τους, χωρίς να εκμεταλλεύονται το υπόλοιπο ποσοστό. Σύμφωνα με την αναφορά [4], οι εταιρίες ξόδεψαν επιπλέον $140 δισεκατομμύρια δολάρια σε εξυπηρετητές έτσι ώστε να μπορούν να αναπτύξουν τις εφαρμογές τους, λόγω της μη εκμετάλλευσης χρήσης των υπολογιστικών πόρων στο μέγιστο, στους υφιστάμενους τους εξυπηρετητές. Με άλλα λόγια οι εταιρίες αυτές θα μπορούσαν να λειτουργούν κανονικά τις εφαρμογές τους για επιπλέον τρία χρόνια χωρίς κανένα επιπλέον κόστος εάν χρησιμοποιούσαν σωστά τις δυνατότητες των υφιστάμενων τους εξυπηρετητών. Στο κέντρο του σχήματος παρουσιάζεται η ανάπτυξη εφαρμογών με ένα στατικό τρόπο κατα-

νομής υπολογιστικών πόρων όπου η κατανομή για κάθε εφαρμογή καθορίζεται πριν από την δημιουργία της χωρίς να μπορεί να αλλάξει κατά την διάρκεια λειτουργίας της. Στα δεξιά του σχήματος παρουσιάζεται ο τρόπος ανάπτυξης εφαρμογών κατά τον οποίο όλες οι εφαρμογές βρίσκονται σε μια φυσική μηχανή - εξυπηρετητή, όπου οι κατανομές αλλάζουν δυναμικά με την ζήτηση σε υπολογιστικούς πόρους. Η πρόκληση στο σημείο αυτό είναι η τυχαιότητα που παρατηρείται στους διάφορους φόρτους εργασίας των εφαρμογών οι οποίοι αλλάζουν δυναμικά και τυχαία βάσει τον αριθμό των αιτημάτων που στέλνονται στους εξυπηρετητές. Βασική προϋπόθεση για την σωστή κατανομή των υπολογιστικών πόρων, είναι να γίνει μια μοντελοποίηση του συστήματος. Όπως αναφέρθηκε και παραπάνω, ένας βασικός παράγοντας για την απόδοση μιας εφαρμογής υπολογιστικής νέφους είναι ο χρόνος απόκρισης. Με αυτό τον βασικό παράγοντα αλλά και με την φύση των φόρτων εργασίας των εφαρμογών θα μπορούμε να μελετήσουμε στη συνέχεια την απόδοση του συστήματός.

2.5 Τεχνολογία εικονικοποίησης

Η τεχνολογία εικονικοποίησης (virtualization) μηχανών είναι το αναγκαίο εργαλείο για τη σωστή κατανομή των υπολογιστικών πόρων στις διάφορες εφαρμογές έτσι ώστε να κρατηθεί το QoS στο σωστό επίπεδο. Η τεχνολογία αυτή έχει την δυνατότητα να μετατρέπει τον κάθε φυσικό υπολογιστικό πόρο μιας μηχανής (π.χ, χρόνος επεξεργαστικής ισχύς, μέγεθος μνήμης , αποθηκευτικός χώρος, εύρος ζώνης δικτύου) σε ενιαίους υπολογιστικούς πόρους που μοιράζονται σε πολλές εικονικές μηχανές. Με τον τρόπο αυτό δημιουργείται ένα στρώμα απομόνωσης των εικονικών μηχανών που παρέχουν τις διάφορες τους εφαρμογές και μπορούν να χρησιμοποιηθούν από τους χρήστες με τον ίδιο ακριβώς τρόπο όπως θα χρησιμοποιούσαν την ίδια εφαρμογή εάν αυτή βρισκόταν σε μια φυσική μηχανή. Επίσης με την τεχνολογία αυτή υπάρχει η δυνατότητα μεταφοράς μιας εικονικής μηχανής (VM), από μια φυσική μηχανή σε κάποια άλλη. Με τον τρόπο αυτό επιτυγχάνεται εξοικονόμηση ενέργειας στα κέντρα δεδομένων αφού θα μπορούν έτσι να φιλοξενούνται πολλές εφαρμογές - VMs σε μια φυσική μηχανή, ξεφεύγοντας έτσι από τον κλασικό τρόπο ανάπτυξης εφαρμογών, όπου κάθε εφαρμογή βρισκόταν αποκλειστικά σε μια φυσική μηχανή.

Στα αριστερά του διαγράμματος φαίνεται ένα κλασικό σύστημα δύο φυσικών μηχανών που φιλοξενούν ένα διακομιστή ιστού και μια βάση δεδομένων αντίστοιχα. Η κάθε φυσική μηχανή παρέχει τους υπολογιστικούς της πόρους στις ανάλογες εφαρμογές ξεχωριστά. Στα δεξιά του διαγράμματος φαίνεται μια φυσική μηχανή η οποία παρέχει προς κοινή χρήση τους φυσικούς υπολογιστικούς



Σχήμα 2.4: Τεχνολογία εικονικοποίησης.

της πόρους στις 2 εικονικές μηχανές με την χρήση του Διαχειριστή Εικονικών Μηχανών (VMM).

Υπάρχουν δύο βασικές αρχιτεκτονικές εικονικοποίησης μηχανών οι οποίες αναπτύσσονται σε πολλές υποκατηγορίες. Η Πρώτου τύπου αρχιτεκτονική (Type 1) ονομάζεται αρχιτεκτονική εικονικοποίησης γυμνού-μετάλλου (bare-metal virtualization), ενώ η Δεύτερου τύπου (Type 2) ονομάζεται αρχιτεκτονική εικονικοποίησης επιφανείας(hosted virtualization).

2.5.1 Εικονικοποίηση επιπέδου υλικού

Στην αρχιτεκτονική πρώτου τύπου, ο υπερεπόπτης φορτώνεται απευθείας πάνω στο υλικό της φυσικής μηχανής, όπως ένα λειτουργικό σύστημα. Στην ουσία είναι ένα τροποποιημένο λειτουργικό σύστημα ή αλλιώς ένας πυρήνας και δεν προϋποθέτει οποιοδήποτε εγκαταστημένο λειτουργικό σύστημα. Για τον λόγο αυτό έχει άμεση επικοινωνία με τους φυσικούς πόρους της μηχανής, πράγμα το οποίο επιτρέπει την διαχείριση και την υψηλή διαθεσιμότητα σε υπολογιστικούς πόρους, την καλύτερη απόδοση και μεγαλύτερη επεκτασιμότητα. Υπάρχουν τρεις τεχνικές εικονικοποίησης σε επίπεδο υλικού όπου αναφέρονται παρακάτω. Μερικά παραδείγματα υπερεπόπτων εικονικοποίησης σε επίπεδο υλικού είναι π.χ., το Xen Hypervisor , το Hyper-V, το ESX/ESXi κ.α.

Πλήρης Εικονικοποίηση - Full Virtualization

Στον συγκεκριμένο τύπο εικονικοποίησης, ένα εξειδικευμένο λογισμικό που λέγεται και υπερεπόπτης (hypervisor) δημιουργεί μια εξομοίωση του υλικού της μηχανής, πάνω στις εικονικές μηχανές. Έτσι όλα τα βασικά χαρακτηριστικά του υλικού (hardware) που συνθέτουν την λειτουργία της φυσικής μηχανής οφείλουν να λειτουργούν κανονικά και στην κάθε εικονική μηχανή. Με αυτόν τον τρόπο η φυσική μηχανή μπορεί να φιλοξενήσει πολλών ειδών λειτουργικά συστήματα.

Παραεικονικοποίηση - Paravirtualization

Στην τεχνική αυτή, ο υπερεπόπτης δεν προσομοιώνει το υλικό της φυσικής μηχανής στις εικονικές μηχανές, αλλά παρέχει σε αυτές μια διεπαφή - διασύνδεση με το υλικό έτσι ώστε να επιτρέπεται η εκτέλεση των διαφόρων λειτουργικών συστημάτων. Η τεχνική αυτή παρέχει υψηλή απόδοση λόγω του ότι δεν προσομοιώνονται οι υπολογιστικοί πόροι αποκλειστικά σε εικονικές μηχανές.

2.5.2 Εικονικοποίηση επιπέδου λειτουργικού συστήματος

Σε αυτή την αρχιτεκτονική, ο υπερεπόπτης δεν τρέχει απευθείας πάνω στο υλικό της φυσικής μηχανής αλλά εγκαθίσταται σαν εφαρμογή στο υφιστάμενο λειτουργικό σύστημα. Για τον λόγο αυτό, θα πρέπει να προηγηθεί η εγκατάσταση κάποιου λειτουργικού συστήματος. Στην αρχιτεκτονική αυτή, οι εικονικές μηχανές επικαλούνται τα προγράμματα οδήγησης (drivers) από το υφιστάμενο λειτουργικό σύστημα έτσι ώστε να τρέξουν τις δικές τους εφαρμογές. Μερικά παραδείγματα λογισμικών εικονικοποίησης επιφανείας είναι το Workstation, το Virtual Server, το Fusion κ.α.

Εικονικοποίηση Λειτουργικού Συστήματος - OS Virtualization

Η εικονικοποίηση Λειτουργικών συστημάτων γίνεται εφικτή με την εγκατάσταση ενός λογισμικού το οποίο λέγεται και λογισμικό στρώματος εικονικοποίησης, πάνω στο υφιστάμενο λειτουργικό σύστημα στο οποίο τρέχει η φυσική μηχανή. Με αυτό τον τρόπο, ο πυρήνας (kernel) του λειτουργικού συστήματος, μπορεί να δημιουργήσει πολλαπλά απομονωμένα εικονικά λειτουργικά συστήματα. Η εικονική πλέον μηχανή που τρέχει πάνω σε αυτό το στρώμα είναι ένα περιβάλλον ενός εικονικού λειτουργικού συστήματος το οποίο περιγράφεται από τα δικά του χαρακτηριστικά και ρυθμίσεις. Η εικονική μηχανή αυτή δεν μπορεί να έχει άμεση επικοινωνία με τις υπόλοιπες εικονικές μηχανές αλλά ούτε και με τους φυσικούς πόρους της φυσικής μηχανής.

Κεφάλαιο 3

Αρχιτεκτονική του συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική του συστήματος το οποίο αναπτύχθηκε και τροποποιήθηκε ανάλογα έτσι ώστε να γίνει η σωστή διακρίβωση της σωστής λειτουργίας.

Αρχικά στην Ενότητα 3.1 παρουσιάζεται ένα γενικό διάγραμμα της υποδομής για την αρχιτεκτονική του συστήματος που υλοποιήθηκε. Στην Ενότητα 3.2 αναφέρονται όλα τα υπολογιστικά συστήματα από πλευράς υλικού που χρησιμοποιήθηκαν αλλά και οι ανάλογες ρυθμίσεις που έγιναν για να μπορέσουν τα υπόλοιπα μέρη του συστήματος να λειτουργούν κανονικά. Η Ενότητα 3.3 αναφέρεται στην τεχνολογία εικονικοποίησης που χρησιμοποιήθηκε στην εργασία αυτή και πιο συγκεκριμένα στον τρόπο εγκατάστασης του υπερεπόπτη Xen [5] ο οποίος χρησιμοποιήθηκε για το σύστημά μας. Τέλος η Ενότητα 3.4 αναφέρει όλη την διαδικασία ανάπτυξης της διαδικτυακής εφαρμογής δημοπρασιών $RUBiS$ για κάθε στοιχείο του συμπλέγματος που το περιγράφει. Επίσης στην Ενότητα 3.4 αναφέρονται και οι ανάλογες τροποποιήσεις που έγιναν για το σύστημα μας αλλά και για τον εξομοιωτή χρηστών έτσι ώστε να εξάγουμε την βασική μέτρηση για την απόδοση της εφαρμογής, δηλαδή το mRT.

3.1 Εισαγωγή στην αρχιτεκτονική του συστήματος

Για να επιτευχθεί η σωστή εξακρίβωση της απόδοσης του συστήματος, εξομοιώθηκε ένα κέντρο δεδομένων στο οποίο τρέχει μια εφαρμογή διαδικτυακών δημοπρασιών που ονομάζεται $RUBiS$. Η εφαρμογή αυτή δέχεται ένα πλήθος αιτημάτων από ένα εξομοιωτή χρηστών (Client Emulator) που εξομοιώνει τις διάφορες συμπεριφορές των χρηστών κατά την περιήγηση τους στην εφαρμογή αυτή. Ο χρόνος απόκρισης των αιτημάτων αυτών καταγράφεται έτσι ώστε να βρεθεί ο αριθμός των χρηστών στον οποίο παραβιάζεται το QoS που καθορίστηκε

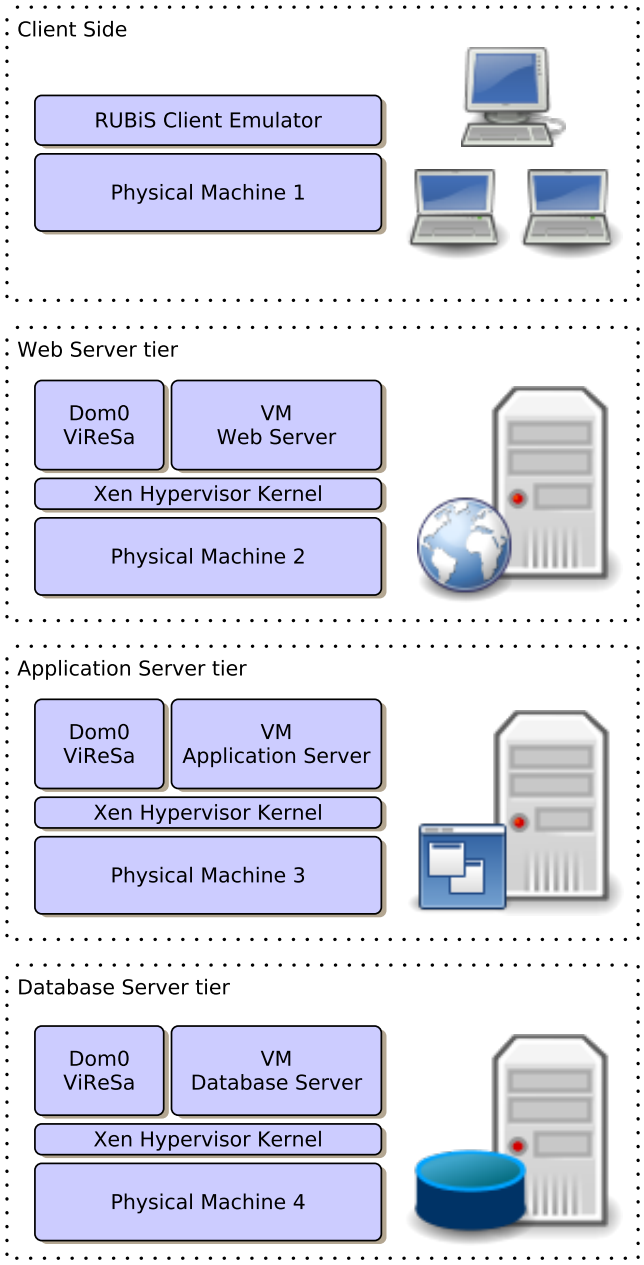
εξ' αρχής. Στη συνέχεια με την ανάπτυξη ενός πηγαίου κώδικα, καταγράφονται οι χρήσεις σε CPU για κάθε στοιχείο της εφαρμογής ανά δευτερόλεπτο, και ανάλογα τα διάφορα συστήματα ελέγχου που σχεδιάστηκαν, προσαρμόζουν δυναμικά την κατανομή των πόρων έτσι ώστε να επιτυγχάνεται εξοικονόμηση σε υπολογιστικούς πόρους για να μπορούν άλλες εφαρμογές να αναπτυχθούν στις ίδιες φυσικές μηχανές.

Όλα τα βασικά μέρη του συστήματος, φαίνονται στο Σχήμα 3.1. Το λογισμικό για κάθε βαθμίδα (tier) εξυπηρετητή (Web Server, Application Server, Database Server) αναπτύχθηκε σε διαφορετικές εικονικές μηχανές αλλά και σε διαφορετικές φυσικές μηχανές έτσι ώστε η διαχείριση των υπολογιστικών πόρων να είναι ανεξάρτητη για κάθε είδος εξυπηρετητή. Αυτή η αρχιτεκτονική είναι η πλέον διαδεδομένη στο χώρο της ανάπτυξης εφαρμογών υπολογιστικής νέφους στα κέντρα δεδομένων, με την διαφορά ότι οι χρήστες δεν προσομοιώνονται αλλά είναι οι πραγματικοί χρήστες που χρησιμοποιούν τις διάφορες εφαρμογές στους περιηγητές ιστού.

3.2 Εγκατάσταση και τροποποίηση της υποδομής

Στην ενότητα αυτή περιγράφεται το υλικό που χρησιμοποιήθηκε, οι διάφορες τροποποιήσεις που έγιναν για την ανάπτυξη του συστήματος, αλλά και οι διάφορες δυσκολίες που βρέθηκαν κατά την διάρκεια έτσι ώστε να εξομοιωθεί το κέντρο δεδομένων. Για την εξομοίωση της λειτουργίας ενός κέντρου δεδομένου χρησιμοποιήθηκαν τρεις φυσικές μηχανές οι οποίες φιλοξενούν τον υπερεπόπτη Xen ξεχωριστά. Η πρώτη φυσική μηχανή $PM1$ χρησιμοποιήθηκε για να στέλνει τα αιτήματα των χρηστών στην εφαρμογή διαδικτυακών δημοπρασιών $RUBiS$ η οποία αποτελείται από δύο βασικά στοιχεία που δημιουργούν το σύμπλεγμα του $RUBiS$. Το πρώτο στοιχείο αναπτύχθηκε στη φυσική μηχανή $PM2$ και φιλοξενεί την εικονική μηχανή όπου εγκαταστάθηκε ο διακομιστής ιστού. Το δεύτερο στοιχείο αναπτύχθηκε στη φυσική μηχανή $PM4$ και φιλοξενεί την εικονική μηχανή στην οποία εγκαταστάθηκε και ρυθμίστηκε η βάση δεδομένων. Η φυσική μηχανή $PM3$ φιλοξενεί κανονικά την δική της εικονική μηχανή όμως στην παρούσα εργασία δεν χρησιμοποιήθηκε η βαθμίδα του εξυπηρετητή εφαρμογών για την εφαρμογή $RUBiS$. Τα χαρακτηριστικά των φυσικών μηχανών που χρησιμοποιήθηκαν στην εργασία αυτή παρουσιάζονται στον Πίνακα 3.1.

Οι τρεις βασικές αυτές μηχανές τοποθετήθηκαν και εγκαταστάθηκαν μαζί με άλλες τρεις μηχανές που ρυθμίστηκαν με τον ίδιο τρόπο προκειμένου να χρησιμοποιηθούν σε μελλοντικές εργασίες, στο δωμάτιο εξυπηρετητών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών



Σχήμα 3.1: Αρχιτεκτονική της υποδομής και στοιχεία της εφαρμογής RUBiS. (PM1: Εξομοιωτής χρηστών, PM2: Διακομιστής ιστού, PM3: Εξυπηρετητές εφαρμογών, PM4: Βάση δεδομένων)

και Πληροφορικής. Αρχικά έγινε κατάλληλη διαμόρφωση των εξυπηρετητών έτσι ώστε να μπορέσουν να φιλοξενήσουν το λειτουργικό σύστημα που επιλέχθηκε, το οποίο είναι Linux και συγκεκριμένα η διανομή Debian Jessie 8.6 [6]. Βασική προϋπόθεση, ήταν η ενεργοποίηση της υποστήριξης τεχνολογίας εικονικοποίησης μέσω του συστήματος βασικών εισόδων και εξόδων (BIOS). Κατά την

Υλικό	Μοντέλο
ΚΜΕ (CPU)	Intel Xeon 5140 @ 2.33GHz
Μνήμη	1Gb
Σκληρός Δίσκος	Western Digital WD800AJS 80Gb
Κάρτα δικτύου	Broadcom NetXtreme BCM5721
Κάρτα γραφικών	Matrox MGA G200e

Πίνακας 3.1: Πίνακας τεχνικών χαρακτηριστικών φυσικών μηχανών.

εγκατάσταση του λειτουργικού συστήματος σε κάθε μηχανή, επιλέχθηκε μια «ελαφριά» έκδοση γραφικού περιβάλλοντος αφού σκοπός είναι οι εξυπηρετητές να ελέγχονται απομακρυσμένα μέσω ssh [7] με εντολές UNIX. Για τον λόγο αυτό, επιλέχθηκε η εγκατάσταση του εξυπηρετητή ssh κατά την εγκατάσταση του λειτουργικού συστήματος.

3.2.1 Διαμόρφωση στατικής διεύθυνσης πρωτοκόλλου διαδικτύου

Η απομακρυσμένη πρόσβαση στο τερματικό εντολών UNIX με την χρήση ssh, απαιτεί μια στατική διεύθυνση πρωτοκόλλου διαδικτύου (IP) ούτως ώστε κατά την επανεκκίνηση των μηχανών, να μην υπάρχει οποιαδήποτε αλλαγή στη διεύθυνση πράγμα που μπορεί να προκαλέσει δυσκολίες στην σύνδεση μέσω ssh και η απομακρυσμένη πρόσβαση στο τερματικό εντολών να μην είναι εφικτή. Για τον λόγο αυτό έγιναν κρατήσεις διευθύνσεων στο δίκτυο για όλες τις μηχανές που χρησιμοποιήθηκαν έτσι ώστε να δοθούν στατικές διευθύνσεις IP στις μηχανές αυτές.

