Ατομική Διπλωματική Εργασία

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΠΕΙΡΑΜΑΤΙΚΟΥ FRAMEWORK ΓΙΑ ΤΗΝ ΜΕΛΕΤΗ CLOUD STREAMING ΕΦΑΡΜΟΓΗΣ ΣΤΟ GOOGLE CLOUD

Δανάκης Χριστοδουλίδης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2018

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός Και Υλοποίηση Πειραματικού
Framework Για Την Μελέτη Cloud Streaming
Εφαρμογής Στο Google Cloud

Δανάκης Χριστοδουλίδης

Επιβλέπων Καθηγητής
Μάριος Δ. Δικαιάκος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2018

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κύριο Μάριο Δικαιάκο που μου έκανε την τιμή και με επέλεξε να συνεργαστούμε στα πλαίσια αυτής της Διπλωματικής εργασίας. Δουλεύοντας μαζί του διεύρυνα τις γνώσεις μου, όχι μόνο στο συγκεκριμένο θέμα που πραγματεύεται η εργασία μου αλλά και σε τομείς που θα μου φανούν χρήσιμοι στην μετέπειτα επαγγελματική μου σταδιοδρομία.

Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον developer του εργαστήριου Αθανάσιο Τρύφωνος και τον μεταπτυχιακό φοιτητή και ερευνητή Ζαχαρία Γεωργίου για την πολύτιμη και καθοριστική καθοδήγηση που μου παρείχαν κατά τη διάρκεια εκπόνησης της Διπλωματικής μου εργασίας.

Τέλος θα ήθελα να ευχαριστήσω το οικογενειακό και στενό φιλικό περιβάλλον μου για την υποστήριξη που μου παρείχαν σε όλους τους τομείς καθόλη τη διάρκεια των σπουδών μου.

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η μελέτη μιας video streaming εφαρμογής στο cloud κάτω από συνθήκες υψηλού φόρτου εργασίας .

Στα πλαίσια της διπλωματικής αυτής εργασίας αναπτύχθηκε μια video streaming εφαρμογή που χρησιμοποιήθηκε ως UseCase της έρευνας αυτής .

Στην συνέχεια έγινε η υλοποίηση της, και αφού έγινε η ανάπτυξη της εφαρμογής ακολούθησε το deployment της στο Cloud για την αξιολόγηση της και έγιναν κάποιες απαραίτητες παρατηρήσεις που αφορούν την συμπεριφορά της εφαρμογής και την αλληλεπίδραση του χρήστη με την εφαρμογή .

Μετά ακολούθησε η φάση της μοντελοποίησης με βάση τις παρατηρήσεις που έχουν προκύψει από την φάση του deployment και γίνετε μια καταγραφή των μετρικών που θα λαμβάνονται υπόψη στο σχεδιασμό του πειράματος αλλά και που θα συλλέγονται κατά την εκτέλεση του πειράματος .

Επίσης σε αυτή την φάση γίνεται μια μοντελοποίηση της video streaming εφαρμογής που αναπτύχθηκε για να είναι εφικτή η συλλογή των μετρικών κατά την εκτέλεση του πειράματος , στην ουσία προκύπτει μια νέα εφαρμογή η οποία μοντελοποιεί την εφαρμογή video streaming που αναπτύχθηκε αρχικά .

Στην συνέχεια έρχεται η φάση του σχεδιασμού του πειράματος όπου εδώ δημιουργείτε ένα πειραματικό Framework το οποίο αποτελείτε από τρία συστατικά στοιχεία , την εφαρμογή η οποία προέκυψε από την μοντελοποίηση της video streaming εφαρμογής , έναν workload generator ο οποίος είναι απαραίτητος για να γίνει εφικτή η προσομοίωση των χρηστών της εφαρμογής και της ποσότητας εργασίας σε πραγματικά σενάρια, και ένα εργαλείο παρακολούθησης και συλλογής μετρικών το οποίο αναπτύχθηκε για την συλλογή των μετρικών , την αναπαράσταση τους , και την αποθήκευση τους.

Εφόσον πλέων υπάρχει το Framework γίνεται υπόθεση κάποιον σεναρίων που αντιστοιχούν σε πραγματικά σενάρια η εκτέλεση τους και παράλληλα γίνετε η συλλογή των μετρικών που θα μας βοηθούν στην μελέτη και στην εξαγωγή των Συμπερασμάτων.

Δανάκης Χριστοδουλίδης – Πανεπιστήμιο Κύπρου, Μάιος 2018

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

Είναι γεγονός ότι το “Cloud” έχει μπει για τα καλά στην ζωή μας και στην καθημερινότητα μας. Ο αριθμός των υπηρεσιών που βασίζονται στο Cloud μεγαλώνει συνεχώς και οι τεχνολογίες του Cloud ξεπροβάλουν σε καινούριους τομείς με ασταμάτητο ρυθμό. Πολλοί άνθρωποι χρησιμοποιούν τέτοιες εφαρμογές και υπηρεσίες χωρίς καν να το γνωρίζουν . Καθώς όμως αυξάνονται οι δυνατότητες των υπηρεσιών και οι χρήστες υπηρεσιών Cloud αυξάνονται και οι υποχρεώσεις των παρόχων των υπηρεσιών διότι ο τελικός χρήστης θέλει να έχει αυτό που πληρώνει . Μια κατηγορία τέτοιων υπηρεσιών είναι οι εφαρμογές που αφορούν το streaming οι οποίες έχουν τεράστιο όγκο δεδομένων και υψηλές απαιτήσεις σε πόρους και σε υπολογιστική ισχύ. Έτσι οι εφαρμογές αυτές πρέπει να παρακολουθούνται με κάποιο τρόπο έτσι ώστε αν και όταν χρειαστεί πρέπει να ληφθούν σημαντικές αποφάσεις που αφορούν την υποδομή τους. Στα πλαίσια της διπλωματικής εργασίας αυτής έχοντας ως στόχο την μελέτη μιας τέτοιας εφαρμογής κάτω από συνθήκες μεγάλου φόρτου εργασίας έχει αναπτυχθεί μια εφαρμογή video streaming και έγινε deploy στο cloud . Έχοντας πλέων την εφαρμογή, σχεδιάστηκε ένα πείραμα και προέκυψε ένα πειραματικό framework το οποίο δίνει την δυνατότητα για τη προσομοίωση της εφαρμογής σε πραγματικά σενίαρα και τη συλλογή μετρικών κατά την διάρκεια των πειραμάτων . Τέλος μετά την εκτέλεση των πειραμάτων

, μέσω των αποτελεσμάτων προκύπτουν συμπεράσματα για τις ενέργειες που μπορεί να πράξει ένας πάροχος τέτοιων υπηρεσιών .

1.2 Motivation

Τα τελευταία χρόνια βλέπουμε μια τάση οι εφαρμογές να παίρνουν την μορφή online υπηρεσιών με τους παρόχους τέτοιων υπηρεσιών να κάνουν την μετάβαση στο “Cloud” όπως το office365[2] και το eclipse che [3]. Το γεγονός ότι οι εταιρίες είτε μικρές είτε μεγάλες έχουν πρώτη τους προτεραιότητα το “Cloud” δεν είναι τυχαίο . Το γεγονός ότι δεν περνάει μέρα και να μην ακούσεις τις λέξεις Spotify[4] και Netflix[5] επίσης δεν είναι τυχαίο. Ο λόγος δεν είναι μόνο τα οικονομικά κέρδη αλλά και το γεγονός ότι το Cloud προσφέρει στις επιχειρήσεις την ευκαιρία να κάνουν περισσότερα πράγματα γρηγορότερα και καλύτερα. Ένα απλό παράδειγμα είναι το Netflix το οποίο ξεκίνησε ως επιχείρηση ενοικίασης και πώλησης DVD και μέσω του Cloud την σήμερον ημέρα είναι μια video streaming υπηρεσία με 135 εκατομμύρια συνδρομητές όπως και το Spotify το οποίο είναι μια audio streaming υπηρεσία με 170 εκατομμύρια χρήστες . Δύστυχος η ευτυχώς πολλοί τομείς που σήμερα δεν μπορεί καν να σκεφτεί το μυαλό μας στο μέλλον θα είναι μια μορφή Cloud υπηρεσίας είτε επί πληρωμή είτε δωρεάν .Όμως όλα αυτά έχουν απαιτήσεις όπως μικρό χρόνο απόκρισης , συνείχες διαθεσιμότητα πράγμα που σημαίνει ότι οι υπηρεσίες αυτές πρέπει να παρακολουθούνται . Η παρακολούθηση είναι σημαντική διότι βοηθά έναν παροχέα να ενημερώνετε όταν η εφαρμογή του δεν είναι διαθέσιμη , και έχοντας ακριβή δεδομένα σχετικά με την απόδοση της εφαρμογής του στον Cloud Server του είναι το κλειδί για τη λήψη σημαντικών αποφάσεων σχετικά με τον τρόπο διατήρησης και βελτιστοποίησης της υποδομής της εφαρμογής του. Έτσι όταν γίνετε συστηματική παρακολούθηση των υπηρεσιών Cloud και γίνονται οι κατάλληλες ενέργειες από του παρόχους όταν χρειαστεί τότε οι τελικοί χρήστες θα απολαμβάνουν στο μέγιστο δυνατό καλύτερες υπηρεσίες και οι παρόχοι θα μεγιστοποιήσουν τα κέρδη τους κάνοντας τη ζωή τους ποιο εύκολη .

1.3 Συνεισφορά

Σκοπός της διπλωματικής εργασίας αυτής είναι η μελέτη μιας Cloud streaming εφαρμογής κάτω από συνθήκες μεγάλου φόρτου εργασίας . Μέσο της προσπάθειας για την κατανόηση του τι γίνεται σε τέτοια περίπτωση και του πώς αντιμετωπίζονται τα προβλήματα που προκύπτουν η έρευνα αυτή συνεισφέρει τα εξής:

- Μια εφαρμογή / πλατφόρμα για video streaming που με μικρής τροποποιήσεις μπορεί να χρησιμοποιηθεί απροβλημάτιστα εφόσον ικανοποιεί όλες τις απαιτήσεις μιας σύγχρονης εφαρμογής (πχ ασφάλεια , απόδοση , επεκτασιμότητα , κατανοητός κώδικας κ.α.) . Προαφαιρεί λειτουργίες στους χρήστες όπως δημιουργία λογαριασμού , login , logout , upload video , αναπαραγωγής βίντεο με προσαρμοζόμενη ποιότητα και διαγραφή βίντεο.
- Μέθοδος συλλογής και αποθήκευσης μετρικών
- Ένα πειραματικό Framework το οποίο δίνει την δυνατότητα σε κάποιον να πειραματιστεί εκτελώντας πειράματα με αντιστοίχου είδους εφαρμογές streaming.
- Εντοπισμός των μετρικών κλειδιών που πρέπει να δίνει έμφαση κάποιος όσο αφορά την επεκτασιμότητα (Scaling) στο Cloud και σε ποιες περιπτώσεις πρέπει να γίνει μια ενέργεια επεκτασιμότητας (Scaling up).

1.4 Δομή εγγράφου

Στο κεφάλαιο 2 γίνεται αναφορά στο υπόβαθρο που βρίσκετε πίσω από την μελέτη αυτή και γίνετε μια εισαγωγή σε έννοιες όπως το cloud computing , παρακολούθηση (monitoring) και ελαστικότητα- επεκτασιμότητα (elasticity - Scalability).

Στο κεφάλαιο 3 παρουσιάζονται κάποια εργαλεία και τεχνολογίες που σχετίζονται με τις έννοιες που παρουσιάζονται στο κεφάλαιο 2 και δίνετε μια γεύση σύγκρισης τους .