3.2.2 Διαμέριση Σκληρού Δίσκου

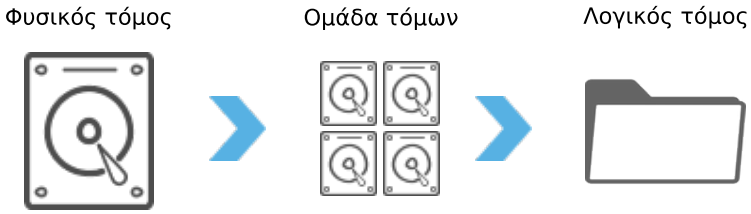
Κατά την διάρκεια της εγκατάστασης του λειτουργικού συστήματος διαμορφώθηκε η διαμέριση (partitioning) του σκληρού δίσκου με σκοπό την απομόνωση των εικονικών μηχανών από την φυσική μηχανή. Στο λειτουργικό σύστημα που χρησιμοποιήθηκε, ο σκληρός δίσκος ονομάστηκε sda και κατά την διαμέριση του προέκυψαν τα διαμερίσματα που παρουσιάζονται στον Πίνακα 3.2.

Για να υπάρξει η αφαιρετικότητα των μελλοντικών εικονικών μηχανών, έγινε η σωστή διαμέριση του σκληρού δίσκου με το να προστεθεί το διαχωριστικό στρώμα (abstraction-layer) ανάμεσα στο λειτουργικό μας σύστημα και στον σκληρό δίσκο που υφίσταται στις μηχανές. Η παραπάνω διαδικασία έγινε με την χρήση του προγράμματος Διαχείρισης Λογικών Τόμων (LVM) που υπάρχει στον πυρήνα του λειτουργικού συστήματος.

Διαμέρισμα	Σημείο προσάρτησης	Μέγεθος
sda1	/boot	250MB
sda2	/	8GB
sda3	swap	2Gb
sda4	reserved for LVM	70GB

Πίνακας 3.2: Πίνακας διαμέρισης σκληρού δίσκου για κάθε φυσική μονάδα.

Αρχικά δημιουργήθηκε ο φυσικός τόμος (Physical Volume) πάνω στο χώρο που επιλέχθηκε κατά την διαμέριση του σκληρού δίσκου, δηλαδή στο $sda4$. Στη συνέχεια δημιουργήθηκε μια ομάδα από τόμους, η οποία ονομάστηκε $vg0$. Σε αυτή την ομάδα μπορούν να δημιουργηθούν οι λογικοί τόμοι όπου κάθε λογικός τόμος θα μπορεί να φιλοξενήσει στη συνέχεια και μια εικονική μηχανή. Αυτό έγινε για κάθε φυσική μηχανή εκτός από την μηχανή $PM1$ η οποία τρέχει το πρόγραμμα εξομοίωσης των πελατών το οποίο στέλνει τα αιτήματα στην εφαρμογή $RUBiS$.



Σχήμα 3.2: Μονάδες Διαχείρισης Λογικών Τόμων.

Αφού εγκαταστάθηκε το LVM2 δημιουργήθηκαν οι ανάλογες μονάδες για όλες τις φυσικές μηχανές. Παρακάτω δίνονται δύο παραδείγματα δημιουργίας φυσικών τόμων, λογικών ομάδων και λογικών τόμων με την χρήση εντολών τερματικού UNIX, για τον διακομιστή ιστού Apache στη φυσική μηχανή $PM2$ και για τον εξυπηρετητή βάσης δεδομένων MySQL στη φυσική μηχανή $PM4$.

```
$ pvcreate /dev/sda4
$ vgcreate vg0 /dev/sda4
$ lvcreate -n Apache -L 10GB vg0
```

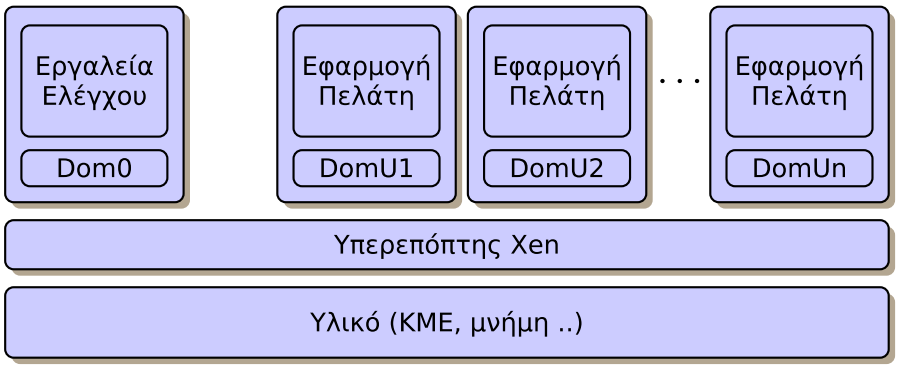
Listing 3.1: Εντολές LVM2, δημιουργία φυσικών τόμων, λογικών ομάδων, λογικού τόμου στην φυσική μηχανή $PM2$

```
$ pvcreate /dev/sda4
$ vgcreate vg0 /dev/sda4
$ lvcreate -n MySQL -L 10GB vg0
```

Listing 3.2: Εντολές LVM2, δημιουργία φυσικών τόμων, λογικών ομάδων, λογικού τόμου στη φυσική μηχανή $PM4$.

3.3 Xen

Το Xen είναι ένα λογισμικό το οποίο μπορεί να προσφέρει διάφορες υπηρεσίες εικονικοποίησης. Αποτελείται από τρεις βασικές οντότητες: α) τον Υπερεπόπτη Xen (Xen Hypervisor - VMM που χρησιμοποιεί ένα μικρο-πυρήνα, microkernel), β) τον τομέα ελέγχου (Dom0), γ) τους φιλοξενούμενους τομείς ή εικονικές μηχανές (DomUs). Το Xen έχει τη δυνατότητα να δημιουργεί, να αποθηκεύει καταστάσεις στον χρόνο, να επανεκκινεί, να τερματίζει ακόμη και να μεταφέρει εικονικές μηχανές σε άλλες φυσικές μηχανές κατά την διάρκεια της λειτουργίας τους.



Σχήμα 3.3: Αρχιτεκτονική του Xen Hypervisor.

3.3.1 Υπερεπόπτης Xen

Ο υπερεπόπτης Xen είναι ένα στρώμα λογισμικού το οποίο μπορεί να παρέχει τις υπηρεσίες εικονικοποίησης, έχοντας άμεση επικοινωνία με το υλικό με αποτέλεσμα να αντικαθιστά το υφιστάμενο λειτουργικό σύστημα. Η ασφάλεια μεταξύ του υλικού με τα φιλοξενούμενα λειτουργικά συστήματα ή αλλιώς εικονικές μηχανές, επιτυγχάνεται με την αρχιτεκτονική του Xen. Αυτό έχει ως αποτέλεσμα την αυξημένη ασφάλεια του συστήματος.

3.3.2 Τομέας ελέγχου - Dom0

Ο τομέας ελέγχου ενεργοποιείται από τον υπερεπόπτη Xen κατά την πρώτη εκκίνηση του συστήματος με τον πυρήνα του Xen. Ο τομέας ελέγχου είναι βασικά το περιβάλλον όπου ο διαχειριστής μπορεί να έχει τον πλήρη έλεγχο των υπολογιστικών πόρων για όλες τις φιλοξενούμενες εικονικές μηχανές.

3.3.3 Φιλοξενούμενοι τομείς - DomUs

Οι φιλοξενούμενοι τομείς ή εικονικές μηχανές είναι ανεξάρτητα λειτουργικά συστήματα τα οποία ελέγχονται από τον τομέα ελέγχου και λειτουργούν ξεχωριστά στο σύστημα. Υπάρχουν δύο τρόποι εικονικοποίησης ενός φιλοξενούμενου τομέα. Ο πρώτος είναι με την τεχνική της Παραεικονικοποίησης (Paravirtualization) η οποία επιτρέπει στο λειτουργικό σύστημα να γνωρίζει ότι τρέχει μέσω ενός υπερεπόπτη αντί απευθείας στο υλικό. Ο δεύτερος τρόπος είναι η Εικονική μηχανή υλικού (Hardware Virtual Machine - HVM) όπου το λειτουργικό σύστημα τρέχει σε ένα εικονικό περιβάλλον αγνοώντας ότι δεν τρέχει απευθείας πάνω στο υλικό. Για αυτή την τεχνική απαιτείται ειδικό υλικό όπως π.χ., επεξεργαστές Intel VT-x και AMD-V.

3.3.4 Χρονοδρομολογητές - Schedulers

Στον υπερεπόπτη Xen παρέχονται διάφοροι χρονοδρομολογητές (schedulers) οι οποίοι είναι ικανοί να αποφασίσουν, μεταξύ των εικονικών επεξεργαστών (vCPUs) των φιλοξενούμενων εικονικών μηχανών, ποιός θα εκτελέσει τις εργασίες που του τέθηκαν να ολοκληρώσει, στον φυσικό επεξεργαστή (pCPU) σε κάποια συγκεκριμένη χρονική στιγμή. Μερικά παραδείγματα χρονοδρομολογητών που χρησιμοποιήθηκαν στο Xen ήταν ο sEDF, BVT, Atropos, Credit2, RTDS. Στη παρούσα εργασία χρησιμοποιήθηκε ο χρονοδρομολογητής Credit αφού ήταν και ο προκαθορισμένος από το Xen.

3.3.5 Διαχείριση υπολογιστικών πόρων

Ο Credit είναι ένας χρονοδρομολογητής ο οποίος έχει την δυνατότητα να διαχειρίζεται του υπολογιστικούς πόρους της κεντρικής μονάδας επεξεργασίας, και να τις κατανέμει ανάλογα με κάποιες παραμέτρους, στις διάφορες εικονικές μηχανές. Μια σημαντική παράμετρος για τον χρονοδρομολογητή είναι η παράμετρος timeslice η οποία καθορίζει τον χρόνο που μπορεί μια εικονική μηχανή να χρησιμοποιήσει την κεντρική μονάδα επεξεργασίας. Αυτό συμβαίνει για κάθε περίοδο της κεντρικής μονάδας επεξεργασίας. Η τυπική τιμή αυτής της παραμέτρου που είναι προεπιλεγμένη από το Xen είναι τα 30ms.

3.3.6 Διαμόρφωση υπερεπόπτη Xen

Στην παρούσα εργασία, αφού έγινε η προσθήκη του πυρήνα Xen στις δύο βασικές φυσικές μηχανές που συνθέτουν την εφαρμογή $RUBiS$, δηλαδή την $PM2$

και $PM4$, δημιουργήθηκαν οι ανάλογες εικονικές μηχανές που θα φιλοξενούν τα λογισμικά του διακομιστή ιστού και της βάσης δεδομένων αντίστοιχα. Βασική προϋπόθεση για την δημιουργία αλλά και τον έλεγχο των εικονικών μηχανών ήταν η εγκατάσταση του $xen - tools$ στον τομέα ελέγχου $Dom0$ της κάθε μηχανής, ο οποίος είναι υπεύθυνος για την εργασία αυτή. Για λόγους παραδείγματος, η δημιουργία της εικονικής μηχανής που φιλοξενεί τον διακομιστή ιστού *Apache* έγινε μέσω του $Dom0$ της φυσικής μηχανής $PM2$. Όπως φαίνεται στις παρακάτω εντολές τερματικού UNIX, το όνομα που επιλέχθηκε για την εικονική μηχανή είναι «*Apache*» για λόγους ευκολίας στον χειρισμό από το σύστημα *ViResA* που θα αναλυθεί στο επόμενο κεφάλαιο. Συγκεκριμένα καθορίστηκαν μεταξύ άλλων, η εικονική μνήμη, ο αριθμός των εικονικών επεξεργαστών *vCPUs*, ο αποθηκευτικός χώρος και η διανομή του λειτουργικού συστήματος της εικονικής μηχανής.

```
$ xen-create-image
       -- hostname Tomcat
       -- memory = 800mb
       -- vcpus = 1
       -- lvm = vg0
       -- dhcp
       -- pygrub
       -- dist = jessie
```

Listing 3.3: Παράδειγμα εντολών τερματικού UNIX για την δημιουργία της εικονικής μηχανής που φιλοξενεί τον διακομιστή ιστού Apache.

Για λόγους απλοποίησης αλλά και της σωστής λειτουργίας των συστημάτων ελέγχου που θα αναφερθούν στην συνέχεια, έγινε ο καθορισμός των $vCPUs$ κάθε εικονικής μηχανής σε ένα μόνο συγκεκριμένο $pCPU$ της φυσικής μηχανής. Λόγω του ότι ο φυσικός επεξεργαστής των μηχανών που χρησιμοποιήθηκαν στην εργασία αυτή, αποτελείται από δύο πυρήνες, το $Dom0$ τέθηκε να λειτουργεί αποκλειστικά με τον ένα πυρήνα, ενώ η εικονική μηχανή που δημιουργήθηκε «*Apache*», στον άλλο πυρήνα του φυσικού επεξεργαστή ($pCPU$).

3.4 *RUBiS*

Αυτή η ενότητα παρουσιάζει την εφαρμογή διαδικτυακών δημοπρασιών $RUBiS$, αλλά και τον τρόπο που αναπτύχθηκε και τροποποιήθηκε το σύμπλεγμα των εικονικών μηχανών έτσι ώστε να επιτευχθεί η σωστή εξαγωγή αποτελεσμάτων. Επίσης η ενότητα αυτή αναφέρεται και στην εφαρμογή δημιουργίας και αποστολής των αιτημάτων (Client Emulator) η οποία εξομοιώνει τους διάφορους χρήστες που χρησιμοποιούν την εφαρμογή $RUBiS$.

3.4.1 Εισαγωγή

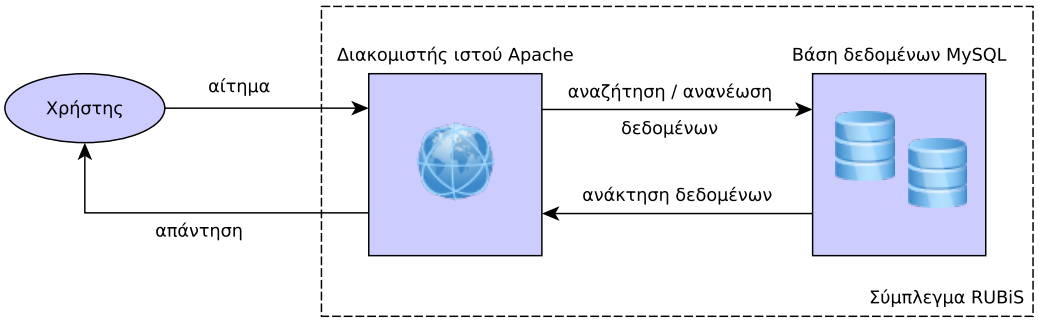
Το RUBiS, Rice University Bidding System [8], είναι μια σελίδα διαδικτυακών δημοπρασιών η οποία δημιουργήθηκε για την μοντελοποίηση του ebay.com. Το RUBiS παρέχει τις περισσότερες δυνατότητες μιας πραγματικής σελίδας διαδικτυακών δημοπρασιών, όπως είναι η αναζήτηση, αγορά, πώληση αντικειμένων, δημιουργία προσφοράς για αντικείμενα και άλλες δυνατότητες όπως προσθήκη σχολίων για άλλους χρήστες, βαθμολόγηση και προβολή άλλων χρηστών.

Αρχικά το RUBiS δημιουργήθηκε για σκοπούς ελέγχου της απόδοσης της εφαρμογής με διάφορες υλοποιήσεις όπως PHP, Java servlets και Enterprise Java Beans (EJB) [9]. Αργότερα η διαδικτυακή εφαρμογή δημοπρασιών RUBiS χρησιμοποιήθηκε καθαρά για σκοπούς αξιολόγησης συστημάτων αυτόματης κατανομής υπολογιστικών πόρων [10], όπως έγινε και στην παρούσα εργασία.

Σε αυτή την ενότητα θα παρουσιαστεί η διαδικτυακή εφαρμογή δημοπρασιών RUBiS και θα αναλυθεί η διάταξη των βαθμίδων της, αλλά και η εφαρμογή εξομοίωσης χρηστών που δημιουργούν τον φόρτο εργασίας στο σύμπλεγμα του RUBiS.

3.4.2 Διάταξη βαθμίδων

Το RUBiS αποτελείται από τρεις βαθμίδες. Η πρώτη είναι η βαθμίδα του διακομιστή ιστού, η δεύτερη η βαθμίδα του εξυπηρετητή εφαρμογής και τέλος η βαθμίδα του εξυπηρετητή της βάσης δεδομένων. Στην παρούσα πτυχιακή εργασία χρησιμοποιήθηκαν μόνο οι δύο βασικές βαθμίδες της εφαρμογής, δηλαδή ο διακομιστής ιστού και ο εξυπηρετητής της βάσης δεδομένων. Στο Σχήμα 3.4 παρακάτω παρουσιάζεται ο τρόπος επικοινωνίας και η «διαδρομή» των αιτημάτων που δημιουργούνται από τους χρήστες καθώς και η επιστροφή τους ως απάντηση πίσω σε αυτούς.



Σχήμα 3.4: Επικοινωνία των διαφόρων βαθμίδων της διαδικτυακής εφαρμογής RUBiS.

Ο διακομιστής ιστού Apache είναι υπεύθυνος να δέχεται τα διάφορα αιτήματα των χρηστών μέσω του περιηγητή ιστού, και με τις ανάλογες εντολές να δημιουργεί και αυτός με την σειρά του τα αντίστοιχα αιτήματα στη βάση δεδομένων MySQL, για αναζήτηση ή ανανέωση δεδομένων που ζήτησε ο χρήστης. Στη συνέχεια τα δεδομένα που ανανεώθηκαν ή αναζητήθηκαν, επιστρέφουν στον διακομιστή ιστού ο οποίος δίνει την απάντηση στον χρήστη σε μορφή σελίδας $HTML$ στο περιβάλλον εργασίας του περιηγητή ιστού. Σε περίπτωση κατά την οποία το αίτημα δεν απαιτεί οποιαδήποτε αλλαγή ή αναζήτηση δεδομένων από την βάση δεδομένων τότε, το αίτημα δεν προωθείται στην βάση δεδομένων για επεξεργασία των δεδομένων, αλλά επιστρέφει η απάντηση απευθείας από τον διακομιστή ιστού.

3.4.3 Βαθμίδα Διακομιστή Ιστού

Όπως αναφέρθηκε και σε προηγούμενα μέρη της εργασίας αυτής, ο διακομιστής ιστού Apache2 εγκαταστάθηκε στο VM που δημιουργήθηκε στην δεύτερη φυσική μηχανή $PM2$. Η υλοποίηση που χρησιμοποιήθηκε για την ανάπτυξη της σελίδας των διαδικτυακών δημοπρασιών $RUBiS$ έγινε με την ευρέως διαδεδομένη γλώσσα προγραμματισμού PHP η οποία είναι ανοικτού κώδικα και είναι η πλέον κατάλληλη για την ανάπτυξη ιστοσελίδων. Μαζί με τον Apache2 εγκαταστάθηκαν τα βοηθητικά προγράμματα για την ανάπτυξη της εφαρμογής του $RUBiS$. Συγκεκριμένα εγκαταστάθηκαν: i) PHP5 ii) PHP-mysql.

Στη συνέχεια ενεργοποιήθηκαν οι κατάλληλες θύρες δικτύου (ports) για τον διακομιστή ιστού Apache2, έτσι ώστε να γίνεται προσβάσιμη η σελίδα στο τοπικό δίκτυο του Πανεπιστημίου όπου είναι εγκατεστημένοι οι εξυπηρετητές.

Σημαντική είναι η αναφορά και της άλλης υλοποίησης που αναπτύχθηκε. Η υλοποίηση αυτή χρησιμοποιεί την γλώσσα Java και παρέχεται στο πακέτο του $RUBiS$. Αυτή η υλοποίηση απαιτεί την εγκατάσταση του διακομιστή ιστού Apache Tomcat. Ωστόσο για λόγους μη σταθερής λειτουργίας αλλά και της περίπλοκης εγκατάστασης και προσαρμογής του $RUBiS$ σε αυτό, προτιμήθηκε η υλοποίηση που αναφέρθηκε παραπάνω δηλαδή με την γλώσσα προγραμματισμού PHP και τον διακομιστή ιστού Apache2.

3.4.4 Βαθμίδα Βάσης Δεδομένων

Ο εξυπηρετητής της βάσης δεδομένων για την εφαρμογή $RUBiS$, αναπτύχθηκε στο VM της τέταρτης φυσικής μηχανής που εγκαταστάθηκε δηλαδή της $PM4$. Αφού εγκαταστάθηκε η βάση δεδομένων MySQL, δημιουργήθηκε ένας λογαριασμός διαχειριστή ο οποίος έχει πλήρη εξουσιοδότηση σε όλες τις λειτουρ-

γίες όπως έχει ένας διαχειριστής βάσεων δεδομένων. Στη συνέχεια με τη χρήση των διαφόρων εντολών που παρέχονται στο πακέτο του RUBiS για τη βάση δεδομένων, δημιουργήθηκαν οι διάφοροι χρήστες, αντικείμενα, προσφορές και σχόλια που θα είναι διαθέσιμα για την δημιουργία του φόρτου εργασίας από τον εξομοιωτή χρηστών. Συγκεκριμένα δημιουργήθηκαν 10000 λογαριασμοί χρηστών, 5000 αντικείμενα και μέγιστος αριθμός προσφορών ανά αντικείμενο, 20. Περισσότερες λεπτομέρειες για το φόρτο εργασίας που δημιουργήθηκε στην εργασία αυτή, υπάρχουν στο αρχείο που περιλαμβάνει τις ιδιότητες του εξομοιωτή χρηστών.

3.4.5 Εξομοιωτής χρηστών

Ο εξομοιωτής χρηστών (Client Emulator) συμπεριλαμβάνεται στο πακέτο του RUBiS και χρησιμοποιείται για την δημιουργία του φόρτου εργασίας στην εφαρμογή. Αυτός ο εξομοιωτής χρηστών, δημιουργεί βασικά πολλαπλά νήματα αιτημάτων για κάθε χρήστη με σκοπό να δημιουργήσει φόρτους εργασίας μοντελοποιώντας έτσι την συμπεριφορά πραγματικών εφαρμογών υπολογιστικής νέφους. Ο φόρτος εργασίας που δημιουργείται είναι ανάλογος των ταυτόχρονων χρηστών που έχουν πρόσβαση στη σελίδα του RUBiS, οι οποίοι στέλνουν τα διάφορα αιτήματά τους περιμένοντας να λάβουν την απάντηση ή τα δεδομένα που ζητούν. Ακόμα μια παράμετρος που επηρεάζει τον φόρτο εργασίας της εφαρμογής είναι ο τύπος των αιτημάτων που δημιουργούνται από τον εξομοιωτή χρηστών. Κάθε χρήστης μπορεί να στείλει πολλών ειδών αιτήματα στον διακομιστή ιστού. Στον Πίνακα 3.3 παρουσιάζονται οι βασικές ιδιότητες του εξομοιωτή χρηστών που χρησιμοποιήθηκαν στην εργασία αυτή.