Στο κεφάλαιο 4 παρουσιάζεται η μεθοδολογία , τα βήματα που ακολουθήθηκαν για την επίτευξη των στόχων της διπλωματικής εργασίας καθώς και η αναλυτική περιγραφή των βημάτων αυτών που είναι ο προσδιορισμός του USECASE ,η υλοποίηση μιας video

streaming εφαρμογής , deployment , μοντελοποίηση , ο σχεδιασμός του πειράματος , εκτέλεση του πειράματος , αποτελέσματα και συμπεράσματα.

Στο κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα από την εκτέλεση των πειραμάτων και γίνονται κάποιες παρατηρήσεις.

Και τέλος στο Κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα και η μελλοντική εργασία που μπορεί να γίνει .

Κεφάλαιο 2

Background Section

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται αναφορά στο υπόβαθρο που βρίσκετε πίσω από την μελέτη αυτή και γίνετε μια παρουσίαση εννοιών όπως το τι είναι το cloud computing , τα χαρακτηριστικά του , οι τύποι των υπηρεσιών του και τύποι ανάπτυξης του .Τι είναι οι cloud εφαρμογές . Τι είναι παρακολούθηση (monitoring) , τι σημαίνει παρακολούθηση απόδοσης μιας εφαρμογής, παρακολούθηση στο cloud και κάποιες καλές πρακτικές που αφορούν την παρακολούθηση. Και στην έννοια της ελαστικότητας - επεκτασιμότητας (elasticity - Scalability) , τι είναι ; , τα είδη της επεκτασιμότητας – ελαστικότητας .

2.2 Cloud Computing

2.2.1 Τι είναι το Cloud?

Πολλοί άνθρωποι στις μέρες μας χρησιμοποιούν υπηρεσίες Cloud χωρίς να το συνειδητοποιούν καθώς βρίσκετε πίσω από :

- αποστολή email
- επεξεργασία εγγράφων
- παρακολούθηση ταινιών
- ακρόαση μουσικής
- αναπαραγωγή παιχνιδιών
- αποθήκευση εικόνων και άλλων αρχείων

και άλλα πολλά . Με ποιο απλά λόγια Cloud Computing σημαίνει αποθήκευση και πρόσβαση σε δεδομένα και προγράμματα μέσω του Internet αντί του σκληρού δίσκου του υπολογιστή .

2.2.2 Χαρακτηριστικά του Cloud Computing

- Κόστος (Cost)

  Το Cloud computing εξαλείφει το κόστος για την αγορά υλικού και λογισμικού, την συντήρηση αλλά και το ηλεκτρικό ρεύμα που χρειάζεται η υποδομή.
- Ταχύτητα (Speed)

  Αν χρειαστεί ανά πάσα χρονική στιγμή περισσότεροι υπολογιστική δύναμη - πόροι μπορούν να διατεθούν σε λίγα λεπτά, με μερικά κλικ , παρέχοντας μεγάλη ευελιξία .
- Παραγωγικότητα (Productivity)

  Το Cloud computing εξαλείφει πολλά καθήκοντα που απαιτεί ένα τοπικό Data Center ,έτσι ώστε οι ομάδες IT να μπορούν να αποσειόσουν χρόνο για την επίτευξη πιο σημαντικών στόχων.
- Απόδοση (Performance)

  Οι υπηρεσίες cloud computing λειτουργούν σε ένα παγκόσμιο δίκτυο ασφαλών κέντρων δεδομένων (data centers), τα οποία αναβαθμίζονται τακτικά στην τελευταία γενιά γρήγορου και αποδοτικού εξοπλισμού. Αυτό προσφέρει πολλά

πλεονεκτήματα όπως την μειωμένη καθυστέρηση δικτύου (Network Latency) , τεράστια υπολογιστική δύναμη κ.α.

- Αξιοπιστία (Reliability)

  Το Cloud computing καθιστά ευκολότερη και λιγότερο δαπανηρή την δημιουργία αντιγράφων ασφαλείας των δεδομένων, την αποκατάσταση δεδομένων σε περίπτωση καταστροφών , επειδή τα δεδομένα μπορούν να κλωνοποιούνται σε πολλαπλές τοποθεσίες στο δίκτυο του παροχέα υπηρεσιών cloud.

- Ελαστικότητα (Elasticity)

  Οι επιχειρήσεις μπορούν να μεγαλώσουν καθώς οι ανάγκες σε υπολογιστές αυξάνονται, άλλα και να μικρύνουν καθώς οι απαιτήσεις μειώνονται. Αυτό εξαλείφει την ανάγκη για μαζικές επενδύσεις σε τοπικές υποδομές, οι οποίες ενδέχεται να παραμείνουν ενεργές ή όχι.

2.2.3 Τύποι των υπηρεσιών Cloud

Οι υπηρεσίες Cloud χωρίζονται στις ποιο κάτω τρείς κατηγορίες .

1. Υποδομή ως υπηρεσία (IaaS - Infrastructure as a service)

   Με το IaaS , μπορεί κάποιος να ενοικιάσει υποδομές πληροφορικής , διακομιστές και εικονικές μηχανές (VM), αποθηκευτικούς χώρους, δίκτυα, λειτουργικά συστήματα - από έναν πάροχο υπηρεσιών σύννεφου .

   π.χ. Amazon EC2[1], Windows Azure[6], Google Compute Engine[7]

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

   Το PaaS αναφέρεται σε υπηρεσίες cloud computing που παρέχουν ένα περιβάλλον για την ανάπτυξη, τον έλεγχο, την παράδοση και τη διαχείριση εφαρμογών λογισμικού. Το PaaS έχει σχεδιαστεί για να διευκολύνει τους προγραμματιστές να δημιουργούν γρήγορα εφαρμογές , χωρίς να ανησυχούν για τη δημιουργία ή τη διαχείριση της υποδομής διακομιστών, χώρου αποθήκευσης, δικτύου και βάσεων δεδομένων που απαιτούνται για την ανάπτυξη.

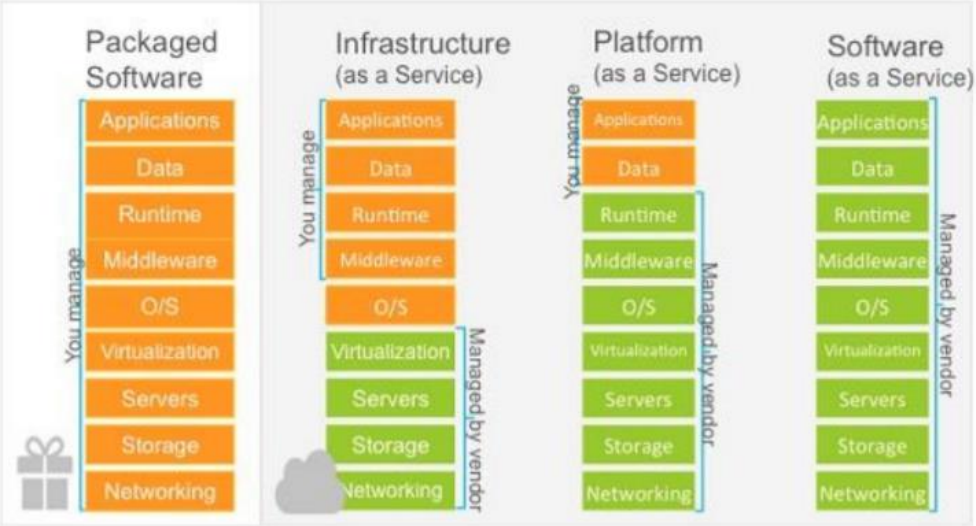
   π.χ Amazon Elastic Beanstalk[8], Cloud Services[9], Google App Engine[10]

3. Λογισμικό ως υπηρεσία (SaaS Software as a service)

   Το SaaS είναι μια μέθοδος για την παράδοση εφαρμογών λογισμικού μέσω του Διαδικτύου. Με το SaaS, οι πάροχοι σύννεφων φιλοξενούν και διαχειρίζονται μία εφαρμογή λογισμικού , την υποδομή και χειρίζονται οποιαδήποτε συντήρηση,

όπως αναβαθμίσεις λογισμικού και ενημερωμένες εκδόσεις ασφαλείας. Οι χρήστες συνδέονται με την εφαρμογή μέσω του Διαδικτύου.

π.χ. Google Apps, Microsoft Office 365, Dropbox[11]



Σχήμα 2. 1 Τύποι των υπηρεσιών Cloud

2.2.4 Τύποι ανάπτυξης του Cloud

Υπάρχουν τρεις διαφορετικοί τρόποι για την ανάπτυξη πόρων του Cloud Computing

Δημόσιο - Public Cloud

Τα Public Cloud λειτουργούν και ανήκουν σε έναν παροχέα υπηρεσιών Cloud, ο οποίος παρέχει τους υπολογιστικούς πόρους. Με ένα Public Cloud, όλο το υλικό, το λογισμικό και άλλη υποστηρικτική υποδομή ανήκει στον πάροχο του Cloud και την διαχειρίζεται αυτός. Μπορεί κάποιος να αποκτήσει πρόσβαση σε αυτές τις υπηρεσίες και να διαχειρίζεται το λογαριασμό του χρησιμοποιώντας ένα πρόγραμμα περιήγησης ιστού.

Ιδιωτικό - Private Cloud

Ένα Private Cloud αναφέρεται σε πόρους υπολογιστικού νέφους που χρησιμοποιούνται αποκλειστικά από μια επιχείρηση ή έναν οργανισμό. Ένα Private Cloud μπορεί να

βρίσκεται στο datacenter της επιχείρησης. Ορισμένες εταιρείες πληρώνουν επίσης τρίτους παρόχους υπηρεσιών για να φιλοξενήσουν το ιδιωτικό τους σύννεφο.

Υβριδικό - Hybrid Cloud

Τα Hybrid Cloud συνδυάζουν δημόσια και ιδιωτικά Cloud, που συνδέονται με τεχνολογία που επιτρέπει την κοινή χρήση δεδομένων και εφαρμογών μεταξύ τους. Επιτρέποντας στα δεδομένα και τις εφαρμογές να μετακινούνται μεταξύ ιδιωτικών και δημόσιων σύννεφων, το Hybrid Cloud παρέχει στις επιχειρήσεις μεγαλύτερη ευελιξία και περισσότερες δυνατότητες ανάπτυξης.

2.3 Cloud Εφαρμογές

Μια Cloud εφαρμογή (Cloud App) είναι ένα πρόγραμμα λογισμικού όπου έχουμε συνεργασία των cloud και τοπικών συστατικών . Το μοντέλο αυτό βασίζετε πάνω σε απομακρυσμένους διακομιστές που επεξεργάζονται την λογική της εφαρμογής όπου συνήθως είναι προσβάσιμη μέσω ενός προγράμματος περιήγησης ιστού. Τα δεδομένα είναι αποθηκεμένα και οι υπολογισμοί γίνονται σε απομακρυσμένα κέντρα δεδομένων (data centers)

2.4 Παρακολούθηση (Monitoring)

Παρακολούθηση είναι η διαδικασία όπου γίνετε συστηματική συλλογή, ανάλυση , αξιολόγηση και η χρήση πληροφοριών για την παρακολούθηση των λειτουργιών ενός προγράμματος , για την επίτευξη των στόχων του και για τη λήψη των αποφάσεων. Γενικά, η παρακολούθηση του λογισμικού παρακολουθεί και καταγράφει όλη την εισερχόμενη / εξερχόμενη κίνηση δικτύου, τις διεργασίες και τις αλληλεπιδράσεις των χρηστών και τις δραστηριότητες της εφαρμογής.

2.4.1 Παρακολούθηση απόδοσης εφαρμογής (Application Performance Monitoring)

Η παρακολούθηση απόδοσης εφαρμογών εστιάζει στην διασφάλιση ότι η εφαρμογή λειτουργεί όπως αναμένεται έτσι ώστε να παρέχεται στους τελικούς χρήστες μια ποιοτική εμπειρία. Τα εργαλεία παρακολούθησης εφαρμογών παρέχουν στους διαχειριστές τα

δεδομένα που χρειάζονται για να ανακαλύψουν γρήγορα, να απομονώσουν και να επιλύσουν προβλήματα που επηρεάζουν αρνητικά την απόδοση μιας εφαρμογής. Τέτοια εργαλεία μπορούν να συλλέγουν δεδομένα σχετικά με τη χρήση του CPU του πελάτη/διακομιστή , τις απαιτήσεις μνήμης , τη διακίνηση δεδομένων, το εύρος ζώνης, τον αριθμό των αιτήσεων και πόσο χρόνο παίρνουν κατά μέσο όρο να εξυπηρετηθούν και τα ποσοστά των σφαλμάτων της εφαρμογής.

2.4.2 Παρακολούθηση στο Cloud (Cloud Monitoring)

Το cloud monitoring είναι η διαδικασία αξιολόγησης, παρακολούθησης και διαχείρισης υπηρεσιών, εφαρμογών και υποδομών που βασίζονται στο Cloud. Η παρακολούθηση στο Cloud γίνετε μέσω εργαλείων όπου επιβλέπουν τους πόρους και τις εφαρμογές . Τα εργαλεία αυτά συνήθως χωρίζονται σε δύο κατηγορίες: (1) Τα εργαλεία που παρέχονται από τον πάροχο του Cloud τα οποία δεν θέλουν κάποια εγκατάσταση, είναι μέρος της υπηρεσίας και (2) Τα εργαλεία που παρέχονται από άλλο παροχέα διαφορετικό από αυτόν του Cloud. Αυτά τα εργαλεία προσφέρουν δεδομένα σχετικά με την απόδοση ,την ασφάλεια και την συμπεριφορά των πελατών έτσι ώστε μέσα από αυτά να είναι εφικτή η γρήγορη ανίχνευση σφαλμάτων , η επίλυση τους ,η βελτίωση της απόδοσης ,της ταχύτητας και γενικά η εμπειρία των χρηστών και της ικανοποίησης τους .

2.4.3 Καλές πρακτικές που αφορούν την παρακολούθηση

- Πρέπει να γίνει προσδιορισμός για το τί πρέπει να παρακολουθείτε και ποιες μετρικές είναι σημαντικές .
- Η Συλλογή των δεδομένων να γίνετε σε μια κεντρική πλατφόρμα έτσι ώστε μέσω των δεδομένων από τις πολλές πηγές να γίνετε ένας ομοιόμορφος υπολογισμός αποτελεσμάτων που να καταλήγει σε μια συνολική εικόνα.
- Πρέπει να γίνετε διαχωρισμός των δεδομένων ,τα δεδομένα παρακολούθησης να μπαίνουν ξεχωριστά από τις εφαρμογές και τις υπηρεσίες για εύκολη πρόσβαση από τους άμεσα ενδιαφερόμενους.
- Μια εφαρμογή είναι καλό να μετριέται στα “άκρα” να πάρουμε δεδομένα για τις ακραίες περιπτώσεις της εφαρμογής και να παρακολουθήσουμε τι γίνετε και

μέσω των δεδομένων παρακολούθησης να μπορέσουμε να ερμηνεύσουμε την συμπεριφορά που παρατηρήσαμε .

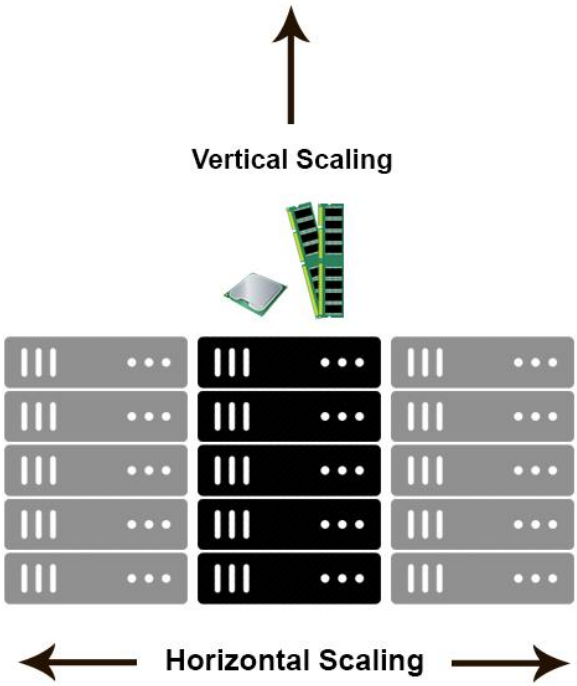
2.5 Επεκτασιμότητα (Scalability)/Ελαστικότητα (Elasticity)

2.5.1 Τι είναι η επεκτασιμότητα (scalability)

Η δυνατότητα επεκτασιμότητας είναι ένα χαρακτηριστικό που περιγράφει την ικανότητα μιας διαδικασίας, λογισμικού ή συστήματος να αναπτυχθεί και να διαχειριστεί την αυξημένη ζήτηση / ποσότητα εργασίας, χωρίς να επηρεάζεται η απόδοση. Η επεκτασιμότητα είναι ένα επιθυμητό χαρακτηριστικό καθώς η κακή κλιμάκωση μπορεί να οδηγήσει σε κακή απόδοση του συστήματος, Για παράδειγμα, ένα σύστημα θεωρείται κλιμακωτό / επεκτάσιμο εάν είναι σε θέση να αυξήσει τη συνολική του απόδοση υπό ένα αυξημένο φορτίο όταν προστίθενται πόροι. Η επεκτασιμότητα είναι συχνά ένα σημάδι σταθερότητας και ανταγωνιστικότητας, καθώς σημαίνει ότι η διαδικασία, το λογισμικό ή το συστήματος είναι έτοιμο να χειριστεί την αυξημένη ζήτηση, την αυξημένη παραγωγικότητα, τις τάσεις, και τις μεταβαλλόμενες ανάγκες.

2.5.2 Κάθετη επεκτασιμότητα (Vertical scaling) / Οριζόντια επεκτασιμότητα (Horizontal scaling)

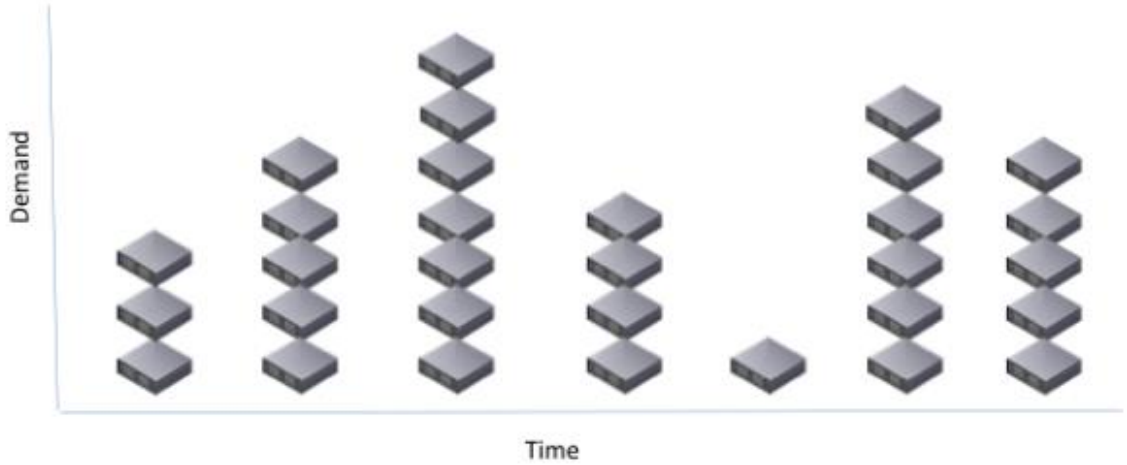
Όταν μιλάμε για επεκτασιμότητα στο cloud computing, ακούμε συχνά δύο βασικούς τρόπους επεκτασιμότητας την οριζόντια και την κάθετη. Όταν κλιμακώνετε ένα σύστημα κάθετα, προσθέτετε περισσότερη ισχύ σε μια υπάρχον instance . Αυτό μπορεί να σημαίνει περισσότερη μνήμη (RAM), ταχύτερη αποθήκευση, όπως μονάδες Solid State Drives (SSD) ή ισχυρότερους επεξεργαστές (CPU). Όταν κλιμακώνετε ένα σύστημα οριζόντια, ουσιαστικά προσθέτουμε περισσότερους διακομιστές για να μεταδίδετε το φορτίο σε πολλαπλές μηχανές.



Σχήμα 2.2 Κάθετη / Οριζόντια επεκτασιμότητα

2.5.3 Τι είναι η ελαστικότητα (elasticity)

Η ελαστικότητα[47] στο Cloud είναι η ικανότητα ανάπτυξης ή συρρίκνωσης των πόρων της υποδομής δυναμικά με αυτόματο τρόπο, ανάλογα με τις ανάγκες, έτσι ώστε να προσαρμόζεται στις αλλαγές του φόρτου εργασίας με αυτόνομο τρόπο μεγιστοποιώντας τη χρήση των πόρων. Αυτό μπορεί να έχει ως αποτέλεσμα εξοικονόμηση κόστους υποδομής συνολικά. Η ελαστικότητα του Cloud είναι ένα χαρακτηριστικό που σχετίζεται με λύσεις οριζόντιας επεκτασιμότητας που επιτρέπουν τη δυναμική προσθήκη ή αφαίρεση πόρων όταν απαιτείται.



Σχήμα 2.3 Ελαστικότητα στο Cloud

Κεφάλαιο 3

Related Work and State of the art

3.1 Εισαγωγή

Πριν προχωρήσουμε είναι σημαντικό να δούμε την state-of-the art αναφορά σχετικά με τις βασικές τεχνολογίες που σχετίζονται με την διπλωματική εργασία αυτή. Σε σχέση με το Κεφάλαιο2 «Background Section» , αυτή η ενότητα παρέχει οδηγό αναφοράς για συγκεκριμένες τεχνολογίες που χρησιμοποιούνται σε αντίστοιχες περιπτώσεις.

3.2 Cloud Computing

Σε αυτή την ενότητα, περιγράφονται ορισμένα από τα προϊόντα υπολογιστικού νέφους (cloud computing).