Ιδιώτητες Εξομοιωτή Χρηστών	
Αριθμός χρηστών για σταθερό φόρτο εργασίας	500
Τύπος αιτημάτων	BR mix
Μέγιστος αριθμός μεταβάσεων	1000
Μέγιστος αριθμός προσφορών ανά αντικείμενο	20
Χρονικό διάστημα μεταξύ δύο αιτημάτων	7s tpc
Αριθμός εγγεγραμμένων χρηστών στη βάση δεδομένων	10000
Αριθμός εγγεγραμμένων αντικειμένων στη βάση δεδομένων	30000
Χρονική διάρκεια πειράματος	200s

Πίνακας 3.3: Πίνακας χαρακτηριστικών φόρτου εργασίας και ιδιοτήτων του εξομοιωτή χρηστών.

Η χρονική απόσταση δύο αιτημάτων, για κάθε χρήστη, περιγράφεται από μια αρνητικά εκθετική κατανομή η οποία έχει μέση τιμή τα 7 δευτερόλεπτα [11]. Στις ρυθμίσεις του εξομοιωτή χρηστών υπάρχει η δυνατότητα να χρησιμοποιείται αυτός ο τρόπος για την δημιουργία των αιτημάτων αλλά υπάρχει και η δυνατότητα κάποιας άλλης κατανομής που ορίζεται μέσω του εξομοιωτή χρηστών με τα διάφορα αρχεία φόρτου εργασίας. Στο $RUBiS$ υπάρχουν δύο βασικοί τύποι αιτημάτων i) Περιήγησης - Browsing Mix (BR) τα οποία απαιτούν μόνο ανάγνωση από τη βάση δεδομένων. ii) Προσφοράς - Bidding Mix (BD) τα οποία απαιτούν ανάγνωση αλλά και εγγραφή στη βάση δεδομένων. Στη παρούσα πτυχιακή εργασία χρησιμοποιήθηκαν αιτήματα περιήγησης (BR). Σε αυτό το σημείο έγιναν κάποιες αλλαγές στον κώδικα του εξομοιωτή χρηστών έτσι ώστε όλοι οι χρόνοι απάντησης των αιτημάτων να καταγράφονται σε αρχείο για μελλοντική επεξεργασία. Μαζί με τους χρόνους των απαντήσεων, καταγράφεται αυτόματα και η αντίστοιχη χρονική στιγμή που επέστρεψε η απάντηση στον χρήστη, έτσι ώστε να γίνεται εφικτή η σύγκριση του mRT για κάθε χρονική στιγμή των πειραμάτων που θα παρουσιαστούν στη συνέχεια της εργασίας.

Κεφάλαιο 4

Ανάλυση και σχεδίαση συστημάτων ελέγχου

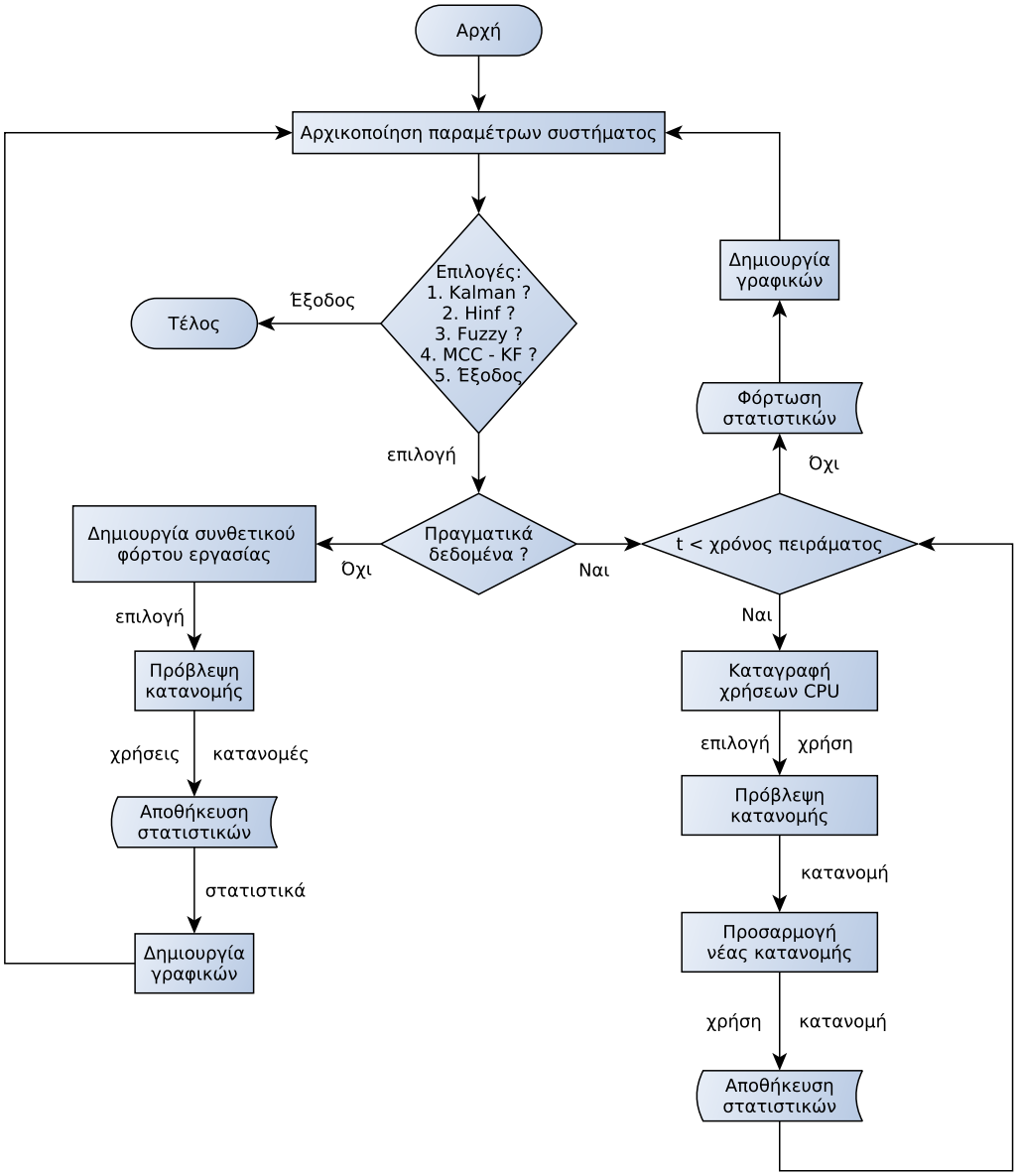
Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη που έγινε για την μοντελοποίηση του προβλήματος αλλά και τον σχεδιασμό των συστημάτων αυτομάτου ελέγχου που χρησιμοποιήθηκαν για την πρόβλεψη της ζήτησης σε υπολογιστικούς πόρους αλλά και την δυναμική κατανομή των πόρων αυτών στους εξυπηρετητές του συστήματος που έχουμε αναπτύξει. Αρχικά περιγράφεται η μοντελοποίηση του προβλήματος της δυναμικής κατανομής των υπολογιστικών πόρων και η σχεδίαση του συστήματος ViResA. Στη συνέχεια περιγράφονται αναλυτικά τα φίλτρα Kalman, το φίλτρο $\mathcal{H}_\infty$, το MCC-KF και τέλος ο ευφυής ελεγκτής ANFIS (Fuzzy Inference).

4.1 Προϋπάρχουσα εργασία (ViResA project)

Όλες οι προβλέψεις της χρήσης των υπολογιστικών πόρων αλλά και οι προσαρμογές των ανάλογων κατανομών έγιναν με την ανάπτυξη του κώδικα ViResA ο οποίος βρίσκεται στο αποθετήριο του συστήματος ελέγχου εκδόσεων Bitbucket [12]. Αρχικά το ViResA δημιουργήθηκε για σκοπούς της εργασίας [13] για να μοντελοποιεί διάφορους φόρτους εργασίας, με συνθετικά δεδομένα, που έχει ένα σύμπλεγμα εφαρμογής υπολογιστικής νέφους έτσι ώστε να προσαρμόζει τις διάφορες κατανομές με τη χρήση φίλτρων Kalman, $\mathcal{H}_\infty$ αλλά και με συστήματα ευφυούς ελέγχου όπως είναι ο ANFIS. Για το σκοπό αυτό τα στατιστικά των χρήσεων αλλά και των κατανομών ήταν συνθετικά, όμως μπορούσαν να δώσουν μια πλήρη εικόνα για την απόδοση και αλλά και την καταλληλότητα των διάφορων ελεγκτών που χρησιμοποιήθηκαν.

Στην εργασία αυτή, αναπτύχθηκε μια νέα έκδοση του ViResA όπου τα στατι-

στικά χρήσης και κατανομής των διάφορων VMs που συνθέτουν την εφαρμογή υπολογιστικής νέφους, που στη περίπτωση αυτή είναι το $RUBiS$, είναι πλέον πραγματικά. Στο Σχήμα 4.1 παρουσιάζεται ένα διάγραμμα ροής της βασικής λειτουργίας του ViResA.



Σχήμα 4.1: Σύστημα δυναμικής κατανομής υπολογιστικών πόρων - Virtualized Resource Allocation (ViResA).

Αρχικά το ViResA, καθορίζει τις αρχικές παραμέτρους του συστήματος ως προς τις διάφορες μεταβλητές του κώδικα, αλλά και τις παραμέτρους των ελεγκτών (π.χ., αρχική κατανομή, διασπορά θορύβου μέτρησης και διαδικασίας κ.λπ). Στη συνέχεια ένα μενού επιλογών εμφανίζεται στο τερματικό εντολών $UNIX$

το οποίο δίνει την δυνατότητα επιλογής του ελεγκτή που θα κάνει τις προβλέψεις αλλά και τις προσαρμογές των κατανομών. Μετά την επιλογή του ελεγκτή, το σύστημα δίνει και πάλι την δυνατότητα επιλογής του τύπου των δεδομένων. Η πρώτη επιλογή που δίνει το σύστημα είναι η επιλογή συνθετικών δεδομένων όπου οι χρήσεις και οι κατανομές δημιουργούνται με βάση κάποιων σεναρίων ζήτησης που έγιναν σε προηγούμενες εργασίες. Η δεύτερη επιλογή του συστήματος είναι η επιλογή για πραγματικά δεδομένα τα οποία εισάγονται στο σύστημα ViResA μέσω εντολών του υπερεπόπτη ή αλλιώς του VMM, όπου στην περίπτωση της εργασίας αυτής είναι ο Xen Hypervisor. Οι χρήσεις σε CPU καταγράφονται σε πραγματικό χρόνο κάθε 1 δευτερόλεπτο και στη συνέχεια περνούν στους ελεγκτές όπου γίνονται οι προβλέψεις για την αντίστοιχη κατανομή. Ακολούθως η προσαρμογή της κατανομής επιτυγχάνεται με την κλήση εντολών του συστήματος προς το Xen έτσι ώστε να προσαρμόζονται στο ποσοστό της νέας κατανομής που αποφάσισε ο αντίστοιχος ελεγκτής. Τέλος όλες οι χρήσεις και κατανομές αποθηκεύονται σε αρχεία για παραγωγή γραφικών παραστάσεων ή για εκπαίδευση του ελεγκτή ANFIS ο οποίος δέχεται τα δεδομένα αυτά σαν δεδομένα εκπαίδευσης. Ο χρόνος για την διαδικασία της δυναμικής κατανομής αυτής καθορίζεται κατά την αρχικοποίηση του συστήματος.

4.2 Μοντέλο Συστήματος

Στην εργασία αυτή, όπως προαναφέρθηκε και στο Κεφάλαιο 3, χρησιμοποιήθηκε ένα σύμπλεγμα εφαρμογής νέφους, το οποίο αποτελείται από δύο βασικούς εξυπηρετητές, ένας για τον διακομιστή ιστού και ένας για την βάση δεδομένων. Κάθε ένα από τα δύο αυτά στοιχεία του συμπλέγματος, εγκαταστάθηκε σαν VM, χρησιμοποιώντας την μεθοδολογία του Κεφαλαίου 3, σε μια φυσική μηχανή ξεχωριστά.

Σύμβολο	Περιγραφή
k	διάστημα διακριτού χρόνου
x_k	κατάσταση του συστήματος στο διάστημα k
w_k	θόρυβος διαδικασίας στο διάστημα k
y_k	μέτρηση της κατάστασης του συστήματος στο διάστημα k
v_k	θόρυβος μέτρησης στο διάστημα k
a_k	κατανομή στο διάστημα k
h	περιθώριο χρήσης - κατανομής

Πίνακας 4.1: Πίνακας σημειογραφίας μοντέλου συστήματος.

Βάσει τις εργασίες [1], [13], [14], [15] η χρήση σε υπολογιστικούς πόρους δίνεται από την γραμμική στοχαστική εξίσωση:

$$x_{k+1} = x_k + w_k, \tag{4.1}$$

όπου $x_k \in \mathbb{R}_+$ είναι το ποσοστό της συνολικής χρήσης σε CPU η οποία καταναλώνεται αποκλειστικά από ένα στοιχείο του συμπλέγματος για τις ανάγκες της εφαρμογής κατά το χρονικό διάστημα k. Ο θόρυβος διαδικασίας μοντελοποιεί την τη μεταβολή του φόρτου εργασίας λόγω της φύσης της εφαρμογής $RUBiS$, όπως η αύξηση αιτημάτων κ.λπ. Ο θόρυβος διαδικασίας αντιπροσωπεύεται από την ανεξάρτητη τυχαία μεταβλητή w_k, όπου $w_k \in \mathbb{R}_+$.

Το ποσοστό της συνολικής χρήσης σε CPU που παρατηρήθηκε στο VM συμπεριλαμβανομένων των διαφόρων εξωτερικών θορύβων από άλλες πηγές εκτός της εφαρμογής αντιπροσωπεύεται από το y_k.

$$y_k = x_k + v_k, \tag{4.2}$$

όπου $y_k \in \mathbb{R}_+$ υποδηλώνεται η χρήση που παρατηρείται στο VM, στην οποία συμπεριλαμβάνονται και οι εξωτερικοί θόρυβοι, όπως π.χ., λογισμικά για την σωστή λειτουργία της εφαρμογής, τα οποία είναι παράγοντες που δεν πρέπει να επηρεάζουν την εφαρμογή όμως υφίστανται αναγκαστικά. Το λειτουργικό σύστημα και διάφορα άλλα λογισμικά που υποστηρίζουν την λειτουργία της εφαρμογής προκαλούν αυτό τον θόρυβο. Το v_k αντιπροσωπεύει τον θόρυβο μέτρησης για την χρήση υπολογιστικών πόρων.

Η χωρητικότητα σε ποσοστό της $pCPU$ της φυσικής μηχανής, η οποία κατανεμήθηκε για ένα συγκεκριμένο VM υποδηλώνεται με το a_k όπου $a_k \in \mathbb{R}_+$. Αντιπροσωπεύει δηλαδή την μέγιστη ποσότητα σε υπολογιστικούς πόρους που μπορεί να χρησιμοποιήσει το συγκεκριμένο VM.

$$a_k = \min\{(1+h)x_k, a_{\max}\}, \tag{4.3}$$

όπου $h \in (0,1)$ αντιπροσωπεύει το περιθώριο (headroom) χρήσης-κατανομής, δηλαδή το ποσοστό των επιπλέον υπολογιστικών πόρων που θα μπορούσε να κατανεμηθεί σε ένα VM έτσι ώστε να επιτυγχάνεται η υψηλή απόδοση του εξυπηρετητή και να μην επηρεάζεται η απόδοση της εφαρμογής.

Όπως αναφέρθηκε σε προηγούμενα κεφάλαια στόχος των ελεγκτών, που θα παρουσιαστούν στη συνέχεια της εργασίας αυτής, είναι να μεταβάλλουν δυναμικά την μέγιστη ποσότητα σε υπολογιστικούς πόρους που μπορεί να χρησιμοποιήσει ένα VM. Αυτό μπορεί να γίνει εφικτό με την δυναμική προσαρμογή της κατανομής σε CPU του VM, πάνω από ένα περιθώριο (headroom) της αντίστοιχης χρήσης. Αυτό απέδειξαν και οι συγγραφείς στις εργασίες [1], [14] και [15],

όπου σχεδιάστηκαν και χρησιμοποιήθηκαν φίλτρα Kalman και φίλτρα $\mathcal{H}_\infty$ αντίστοιχα έτσι ώστε να προβλέπουν την χρήση σε CPU για κάθε στοιχείο, και να κατανέμουν ανάλογα το αντίστοιχο ποσοστό σε υπολογιστικούς πόρους έτσι ώστε να επιτυγχάνεται η υψηλή απόδοση των εξυπηρετητών αλλά και η δυνατότητα φιλοξενίας άλλων εφαρμογών στην ίδια φυσική μηχανή.

Θεωρώντας ότι το $\mathcal{Y}_k$ υποδηλώνει το σύνολο όλων των μετρήσεων για όλη την χρονική διάρκεια k, η εκ των υστέρων ανανέωση $a\ posteriori$ παρουσιάζεται με $\hat{x}_{k|k} = \mathbb{E}\{x_k|\mathcal{Y}_k\}$ ενώ η εκ των προτέρων εκτίμηση $a\ priori$ κατάστασης παρουσιάζεται με $\hat{x}_{k+1|k} = \mathbb{E}\{x_{k+1}|\mathcal{Y}_k\}$. Άρα η πρόβλεψη για την χρονική στιγμή $k+1$ θα είναι $\hat{x}_{k+1|k}$ και έτσι η αντίστοιχη κατανομή σε CPU που θα δώσει ο ελεγκτής δίνεται από την σχέση:

$$a_{k+1} = \max\left\{0, \min\{(1+h)\hat{x}_{k+1|k}, a_{\max}\}\right\}. \tag{4.4}$$

4.3 Φίλτρο Kalman

Στην ενότητα αυτή παρουσιάζεται και αναλύεται το φίλτρο Kalman [16] το οποίο χρησιμοποιήθηκε για την επίτευξη της πρόβλεψης αλλά και της δυναμικής προσαρμογής της κατανομής των υπολογιστικών πόρων. Το συγκεκριμένο φίλτρο, υπολογίζει την κατάσταση ενός στοχαστικού γραμμικού συστήματος με αναδρομικό τρόπο με βάση τις μετρήσεις θορύβου. Το φίλτρο Kalman είναι η βέλτιστη μέθοδος όταν το μοντέλο του συστήματος είναι γραμμικό και όταν ο θόρυβος διαδικασίας και μέτρησης είναι λευκός και γκαουσιανής μορφής.

Στην εργασία αυτή το φίλτρο Kalman προσαρμόστηκε έτσι ώστε να προβλέπει και να προσαρμόζει την κατανομή σε υπολογιστικούς πόρους βάσει προηγούμενων μετρήσεων, για μόνο ένα στοιχείο του συμπλέγματος της εφαρμογής $RUBiS$, δηλαδή σε ένα VM. Η κατανομή, a για ένα στοιχείο του συμπλέγματος, καθορίστηκε ως το ποσοστό από την συνολική κατανομή CPU μιας φυσικής μηχανής.

Η δυναμική του συστήματος περιγράφεται από τις παρακάτω εξισώσεις:

$$x_{k+1} = Ax_k + w_k, \tag{4.5α'}$$

$$y_k = Cx_k + v_k, \tag{4.5β'}$$

όπου $x_k \in \mathbb{R}^{n_x}$ υποδηλώνεται η κατάσταση του συστήματος, $u_k \in \mathbb{R}^{n_u}$ αποτελεί την είσοδο του ελεγκτή, $w_k \in \mathbb{R}^{n_x}$ είναι μια τυχαία κατανομή με μηδενικό μέσο όρο και πεπερασμένης δευτέρας τάξης πίνακα W_k, όπου $y_k \in \mathbb{R}^{n_y}$ είναι η μέτρηση της κατάστασης x_k του συστήματος, $v_k \in \mathbb{R}^{n_y}$ είναι τυχαία κατανομή με μηδενικό μέσο όρο και πεπερασμένης δευτέρας τάξης πίνακα V_k, και A και C είναι οι πίνακες με τις κατάλληλες διαστάσεις.