3.2.1 Amazon EC2

Το Amazon Web Services (AWS)[12] είναι ένα σύνολο υπηρεσιών cloud που παρέχουν λειτουργίες που επιτρέπουν σε οργανισμούς και ιδιώτες να αναπτύξουν εφαρμογές και υπηρεσίες στο cloud . Οι υπηρεσίες του AWS είναι προσβάσιμες μέσω HTTP, χρησιμοποιώντας τα πρωτόκολλα REST και SOAP. Το Amazon Elastic Compute Cloud (Amazon EC2) επιτρέπει στους χρήστες του να εκκινήσουν και να διαχειριστούν server

instances σε κέντρα δεδομένων χρησιμοποιώντας API ή διαθέσιμα εργαλεία και βοηθητικά προγράμματα. Μετά τη δημιουργία και την εκκίνηση ενός instance, οι χρήστες μπορούν να μεταφορτώσουν λογισμικό και να πραγματοποιήσουν αλλαγές σε αυτό. Ένα ανάλογο αντίγραφο μπορεί στη συνέχεια να ξεκινήσει ανά πάσα στιγμή. Οι χρήστες έχουν σχεδόν πλήρη έλεγχο ολόκληρης της στοίβας λογισμικού στα EC2 instances που μοιάζουν με hardware για αυτούς. Το EC2 παρέχει τη δυνατότητα τοποθέτησης των instances σε πολλαπλές τοποθεσίες. Οι εικόνες μηχανής EC2 (EC2 machine images) αποθηκεύονται και ανακτώνται από την υπηρεσία απλής αποθήκευσης Amazon (Amazon Simple Storage Service -Amazon S3[13]). Το Virtual Private Cloud (VPC)[14] του Amazon είναι μια ασφαλής γέφυρα ανάμεσα στην ήδη υπάρχουσα υποδομή πληροφορικής της επιχείρησης και το AWS cloud. Το Amazon VPC επιτρέπει στις επιχειρήσεις να συνδέουν την υπάρχουσα υποδομή τους με ένα σύνολο απομονωμένων υπολογιστικών πόρων AWS μέσω σύνδεσης Virtual Private Network (VPN) και να επεκτείνουν τις υπάρχουσες δυνατότητες διαχείρισης τους, όπως υπηρεσίες ασφαλείας, τείχη προστασίας (Firewalls) και συστήματα ανίχνευσης εισβολών Το Amazon CloudWatch[15] είναι ένα χρήσιμο εργαλείο διαχείρισης που συλλέγει δεδομένα από συνεργαζόμενες υπηρεσίες AWS όπως το Amazon EC2 και στη συνέχεια επεξεργάζεται τις πληροφορίες σε μετρήσεις σχεδόν σε πραγματικό χρόνο. Οι μετρήσεις για το EC2 περιλαμβάνουν, για παράδειγμα, τη χρήση της CPU, Network I/O (bytes), λειτουργίες Disks I/O (bytes), κ.λπ.

3.2.2 Microsoft Azure platform

Η πλατφόρμα Azure της Microsoft αποτελείται από τρία συστατικά στοιχεία και κάθε ένα από αυτά παρέχει ένα συγκεκριμένο σύνολο υπηρεσιών για τους χρήστες cloud. Το microsoft Azure παρέχει ένα περιβάλλον βασισμένο σε Windows[16] για την εκτέλεση εφαρμογών και την αποθήκευση δεδομένων σε διακομιστές σε κέντρα δεδομένων. Το SQL Azure[17] παρέχει υπηρεσίες δεδομένων στο cloud βασισμένες στον SQL Server. Οι υπηρεσίες .NET (.NET Services) προσφέρουν υπηρεσίες κατανεμημένης υποδομής (distributed infrastructure services) σε εφαρμογές που βασίζονται στο cloud αλλά και τοπικές εφαρμογές. Η πλατφόρμα Windows Azure μπορεί να χρησιμοποιηθεί τόσο από εφαρμογές που εκτελούνται στο cloud όσο και από εφαρμογές που εκτελούνται τοπικά. Το Windows Azure υποστηρίζει επίσης εφαρμογές που βασίζονται στο .NET

Framework[19] και άλλες συνήθεις γλώσσες που υποστηρίζονται σε συστήματα Windows, όπως C #[20], Visual Basic[21], C ++[22] και άλλες. Οι προγραμματιστές μπορούν να δημιουργούν εφαρμογές ιστού χρησιμοποιώντας τεχνολογίες όπως το ASP.NET[18]. Τα αποθηκεμένα δεδομένα στο Windows Azure είναι προσπελάσιμα μέσω RESTful (HTTP ή HTTPS) . Η SQL Azure βάση δεδομένων βασίζεται στον Microsoft SQL Server, παρέχοντας ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) στο cloud. Οι υπηρεσίες .NET διευκολύνουν τη δημιουργία κατανεμημένων εφαρμογών. Το service bus βοηθά μια εφαρμογή να ορίσει endpoints που μπορούν να προσεγγιστούν από άλλες εφαρμογές. Για κάθε endpoint υπάρχει ένα URI για να μπορούν οι πλάτες να έχουν πρόσβαση και να χρησιμοποιήσουν μια υπηρεσία. Όλοι οι φυσικοί πόροι, τα VM και οι εφαρμογές στο κέντρο δεδομένων παρακολουθούνται από λογισμικό που ονομάζεται fabric controller. Με κάθε εφαρμογή, οι χρήστες ανεβάζουν ένα αρχείο ρυθμίσεων που παρέχει μια περιγραφή σε XML μορφή που περιγράφει τις ανάγκες της εφαρμογής. Με βάση αυτό το αρχείο, ο fabric controller αποφασίζει πού θα πρέπει να τρέχουν οι νέες εφαρμογές, επιλέγοντας φυσικούς διακομιστές για να βελτιστοποιήσει τη χρήση του hardware.

3.2.3 Google Cloud Platform (GCP)

Με το Google Cloud Platform[23] μπορεί κάποιος να δημιουργήσει , να δοκιμάσει και να αναπτύξει εφαρμογές στην υψηλά επεκτάσιμη και αξιόπιστη υποδομή της Google για λύσεις που αφορούν το web , mobile ,και το backend . Το GCP αποτελείται από ένα σύνολο φυσικών πόρων, όπως υπολογιστές και μονάδες σκληρού δίσκου αλλά και εικονικούς πόρους, όπως οι εικονικές μηχανές (VM), που περιέχονται στα κέντρα δεδομένων της Google σε όλο τον κόσμο. Η κατανομή των πόρων του GCP παρέχει πολλά οφέλη, συμπεριλαμβανομένης της απόλυσης σε περίπτωση αποτυχίας και μειωμένης καθυστέρησης, εντοπίζοντας πόρους πιο κοντά στους πελάτες. Ο κατάλογος των διαθέσιμων υπηρεσιών και εργαλείων στο GCP είναι μακρύς και συνεχίζει να αυξάνεται. Όταν κάποιος αναπτύσσει έναν ιστότοπο ή εφαρμογή στο GCP, βρίσκει και χρησιμοποιεί αυτές τις υπηρεσίες σε συνδυασμούς που του παρέχουν την υποδομή που χρειάζεται και στη συνέχεια προσθέτει τον κώδικα για να ενεργοποιήσει τα σενάρια που θέλει να δημιουργήσει. Οποιοσδήποτε πόρος του GCP πρέπει να ανήκει σε ένα project. Ένα project αποτελείται από τις ρυθμίσεις, τα δικαιώματα και άλλα μεταδεδομένα που

περιγράφουν τις εφαρμογές. Όταν είναι ενεργοποιημένη η χρέωση, κάθε έργο συνδέεται με έναν λογαριασμό χρέωσης. Πολλά έργα μπορούν να χρεώνουν τη χρήση των πόρων στον ίδιο λογαριασμό. Το GCP παρέχει τρεις βασικούς τρόπους αλληλεπίδρασης με τις υπηρεσίες και τους πόρους. Η Κονσόλα πλατφόρμας Google Cloud παρέχει μια διαδικτυακή γραφική διεπαφή χρήστη που μπορεί κάποιος να την χρησιμοποιήσει για τη διαχείριση των έργων και πόρων του GCP. Εάν κάποιος προτιμά να εργάζεται σε παράθυρο τερματικού (terminal), το Google Cloud SDK παρέχει το gcloud command-line tool που θα του δώσει πρόσβαση στις εντολές που χειριάζετε . Το Cloud SDK περιλαμβάνει βιβλιοθήκες πελατών (client libraries) που σας επιτρέπουν την δημιουργία και την εύκολη διαχείριση πόρων. Η πλατφόρμα Google Cloud παρέχει επιπλέον εργαλεία , υπηρεσίες και υποστήριξη , για να βοηθήσει τον χρήστη να αναπτύξει πιο εύκολα μια εφαρμογή.

3.2.4 Σύνοψη χαρακτηριστικών

	Google Cloud	amazon web services	Microsoft Azure
Cloud Monitoring	Stackdriver	CloudWatch	Monitor
Load balancing configuration	Cloud Load Balancing	Elastic Load Balancing	Load Balancer Application Gateway
Automatically scale instances	Instance Groups	Auto Scaling	Virtual Machine Scale Sets App Service Scale Capability (PAAS) AutoScaling
Deploy, manage, and maintain virtual servers	Compute Engine	Elastic Compute Cloud (EC2)	Virtual Machines
Platform-as-a-Service	Google App Engine	Elastic Beanstalk	Cloud Services
Object storage service for use cases	Google Cloud Storage	Simple Storage Services (S3)	Storage(Block Blob)

Automatic protection and disaster recovery	N/A	Disaster Recovery	Site Recovery
Relational/SQL Database	Cloud SQL Cloud Spanner	RDS	-SQL Database - Database for MySQL -Database for PostgreSQL

Πίνακας 3. 1 Σύνοψη χαρακτηριστικών προϊόντων υπολογιστικού νέφους

3.3 Παρακολούθηση (Monitoring)

Τα εργαλεία παρακολούθησης γενικού σκοπού, όπως το Nagios[24] , παραδοσιακά χρησιμοποιούνται από διαχειριστές συστημάτων για την παρακολούθηση σταθερών κατανεμημένων υποδομών, όπως δίκτυα υπολογιστών και συμπλέγματα διακομιστών. Οι χρήστες Cloud τείνουν να υιοθετούν τέτοιες λύσεις για την παρακολούθηση των εφαρμογών τους. Όμως επειδή στα πλαίσια της αυτής της μελέτης χρειαζόμασταν περισσότερη ευελιξία και παραμετροποίηση στον τρόπο συλλογής των μετρικών , και των παραγόμενών γραφικών παρατάσεων έτσι η προαναφερθείσα κατηγορία εργαλείων ήταν ακατάλληλα για την συγκεκριμένη περίπτωση. Υπάρχουν και τα εργαλεία που παρέχονται από τον πάροχο του Cloud τα οποία δεν θέλουν κάποια εγκατάσταση, είναι μέρος της υπηρεσίας, οπός το Amazon CloudWatch και το Stackdriver του Google Cloud Compute Engine, Παρά το γεγονός ότι αυτά τα εργαλεία είναι εύχρηστα, το μεγαλύτερο μειονέκτημα τους είναι ότι περιορίζουν τη λειτουργία τους σε συγκεκριμένους παρόχους Cloud και είναι περιοριστικά στο χρόνο συλλογής των μετρικών αλλά και στο είδος των μετρικών. Τέλος υπάρχει, και το JCatascopia [26] το οποίο είναι ανοιχτού κώδικα πολυεπίπεδο , διαλειτουργικό σύστημα παρακολούθησης Cloud.

3.4 Ελαστικότητα (Elasticity)

Η ελαστικότητα είναι ένα βασικό σημείο που αφορά την ποιότητα σε υπηρεσίες Cloud . Χρησιμοποιείται για να αποφευχθεί η ανεπαρκής παροχή πόρων και η υποβάθμιση της απόδοσης του συστήματος, επιτυγχάνοντας ταυτόχρονα μείωση του κόστους . Υπάρχουν οι μηχανισμοί που αφορούν την ελαστικότητα στο Cloud που παρέχονται από τους παρόχους υποδομών Cloud όπως το Google Cloud Platform autoscaler , το Microsoft Azure Autoscale και το Amazon Auto Scaling Service και συνήθως χρησιμοποιούν κανόνες που σχετίζονται με τους πόρους της υποδομής για παράδειγμα αν η χρήση του κεντρικού επεξεργαστή μίας εικονικής μηχανής στο cloud πάει πάνω από 80% τότε προστίθεται ακόμα μία εικονική μηχανή .

Κεφάλαιο 4

Μεθοδολογία

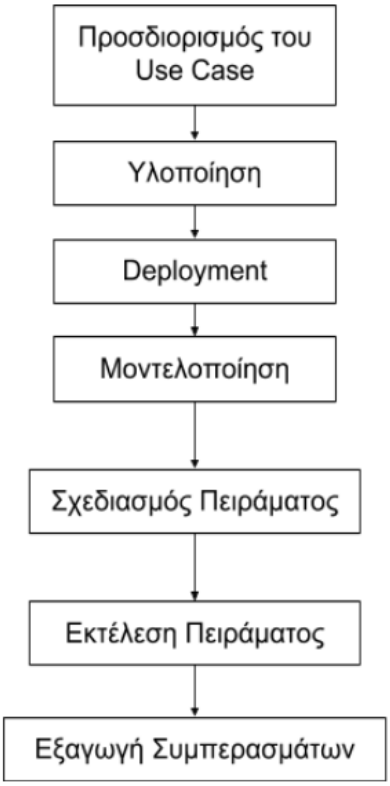
4.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται η μεθοδολογία , τα βήματα που ακολουθήθηκαν για την επίτευξη των στόχων της διπλωματικής εργασίας καθώς και η αναλυτική περιγραφή των βημάτων αυτών που είναι ο προσδιορισμός του USECASE ,η υλοποίηση μιας video streaming εφαρμογής για την οποία περιγράφετε ο τρόπος υλοποίησης της εφαρμογής αυτής οι τεχνολογίες που χρησιμοποιηθήκαν και γίνετε μια παρουσίαση της. Στην συνέχεια περιγράφεται ο τρόπος με τον οποίο έγινε deploy η εφαρμογή στο cloud . Επιπρόσθετα περιγράφονται οι μετρικές που θα συλλέγονται κατά την διάρκεια της εκτέλεσης των πειραμάτων αλλά και η μοντελοποίηση της video streaming εφαρμογής. Και τέλος ο σχεδιασμός του πειράματος από τον οποίο προέκυψε ένα πειραματικό framework με το οποίο μπορούν να εκτελεστούν τα πειράματα για τέτοιου είδους εφαρμογές

4.2 Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε αρχίζει με το προσδιορισμό του Use Case , την υλοποίηση – εφαρμογή video streaming, στην συνέχεια το deployment της εφαρμογής αυτής στο cloud , ακολουθεί η φάση της μοντελοποίησης, μετά γίνετε ο σχεδιασμός του πειράματος και τέλος η εκτέλεση του πειράματος και η εξαγωγή των συμπερασμάτων. Όσο αφορά το Use Case του πειράματος έπρεπε να χρησιμοποιηθεί μια εφαρμογή η οποία θα γινόταν deploy στο cloud η οποία ήταν επιθυμητό να έχει μεγάλες απαιτήσεις σε υπολογιστικούς πόρους κάτω από συνθήκες δοκιμών με αυξημένη ζήτηση και ποσότητα εργασίας. Έχοντας στο μυαλό ένα από τα αρχικά κίνητρα που ήταν οι Video Streaming εφαρμογές πάρθηκε η απόφαση να χρησιμοποιήσουμε στα πλαίσια του πειράματος μια video streaming εφαρμογή. Έγινε μια μελέτη για το τι λογισμικό υπάρχει είδη διαθέσιμο εντοπίστηκαν κάποιες έτοιμες εφαρμογές όπως το Streama[27] αλλά λόγω του γεγονότος ότι τα έτοιμα λογισμικά αποτελούνται από χιλιάδες γραμμές κώδικα, είναι πολύπλοκα , έχουν πολλές λειτουργίες που δεν χρειάζονταν για το πείραμα αυτό και ήταν απαραίτητο ο κώδικας τους να ήταν πλήρης κατανοητός για το τι ακριβώς κάνει και πώς το κάνει, για να έχουμε στο τέλος όσο ποιο σωστά αποτελέσματα γίνετε, η λύση του να χρησιμοποιηθεί κάποιο έτοιμο λογισμικό για Use Case του πειράματος απορρίφθηκε. Τι

πιο καλό για να μελετήσει κάποιος μια video streaming εφαρμογή από το να αναπτύξει μια , έτσι πάρθηκε η απόφαση να γίνει η ανάπτυξη μιας video streaming εφαρμογής όπου ο κώδικας , οι λειτουργίες της και ο τρόπος λειτουργίας της θα ήταν γνωστά και πλήρης κατανοητά. Έτσι περνούμε στην φάση της υλοποίησης και αφού έγινε η ανάπτυξη της εφαρμογής ακολουθεί το deployment της σε μια πλατφόρμα cloud provider έτσι ώστε να γίνει μια αξιολόγηση και κάποιες απαραίτητες παρατηρήσεις που αφορούν την συμπεριφορά της εφαρμογής και την αλληλεπίδραση του χρήστη με την εφαρμογή , που θα χρησιμοποιηθούν στην συνέχεια για την μοντελοποίηση της εφαρμογής και στο σχεδιασμό του πειράματος. Στην φάση της μοντελοποίησης με βάση τις παρατηρήσεις που έχουν προκύψει από την φάση του deployment γίνετε μια καταγραφή των μετρικών που θα λαμβάνονται υπόψη στη φάση της Σχεδίασης του πειράματος αλλά και θα συλλέγονται κατά την εκτέλεση του πειράματος . Επίσης σε αυτή την φάση γίνεται μοντελοποίηση της video streaming εφαρμογής που αναπτύχθηκε για να είναι εφικτή η συλλογή των μετρικών κατά την εκτέλεση του πειράματος , στην ουσία προκύπτει μια νέα εφαρμογή η οποία μοντελοποιεί την εφαρμογή video streaming που αναπτύχθηκε αρχικά .Στην συνέχεια έρχεται η φάση του σχεδιασμού του πειράματος όπου εδώ δημιουργείτε ένα πειραματικό Framework το οποίο αποτελείτε από τρία συστατικά στοιχεία , την εφαρμογή η οποία προέκυψε από την μοντελοποίηση της video streaming εφαρμογής , έναν workload generator ο οποίος είναι απαραίτητος για να γίνει εφικτή η προσομοίωση των χρηστών της εφαρμογής και της ποσότητας εργασίας σε πραγματικά σενάρια, και ένα εργαλείο παρακολούθησης (monitoring tool) το οποίο αναπτύχθηκε για την συλλογή των μετρικών , την αναπαράσταση τους , και την αποθήκευση τους. Εφόσον πλέων υπάρχει το Framework γίνεται υπόθεση κάποιον σεναρίων που αντιστοιχούν σε πραγματικά σενάρια η εκτέλεση τους και παράλληλα γίνετε η συλλογή των μετρικών που θα μας βοηθήσουν στην μελέτη και στην εξαγωγή των Συμπερασμάτων



Σχήμα 4.1 Ροή της μεθοδολογίας

4.3 Υλοποίηση

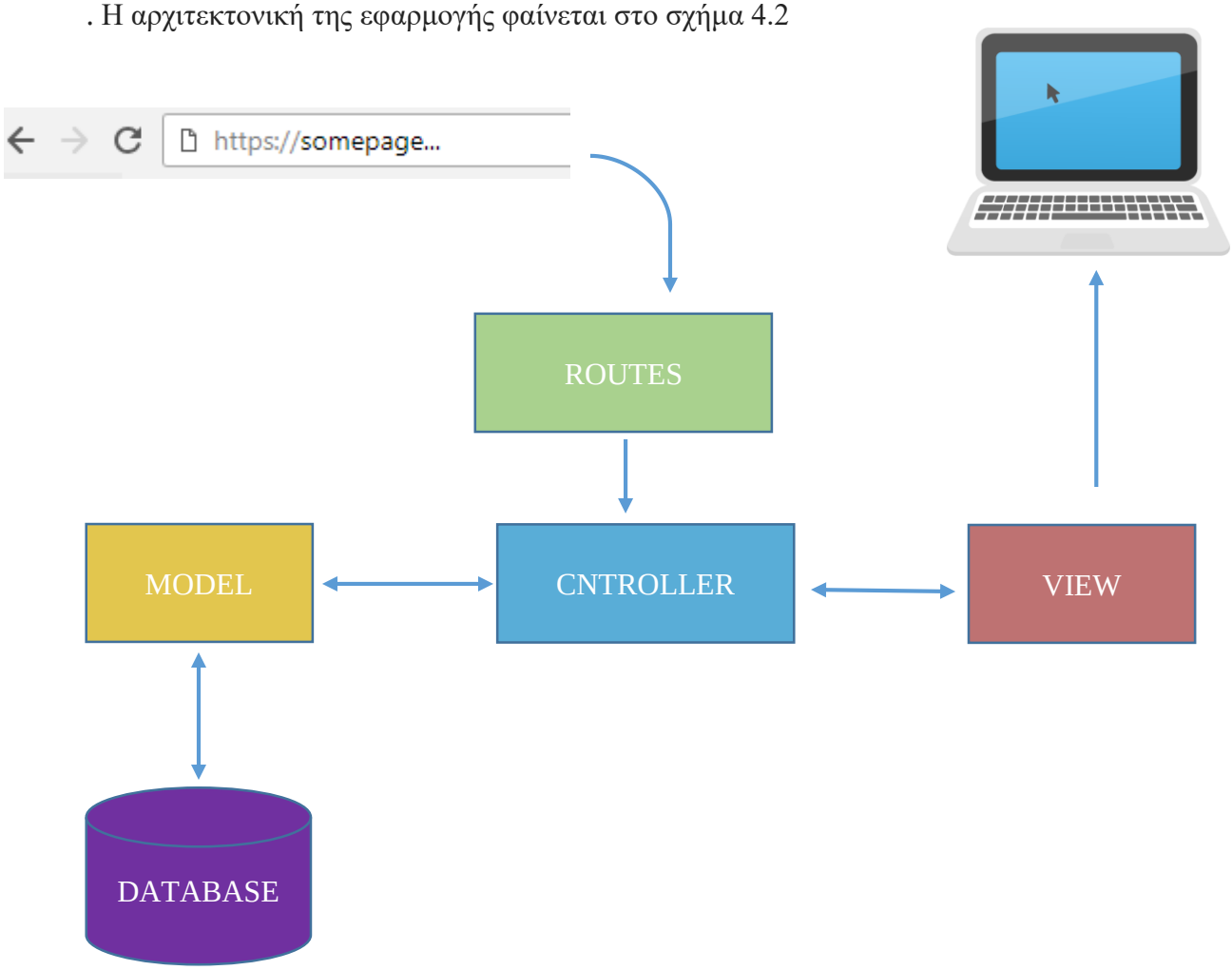
4.3.1 Σχεδιασμός Εφαρμογής

Για την ανάπτυξη της video streaming εφαρμογής χρειάζεται ένας εξυπηρετητής όπου θα είναι αποθηκευμένα τα βίντεο των χρηστών καθώς και μια βάση Δεδομένων όπου θα κρατά πληροφορίες σχετικά με τους χρήστες και τα βίντεο της εφαρμογής για να μπορούν να υλοποιηθούν οι ποιο κάτω λειτουργίες :

- Δυνατότητα εγγραφής (register) χρηστών στο σύστημα.
- Δυνατότητα εισόδου (login) χρηστών στο σύστημα.
- Δυνατότητα εξόδου (logout) χρηστών στο σύστημα.
- Δυνατότητα ενός χρήστη να μπορεί να μεταφορτώσει (Upload) τα δικά του βίντεο στο σύστημα.
- Δυνατότητα ενός χρήστη να μπορεί να δει μια λίστα με τα δικά του βίντεο.