Η συμπεριφορά της κατανομής σε CPU περιγράφεται από την σχέση:

$$x_{k+1} = x_k + w_k, \tag{4.6}$$

ενώ το σήμα το οποίο περιγράφει την χρήση CPU σε υπολογιστικούς πόρους περιγράφεται από την εξίσωση:

$$y_k = cx_k + v_k, \tag{4.7}$$

όπου το c υποδηλώνει την επιπλέον ποσότητα σε υπολογιστικούς πόρους που θα μπορούσε ένα VM να χρησιμοποιήσει. Οι ανεξάρτητες τυχαίες μεταβλητές w_k και v_k υποδηλώνουν τον θόρυβο διαδικασίας και τον θόρυβο μέτρησης αντίστοιχα. Η κατανομή τους είναι γκαουσιανής μορφής, δηλαδή έχουν μέση τιμή 0, διασπορά θορύβου διαδικασίας Q και διασπορά θορύβου μέτρησης R αντίστοιχα.

$$p(w) \sim \mathcal{N}(0, Q),$$

$$p(v) \sim \mathcal{N}(0, R).$$

Το φίλτρο Kalman χρησιμοποιείται έτσι ώστε η νέα κατανομή που προβλέφθηκε να «ακολουθεί» τις χρήσεις σε CPU x_k, βάσει προηγούμενων μετρήσεων. Η $a\ priori$ (εκ των προτέρων πρόβλεψη) για την κατανομή σε CPU για το χρονικό διάστημα k δίνεται από την σχέση:

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1}, \tag{4.8}$$

Για τον υπολογισμό της κατάστασης $\hat{x}_{k-1|k-1}$ απαιτείται η διόρθωση ή αλλιώς ανανέωση από το προηγούμενο διάστημα χρησιμοποιώντας την $a\ posteriori$ (εκ των υστέρων ανανέωση):

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - c\hat{x}_{k|k-1}), \tag{4.9}$$

όπου $\hat{x}_{k|k-1}$ ορίζεται σαν η $a\ priori$ πρόβλεψη για την κατανομή στο διάστημα k, βάσει προηγούμενων μετρήσεων, όπου $\hat{a}_{k-1|k-1}$ ορίζεται σαν η $a\ posteriori$ ανανέωση της κατανομής σε CPU η οποία είναι πλέον διορθωμένη, βάσει των νέων μετρήσεων. Το κέρδος διόρθωσης μεταξύ της πραγματικής μέτρησης με την προβλέψιμη εκτίμηση για το φίλτρο Kalman δίνεται από την σχέση:

$$K_k = cP_{k|k-1}(c^2P_{k|k-1} + R)^{-1} \tag{4.10}$$

Το σφάλμα της $a\ posteriori$ ανανέωσης:

$$P_{k|k} = (1 - cK_k)P_{k|k-1} \tag{4.11}$$

ενώ το σφάλμα της $a\ priori$ πρόβλεψης:

$$P_{k|k-1} = P_{k-1|k-1} + Q \tag{4.12}$$

4.4 Φίλτρο $\mathcal{H}_\infty$

Στην ενότητα αυτή παρουσιάζεται μια νέα προσέγγιση στο πρόβλημα της πρόβλεψης της κατάστασης της κατανομής υπολογιστικών πόρων, εισάγοντας την ευρωστία στο σύστημα μέσω του φίλτρου $\mathcal{H}_\infty$. Το φίλτρο $\mathcal{H}_\infty$ μειώνει το σφάλμα της χειρότερης πρόβλεψης ($minmax$), και σαν αποτέλεσμα μπορεί να δώσει λύση σε θέματα ευρωστίας για την πρόβλεψη της κατάστασης.

Η σχέση που περιγράφει την συνάρτηση κόστους για το πρόβλημα:

$$J = \frac{\sum_{k=0}^{N-1} \| x_k - \hat{x}_k \|_2^2}{\| x_o - \hat{x}_o \|_{P_0^{-1}}^2 + \sum_{k=0}^{N-1} \left(\| w_k \|_{Q_k^{-1}}^2 + \| v_k \|_{R_k^{-1}}^2 \right)}, \tag{4.13}$$

όπου $P_0 \in \mathbb{R}^{N\times N}$, $Q_k \in \mathbb{R}^{N\times N}$ και $R_k \in \mathbb{R}^{N\times N}$ είναι συμμετρικοί και θετικά καθορισμένοι πίνακες. Ο πίνακας P_0 υποδηλώνει το αρχικό σφάλμα συνδιασποράς. Ο πίνακας Q_k υποδηλώνει την συνδιασπορά θορύβου διαδικασίας ενώ ο πίνακας R_k την συνδιασπορά θορύβου μέτρησης, για το χρονικό διάστημα k. Η εκτίμηση για την κατανομή στο διάστημα k παρουσιάζεται με το $\hat{x}_k$. Λόγω του ότι η ελαχιστοποίηση της συνάρτησης κόστους J δεν είναι άμεση, τέθηκε ένα όριο στον σχεδιασμό του ελεγκτή. Το όριο αυτό περιγράφεται από την συνθήκη:

$$J < \frac{1}{\theta}, \tag{4.14}$$

όπου η τιμή του θ καθορίζεται έτσι ώστε το επιθυμητό mRT να είναι χαμηλότερο από αυτό που καθορίστηκε από τον χρήστη. Η συνάρτηση κόστους για το φίλτρο $\mathcal{H}_\infty$ σταθερής κατάστασης δίνεται από την σχέση:

$$J = \lim_{N\to\infty} \frac{\sum_{k=0}^{N-1} \| x_k - \hat{x}_k \|_2^2}{\sum_{k=0}^{N-1} \left(\| w_k \|_{Q_k^{-1}}^2 + \| v_k \|_{R_k^{-1}}^2 \right)}, \tag{4.15}$$

Στην (4.16), το G_{xe} υποδηλώνει το σύστημα με είσοδο $e = [w \times v]^T$ και έξοδο το $\hat{x}$. Με βάση την εργασία [17] και αφού το φίλτρο $\mathcal{H}_\infty$ ικανοποιεί την συνθήκη της (4.14), τότε θα ισχύει ότι:

$$\| G_{\hat{x}e} \|_\infty^2 = \sup_\zeta \frac{\| x - \hat{x} \|_2^2}{\| w \|_{Q^{-1}}^2 + \| v \|_{R^{-1}}^2} \leq \frac{1}{\theta}, \tag{4.16}$$

όπου ζ η φάση του $\| w \|_{Q^{-1}}^2 + \| v \|_{R^{-1}}^2$ συμπεριλαμβανομένου του χρόνου δειγματοληψίας του συστήματος αλλά και της συχνότητας των σημάτων. Αφού το mRT δεν πρέπει να ξεπερνά το όριο που τέθηκε δηλαδή το 1 δευτερόλεπτο στην παρούσα εργασία, η (4.16) θα προσαρμοστεί στην σχέση:

$$\sup_\zeta \frac{\| \Phi - C \|_2^2}{\| w \|_{Q^{-1}}^2 + \| v \|_{R^{-1}}^2} \leq \frac{1}{\theta}, \tag{4.17}$$

ή αλλιώς:

$$\theta \leq \inf_{\zeta} \frac{\| w \|^2_{Q^{-1}} + \| v \|^2_{R^{-1}}}{\| \Phi - C \|^2_2}, \tag{4.18}$$

όπου το Φ *και το* C *υποδηλώνουν διαγώνιους πίνακες με περιθώρια τις τιμές* ϕ_i *και* c_i *αντίστοιχα για κάθε στοιχείο του συμπλέγματος.*

Απαραίτητη είναι η συνθήκη για την ευστάθεια του συστήματος για το φίλτρο $\mathcal{H}_\infty$*, αλλά και του ορισμού της τιμής του* P_k *ως θετικής. Για τους λόγους αυτούς η (4.19) πρέπει να ικανοποιείται.*

$$I - \theta P_{k|k-1} + C^T {R_k}^{-1} C P_{k|k-1} \succ 0. \tag{4.19}$$

Έτσι με βάση τη συνάρτηση κόστους (4.13), το φίλτρο $\mathcal{H}_\infty$ *περιγράφεται από τις:*

$$K_k = P_{k|k-1}[I - \theta P_{k|k-1} + C^T {R_k}^{-1} C P_{k|k-1}]^{-1} C^T R_k^{-1}, \tag{4.20}$$

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + K_k \left(y_k - C\hat{x}_k\right), \tag{4.21}$$

$$P_{k|k} = P_k[I - \theta P_k + C^T {R_k}^{-1} C P_k]^{-1} + Q_k, \tag{4.22}$$

όπου το K_k *υποδηλώνει τον πίνακα κέρδους, ενώ το* P_k *υποδηλώνει τον πίνακα σφάλματος συνδιασποράς ο οποίος είναι και θετικά ορισμένος.*

Οι παραπάνω σχέσεις απλοποιούνται στις σχέσεις (4.23),(4.24),(4.25),(4.26) αφού το μοντέλο που χρησιμοποιήθηκε στην εργασία αυτή ήταν μονής εισόδου μονής εξόδου ($SISO$*).*

$$P_{k|k-1} = P_{k-1|k-1} + Q_k, \tag{4.23}$$

$$K_k = \frac{cP_{k|k-1}}{R\left(1 + c^2 P_{k|k-1} R^{-1} - \theta\right)}, \tag{4.24}$$

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + K_k \left(y_k - c\hat{x}_k\right), \tag{4.25}$$

$$P_{k|k} = \frac{P_{k+1}}{1 + c^2 P_{k|k-1} R^{-1} - \theta}. \tag{4.26}$$

4.5 Φίλτρο MCC-KF

Σε αυτή την ενότητα παρουσιάζεται μια νεα προσέγγιση του φίλτρου Kalman, η οποία χρησιμοποιεί το κριτήριο της μέγιστης συνεντροπίας (Maximum Correntropy Criterion - MCC) για τον υπολογισμό της κατάστασης [18] και [19]. Το κριτήριο αυτό μετράει την ομοιότητα δύο τυχαίων μεταβλητών χρησιμοποιώντας

πληροφορία από υψηλής τάξης σήματα στατιστικών [20, 21, 22, 23]. Λόγω του ότι το φίλτρο Kalman χρησιμοποιεί μόνο δεύτερης τάξης πληροφορία σήματος, δεν είναι το καταλληλότερο για προβλέψεις όπου ο θόρυβος διαδικασίας και ο θόρυβος μέτρησης δεν είναι Γκαουσιανής κατανομής.

Η δυναμική του συστήματος περιγράφεται από τις εξισώσεις:

$$x_{k+1} = Ax_k + w_k, \tag{4.27α'}$$

$$y_k = Cx_k + v_k, \tag{4.27β'}$$

όπως μπορεί να παρατηρηθεί οι εξισώσεις για την δυναμική του συστήματος είναι ίδιες με του φίλτρου Kalman, βλπ Εξίσωση 4.5.
Το σφάλμα συνδιασποράς για την $a\ posteriori$ διορθώση και την $a\ priori$ πρόβλεψη δίνεται από τις σχέσεις:

$$P_{k|k} = \mathbb{E}\left\{(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T|\mathcal{Y}_k\right\},$$
$$P_{k+1|k} = \mathbb{E}\left\{(x_k - \hat{x}_{k+1|k})(x_k - \hat{x}_{k+1|k})^T|\mathcal{Y}_k\right\}.$$

Οι γενικές εξισώσεις για το φίλτρο $MCC-KF$ παρουσιάζονται παρακάτω βάση της εργασίας [19]. Η φάση της $a\ priori$ πρόβλεψης δίνεται από τις:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1}, \tag{4.28}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + W_k, \tag{4.29}$$

ενώ η φάση της $a\ posteriori$ ανανέωσης δίνεται από τις:

$$L_k = \frac{G_\sigma\left(\| y_k - C\hat{x}_{k|k-1}\|_{V_k^{-1}}\right)}{G_\sigma\left(\| \hat{x}_{k|k-1} - A\hat{x}_{k-1|k-1}\|_{P_{k|k-1}^{-1}}\right)}, \tag{4.30}$$

$$K_k = (P_{k|k-1}^{-1} + L_kC^TV_k^{-1}C)^{-1}L_kC^TV_k^{-1}, \tag{4.31}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}), \tag{4.32}$$

$$P_{k|k} = (I - K_kC)P_{k|k-1}(I - K_kC)^T + K_kV_kK_k^T, \tag{4.33}$$

όπου το G_σ υποδηλώνει τον γκαουσιανό πυρήνα, ενώ το L_k και K_k την συνάρτηση κόστους και το κέρδος αντίστοιχα.

Οι γενικές σχέσεις (4.28-4.33) απλοποιούνται στις σχέσεις (4.34-4.39) αφού το μοντέλο που χρησιμοποιήθηκε στην εργασία αυτή ήταν μονής εισόδου μονής εξόδου $(SISO)$.

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1}, \tag{4.34}$$

$$P_{k|k-1} = P_{k-1|k-1} + W_k, \tag{4.35}$$

$$L_k = \frac{G_\sigma\left(\| y_k - \hat{x}_{k|k-1}\|_{V_k^{-1}}\right)}{G_\sigma\left(\| \hat{x}_{k|k-1} - \hat{x}_{k-1|k-1}\|_{P_{k|k-1}^{-1}}\right)}, \tag{4.36}$$

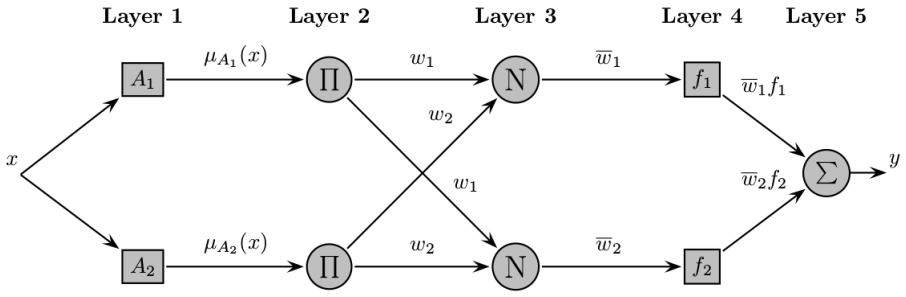
$$K_k = \frac{L_k}{(P_{k|k-1}^{-1} + L_k V_k^{-1})V_k}, \tag{4.37}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - \hat{x}_{k|k-1}), \tag{4.38}$$

$$P_{k|k} = (1 - K_k)^2 P_{k|k-1} + {K_k}^2 V_k. \tag{4.39}$$

4.6 Προσαρμοστικός Νευροασαφής Ελεγκτής (ANFIS)

Εκτός από τις τεχνικές του βέλτιστου και εύρωστου ελέγχου μέσω φίλτρων Kalman και $\mathcal{H}_\infty$ αντίστοιχα, ο έλεγχος και η κατανομή υπολογιστικών πόρων δοκιμάστηκε και με τεχνικές ευφυούς ελέγχου και συγκεκριμένα με χρήση ενός προσαρμοστικού νευροασαφούς (Adaptive Neuro Fuzzy Inference System - ANFIS). Στην εργασία [13], οι συγγραφείς χρησιμοποίησαν έναν προσαρμοστικό νευροασαφή ελεγκτή ο οποίος προηγουμένως είχε εκπαιδευτεί με τα αποτελέσματα των φίλτρων Kalman και $\mathcal{H}_\infty$ και κατέληξαν ότι ο νευροασαφής ελεγκτής δίνει καλύτερα αποτελέσματα σε σύγκριση με τους προηγούμενους δύο ελεγκτές. Ο ελεγκτής Fuzzy αποτελείται από 5 επίπεδα όπως φαίνεται και στο Σχήμα 4.2:



Σχήμα 4.2: Δομή συμπερασμού ασαφής λογικής 5 επιπέδων μονής εισόδου μονής εξόδου (SISO).

4.6.1 Επίπεδο 1

Το πρώτο επίπεδο του ελεγκτή ονομάζεται επίπεδο ασαφοποίησης. Στο επίπεδο αυτό κάθε νευρώνας αντιστοιχεί τα δεδομένα εισόδου του συστήματος σε ένα ασαφές διάστημα $[0, 1]$ ενός ασαφούς συνόλου A, δηλαδή τα μετατρέπει σε ασαφή μορφή. Η αντιστοίχηση αυτή γίνεται μέσω μιας συνάρτησης συμμετοχής $\mu_A(x)$ η οποία περιγράφει και τον βαθμό συμμετοχής της εισόδου x, στο σύνολο A.

$$\mu_A(x; l, c, r) = \begin{cases} 1 - \frac{c-x}{c-l}, & \text{καθώς } \ l < x \leq c, \\ 1 - \frac{x-c}{r-c}, & \text{καθώς } \ c < x < r, \\ 0, & \text{αλλιώτικα.} \end{cases} \tag{4.40}$$

4.6.2 Επίπεδο 2

Το Επίπεδο 2 αποτελεί την βάση κανόνων όπου κάθε νευρώνας είναι ισοδύναμος με ένα κανόνα της μεθόδου Takagi-Sugeno (T-S inference rule) [24]. Η μέθοδος T-S, χωρίζει το μη-γραμμικό σύστημα σε δύο περιοχές ασάφειας όπου κάθε περιοχή έχει ένα τοπικό γραμμικό μοντέλο. Ο κανόνας που χρησιμοποιεί η μέθοδος T-S είναι της μορφής:

$$R_i \ : \ IF \ x \ IS \ A_i \ THEN \ y_i, \tag{4.41}$$

όπου x είναι η μεταβλητή εισόδου του συστήματος, A^i η είσοδος του ασαφούς συνόλου. Η έξοδος κάθε κανόνα υποδηλώνεται με $y^i = c_0^i + c_1^i x_1, \ i = 1, 2, ..., m$ όπου m ο ολικός αριθμός των κανόνων και c_k^i είναι οι παραμέτροι σταθερές της εξόδου. Αφού το Επίπεδο 2 λάβει τις ασαφοποιημένες εισόδους από το Επίπεδο 1, τότε μπορούν να υπολογιστούν οι βαθμοί ενεργοποίησης (firing strengths) του κανόνα μ_{ij}. Το αλγεβρικό γινόμενο $w_i = \prod_{j=1}^{k} \mu_{ij}$, το οποίο εξακριβώνει τις συζεύξεις των εισόδων του κανόνα, απλοποιείται σε $w_i = \mu_i$, αφού το σύστημα που χρησιμοποιήθηκε στην εργασία αυτή είναι $SISO$.

4.6.3 Επίπεδο 3

Αφού το Επίπεδο 3 έλαβε τις εισόδους από το Επίπεδο 2, στη συνέχεια θα καθορίσει τους κανονικοποιημένους βαθμούς ενεργοποίησης ή την συμμετοχή $\overline{w}_i$, για κάθε κανόνα, στο τελικό αποτέλεσμα:

$$\overline{w}_i = \frac{\mu_i}{\sum_{j=1}^{k} \mu_i}, \tag{4.42}$$

4.6.4 Επίπεδο 4

Στο Επίπεδο 4 γίνεται η απο-ασαφοποίηση, δηλαδή μετατρέπονται τα αποτελέσματα που προκύψανε από το προηγούμενο επίπεδο, σε σαφή μορφή, για τις διάφορες διεργασίες απόφασης από επόμενα συστήματα.

$$y_i = \sum_{i=1}^{n} \overline{w}_i f_i, \tag{4.43}$$

όπου f_i η έξοδος κάθε κανόνα που υποδηλώνεται ως πρώτης τάξης πολυώνυμο, $f_i = c_{0i} + c_{1i}x_1 + ... + c_{ji}x_j$, $i = 1, ..., n$ και $j = 1, ..., m$. Λόγω του ότι στην έξοδο χρησιμοποιούμε μόνο τις σταθερές c_0^i, έχουμε την λεγόμενη «απλοποιημένη συναρτησιακή συλλογιστική Takagi-Sugeno» ή Takagi-Sugeno μηδενικής τάξης (T-S type zero-order). Στην εργασία αυτή λόγω του $SISO$ συστήματος που χρησιμοποιήθηκε, το μοντέλο f απλοποιείται σε $f_i = c_{0i}, i = 1, ..., n$.

4.6.5 Επίπεδο 5

Το Επίπεδο 5 δίνει το ολικό συμπέρασμα αφού προσθέσει όλες τις εξόδους από τους νευρώνες απο-ασαφοποίησης και υπολογίζει την μέση τιμή των εξόδων y_i με βάρος μ_i όπως δίνεται από την σχέση:

$$y = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}, \tag{4.44}$$

Κεφάλαιο 5

Αξιολόγηση αποτελεσμάτων

Όπως αναφέρθηκε σε προηγούμενα κεφάλαια, η πιο αντιπροσωπευτική παράμετρος μέτρησης της απόδοσης μιας εφαρμογής υπολογιστικής νέφους, όπου στη συγκεκριμένη περίπτωση είναι η εφαρμογή δημοπρασιών $RUBiS$, είναι ο μέσος χρόνος απάντησης στα αιτήματα των χρηστών (mRT). Στην εργασία αυτή, ο χρόνος απάντησης για κάθε αίτημα καταγράφεται σε αρχείο μαζί με την χρονική στιγμή της κάθε απάντησης, και έτσι μέσω του κώδικα ViResA που αναπτύχθηκε, να υπολογίζεται το mRT για κάθε δευτερόλεπτο του πειράματος. Η απόδοση μιας εφαρμογής υπολογιστικής νέφους, εξαρτάται από τρεις κύριους παράγοντες. Ωστόσο η απόδοση των εφαρμογών αυτών διαφέρει λόγω της φύσης των εφαρμογών αλλά και λόγω των διαφόρων φόρτων εργασιών που δημιουργούνται. Αυτό όμως δεν επηρεάζει τα κύρια χαρακτηριστικά για τις τιμές του mRT [25]. Έτσι και η απόδοση της εφαρμογής $RUBiS$ η οποία χρησιμοποιήθηκε για την αξιολόγηση των αποτελεσμάτων μας χωρίζεται σε τρεις βασικές περιοχές για το mRT:

(α) όταν η εφαρμογή έχει πληθώρα από υπολογιστικούς πόρους τότε όλα τα αιτήματα εξυπηρετούνται σε πολύ γρήγορους χρόνους και έτσι το mRT παραμένει χαμηλό;

(β) καθώς η χρήση σε υπολογιστικούς πόρους προσεγγίζει το 100% , για παράδειγμα 70-80% των δυνατοτήτων της εκάστοτε εικονικής μηχανής τότε το mRT αρχίζει να αυξάνεται λόγω του ότι μειώνονται οι διαθέσιμοι υπολογιστικοί πόροι και έτσι τα καινούρια αιτήματα χρειάζονται περισσότερο χρόνο για να εξυπηρετηθούν;

(γ) όταν η χρήση σε υπολογιστικούς πόρους φθάσει το 100% τότε οι υπολογιστικοί πόροι που απομένουν διαθέσιμοι είναι τόσο λίγοι και έτσι το mRT αυξάνεται δραματικά λόγω του ότι δεν υπάρχουν σχεδόν καθόλου διαθέσιμοι πόροι για την εξυπηρέτηση καινούριων αιτημάτων.

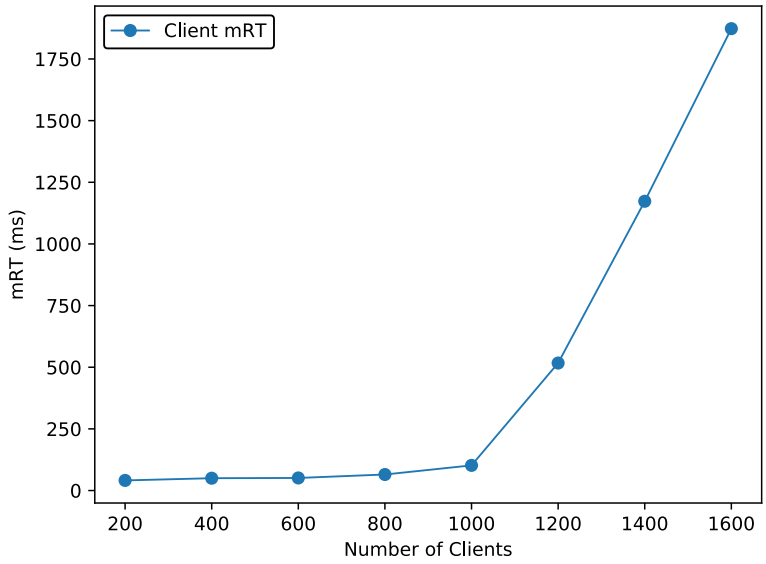
Για την επίτευξη της καλής απόδοσης των εξυπηρετητών στους οποίους αναπτύχθηκε η εφαρμογή $RUBiS$, απαιτείται η εύρεση του περιθωρίου (headroom). Το περιθώριο αυτό αποτελεί το ποσοστό χρήσης σε υπολογιστικούς πόρους από την συνολική χωρητικότητα του CPU δηλαδή το 100%. Όταν το περιθώριο αυτό έχει τιμές οι οποίες κυμαίνονται μεταξύ των περιοχών β) και γ), τότε οι εξυπηρετητές είναι αρκετά εφοδιασμένοι με υπολογιστικούς πόρους και έτσι το mRT παραμένει χαμηλό. Λόγω μεγάλων μεταβολών στον φόρτο εργασίας της εφαρμογής, η χρήση σε υπολογιστικούς πόρους μπορεί να αυξηθεί απότομα με αποτέλεσμα το όριο του περιθωρίου να ξεπερασθεί και έτσι τα συστήματα ελέγχου θα πρέπει να δώσουν στους εξυπηρετητές περισσότερους υπολογιστικούς πόρους.

5.1 Αναγνώριση συστήματος

Στόχος του παρόντος συστήματος είναι η δυναμική προσαρμογή της κατανομής των υπολογιστικών πόρων ανάλογα με την χρήση σε CPU έτσι ώστε η εφαρμογή να πληρεί τον στόχο της σχετικά με το QoS, καθώς ο φόρτος εργασίας αλλάζει. Βάσει προηγούμενων εργασιών [26] θέσαμε το όριο για το mRT στο 1 δευτερόλεπτο και με αυτό τον τρόπο βασικά τέθηκε και το όριο για το QoS το οποίο θα πρέπει να τηρείται.

Αρχικά, μετρήσαμε την απόδοση των εξυπηρετητών καθώς ήταν πλήρως εφοδιασμένοι με το 100% των υπολογιστικών πόρων που προκαθορίστηκαν για τις εικονικές μηχανές του διακομιστή ιστού και της βάσης δεδομένων. Οι μετρήσεις αυτές έγιναν χωρίς καμία εμπλοκή των συστημάτων ελέγχου έτσι ώστε να μην επηρεαστεί καθόλου η απόδοση των εξυπηρετητών. Στο πείραμα αυτό, χρησιμοποιήθηκε σταθερός φόρτος εργασίας με αιτήματα περιήγησης (BR) όσο αναφορά το είδος των αιτημάτων, αυξάνοντας όμως τον αριθμό των ταυτόχρονων χρηστών που δημιουργούν αιτήματα στην εφαρμογή. Για το σκοπό αυτό τροποποιήσαμε ανάλογα τα αρχεία διαμόρφωσης που καθορίζουν τον τύπο των αιτημάτων και τον αριθμό των χρηστών. Το παρόν πείραμα είχε διάρκεια τρία λεπτά για κάθε αριθμό χρηστών, ξεκινώντας από 200 χρήστες μέχρι τους 1600, με διαφορά 200 χρηστών για κάθε εξομοίωση. Κάθε εξομοίωση των διαδικτυακών δημοπρασιών του $RUBiS$ αποτελείται από τρεις διαφορετικές περιοχές όπου δημιουργούνται τα αιτήματα (π.χ, up-ramp, session-run, down-ramp). Το mRT στο συγκεκριμένο πείραμα μετρήθηκε με βάση τους μέσους χρόνους απάντησης των αιτημάτων και για τις τρεις περιοχές. Στο Σχήμα 5.1 παρουσιάζονται τα αποτελέσματα των μέσων χρόνων απάντησης στα αιτήματα των χρηστών σε σχέση με τον αριθμό των χρηστών.

Όπως φαίνεται στο Σχήμα 5.1, το mRT χωρίζεται σε τρεις βασικές περιοχές. Η πρώτη περιοχή είναι αυτή κατά την οποία το mRT παραμένει χαμηλό και έτσι επιτυγχάνεται η υψηλή απόδοση της εφαρμογής $RUBiS$. Η περιοχή αυτή συμπεριλαμβάνει τις εξομοιώσεις με αριθμούς ταυτόχρονων χρηστών 200 μέχρι και 1000. Αυτό επιβεβαιώνει ότι η εφαρμογή $RUBiS$ μπορεί να εξυπηρετεί ταυτόχρονα μέχρι και 1000 χρήστες χωρίς αυτό να επιδρά αρνητικά στο mRT, και άρα να αποδίδει πολύ καλά χωρίς να παραβιάζει το QoS. Η δεύτερη περιοχή συμπεριλαμβάνει τα mRT τα οποία ξεκινούν να έχουν αύξηση στις τιμές τους χωρίς όμως να παραβιάζεται το QoS που έχει προκαθοριστεί κατά το SLA το οποίο είναι στο 1 δευτερόλεπτο. Η περιοχή αυτή παρατηρείται όταν οι χρήστες που στέλνουν ταυτόχρονα αιτήματα είναι από 1000 μέχρι 1300. Στη τρίτη και τελευταία περιοχή βρίσκονται τα mRT τα οποία ξεπερνούν το QoS δηλαδή το 1 δευτερόλεπτο, όπου πλέον παρατηρείται και η απότομη αύξηση τους. Αυτό συμβαίνει όταν ο αριθμός των ταυτόχρονων χρηστών που στέλνουν αιτήματα στους εξυπηρετητές ξεπεράσουν τους 1300.

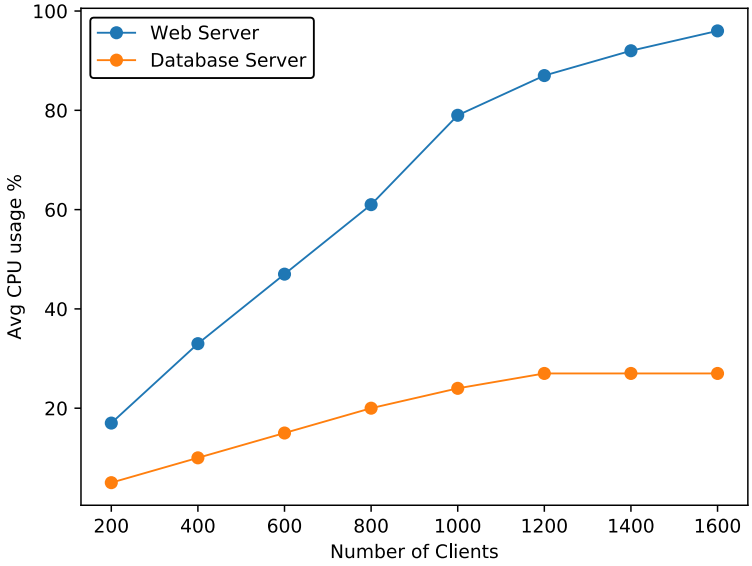


Σχήμα 5.1: Μέση τιμή χρόνων απάντησης (mRT) ανά αριθμό χρηστών που στέλνουν ταυτόχρονα αιτήματα στους εξυπηρετητές.

Παράλληλα δημιουργήθηκε ένα περιβάλλον το οποίο κατέγραφε τις χρήσεις σε CPU για κάθε στοιχείο της εφαρμογής $RUBiS$ (π.χ., Web Server, Database Server) και υπολόγιζε τον μέσο όρο έτσι ώστε να υπάρχει μια εικόνα η οποία να συσχετίζει τους μέσους χρόνους απάντησης στα αιτήματα των χρηστών (mRT) μαζί με το ποσοστό της χρήσης σε CPU. Αυτό επιτεύχθηκε με την ανάπτυξη ενός κώδικα Python ο οποίος καταγράφει τα ποσοστά αυτά μέσω της εντολής $xl\ top$ του Xen Hypervisor σε ένα αρχείο. Στη συνέχεια μια συνάρτηση στον ίδιο κώδικα υπολόγιζε την μέση χρήση σε CPU για κάθε εξομοίωση. Η μέση χρήση

σε CPU ανά αριθμό χρηστών παρουσιάζεται στο Σχήμα 5.2 για κάθε στοιχείο του συμπλέγματος RUBiS που υλοποιήθηκε.

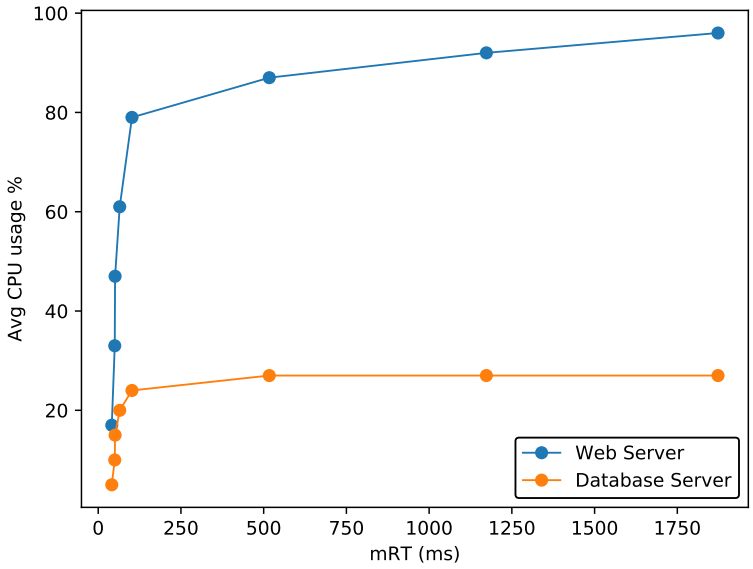
Στο Σχήμα 5.2 φαίνεται η χρήση της CPU για κάθε εξομοίωση με διαφορετικό αριθμό χρηστών. Αρχικά με την αύξηση των χρηστών που στέλνουν ταυτόχρονα αιτήματα στην εφαρμογή, αυξάνεται σχεδόν γραμμικά και το ποσοστό της χρήσης της CPU για κάθε στοιχείο (εξυπηρετητή). Φυσικά, όπως φαίνεται, το στοιχείο της βάσης δεδομένων δεν χρησιμοποιεί πολύ υψηλά ποσοστά χρήσης σε CPU όσο ο διακομιστής ιστού. Σχετικά με τον διακομιστή ιστού παρατηρείται ότι μέχρι και τους 1000 χρήστες, το ποσοστό αυξάνεται σχεδόν γραμμικά ενώ στο Σχήμα 5.1 το mRT παραμένει σχεδόν σταθερό σε χαμηλές τιμές. Αυτό οδηγεί στο συμπέρασμα ότι όσο το ποσοστό χρήσης σε CPU είναι μικρότερο από 80% τότε το mRT παραμένει χαμηλό και εξυπηρετούνται περίπου 1000 χρήστες ταυτόχρονα με μηδαμινές επιπτώσεις στην επίδοση του RUBiS. Από τη στιγμή όμως που το ποσοστό χρήσης σε CPU ξεπεράσει το 80% λόγω της αύξησης των ταυτόχρονων χρηστών που στέλνουν τα αιτήματά τους, τότε σύμφωνα με το Σχήμα 5.1, το mRT ξεκινά να αυξάνεται επιδρώντας έτσι αρνητικά την επίδοση του RUBiS. Από το σημείο όπου οι χρήστες ξεπερνούν τους 1300, το ποσοστό χρήσης σε CPU πλησιάζει το 100% και άρα απομένουν ελάχιστοι διαθέσιμοι υπολογιστικοί πόροι για την εφαρμογή αυξάνοντας δραματικά το mRT.



Σχήμα 5.2: Μέση τιμή χρήσης σε CPU ανά αριθμό χρηστών που στέλνουν ταυτόχρονα αιτήματα στους εξυπηρετητές.

Η σχέση μεταξύ του ποσοστού της μέσης χρήσης σε CPU και του mRT παρουσιάζεται πιο ξεκάθαρα στο Σχήμα 5.3. Στο συγκεκριμένο σχήμα μπορεί να παρατηρηθεί η απότομη αύξηση του mRT καθώς η μέση χρήση σε CPU ξεπερνά

το 80%. Από το σημείο αυτό το mRT αυξάνεται υπερβολικά και έτσι η απόδοση της εφαρμογής $RUBiS$ μειώνεται κατακόρυφα.



Σχήμα 5.3: Μέση τιμή χρήσης σε CPU - μέσος χρόνος απάντησης στα αιτήματα χρηστών (mRT).

5.2 Λόγος χρήσης - κατανομής

Μια σημαντική παράμετρος που παίζει καθοριστικό ρόλο στην απόδοση της δυναμικής κατανομής των υπολογιστικών πόρων είναι ο λόγος των ποσοστών της χρήσης και της κατανομής του CPU ή αλλιώς το "c parameter". Η παράμετρος αυτή έχει άμεση σχέση με το περιθώριο δηλαδή την ποσότητα των επιπλέον υπολογιστικών πόρων που παρέχονται από τα συστήματα ελέγχου που θα μελετηθούν στη συνέχεια της εργασίας.

Η σχέση μεταξύ του περιθωρίου και της "παραμέτρου c" παρουσιάζεται στην εξίσωση:

$$c = \frac{1}{1+h}, \tag{5.1}$$

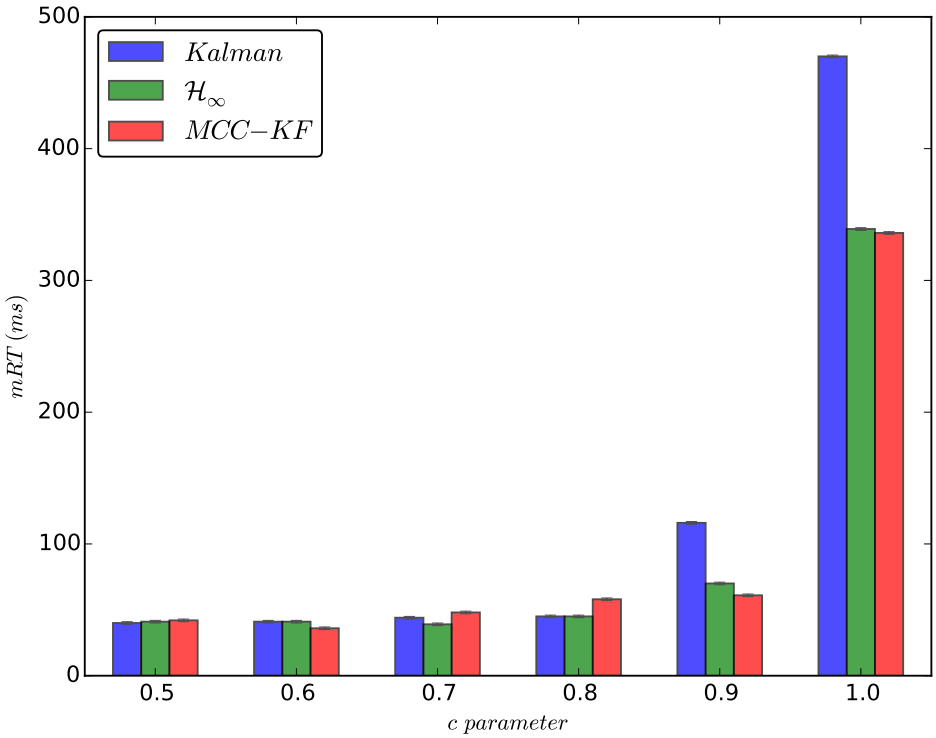
όπου

$$h \in (0, 1).$$

Η Σχέση 5.1 δείχνει ότι καθώς μικραίνει το περιθώριο h τότε η παράμετρος c αυξάνεται και έτσι περισσότεροι υπολογιστικοί πόροι εξοικονομούνται για την λειτουργία άλλων εφαρμογών στον ίδιο εξυπηρετητή. Παρόλα αυτά η προσέγγιση της παραμέτρου c στην τιμή 1 μπορεί να επιδράσει αρνητικά στην απόδοση

της υφιστάμενης εφαρμογής αφού το περιθώριο h των επιπλέον υπολογιστικών πόρων θα είναι πολύ μικρό με αποτέλεσμα την έλλειψη πόρων προκειμένου να ολοκληρωθούν τα αιτήματα των χρηστών της εφαρμογής.

Στο συγκεκριμένο πείραμα τέθηκε σταθερός αριθμός χρηστών στους 500, έτσι ώστε να εξακριβωθεί η παράμετρος c κατά την οποία το mRT αρχίζει να αυξάνεται και η εφαρμογή $RUBiS$ να επηρεάζεται αρνητικά. Επίσης η πρόβλεψη και η προσαρμογή των ανάλογων κατανομών έγινε με την βοήθεια του κώδικα *ViResA* που αναπτύχθηκε για τον σκοπό αυτό. Στο Σχήμα 5.4 παρουσιάζεται ο μέσος χρόνος απάντησης των αιτημάτων για κάθε εξομοίωση όπου μεταβάλλεται η παράμετρος c με αλλαγή στην αντίστοιχη μεταβλητή του κώδικα *ViResA* για κάθε έναν από τους ελεγκτές που χρησιμοποιήθηκαν στην εργασία αυτή.



Σχήμα 5.4: Μέσος χρόνος απάντησης στα αιτήματα χρηστών σε σχέση με την παράμετρο c.

Όπως μπορεί να παρατηρηθεί από το Σχήμα 5.4, το φίλτρο Kalman έχει μεγαλύτερο συνολικό mRT από τα άλλα δύο φίλτρα όταν η παράμετρος c ξεπεράσει την τιμή 0.8. Τα φίλτρα $\mathcal{H}_\infty$ και MCC-KF ξεκινούν να επιδρούν αρνητικά στην απόδοση της εφαρμογής όταν η παράμετρος c ξεπεράσει την τιμή 0.9 αφού τότε αυξάνεται και για αυτό το mRT κατακόρυφα. Άρα τα φίλτρα $\mathcal{H}_\infty$ και MCC-KF μπορούν να εξοικονομήσουν περισσότερους υπολογιστικούς πόρους για την λειτουργία άλλων εφαρμογών στον ίδιο εξυπηρετητή χωρίς να επηρεάζεται σημαντικά η απόδοση της υπάρχουσας εφαρμογής $RUBiS$. Για τον λόγο αυτό κα-

θορίστηκε η παράμετρος c στην τιμής 0.8 για τα επόμενα πειράματα.

5.3 Προσαρμογή ελεγκτών στο σύστημα

Στην ενότητα αυτή θα παρουσιαστούν τα αποτελέσματα των επιδόσεων για όλους τους ελεγκτές που χρησιμοποιήθηκαν στην εργασία αυτή, δηλαδή για το φίλτρο Kalman, $\mathcal{H}_\infty$, MCC-KF και τον ευφυή ελεγκτή ANFIS. Στα πειράματα που θα αναφερθούν στη συνέχεια, χρησιμοποιήθηκε σταθερός φόρτος εργασίας καθ' όλη την διάρκεια των πειραμάτων, εκτός από κάποια συγκεκριμένη χρονική στιγμή όπου δημιουργούνταν επιπλέον αιτήματα από τον εξομοιωτή χρηστών έτσι ώστε να παρατηρηθεί η επίδοση, στις στιγμές των ξαφνικών αλλαγών του φόρτο εργασίας, της εφαρμογής $RUBiS$.

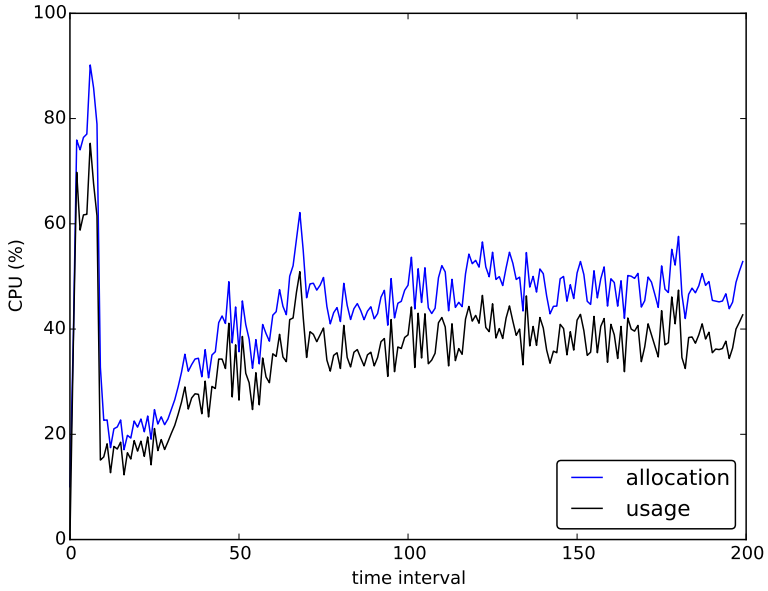
Για την πειράματα αυτά χρησιμοποιήθηκαν 500 χρήστες οι οποίοι δημιουργούσαν αιτήματα μέσω της σελίδας διαδικτυακών δημοπρασιών $RUBiS$. Ο τύπος των αιτημάτων που χρησιμοποιήθηκε ήταν του είδους περιήγησης (Browsing Mix - BR). Η παράμετρος c τέθηκε στο 0.8. Για όλα τα συστήματα ελέγχου καθορίστηκαν οι ακόλουθες αρχικές τιμές: αρχική συνδιασπορά σφάλματος P_o στο 10, η διασπορά του θορύβου διαδικασίας W τέθηκε στην τιμή 4 ενώ η διασπορά του θορύβου μέτρησης V τέθηκε στην τιμή 1. Οι μετρήσεις που θα παρουσιαστούν στις επόμενες ενότητες αφορούν αποκλειστικά τις χρήσεις και κατανομές που προσαρμόστηκαν, μόνο για το στοιχείο του διακομιστή ιστού που βρίσκεται στη φυσική μηχανή $PM2$.