- Δυνατότητα Αναπαραγωγή ενός βίντεο από την λίστα με τα βίντεο του χρήστη.
- Δυνατότητα διαγραφής ενός βίντεο από την λίστα με τα βίντεο του χρήστη.
- Εμφάνιση κατάλληλων ενημερωτικών μηνυμάτων και μνήματα λάθους στο χρήστη όταν χρειάζεται.

Όσο αφορά την αρχιτεκτονική της εφαρμογής θα ακολουθηθεί σχεδιαστικό μοτίβο MVC (Model–View-Controller) στο οποίο έχουμε τους Ελεγκτές - controllers οι οποίοι χειρίζονται τα αιτήματα των χρηστών και ανακτούν τα δεδομένα αξιοποιώντας τα μοντέλα (Models). Τα Μοντέλα (Models) αλληλοεπιδρούν με τη βάση δεδομένων και ανακτούν πληροφορίες σχετικά με τα αντικείμενα της εφαρμογής. Τα Views χρησιμοποιούνται για την προβολή πληροφοριών στο χρήστη και την απόδοση των σελίδων . Επιπλέον έχουμε τις διαδρομές (routes) οι οποίες χρησιμοποιούνται για την χαρτογράφηση διευθύνσεων URL σε καθορισμένες ενέργειες ενός ελεγκτή (controller) . Η αρχιτεκτονική της εφαρμογής φαίνεται στο σχήμα 4.2



Σχήμα 4.2 Αρχιτεκτονική της εφαρμογής.

Έτσι όταν δημιουργείται ένα αίτημα (δηλαδή ο χρήστης εισάγει μια διεύθυνση URL που σχετίζεται με την εφαρμογή) υπάρχει μία διαδρομή που σχετίζεται με την συγκεκριμένη διεύθυνση URL και στην συνέχεια αντιστοιχεί την διεύθυνση σε μια ενέργεια ελεγκτή . Αυτή η ενέργεια του ελεγκτή χρησιμοποιεί τα απαραίτητα μοντέλα για την ανάκτηση πληροφοριών από τη βάση δεδομένων και στη συνέχεια διαβιβάζει αυτά τα δεδομένα σε ένα view.

4.3.2 Υλοποίηση εφαρμογής

4.3.2.1 Τεχνολογίες που χρησιμοποιήθηκαν

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν κάποιες τεχνολογίες που αφορούν την λογική της εφαρμογής , την γραφική διεπαφή του χρήστη την βάση δεδομένων αλλά και το κομμάτι του video streaming.

- Για την λογική και την υλοποίηση της αρχιτεκτονικής της εφαρμογής η οποία περιγράφετε στην ποιο πάνω στην υποενότητα (4.3.1) χρησιμοποιήθηκε το LARAVEL PHP FRAMEWORK[28], το οποίο είναι ένα μοντέρνο πανίσχυρο εργαλείο το οποίο χρησιμοποίει το αρχιτεκτονικό μοτίβο MVC , σκοπεύει να μειώσει τον χρόνο ανάπτυξης μιας εφαρμογής και να βελτιώσει την ποιότητα του κώδικα. Γιατί PHP[29]; Η PHP είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού με την πρόσφατη έκδοση του PHP 7 έκανε την γλώσσα καλύτερη και πιο σταθερή από ποτέ.

  “Η PHP χρησιμοποιείται από το 82,7% όλων των ιστότοπων που γνωρίζουμε των οποίων η γλώσσα προγραμματισμού από την πλευρά του διακομιστή.”

  ~ W3techs.com
- Όσο αφορά την διεπαφή του χρήστη (Interface) και τα views της εφαρμογής χρησιμοποιήθηκαν οι “αυτονόητες” τεχνολογίες:
  - HTML 5 [30]
  - CSS 3 [31]
  - JAVA SCRIPT [32]

Και επίσης το Front End Framework BOOSTRAP[33] για ταχύτερη και ευκολότερη ανάπτυξη web Development το οποίο χρησιμοποιείτε σαν προεπιλογή από το LARAVEL Framework.

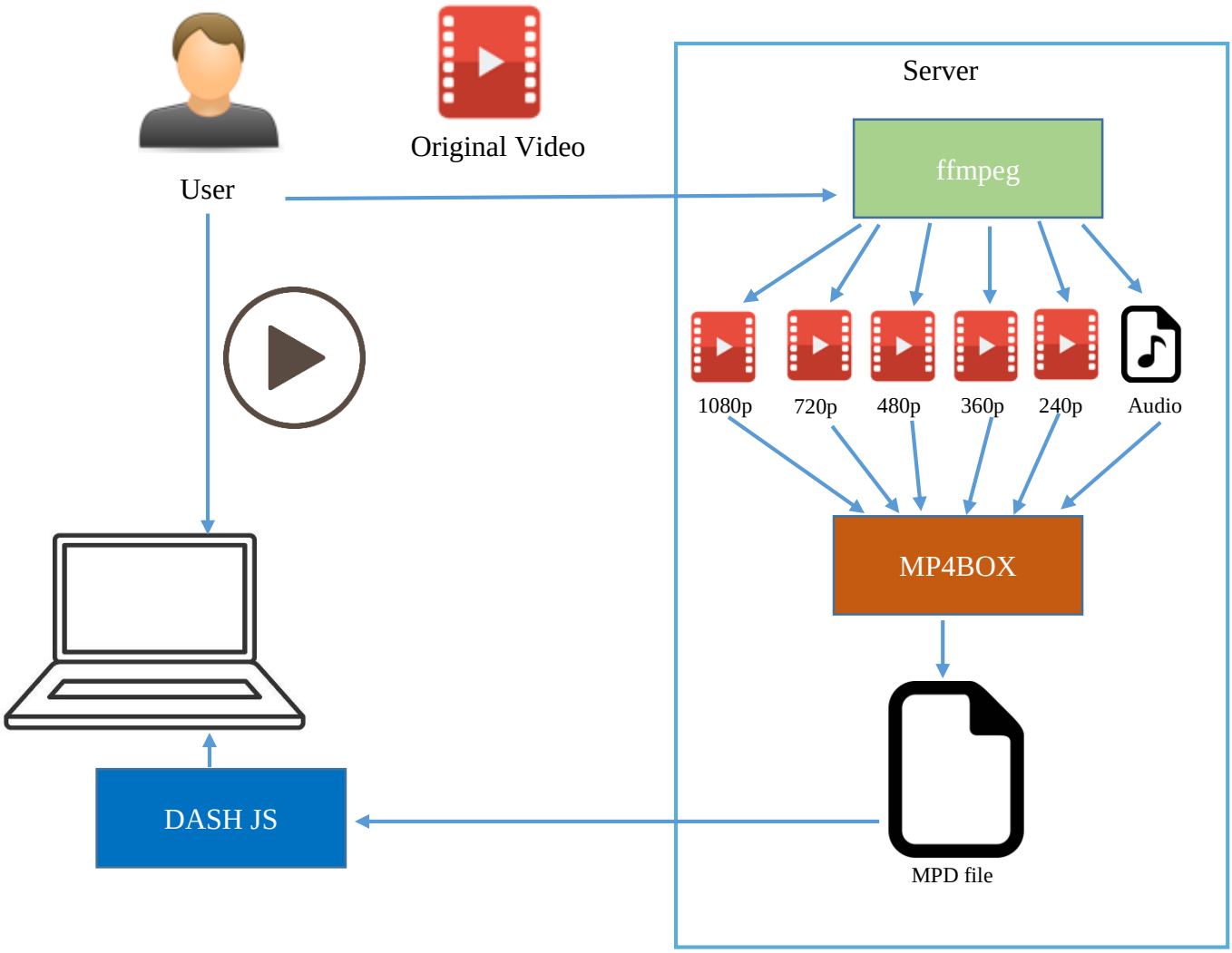
- Όσο αφορά την βάση δεομένων της εφαρμογής χρησιμοποιήθηκε η MYSQL[34].

Για την υλοποίηση του video streaming αρχικά έγινε μια υλοποίηση σε PHP η οποία η οποία αναλάβανε το streaming ενός video. Σε αυτή την υλοποίηση γινόταν η δημιουργία ενός HTTP request και μιας απάντησης HTTP response όπου το περιεχόμενο δηλαδή το βίντεο στελνόταν σιγά σιγά δηλαδή ξεκινούσε η αναπαραγωγή του βίντεο κανονικά από την αρχή που το επέλεγε ο χρήστης και κατά την διάρκεια της αναπαραγωγής φορτώνονταν και το υπόλοιπο. Αξιολογώντας την λύση αυτή κρίθηκε ως ακατάλληλη για την περίπτωση μας διότι με αυτήν την υλοποίηση δεν Θα ήταν δυνατή η συλλογή όλων των επιθυμητών μετρικών , και λόγο του αρχικού στόχου τα αποτελέσματα της έρευνας αυτής να είναι όσο το δυνατών ποιο σωστά γίνετε αντικατοπτρίζοντας πραγματικά σενάρια πάρθηκε η απόφαση το κομμάτι του Video streaming να γίνει με πιο σωστό τρόπο , με τρόπο που χρησιμοποιείτε από λογισμικά video streaming στις μέρες μας δηλαδή το βίντεο να στέλνεται σε πακέτα όπου για κάθε πακέτο να αντιστοιχεί και σε ένα http request / response .Για την υλοποίηση αυτή χρησιμοποιήθηκαν οι τεχνολογίες που ακολουθούν.