5.3.1 Φίλτρο Kalman

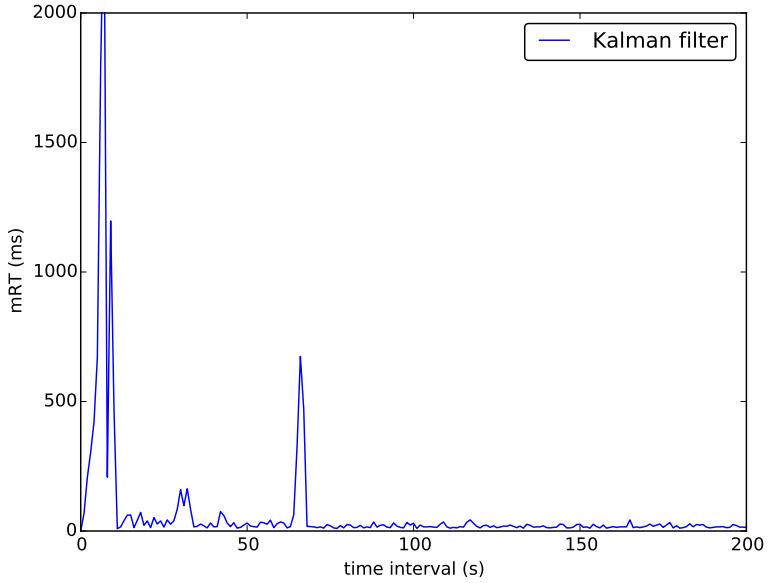
Το φίλτρο Kalman χρησιμοποιήθηκε στην παρούσα εργασία έτσι ώστε να «ακολουθεί» το ποσοστό των χρήσεων σε υπολογιστικούς πόρους και στη συνέχεια να ανανεώνει την αντίστοιχη κατανομή. Συγκεκριμένα το φίλτρο αυτό χρησιμοποιήθηκε και στις εργασίες [1] και [14]. Το φίλτρο Kalman αποτελεί τον πιο συνηθισμένο τρόπο για τέτοιου είδους προβλήματα λόγω της απλότητας, ευστροφίας αλλά και της απόδοσης του. Παρακάτω παρουσιάζεται η απόδοση του φίλτρου Kalman καθώς αλλάζει απότομα ο φόρτος εργασίας της εφαρμογής $RUBiS$.

Στο Σχήμα 5.5 παρουσιάζεται με γραμμή μαύρου χρώματος, το ποσοστό της χρήσης σε CPU για το VM του διακομιστή ιστού, καθώς δημιουργούνται αιτήματα από τους χρήστες που εξομοιώνει ο εξομοιωτής χρηστών της φυσικής μηχανής $PM1$. Με μπλε χρώμα φαίνονται οι αντίστοιχες προβλέψεις για την δυναμική κατανομή που υπολόγισε και στη συνέχεια προσάρμοσε το φίλτρο

Kalman στο VM του διακομιστή ιστού.



Σχήμα 5.5: Χρήση και κατανομή CPU (Kalman Filter).



Σχήμα 5.6: Μέσος χρόνος απάντησης στα αιτήματα των χρηστών (mRT) (Kalman Filter).

Στην αρχή του πειράματος δημιουργούνται 500 χρήστες που στέλνουν αιτήματα στην εφαρμογή $RUBiS$ και έτσι αυξάνεται απότομα η χρήση σε υπολογιστικούς πόρους, επηρεάζοντας αρνητικά την απόδοση της εφαρμογής κατά την χρονική διάρκεια αυτή. Την χρονική στιγμή 60 προστίθενται επιπλέον 200 χρήστες στην εφαρμογή $RUBiS$ και έτσι σαν αποτέλεσμα αυξάνεται σχετικά απότομα το ποσοστό της χρήσης σε υπολογιστικούς πόρους. Όπως μπορεί να παρατηρηθεί από το Σχήμα 5.5 το φίλτρο Kalman συμπεριφέρεται αρκετά καλά όταν

οι διακυμάνσεις της χρήσης είναι σχετικά μικρές. Αυτό συμβαίνει όταν ο αριθμός των χρηστών είναι σταθερός στους 500 ή ακόμη και μετά που προστέθηκαν οι επιπλέον 200 χρήστες. Γενικά οι προβλέψεις του φίλτρου Kalman δίνουν αρκετά καλά αποτελέσματα και έτσι η κατανομή ακολουθεί πιστά τις αντίστοιχες χρήσεις. Το Σχήμα 5.6 παρουσιάζει την απόδοση της εφαρμογής $RUBiS$ με την βοήθεια του mRT. Όπως μπορεί να παρατηρηθεί το mRT παραμένει χαμηλό καθώς ο αριθμός των χρηστών που στέλνουν αιτήματα στην εφαρμογή είναι σταθερός. Κατά την χρονική διάρκεια των 60 - 65 δευτερολέπτων όπου προστίθενται επιπλέον 200 χρήστες στην εφαρμογή, τότε το mRT ξεπερνά τα 700ms και επιστρέφει ξανά στα χαμηλά του επίπεδα όταν ο αριθμός των χρηστών σταθεροποιηθεί στους 700 δηλαδή μετά τα 65 δευτερόλεπτα. Η μεγάλη αύξηση του mRT συμβαίνει λόγου του ότι, την συγκεκριμένη χρονική στιγμή που προσθέτονται οι επιπλέον χρήστες, η ζήτηση για υπολογιστικούς πόρους αυξάνεται. Για τον λόγο αυτό το φίλτρο Kalman, πρόβλεψε την νέα κατανομή βάση των σχετικά σταθερών αυξομειώσεων της χρήσης και έτσι προσάρμοσε μια κατανομή η οποία είχε σχετικά μικρό περιθώριο μεταξύ της αντίστοιχης χρήσης. Το μικρό περιθώριο αυτό επηρεάζει την απόδοση της εφαρμογής $RUBiS$ αφού αυξάνεται το mRT.

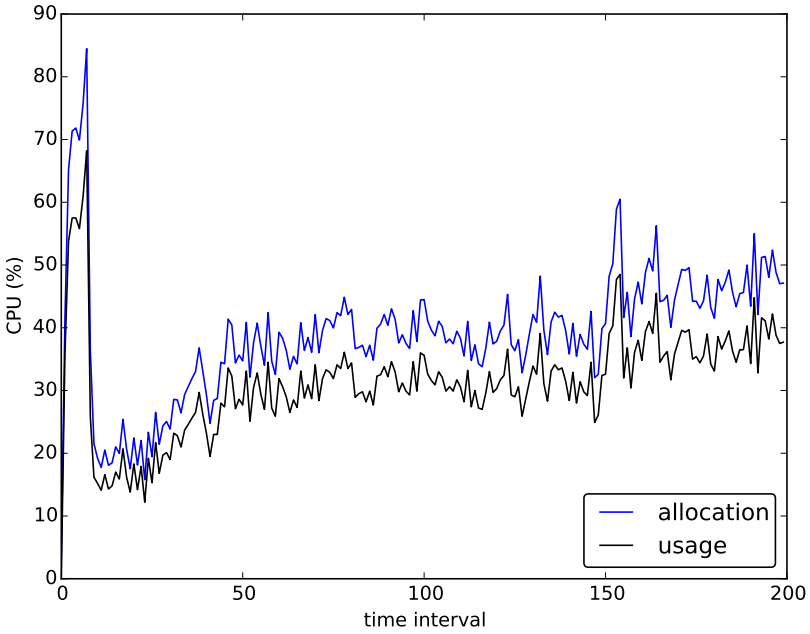
5.3.2 Φίλτρο $\mathcal{H}_\infty$

Η τυχαιότητα του προβλήματος της δυναμικής κατανομής των υπολογιστικών πόρων, αλλά και ο θόρυβος οδήγησε σε μια εναλλακτική λύση για την πρόβλεψη και την προσαρμογή της κατανομής. Οι εκδότες στο [15] πρότειναν την λύση του προβλήματος της δυναμικής κατανομής υπολογιστικών πόρων χρησιμοποιώντας το φίλτρο $\mathcal{H}_\infty$. Χρησιμοποιώντας το φίλτρο αυτό κατάφεραν να μειώσουν το μέγιστο σφάλμα λόγω της στοχαστικής φύσης του μοντέλου.

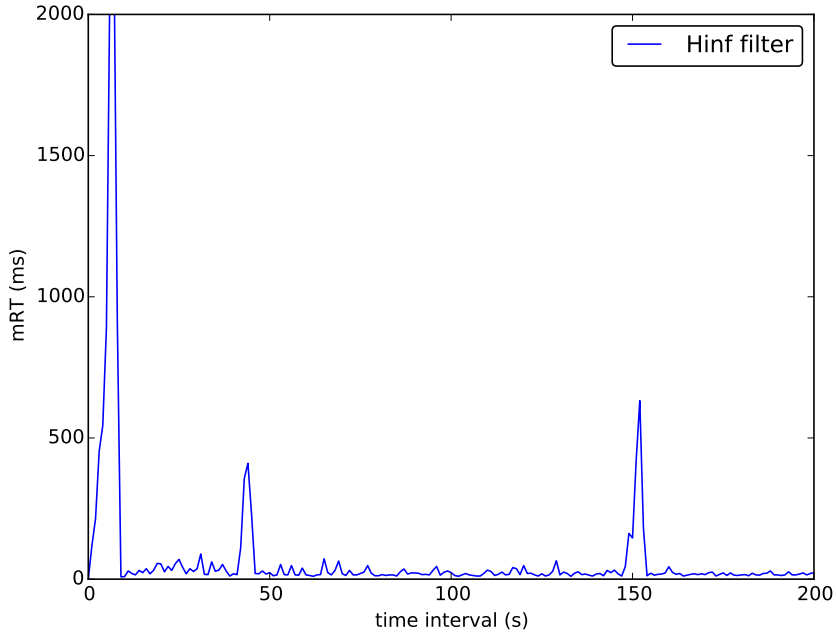
Στο Σχήμα 5.7 παρουσιάζεται το ποσοστό της χρήσης σε CPU για το VM του διακομιστή ιστού, με γραμμή μαύρου χρώματος, καθώς δημιουργούνται αιτήματα από τους χρήστες που εξομοιώνει ο εξομοιωτής χρηστών της φυσικής μηχανής $PM1$. Με μπλε χρώμα φαίνονται οι αντίστοιχες προβλέψεις για την δυναμική κατανομή που υπολόγισε και στη συνέχεια προσάρμοσε το φίλτρο $\mathcal{H}_\infty$ στο VM του διακομιστή ιστού.

Αρχικά δημιουργούνται 500 χρήστες, μέσω του εξομοιωτή χρηστών, και ξεκινούν την περιήγηση τους στην εφαρμογή $RUBiS$ στέλνοντας τα αιτήματά τους. Για τον λόγο αυτό αυξάνεται απότομα η χρήση σε υπολογιστικούς πόρους, επηρεάζοντας αρνητικά την απόδοση της εφαρμογής κατά την χρονική διάρκεια 0 - 10. Στη συνέχεια κατά την χρονική διάρκεια 45 - 50 λόγω κάποιας τυχαίας μεταβολής της ζήτησης των 500 χρηστών, αυξάνεται το mRT στα 450ms όπως φαίνε-

ται στο Σχήμα 5.8. Την χρονική στιγμή 150 προστίθενται επιπλέον 200 χρήστες στην εφαρμογή $RUBiS$ με αποτέλεσμα να αυξάνεται ξαφνικά το ποσοστό της χρήσης σε υπολογιστικούς πόρους, λόγω της σχετικά υψηλής ζήτησης.



Σχήμα 5.7: Χρήση και κατανομή CPU ($\mathcal{H}_\infty$ Filter).



Σχήμα 5.8: Μέσος χρόνος απάντησης στα αιτήματα των χρηστών (mRT) ($\mathcal{H}_\infty$ Filter).

Το Σχήμα 5.8 παρουσιάζει τα αποτελέσματα του mRT ανά δευτερόλεπτο, κατά την διάρκεια του πειράματος. Στο σχήμα αυτό μπορούν να φανούν τα κρίσιμα σημεία όπου υπάρχουν απότομες αλλαγές στην ζήτηση υπολογιστικών πό-

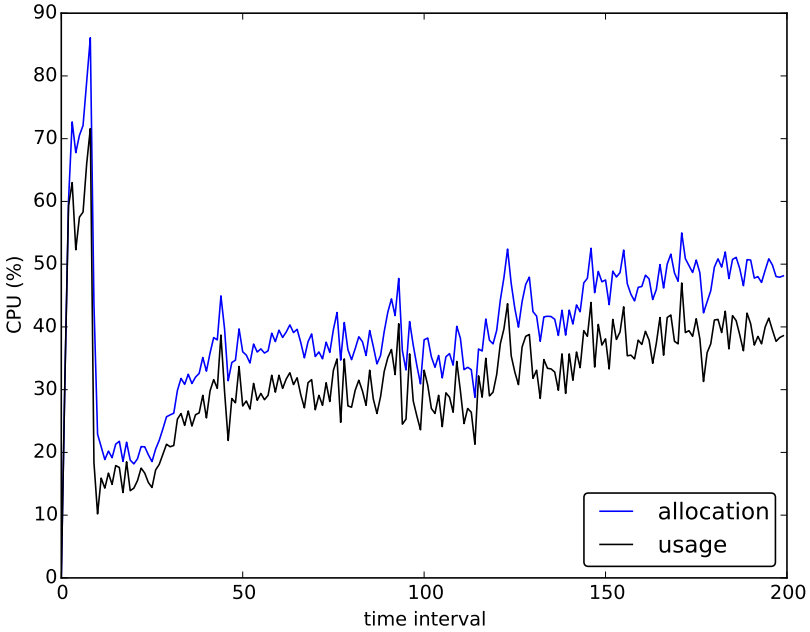
ρων από τους χρήστες. Στην αρχή του πειράματος το mRT αυξάνεται κατακόρυφα στα 2000ms λόγω της ξαφνικής φόρτωσης της εφαρμογής με 500 χρήστες. Στη συνέχεια λόγω κάποιων τυχαίων αλλαγών στην ζήτηση των 500 χρηστών, το mRT αυξάνεται στα 450ms δηλαδή στα κανονικά πλαίσια μικρής αλλαγής του φόρτου εργασίας. Την χρονική στιγμή 150s παρατηρείται η προσθήκη των επιπλέον 200 χρηστών, με αποτέλεσμα το mRT να φτάνει σε ψηλά επίπεδα των 650ms. Ωστόσο καθ' όλη την υπόλοιπη διάρκεια του πειράματος, το φίλτρο $\mathcal{H}_\infty$ συμπεριφέρεται καλά κρατώντας χαμηλό το mRT.

5.3.3 Φίλτρο MCC-KF

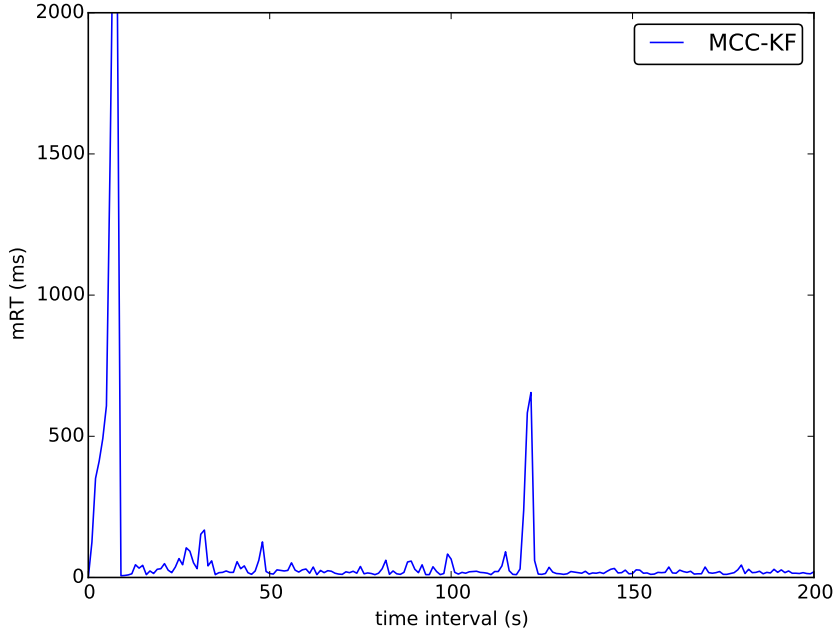
Στην ενότητα αυτή παρουσιάζονται τα αποτελέσματα της πρόβλεψης και προσαρμογής της κατανομής υπολογιστικών πόρων στον διακομιστή ιστού, χρησιμοποιώντας τον ελεγκτή Kalman με το κριτηρίο μέγιστης συνεντροπίας. Το φίλτρο αυτό είναι παρόμοιο με το κλασσικό φίλτρο Kalman, με την διαφορά ότι συμπεριφέρεται αρκετά καλύτερα όταν ο θόρυβος της διαδικασίας δεν είναι γκαουσιανής μορφής, λόγω του κριτηρίου μέγιστης συνεντροπίας που χρησιμοποιεί. Όπως και στα προηγούμενα πειράματα, έτσι και αυτό ξεκινά με την δημιουργία 500 χρηστών στην εφαρμογή $RUBiS$. Στο Σχήμα 5.9, οι χρήσεις σε CPU παρουσιάζονται με μαύρο χρώμα ενώ οι αντίστοιχες κατανομές που προβλέφθηκαν και προσαρμόστηκαν στην βαθμίδα του διακομιστή ιστού του VM από το MCC-KF με μπλε χρώμα.

Αρχικά παρατηρείται ένα μεγάλο ποσοστό χρήσης CPU λόγω της απότομης αύξησης της ζήτησης σε υπολογιστικούς πόρους με την δημιουργία 500 νέων χρηστών. Την χρονική στιγμή 115 ο εξομοιωτής χρηστών δημιουργεί επιπλέον 200 χρήστες οι οποίοι κάνουν περιήγηση στη σελίδα του $RUBiS$ δημιουργώντας έτσι τα αιτήματα τους και παράλληλα επιπλέον φόρτο εργασίας στα στρώματα της εφαρμογής. Συγκεκριμένα ο αριθμός των χρηστών κατά την χρονική διάρκεια των 110 - 115 δευτερολέπτων, αυξάνεται από 500 σε 700 όπως και στα προηγούμενα πειράματα, αυξάνοντας έτσι και την ζήτηση σε υπολογιστικούς πόρους.

Η αύξηση του mRT για το φίλτρο MCC-KF μπορεί να παρατηρηθεί από το Σχήμα 5.10, το οποίο δείχνει το mRT να αυξάνεται κατά τις χρονικές διάρκειες 0 - 10 και 115 - 120. Συγκεκριμένα στην αρχή του πειράματος, (0 - 10 δευτερόλεπτα), το mRT αυξάνεται στα 2000ms όπως και στην περίπτωση του κλασσικού φίλτρου Kalman αλλά και του φίλτρου $\mathcal{H}_\infty$. Την χρονική διάρκεια 115 - 120 το mRT αυξάνεται στα 600ms, και έτσι επιτυγχάνεται η τήρηση του ορίου του 1s που ορίστηκε σαν QoS. Όταν ο φόρτος εργασίας ήταν σταθερός δηλαδή ο αριθμός των ταυτόχρονων συνδεδεμένων χρηστών ήταν ο ίδιος τότε το mRT βρι-



Σχήμα 5.9: Χρήση και κατανομή CPU (MCC-Kalman Filter).



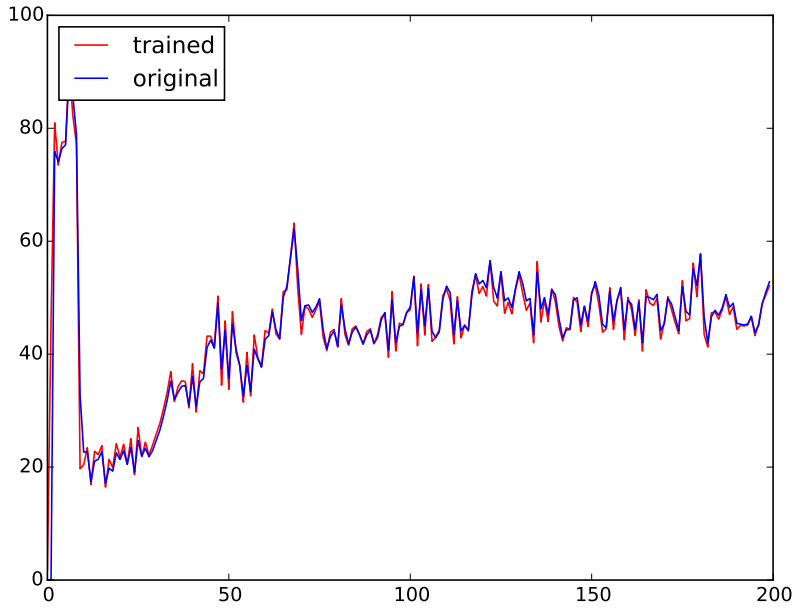
Σχήμα 5.10: Μέσος χρόνος απάντησης στα αιτήματα των χρηστών (mRT) (MCC-Kalman Filter).

σκόταν στα χαμηλά του επίπεδα όπως ήταν λογικό και από τα προηγούμενα φίλτρα.

5.3.4 Προσαρμοστικός Νευροασαφής Ελεγκτής (ANFIS)

Στην υποενότητα αυτή, εξακριβώθηκε η λειτουργία του ελεγκτή ασαφής λογικής ANFIS - Fuzzy Inference, αφού έγινε εκπαίδευση με βάση τα στατιστικά των χρήσεων - κατανομών προηγούμενων πειραμάτων με το φίλτρο Kalman. Στο πείραμα αυτό, χρησιμοποιήθηκε ο ευφυής ελεγκτής Fuzzy Inference με το μοντέλο που υλοποιήθηκε στην εργασία [13] όπου η ζήτηση και τα στατιστικά ήταν συνθετικά. Στην παρούσα όμως εργασία, αναπτύχθηκε ο κώδικας ViResA έτσι ώστε ο έλεγχος να γίνεται με πραγματικά πλέον δεδομένα, χάριν στην εφαρμογή $RUBiS$ που αναπτύχθηκε στην υποδομή που υλοποιήθηκε.

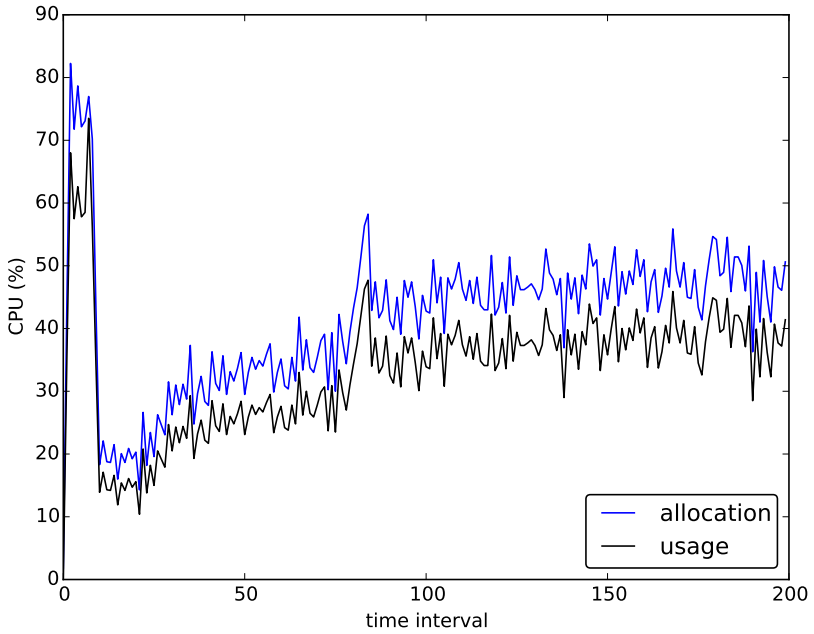
Αρχικά για το πείραμα αυτό, φορτώθηκαν τα στατιστικά του φίλτρου Kalman, στον ελεγκτή ANFIS, έτσι ώστε να γίνει εκπαίδευση με βάση τα δεδομένα παρόμοιου φόρτου εργασίας. Στο Σχήμα 5.11 παρουσιάζονται τα αποτελέσματα εκπαίδευσης για την κατανομή χρησιμοποιώντας τα στατιστικά των κατανομών που προσαρμόστηκαν στο φίλτρο Kalman. Με μπλε χρώμα οι κατανομές προς εκπαίδευση ενώ με κόκκινο χρώμα οι εκπαιδευμένες κατανομές για τον ελεγκτή ANFIS. Όπως μπορεί να γίνει αντιληπτό, η εκπαίδευση του ANFIS, «ακολουθεί» πιστά τις κατανομές του φίλτρο Kalman.



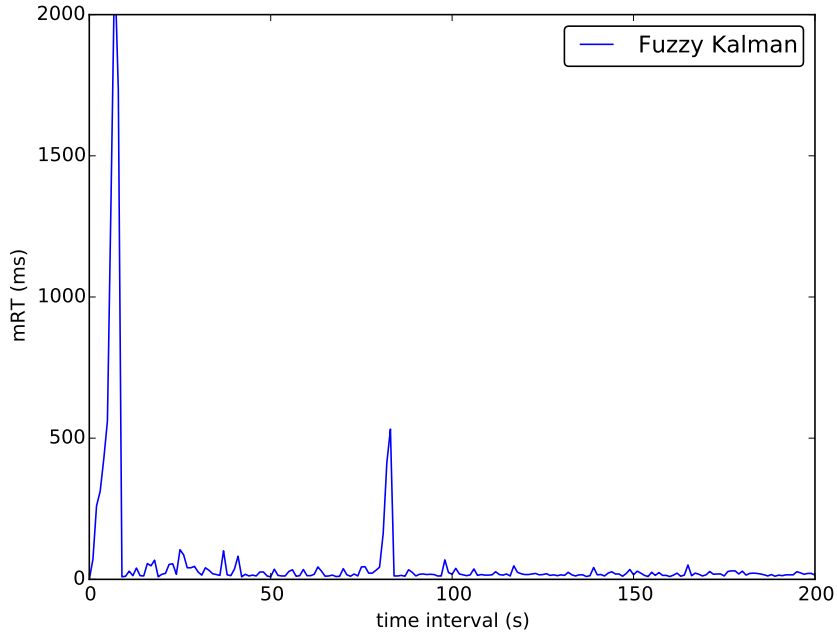
Σχήμα 5.11: Αποτελέσματα εκπαίδευσης Fuzzy με βάση προηγούμενα στατιστικά του φίλτρου Kalman.

Στη συνέχεια αφού έγινε η εκπαίδευση του ελεγκτή ANFIS, δημιουργήθηκε φόρτος εργασίας για 500 χρήστες στην εφαρμογή $RUBiS$, έτσι ώστε να εξακριβωθεί η απόδοση της λειτουργίας του ελεγκτή. Στο Σχήμα 5.12 παρουσιάζονται με μαύρο χρώμα το ποσοστό χρήσης σε CPU, ενώ με μπλε χρώμα οι αντίστοι-

χες κατανομές που προσαρμόστηκαν με τον ελεγκτή ANFIS. Στο Σχήμα 5.13 φαίνεται η απόδοση της εφαρμογής $RUBiS$ μέσω των mRT ανά δευτερόλεπτο του πειράματος.



Σχήμα 5.12: Χρήση και κατανομή CPU Fuzzy Controller.



Σχήμα 5.13: Μέσος χρόνος απάντησης στα αιτήματα των χρηστών (mRT) - Fuzzy Controller.

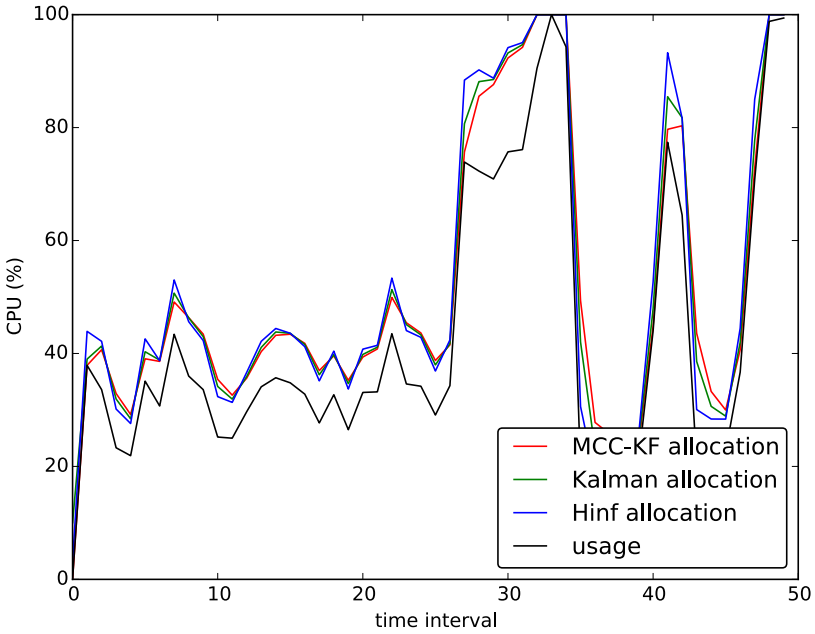
Στην αρχή του πειράματος παρατηρείται μια απότομη αύξηση του ποσοστού χρήσης CPU λόγω του ξαφνικού φόρτου εργασίας ο οποίος δημιουργή-

θηκε λόγω των 500 χρηστών που άρχισαν να στέλνουν αιτήματα στην εφαρμογή $RUBiS$. Λόγω αυτής της απότομης αλλαγής στον φόρτο εργασίας στο διάστημα αυτό, παρατηρείται μια αύξηση στον χρόνο απάντησης στα αιτήματα των χρηστών περίπου στα 2000ms, όπως φαίνεται στο Σχήμα 5.12. Στη συνέχεια από το χρονικό διάστημα 10 - 80 ο φόρτος εργασίας είναι σχετικά σταθερός και έτσι ο ελεγκτής ANFIS κατανέμει αρκετά καλά τους υπολογιστικούς πόρους κρατώντας χαμηλό το mRT. Την χρονική στιγμή 80s, αυξάνεται ο φόρτος εργασίας με την δημιουργία επιπλέον 200 χρηστών στην εφαρμογή και έτσι παρατηρείται μερική αύξηση στις χρήσεις και αντίστοιχα στις κατανομές σε CPU, οι οποίες θα δημιουργήσουν μια αύξηση του mRT στα 500ms. Από την χρονική στιγμή 90s μέχρι και το τέλος του πειράματος ο αριθμός των χρηστών παραμένει σταθερός στους 700 και έτσι το mRT παραμένει στις χαμηλές του τιμές.

5.4 Σύγκριση

Εκτός από την απόδοση της εφαρμογής του $RUBiS$ μέσω του mRT που πρέπει να κρατηθεί κάτω από το όριο που θέσαμε σαν QoS, μελετήθηκε και η απόδοση στην πρόβλεψη των κατανομών με βάση κοινού φόρτου εργασίας. Στο πείραμα αυτό οι κατανομές που προβλέφθηκαν από τα φίλτρα, δεν προσαρμόστηκαν στο σύστημα με την χρήση του Xen Hypervisor όπως τα προηγούμενα πειράματα για τον λόγο ότι μόνο μια προσαρμογή κατανομής μπορεί να γίνει κάθε φορά σε ένα στοιχείο του συμπλέγματος $RUBiS$. Ωστόσο το πείραμα αυτό παρουσιάζει ξεκάθαρα τον τρόπο με τον οποίο το κάθε φίλτρο ανταποκρίνεται στις ξαφνικές αλλαγές της ζήτησης, προβλέποντας τις ανάλογες κατανομές με βάση τις χρήσεις. Για τον λόγο αυτό, το συγκεκριμένο πείραμα μπορεί να δώσει μια πλήρη εικόνα για την απόδοση των ελεγκτών.

Το Σχήμα 5.14 παρουσιάζει ένα φόρτο εργασίας σε ποσοστά χρήσης CPU, για 500 χρήστες. Όπως φαίνεται στο σχήμα αυτό, κατά την χρονική διάρκεια 25 - 50 υπάρχουν μεγάλες αλλαγές στα ποσοστά χρήσης CPU λόγω της ξαφνικής μεταβολής του φόρτου εργασίας που τέθηκε. Όπως μπορεί να παρατηρηθεί από το Σχήμα 5.14, το φίλτρο $\mathcal{H}_\infty$ παρέχει περισσότερους υπολογιστικούς πόρους από τα υπόλοιπα φίλτρα, ενώ το φίλτρο MCC-KF παρέχει λιγότερους υπολογιστικούς πόρους βάση των κατανομών που φαίνεται να προβλέπει. Ωστόσο έχει περίπου την ίδια απόδοση όπως αναφέρθηκε στην προηγούμενη ενότητα σε σχέση με το mRT, πράγμα που τον καθιστά πιο αποδοτικό στην εξοικονόμηση υπολογιστικών πόρων για άλλες εφαρμογές χωρίς να μειώνεται αισθητά η απόδοση της υφιστάμενης εφαρμογής $RUBiS$.



Σχήμα 5.14: Προβλέψεις της κατανομής για κοινό φόρτο εργασίας ανά ελεγκτή.

5.5 Συμπεράσματα

Σύμφωνα με τα παραπάνω αποτελέσματα που παρουσιάστηκαν, μπορεί να γίνει κατανοητό ότι το φίλτρο MCC-KF συμπεριφέρεται αρκετά καλά όσο και το φίλτρο $\mathcal{H}_\infty$ στις απότομες αλλαγές του φόρτου εργασίας. Ωστόσο, η φύση του φίλτρου $\mathcal{H}_\infty$ δίνει περισσότερους πόρους έτσι ώστε να κρατάει το mRT όσο πιο χαμηλά γίνεται. Εν αντιθέσει το φίλτρο MCC-KF εξοικονομεί πόρους για άλλες εφαρμογές που ίσως θα μπορούσαν να αναπτυχθούν στην ίδια φυσική μηχανή, χωρίς να επηρεάζεται σχεδόν καθόλου η απόδοση της υπάρχουσας εφαρμογής. Το φίλτρο Kalman, συμπεριφέρεται αρκετά καλά καθ' όλη την διάρκεια των πειραμάτων με την διαφορά ότι στις απότομες αλλαγές του φόρτου εργασίας δεν καταφέρνει να έχει απόδοση τόσο καλή όσο οι υπόλοιποι ελεγκτές. Σε ξεχωριστή κατηγορία, μπορούμε να πούμε ότι ο βέλτιστος τρόπος κατανομής υπολογιστικών πόρων είναι με την χρήση του ευφυούς ελεγκτή ANFIS, ο οποίος είχε την καλύτερη απόδοση στα πειράματα που παρουσιάστηκαν στο κεφάλαιο αυτό. Ωστόσο, ένας τέτοιου είδους ελεγκτής απαιτεί εκπαίδευση με μοτίβα ζήτησης παρόμοια με αυτά που πρόκειται να προσαρμόσει τις ανάλογες κατανομές.

Κεφάλαιο 6

Σχετικές Εργασίες

Στο κεφάλαιο αυτό αναφέρονται παρόμοιες εργασίες με την παρούσα διπλωματική εργασία, που έγιναν μέχρι στιγμής. Η Ενότητα 6.1 αναφέρεται σε έλεγχο της κατανομής υπολογιστικών πόρων με τη χρήση ελεγκτών ανάδρασης. Στον έλεγχο με βάση μελλοντικές προβλέψεις χρήσης πόρων αναφέρεται η Ενότητα 6.2. Η Ενότητα 6.3 παρουσιάζει εργασίες όπου ο έλεγχος γινόταν με την χρήση φίλτρων. Τέλος η Ενότητα 6.4 αναφέρεται σε εργασίες που έγιναν με ευφυή έλεγχο ή με την χρήση μεθόδων μηχανικής μάθησης.

6.1 Έλεγχος με ανάδραση

Ο έλεγχος με συστήματα ανάδρασης συζητήθηκε στις εργασίες [27], [28] και [29]. Στις εργασίες αυτές χρησιμοποιήθηκαν διάφοροι ελεγκτές έτσι ώστε να ρυθμίζουν το mRT σε περιπτώσεις όπου κάποιο στοιχείο του συμπλέγματος είχε περίσσεια ή έλλειψη από υπολογιστικούς πόρους.

Συγκεκριμένα στην εργασία [27] παρουσιάστηκε ένας ελεγκτής Αναλογικής Ολοκλήρωσης (Proportional-Integral PI) ο οποίος ρύθμιζε τους μέσους χρόνους απάντησης αιτημάτων των χρηστών με τον έλεγχο της ανάλογης κατανομής. Ο ελεγκτής που χρησιμοποιήθηκε στην συγκεκριμένη εργασία συμπεριφερόταν καλά αφού προσαρμοζόταν ανάλογα με την αλλαγή των διαφόρων παραμέτρων του συστήματος.

Στην εργασία [28] μελετήθηκε η συμπεριφορά τριών διαφορετικών ελεγκτών σε συνθήκες υπερφόρτωσης ή υποφόρτωσης υπολογιστικών πόρων. Ο πρώτος ελεγκτής που χρησιμοποιήθηκε ήταν και αυτός Αναλογικής Ολοκλήρωσης (Proportional-Integral PI). Λόγω της μη γραμμικής συμπεριφοράς του προβλήματος όταν οι υπολογιστικοί πόροι άλλαζαν από την περιοχή όπου υπήρχε περίσσεια υπολογιστικών πόρων στην περιοχή όπου υπήρχε έλλειψη υπολογιστικών

πόρων τότε ο ελεγκτής αποτύγχανε να προσαρμόσει τις κατανομές. Ο δεύτερος ελεγκτής στην εργασία αυτή ήταν ένας μη γραμμικός Ολοκληρωτικός (Integral I) ο οποίος μπορούσε να προσαρμόσει το κέρδος ανάλογα. Ο ελεγκτής αυτός ρύθμιζε την σχετική χρήση CPU, σε μια τιμή αναφοράς, και προσάρμοζε το κέρδος του ανάλογα έτσι ώστε να ανταποκρίνεται γρήγορα σε υπερφορτώσεις εργασίας και πιο αργά στις περιοχές όπου υπήρχε περίσσεια υπολογιστικών πόρων. Ο τρίτος ελεγκτής που προτάθηκε ήταν κι αυτός προσαρμοστικός και μπορούσε να ελέγχει την αντίστροφο τιμή του mRT και την σχετική χρήση στις περιοχές όπου οι υπολογιστικοί πόροι ήταν περισσότεροι από όσοι πραγματικά χρειάζονταν.

Οι εκδότες στο [29] παρουσίασαν ένα ένθετο σύστημα ελέγχου. Ο εσωτερικός βρόχος του συστήματος αποτελείτο από ένα ελεγκτή Ολοκλήρωσης (Integral I) προσαρμοστικού κέρδους ο οποίος ρύθμιζε τη σχετική χρήση. Ο εξωτερικός βρόχος του συστήματος αποτελείτο και αυτός από ένα ελεγκτή Ολοκλήρωσης ο οποίος ρύθμιζε το σημείο αναφοράς της σχετικής χρήσης έτσι ώστε να κρατηθεί το mRT σε ένα συγκεκριμένο όριο που θα μπορούσε να καθορίσει ο χρήστης. Η έξοδος του εξωτερικού βρόχου του συστήματος οδηγούσε στην είσοδο του εσωτερικού βρόχου χωρίς όμως να δοθεί η αιτιολόγηση του εξωτερικού βρόχου από τους εκδότες.

6.2 Έλεγχος με πρόβλεψη

Στην ενότητα αυτή παρουσιάζεται η εργασια που έγινε χρησιμοποιώντας προληπτικές μεθόδους για την ρύθμιση των παραμέτρων, προβλέποντας μελλοντικές χρήσεις βάση προηγούμενων μετρήσεων.

Στην εργασία [30] παρουσιάστηκε ένας ελεγκτής ο οποίος ρύθμιζε την σχετική χρήση, με την χρήση τριών αλγορίθμων σειριακών προβλέψεων. Με βάση αυτούς τους αλγορίθμους, το μοντέλο της πρόβλεψης ανταποκρινόταν αρκετά καλά και γρήγορα όταν οι προηγούμενες μετρήσεις ήταν πολύ κοντά στις προβλέψιμες. Όταν όμως οι τιμές των προηγούμενων μετρήσεων ήταν μακριά από τις τιμές των επόμενων προβλέψεων, δηλαδή όταν υπήρχαν νέες τάσεις ως προς την χρήση CPU, τότε οι ελεγκτές δεν είχαν καλή απόδοση.

6.3 Έλεγχος με τη χρήση φίλτρων

Ένα φίλτρο Εκθετικά Σταθμισμένου Κινούμενου Μέσου Όρου (EWMA) χρησιμοποιήθηκε στην εργασία [31] με σκοπό τον υπολογισμό μελλοντικών ζητήσεων σε υπολογιστικούς πόρους με βάση πρηγούμενων μετρήσεων. Το φίλτρο

αυτό μπορούσε να λειτουργήσει με ένα από τους πολλούς τρόπους απόκρισης. Δηλαδή με βάση την τιμή της παραμέτρου εξομάλυνσης, το φίλτρο μπορούσε να αποκρίνεται απότομα ή πιο ομαλά στις αλλαγές της χρήσης. Ωστόσο μια τέτοια παράμετρος καθιστά το φίλτρο να μην προσαρμόζεται δυναμικά σε άλλες συνθήκες.

Οι εκδότες της εργασίας [32], χρησιμοποίησαν ένα φίλτρο flop-flip βασισμένο στα φίλτρα flip-flop που παρουσιάστηκαν στην εργασία [33]. Συγκεκριμένα το φίλτρο αυτό χρησιμοποιούσε ένα Κινούμενο Μέσο Όρο για χρονική διάρκεια 30 δευτερολέπτων κατά την οποία ο μέσος όρος αυτός άλλαζε εάν ο υπολογισμός είχε μεγαλύτερη τιμή από μια καθορισμένη τυπική απόκλιση.

6.4 Ευφυής έλεγχος και μηχανική μάθηση

Για την διαχείριση υπολογιστικών πόρων χρησιμοποιήθηκαν και άλλες τεχνικές όπως ο ευφυής έλεγχος και η μηχανική μάθηση. Στην εργασία [34] παρουσιάστηκε ένα σύστημα διαχείρισης υπολογιστικών πόρων χρησιμοποιώντας μοντελοποίηση ANFIS έτσι ώστε να εκπαιδευτεί σε διάφορες καταστάσεις φόρτου εργασίας και ζήτησης υπολογιστικών πόρων.

Η εργασία [35] είχε να κάνει με την εφαρμογή της μεθόδου Ενίσχυσης Μάθησης (Reinforcement Learning) για το πρόβλημα της κατανομής των υπολογιστικών πόρων. Το σύστημα αυτό αποτελείτο από δύο στρώματα τα οποία εφαρμόζονταν σε ένα κέντρο δεδομένων. Το πρώτο στρώμα έπρεπε να παρέχει μια γραφική της χρήσης για κάθε εφαρμογή του κέντρου δεδομένων ενώ το δεύτερο στρώμα αποφάσιζε πως να κατανέμει τους υπολογιστικούς πόρους για κάθε εφαρμογή με βάση των χρήσεων του πρώτου στρώματος.