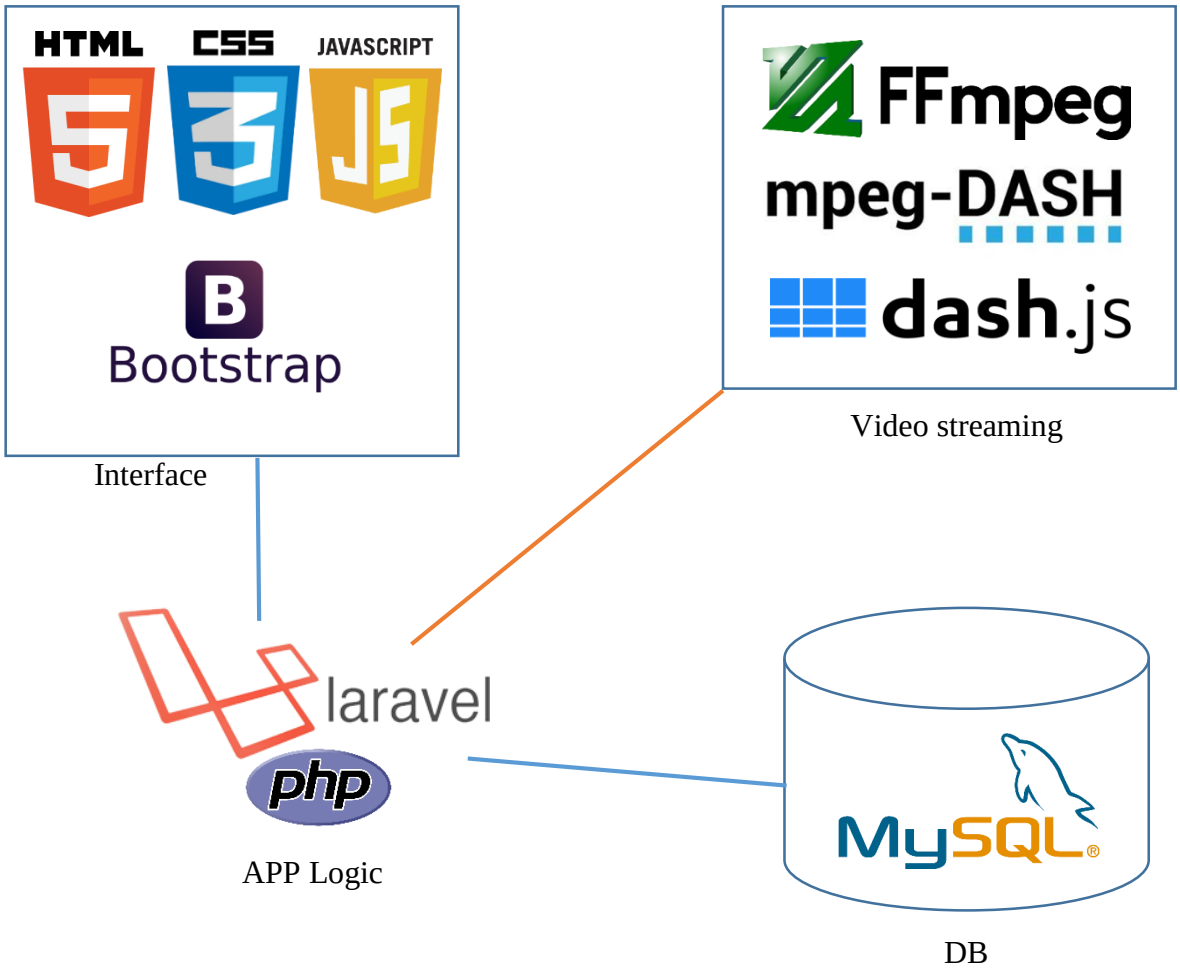
- Dynamic Adaptive Streaming over HTTP (DASH), γνωστό και ως MPEG-DASH[35] είναι μια προσαρμοστική τεχνολογία ροής bitrate (adaptive bitrate streaming) όπου ένα αρχείο πολυμέσων είναι χωρισμένο σε ένα ή περισσότερα τμήματα και παραδίδεται σε ένα client χρησιμοποιώντας HTTP. Ένα αρχείο περιγραφής παρουσίασης πολυμέσων (Media Presentation Description MPD) περιγράφει πληροφορίες τμήματος (χρονοδιάγραμμα, URL , χαρακτηριστικά μέσων όπως ανάλυση βίντεο και bitrates) . Τα τμήματα αυτά μπορούν να περιέχουν οποιαδήποτε δεδομένα πολυμέσων.
- Το FFmpeg[36] είναι ένα project ελεύθερου λογισμικού , όπου είναι μια τεράστια σουίτα λογισμικού από βιβλιοθήκες και προγράμματα για τη διαχείριση βίντεο, ήχου και άλλων αρχείων πολυμέσων και ροών (streams). Χρησιμοποιείται ευρέως για μετασχηματισμό μορφής , βασική επεξεργασία (κοπή και συγκόλληση), κλιμάκωση βίντεο και εφέ μετά την παραγωγή βίντεο.

- Το GPAC[37] είναι ένα πλαίσιο πολυμέσων ανοιχτού κώδικα. Το GPAC χρησιμοποιείται για ερευνητικούς και ακαδημαϊκούς σκοπούς και πέρα από τις βιομηχανικές συνεργασίες! Το έργο καλύπτει διάφορες πτυχές των πολυμέσων, με έμφαση στις τεχνολογίες παρουσίασης (γραφικά, κινούμενα σχέδια και διαδραστικότητα) και σε μορφές πολυμέσων όπως το MP4.Το MP4Box[38] είναι ένας συσκευαστής πολυμέσων (multimedia packager του GPAC), με τεράστιο αριθμό λειτουργιών: μετατροπή, διαίρεση, υπαινιγμό, ντάμπινγκ και άλλα. Πρόκειται για ένα εργαλείο γραμμής εντολών .
- Το dash.js[39] είναι ένα framework για την αναπαραγωγή βίντεο και ήχου που αναπαράγουν περιεχόμενο MPEG-DASH χρησιμοποιώντας βιβλιοθήκες JavaScript από την πλευρά του πελάτη. Εφαρμόζει τις βέλτιστες πρακτικές στην αναπαραγωγή του MPEG DASH και είναι ανθεκτικό σε πραγματικό περιβάλλον παραγωγής.

Οι τεχνολογίες οι οποίες προαναφερθήκαν χρησιμοποιήθηκαν με τον εξής τρόπο για την διεκπεραίωση του video streaming.Ο χρήστης αφού προηγουμένως μεταφόρτωσε το βίντεο του μέσο της εφαρμογής μέσο του ffmpeg γίνετε η μετατροπή του βίντεο σε 5 ποιότητες (1080p ,720p, 460p ,360p ,240p) και προκύπτουν αυτά τα 5 αρχεία ακόμα ένα αρχείο με τον ήχο του βίντεο.Στην συνέχεια αυτά τα αρχεία μπαίνουν σαν είσοδο στο MP4BOX και παράγετε το manifest αρχείο MPD τεχνολογίας MPEG DASH και τέλος στο view του χρήστη όπου θα γίνει η αναπαραγωγή του βίντεο μέσο του dash js που παίρνει σαν είσοδο αυτό το αρχείο MPD επιτυγχάνετε η αναπαραγωγή του βίντεο. Η διαδικασία αύτη φαίνεται στο ποιο κάτω σχήμα 4.3.



Σχήμα 4.3 Ροή Λειτουργείας Video Streaming

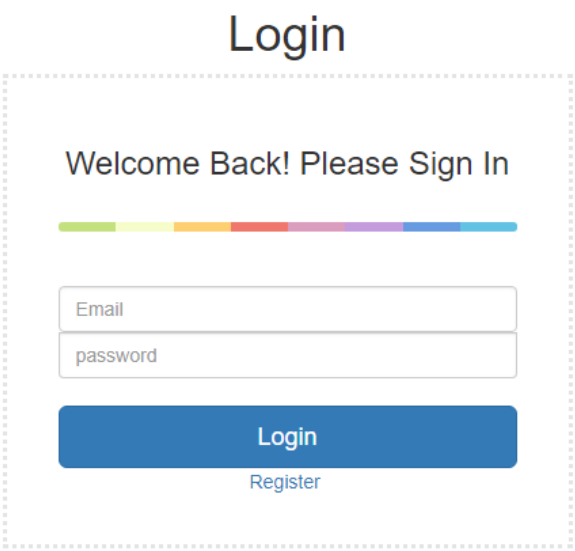


Σχήμα 4.4 Τεχνολογίες που χρησιμοποιήθηκαν για τη λογική , γραφική διεπαφή, βάσης δεδομένων το video streaming.

4.3.2.2 Views Εφαρμογής

Σε αυτή την υποενότητα παρουσιάζονται και αναλύονται όλα τα views της video streaming εφαρμογής με τα οποία αλληλεπιδρά ένας χρήστης της εφαρμογής.

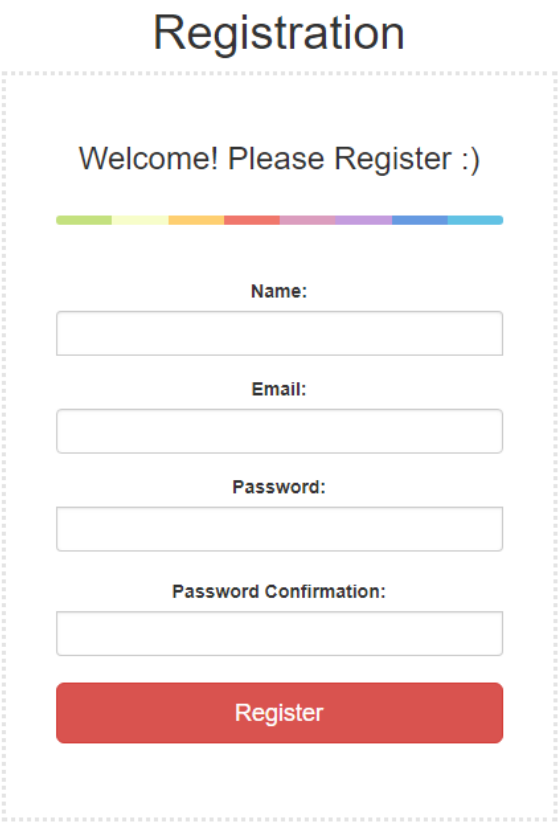
4.3.2.2.1 Login View



Σχήμα 4.5 Login view

Στο Login view του χρήστη υπάρχει μία φόρμα την οποία πρέπει να συμπληρώσει με τα σωστά στοιχεία ο χρήστης έτσι ώστε να έχει πρόσβαση στην εφαρμογή . Τα στοιχεία τα οποία είναι απαραίτητα για την ταυτοποίηση του χρήστη είναι το email και το password τα οποία ο χρήστης έχει δώσει όταν έκανε για πρώτη φορά Register στην εφαρμογή. Υπάρχει και η δυνατότητα αν κάποιος χρήστης δεν έχει λογαριασμό να επιλέξει το "Register" το οποίο θα τον οδηγήσει στην φόρμα για να κάνει εγράφη στην εφαρμογή. Γίνονται όλοι οι απαραίτητοι έλεγχοι για τα στοιχεία που δίνει ο χρήστης και αν χρειαστεί γίνεται εμφάνιση κατάλληλων ενημερωτικών μηνυμάτων και μνήματα λάθους . Αυτό το view είναι το αρχικό view που θα δει κάποιος όταν θα επισκεφτεί την εφαρμογή .

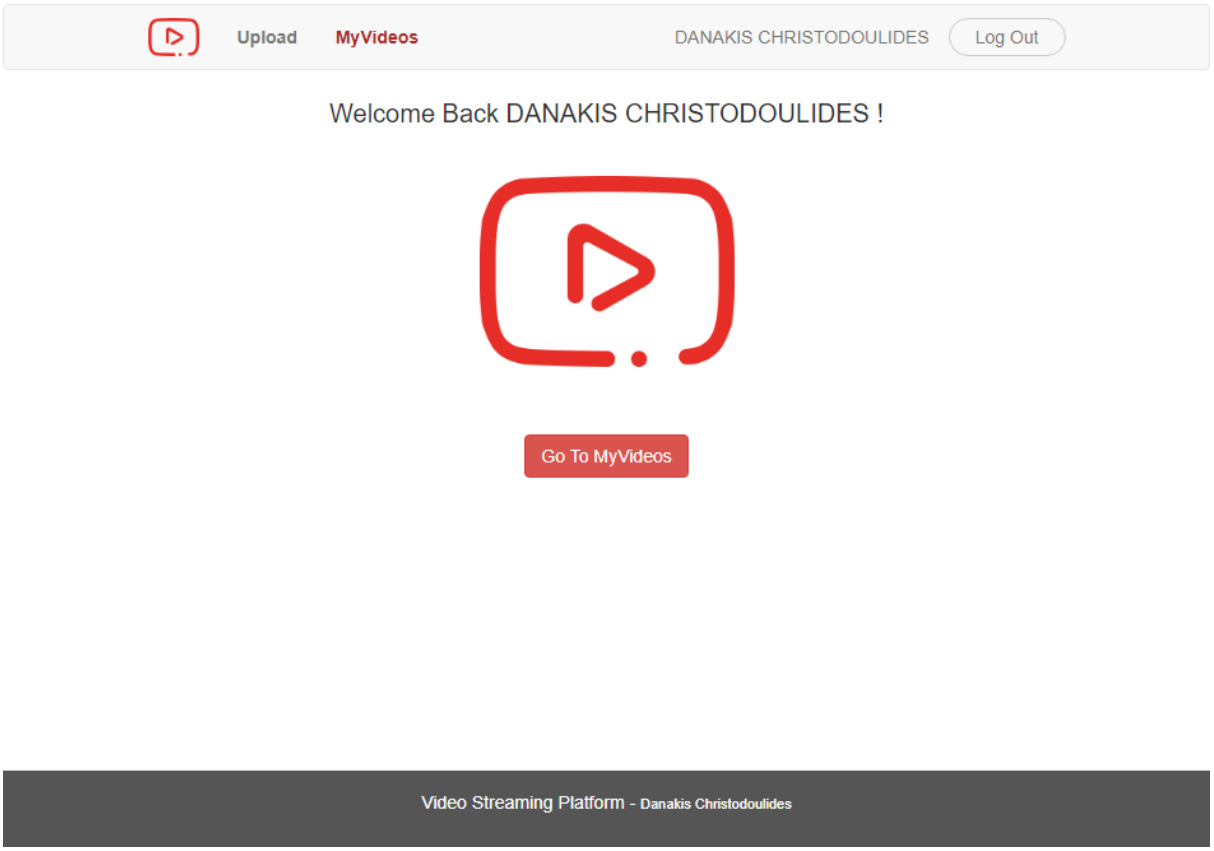
4.3.2.2.2 Register View



Σχήμα 4.6 Register view

Στο Register view θα μεταφερθεί ο χρήστης όταν πατήσει το “Register” στο login view (4.2.3.2.1) του χρήστη υπάρχει μία φόρμα την οποία πρέπει να συμπληρώσει με τα σωστά στοιχεία ο χρήστης έτσι ώστε να δημιουργηθεί ο λογαριασμός του χρήστη . Τα στοιχεία που πρέπει να εισάγει ο χρήστης είναι ένα όνομα (name) , Email , κωδικό (Password) , και επιβεβαίωση κωδικού (Password Confirmation) . Γίνονται όλοι οι απαραίτητοι έλεγχοι για τα στοιχεία που δίνει ο χρήστης και αν χρειαστεί γίνεται εμφάνιση κατάλληλων ενημερωτικών μηνυμάτων και μνήματα λάθους .