Στην εργασία [13] οι εκδότες υλοποίησαν ένα σύστημα το οποίο μοντελοποιούσε διάφορα σενάρια ζήτησης έτσι ώστε εκπαιδεύοντας ένα ελεγκτή ANFIS με βάση προηγούμενες μετρήσεις να κατανέμονταν οι υπολογιστικοί πόροι ανάλογα. Οι εκδότες στην εργασία αυτή απέδειξαν ότι το σύστημα αυτό που υλοποίησαν ανταποκρίνεται καλύτερα από τα φίτλρα Kalman και $\mathcal{H}_\infty$, αφού έκαναν μια σύγκριση μεταξύ των ελεγκτών.

Κεφάλαιο 7

Επίλογος

Η τεχνολογία της εικονικοποίησης αναπτύσσεται παρέχοντας δυνατότητες δυναμικής κατανομής υπολογιστικών πόρων για κάθε στοιχείο που συνθέτει μια εφαρμογή που φιλοξενείται σε ένα υπολογιστικό κέντρο δεδομένων. Με την σωστή εκμετάλλευση των δυνατοτήτων αυτών, θα πρέπει να γίνεται σωστός έλεγχος έτσι ώστε να επιτυγχάνεται εξοικονόμηση υπολογιστικών πόρων χωρίς να επηρεάζεται η ποιότητα υπηρεσίας από τον παροχέα. Οι τεχνολογίες ενοποίησης των εικονικών μηχανών, μοιράζουν πόρους μεταξύ τους, ενεργοποιώντας έτσι όλες τις δυνατότητες για την αποτελεσματική διαχείριση τους. Εκτός αυτού, με τη σωστή διαχείριση των υπολογιστικών πόρων, η φιλοξενία περισσότερων εφαρμογών σε κοινή μηχανή αφήνει περιθώρια βελτίωσης στη γρήγορη και σωστή διακίνηση των δεδομένων χωρίς να παραβιάζεται οποιοδήποτε πρωτόκολλο ασφάλειας. Με την συμβολή της εργασίας αυτής έγινε ένα πρώτο βήμα στον ευφυή έλεγχο της κατανομής υπολογιστικών πόρων σε εικονικοποιημένους εξυπηρετητές με την ανάπτυξη του συστήματος ViResA.

7.1 Περίληψη

Στο Κεφάλαιο 1 έγινε μια εισαγωγή στον στόχο και στα κίνητρα που δόθηκαν για την μελέτη στο θέμα της δυναμικής κατανομής υπολογιστικών πόρων σε κέντρα δεδομένων χρησιμοποιώντας σαν πλατφόρμα αξιολόγησης μια εφαρμογή υπολογιστικής νέφους. Παρουσιάστηκε επίσης και η οργάνωση της δομής αυτής της πτυχιακής εργασίας.

Όλο το θεωρητικό υπόβαθρο σχετικά με την τεχνολογία υπολογιστικής νέφους παρουσιάζονται στο Κεφάλαιο 2. Στη συνέχεια δίνονται τα διάφορα χαρακτηριστικά της αρχιτεκτονικής των κέντρων δεδομένων, τις διάφορες υπηρεσίες που παρέχουν αλλά και τα μέρη τα οποία το συνθέτουν. Επίσης στο κεφά-

λαιο αυτό γίνεται μια εξήγηση των μέτρων αξιολόγησης μιας εφαρμογής υπολογιστικής νέφους και της κατανομής υπολογιστικών πόρων. Τέλος παρουσιάζονται οι διάφοροι τύποι εικονικοποίησης και δίνονται διάφορα παραδείγματα και πλεονεκτήματα για το κάθε είδος.

Στο Κεφάλαιο 3 παρουσιάζεται η αρχιτεκτονική του συστήματος και των υποσυστημάτων από τα οποία αποτελείται. Συγκεκριμένα αναλύεται η υποδομή του υλικού που χρησιμοποιήθηκε όπως π.χ. φυσικών μηχανών - εξυπηρετητών, δικτυακών συσκευών αλλά και όλες οι τροποποιήσεις που έγιναν με βάση το μοντέλο που υλοποιήθηκε. Στη συνέχεια παρουσιάζεται και αναλύεται ο υπερεπόπτης Xen που χρησιμοποιήθηκε για την δυναμική κατανομή των υπολογιστικών πόρων αλλά και για την δημιουργία και διαχείριση των εικονικών μηχανών. Τέλος στο κεφάλαιο αυτό γίνεται μια εξήγηση του τρόπου λειτουργίας της εφαρμογής $RUBiS$ και αναλύονται όλες οι λεπτομέρειες σχετικά με τον τρόπο εγκατάστασης και τροποποίησης της έτσι ώστε να εξάγουμε τα μέτρα αξιολόγησης όπως π.χ. το mRT.

Το Κεφάλαιο 4 δίνει μια πλήρη εικόνα του συστάματος δυναμικής κατανομής υπολογιστικών πόρων ViResA που αναπτύχθηκε. Στη συνέχεια εξάγονται οι διάφορες εξισώσεις που περιγράφουν την φύση του προβλήματος με βάση την θεωρεία συστημάτων ελέγχου. Τέλος περιγράφονται οι ελεγκτές που χρησιμοποιήθηκαν για την πρόβλεψη της ζήτησης και την προσαρμογή της κατανομής δηλαδή το φίλτρο Kalman, το φίλτρο $\mathcal{H}_\infty$,το φίλτρο MCC-KF και ο ελεγκτής ασαφής λογικής Fuzzy.

Στο Κεφάλαιο 5 παρουσιάζονται όλα τα πειράματα που έγιναν για την εξαγωγή των αποτελεσμάτων που εξάχθηκαν για την εξακρίβωση της σωστής λειτουργίας του συστήματος. Αρχικά παρουσιάζονται τα πειράματα που έγιναν για την αναγνώριση του συστήματος με σκοπό να τεθεί σαν σημείο αναφοράς για μελλοντικά πειράματα. Στη συνέχεια γίνεται μια περιγραφή με βάση τα αποτελέσματα του πειράματος για εξαγωγή της τιμής του περιθωρίου η οποία θα έπαιζε καθοριστικό ρόλο στην αποδοτική κατανομή των πόρων από τα συστήματα ελέγχου. Τέλος αφού έγινε μια παρουσίαση όλων των αποτελεσμάτων για το mRT, τις χρήσεις και κατανομές για κάθε ελεγκτή ξεχωριστά, έγινε μια σύγκριση μεταξύ τους για την εξαγωγή συμπερασμάτων ως προς τον τρόπο απόκρισης και απόδοσης του καθενός.

Το Κεφάλαιο 6 παρουσιάζει συνοπτικά, σχετικές εργασίες που έγιναν στο παρελθόν για την βελτιστοποίηση της δυναμικής κατανομής υπολογιστικών πόρων σε κέντρα δεδομένων. Συγκεκριμένα αναφέρονται οι εργασίες ανάλογα με τον τρόπο που γίνεται ο έλεγχος αλλά και τον τρόπο εξαγωγής των μέτρων αξιολόγησης τους.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί περαιτέρω, τουλάχιστον ως προς τρεις κατευθύνσεις. Συγκεκριμένα, το σύστημα θα μπορούσε να βελτιωθεί σε σχέση με την πλατφόρμα αξιολόγησης των αποτελεσμάτων, την αρχιτεκτονική του συστήματος για ελεγκτές πολλαπλών εισόδων πολλαπλών εξόδων (MIMO) αλλά και την δυναμική κατανομή άλλων φυσικών πόρων όπως μνήμης (RAM). Επίσης σημαντικό είναι να εξάγουμε την παράμετρο c, ή και το περιθώριο, και να μπορεί να προσαρμοστεί δυναμικά καθώς ο φόρτος εργασίας αλλάζει.

7.2.1 Βελτιστοποίηση του συστήματος αξιολόγησης.

Με την τριβή στην εφαρμογή RUBiS που χρησιμοποιήθηκε στην παρούσα εργασία, παρατηρήθηκαν πολλές δυσκολίες ως προς την ανάπτυξη της. Συγκεκριμένα η εφαρμογή αυτή δεν παρέχει πλέον την δυνατότητα στους προγραμματιστές να αναπτύξουν όλα τα στοιχεία της εφαρμογής λόγω της έλλειψης συμβατότητας για τα διάφορα λογισμικά που χρησιμοποιεί το κάθε στοιχείο. Σχετικά με το σύστημα δυναμικής κατανομής υπολογιστικών πόρων ViResA, θα πρέπει να αναπτυχθεί αναλόγως έτσι ώστε να καταγράφονται περισσότερα στατιστικά ως προς την απόδοση της εφαρμογής αλλά και των ελεγκτών που χρησιμοποιούνται. Επίσης σε επόμενες εργασίες θα πρέπει να αναπτυχθεί ένας αλγόριθμος ο οποίος θα μπορεί να «ανιχνεύει» το σωστό περιθώριο χρήσης-κατανομής, κατά την διάρκεια λειτουργίας της εφαρμογής και να το προσαρμόζει ανάλογα στους ελεγκτές.

7.2.2 Σχεδιασμός και ανάπτυξη για συστήματα MIMO.

Στην εργασία αυτή χρησιμοποιήθηκαν ελεγκτές μονής εισόδου μονής εξόδου SISO, όπου ο κάθε ελεγκης μπορούσε να έχει σαν είσοδο τις χρήσεις για ένα μόνο στοιχείο της εφαρμογής. Τέτοιου είδους ελεγκτές σχεδιάστηκαν και υλοποιήθηκαν στην εργασία αυτή και προσαρμόστηκαν συγκεκριμένα για τον διακομιστή ιστού. Μια άλλη προσέγγιση όμως που θα έδινε αυξημένη απόδοση και ως προς τις εφαρμογές αλλά και ως προς την σωστή κατανομή των πόρων των φυσικών μηχανών, θα ήταν ένα σύστημα πολλαπλών εισόδων πολλαπλών εξόδων MIMO. Με αυτό το μοντέλο, οι ελεγκτές θα έχουν την δυνατότητα να κατανέμουν ανάλογα και πιο αποδοτικά τους υπολογιστικούς πόρους σε περισσότερα VMs ταυτόχρονα, με βάση τις αντίστοιχες χρήσεις που θα περνούν στην

είσοδό τους. Με τον τρόπο αυτό αυξάνεται και η δυνατότητα φιλοξενίας περισσοτέρων VMs σε μια φυσική μηχανή με αποτέλεσμα και την πιο οικονομική λύση στα προβλήματα των κέντρων δεδομένων.

7.2.3 Κατανομή πόρων μνήμης RAM.

Ένας άλλος σημαντικός παράγοντας στην απόδοση μιας εφαρμογής υπολογιστικής νέφους είναι η χρήση μνήμης από τα ανάλογα στοιχεία που συνθέτουν την εφαρμογή. Η εργασία αυτή δεν αναφέρθηκε σε άλλου είδους φυσικών πόρων εκτός από υπολογιστικούς πόρους CPU αλλά σε μελλοντικά στάδια θα μελετηθούν περαιτέρω άλλοι παράγοντες που επηρεάζουν την απόδοση των εφαρμογών νέφους.

7.3 Συμπεράσματα

Με την ολοκλήρωση της εργασίας αυτής οδηγηθήκαμε στα ακόλουθα συμπεράσματα:

1. *Η δυναμική κατανομή υπολογιστικών πόρων σε κέντρα δεδομένων γίνεται εφικτή με την υιοθέτηση φίλτρων ή ελεγκτών που μπορούν να προβλέπουν την μελλοντική χρήση σε υπολογιστικούς πόρους, με βάση προηγούμενες μετρήσεις και να τους κατανέμουν ανάλογα για την αποτελεσματική χρήση τους από τις διάφορες εφαρμογές υπολογιστικής νέφους.*

2. *Το φίλτρο Kalman εκτός από την απλότητα που χαρακτηρίζεται, συμπεριφέρεται και αρκετά καλά όσο οι αλλαγές στον φόρτο εργασίας παραμένουν σχετικά μικρές, αλλά λόγω απότομων αλλαγών στον φόρτο εργασίας, δεν μπορεί να ανταποκριθεί άμεσα αυξάνοντας έτσι το mRT. Το φίλτρο $\mathcal{H}_\infty$ αποδίδει αρκετά καλά ακόμη και σε πιο απότομες αλλαγές του φόρτου εργασίας με το να κατανέμει περισσότερους πόρους για τις διάφορες λειτουργίες της εφαρμογής. Το φίλτρο MCC-KF ανταποκρίνεται αρκετά καλά όπως και το φίλτρο $\mathcal{H}_\infty$ με την διαφορά ότι κατανέμει πιο λίγους υπολογιστικούς πόρους, αυξάνοντας τα περιθώρια για ανάπτυξη επιπλέον εφαρμογών στην ίδια φυσική μηχανή.*

3. *Με την μικρότερη δυνατή διαφορά περιθωρίου χρήσης-κατανομής εξοικονομούνται αρκετοί υπολογιστικοί πόροι, παρέχοντας την δυνατότητα στους εξυπηρετητές για να φιλοξενούν περισσότερες εφαρμογές που μοιράζονται τους φυσικούς της πόρους.*

Παράρτημα Α'

Πηγαίος Κώδικας συστήματος ViResA

Α'.1 Κώδικας MCC Kalman filter

```
#!/usr/bin/python3

def mcc_kf_siso(P, Q, R, H, y, x):
    # Prior estimation
    F=1
    x_n = F*x
    P_n = F*P + Q

    # Posterior estimation
    R_inv = R
    P_p = P_n
    r = y - H*x # innovation term
    r_norm = np.sqrt(r*R_inv*r)
    sigma = 1 * r_norm
    L = np.exp(-(r_norm**2) / (2*sigma**2))
    K = 1/( (1/P_p) + L*H*R_inv*H ) * (L*H*R_inv)
    x_n = x_n + K*r
    P_n = (1 - K*H)*P_p*(1 - K*H) + K*R*K

    if x_n > 100:
        x_n = 100
    elif x_n < 0:
        x_n = 0
    elif x_n < y:
        x_n = y

    return (x_n,P_n,K)
```

Ο παραπάνω κώδικας αποτελεί την συνάρτηση που υλοποιεί το φίλτρο Kalman με την χρήση του κριτηρίου μέγιστης συνεντροπίας, ο οποίος βρίσκεται καταχωρημένος στο αποθετήριο του Bitbucket μαζί με τις υπόλοιπες συναρτήσεις που αποτελούν τον ολοκληρωμένο σχεδιασμό του κώδικα ViResA.

Βιβλιογραφία

[1] *E. Kalyvianaki, T. Charalambous, and S. Hand, "Adaptive resource provisioning for virtualized servers using Kalman filters," ACM Transaction on Autonomous and Adaptive Systems, vol. 9, no. 2, pp. 10:1–10:35, July 2014.*

[2] *"Hypertext Transfer Protocol," Apr. 2017, page Version ID: 777712636. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=777712636*

[3] *M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," IEEE Network, vol. 14, no. 3, pp. 30–37, May/June 2000.*

[4] *IDC, "Virtualization and multicore innovations disrupting the worldwide server market," March 2007.*

[5] *"Xen Project 4.4.1." [Online]. Available: https://www.xenproject.org/downloads/xen-archives/supported-xen-44-series/xen-441.html*

[6] *"Debian – News – Updated Debian 8: 8.6 released." [Online]. Available: https://www.debian.org/News/2016/20160917*

[7] *"SSH," Jun. 2016, page Version ID: 5914736. [Online]. Available: https://el.wikipedia.org/w/index.php?title=SSH&oldid=5914736*

[8] *C. Amza, E. Cecchet, A. Chanda, A. L. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel, "Specification and implementation of dynamic web site benchmarks," in 5th Workshop on Workload Characterization, no. LABOS-CONF-2005-016, 2002.*

[9] *E. Cecchet, J. Marguerite, and W. Zwaenepoel, "Performance and scalability of ejb applications," in ACM Sigplan Notices, vol. 37, no. 11. ACM, 2002, pp. 246–261.*

[10] *P. Padala, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive Control of Virtualized Resources in Utility Computing Environments," in Proceedings of the European Conference on Computer Systems (EuroSys), 2007, pp. 289–302.*

[11] "TPC-Homepage V5." [Online]. Available: http://www.tpc.org/default.asp

[12] Atlassian, "Bitbucket | The Git solution for professional teams." [Online]. Available: https://bitbucket.org

[13] K. M. Deliparaschos, T. Charalambous, E. Kalyvianaki, and C. Makarounas, "On the use of fuzzy logic controllers to comply with virtualized application demands in the cloud," in European Control Conference (ECC), June 2016, pp. 649–654.

[14] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-Adaptive and Self-Configured CPU Resource Provisioning for Virtualized Servers using Kalman Filters," in Proceedings of the 6th International Conference on Autonomic Computing (ICAC). New York, NY, USA: ACM, 2009, pp. 117–126.

[15] T. Charalambous and E. Kalyviannaki, "A min-max framework for CPU resource provisioning in virtualized servers using $\mathcal{H}_\infty$ Filters," in 49th IEEE Conference on Decision and Control (CDC), Dec. 2010, pp. 3778–3783.

[16] R. E. Kalman et al., "A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[17] D. Simon, Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches. John Wiley & Sons, 2006.

[18] B. Chen, X. Liu, H. Zhao, and J. C. Príncipe, "Maximum Correntropy Kalman Filter," arXiv:1509.04580 [cs, stat], Sep. 2015, arXiv: 1509.04580. [Online]. Available: http://arxiv.org/abs/1509.04580

[19] R. Izanloo, S. A. Fakoorian, H. S. Yazdi, and D. Simon, "Kalman filtering based on the maximum correntropy criterion in the presence of non-Gaussian noise," in Annual Conference on Information Science and Systems (CISS), Mar. 2016, pp. 500–505.

[20] W. Liu, P. P. Pokharel, and J. C. Príncipe, "Correntropy: A Localized Similarity Measure," in The 2006 IEEE International Joint Conference on Neural Network Proceedings, 2006, pp. 4919–4924.

[21] ——, "Correntropy: Properties and Applications in Non-Gaussian Signal Processing," IEEE Transactions on Signal Processing, vol. 55, no. 11, pp. 5286–5298, Nov. 2007.

[22] R. He, W. S. Zheng, and B. G. Hu, "Maximum Correntropy Criterion for Robust Face Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 8, pp. 1561–1576, Aug. 2011.

[23] *A. Singh and J. C. Principe, "Using Correntropy As a Cost Function in Linear Adaptive Filters," in Proceedings of the 2009 International Joint Conference on Neural Networks, ser. IJCNN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1699–1704. [Online]. Available: http://dl.acm.org/citation.cfm?id=1704175.1704421*

[24] *T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. on Syst., Man, and Cyber., vol. 15, no. 1, pp. 116–132, Feb. 1985.*

[25] *L. Kleinrock, Queueing Systems, Volume 1, Theory. Wiley-Interscience, 1975.*

[26] *E. Kalyvianaki, "Resource provisioning for virtualized server applications," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-762, Nov. 2009. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-762.pdf*

[27] *X. Liu, X. Zhu, S. Singhal, and M. Arlitt, "Adaptive Entitlement Control of Resource Containers on Shared Servers," in Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, 2005, pp. 163–176.*

[28] *Z. Wang, X. Zhu, and S. Singhal, "Utilization and SLO-Based Control for Dynamic Sizing of Resource Partitions," in Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM), October 2005, pp. 133–144.*

[29] *X. Zhu, Z. Wang, and S. Singhal, "Utility-Driven Workload Management using Nested Control Design," in Proceedings of the American Control Conference (ACC), 2006, pp. 6033–6038.*

[30] *W. Xu, X. Zhu, S. Singhal, and Z. Wang, "Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers," in Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2006, pp. 115–126.*

[31] *B. Urgaonkar and P. Shenoy, "Sharc: Managing CPU and Network Bandwidth in Shared Clusters," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 1, pp. 2–17, 2004.*

[32] *J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle, "Managing Energy and Server Resources in Hosting Centres," in Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), 2001, pp. 103–116.*

[33] M. Kim and B. Noble, "Mobile Network Estimation," in Proceedings of the ACM Annual Conference on Mobile Computing and Networking (MobiCom), 2001, pp. 298–309.

[34] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "On the Use of Fuzzy Modeling in Virtualized Data Center Management," in Proceedings of the Fourth International Conference on Autonomic Computing (ICAC). Washington, DC, USA: IEEE Computer Society, 2007.

[35] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "On the Use of Hybrid Reinforcement Learning for Autonomic Resource Allocation," Cluster Computing, vol. 10, no. 3, pp. 287–299, 2007.

Συντομογραφίες - Αρκτικόλεξα - - Ακρωνύμια

βλπ	*βλέπε*
κ.α.	*και άλλα*
π.χ.	*παραδείγματος χάριν*
κ.λπ.	*και λοιπά*
κ.ο.κ	*και ούτω καθεξής*
PM	*Physical Machine*
VM	*Virtual Machine*
VMM	*Virtual Machine Manager*
CPU	*Central Processing Unit*
pCPU	*physical Central Processing Unit*
vCPU	*virtual Central Processing Unit*
BIOS	*Basic Input Output System*
RAM	*Random Access Memory*
HTTP	*Hypertext Transfer Protocol*
HTML	*Hypertext Markup Language*
ssh	*secure shell*
IP	*Internet Protocol*
LVM	*Logical Volume Manager*
Dom0	*Domain 0*
RUBiS	*Rice University Bidding System*
mRT	*mean Response Time*
QoS	*Quality of Service*
SLA	*Service Level Agreement*
SISO	*Single Input Single Output*
MIMO	*Multiple Input Multiple Output*
ANFIS	*Adaptive Neuro Fuzzy Inference System*

Απόδοση ξενόγλωσσων όρων

Απόδοση	*Ξενόγλωσσος όρος*
εξυπηρετητής	*server*
επεκτασιμότητα	*scalability*
εικονικοποίηση	*virtualization*
υπερεπόπτης	*hypervisor*
αιτήματα	*requests*
βαθμίδα	*tier*
στοιχείο	*component*
εξομοιωτής χρηστών	*client emulator*
απομακρυσμένη πρόσβαση	*remote access*
διαμέριση	*partitioning*
διαχωριστικό στρώμα	*abstraction layer*
φυσικός τόμος	*physical volume*
ομάδα τόμων	*volume group*
λογικός τόμος	*logical volume*
παραεικονικοποίηση	*paravirtualization*
περιθώριο	*headroom*
βάση δεδομένων	*database*