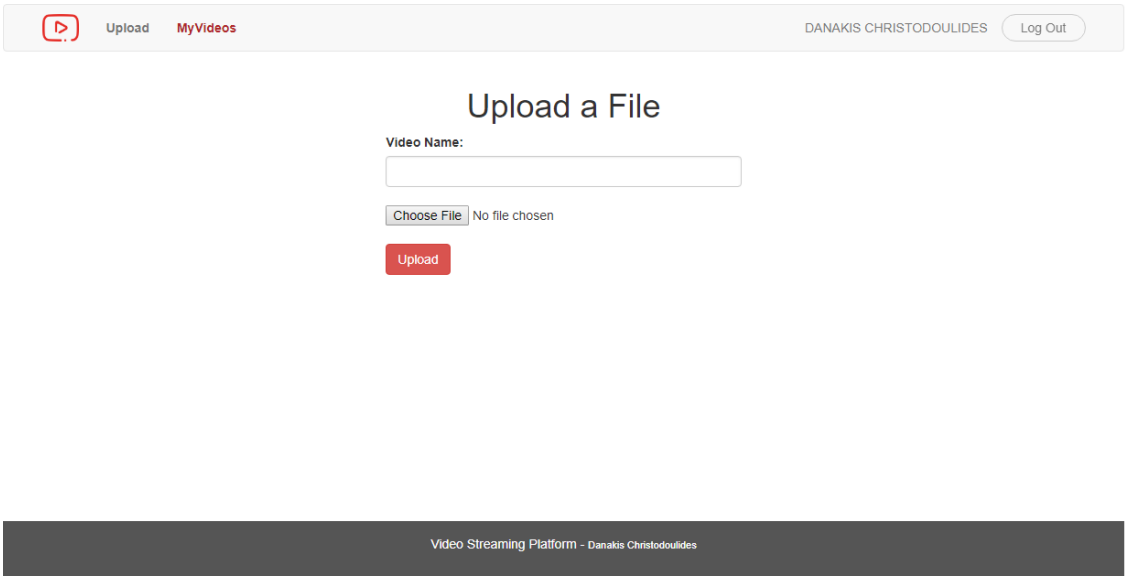
4.3.2.2.3 Main Dashboard View



Σχήμα 4.7 Main Dashboard View

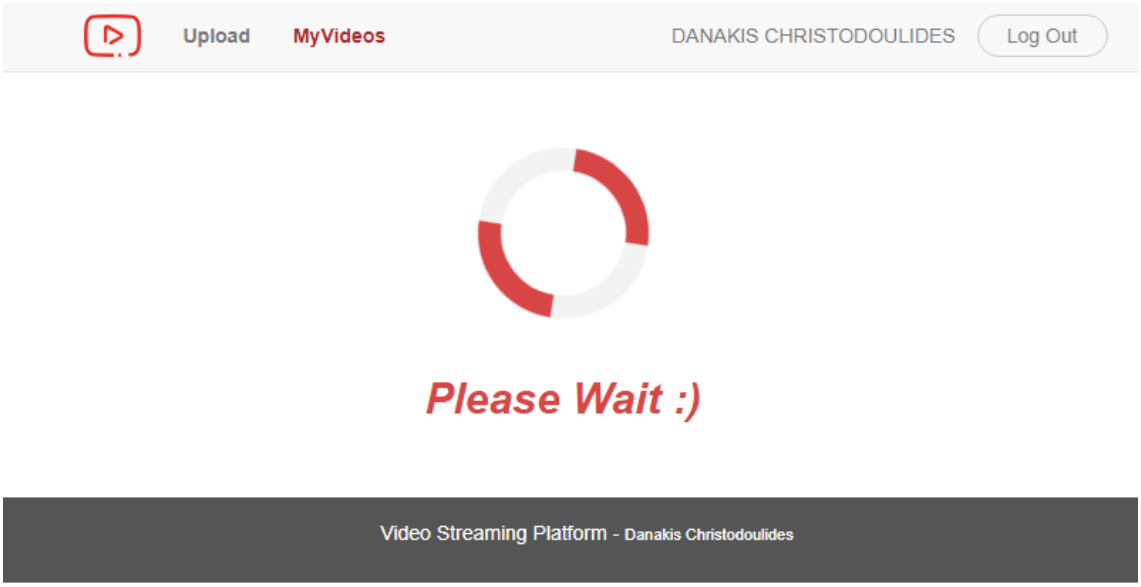
Σε αυτό το view θα μεταφερθεί ο χρήστης κάνει με ευτυχία login ή register για πρώτη φορά. Είναι η κεντρική οθόνη-view της εφαρμογής όπου ο χρήστης έχει πρόσβαση στις περισσότερες λειτουργίες της. Ο Χρήστης επιλέγοντας το “Upload” πάνω αριστερά στην μπάρα πλοήγησης θα μεταφερθεί στο view με την φόρμα όπου έχει την δυνατότητα να κάνει μεταφόρτωση κάποιου video.Επίσης πατώντας το κεντρικό κουμπί “Go To MyVideos” η επιλέγοντας το “MyVideos” από την μπάρα πλοήγησης ο χρήστης θα μεταφερθεί στο view που έχει την δυνατότητα να δει μια λίστα με τα βίντεο τα οποία έχει ανεβάσει μέχρι τώρα. Επιπρόσθετα υπάρχει και η δυνατότητα αποσύνδεσης με το κουπί που βρίσκετε πάνω δεξιά στην μπάρα πλοήγησης.

4.3.2.2.4 Upload View



Σχήμα 4.8 Main Dashboard View

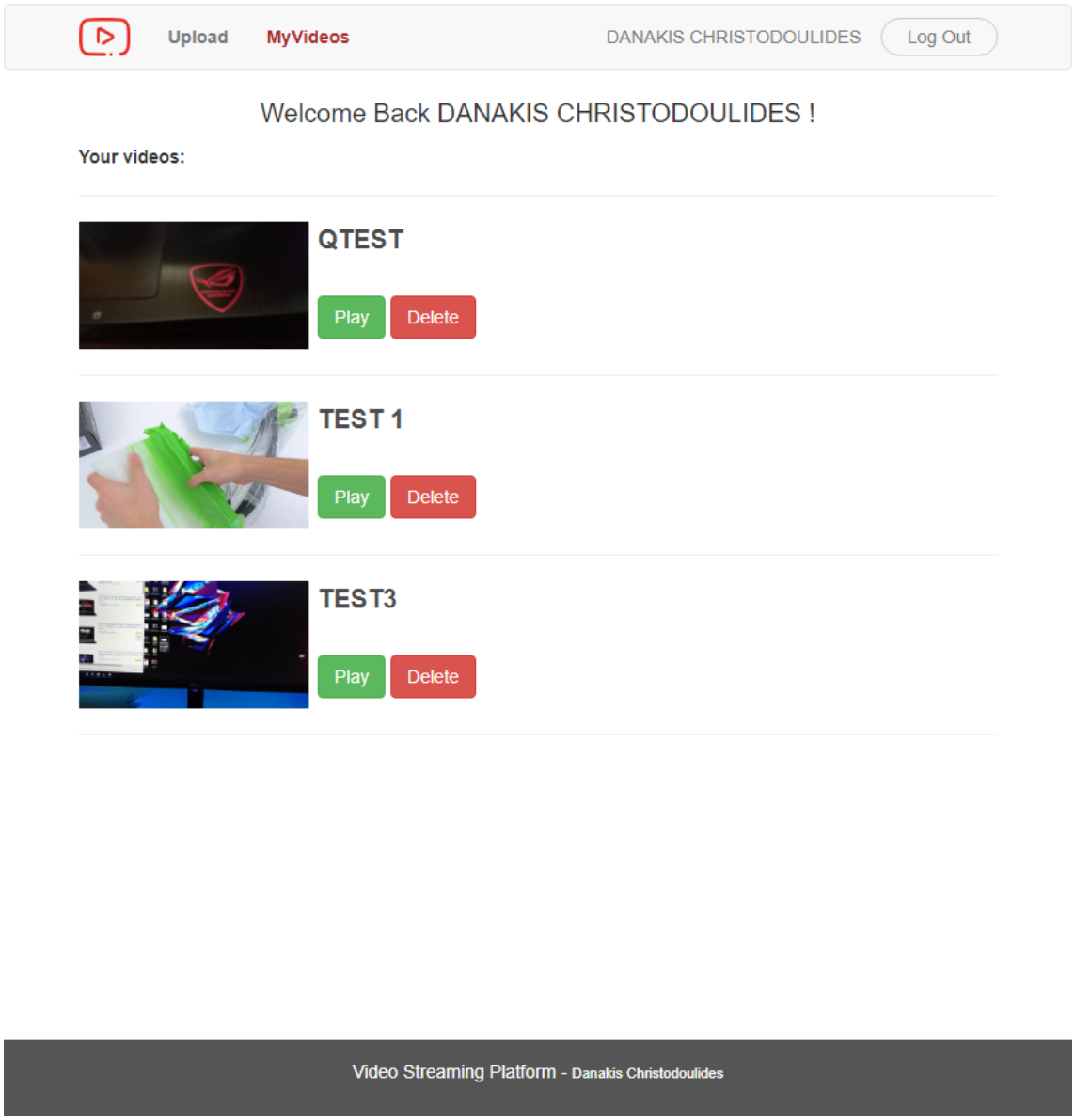
Στο Upload view θα μεταφερθεί ο χρήστης επιλέγοντας το "Upload" πάνω αριστερά στην μπάρα πλοήγησης . Σε αυτό το view υπάρχει μία φόρμα όπου μέσω του κουμπιού "Choose File" ο χρήστης μπορεί να επιλέξει ένα βίντεο από την συσκευή του το οποίο θέλει να μεταφορτώσει και αφού δώσει ένα όνομα πατώντας το upload θα ξεκινήσει η διαδικασία του upload και μέχρι να τελειώσει ο χρήστης θα βλέπει ένα loading animation όπως φαίνεται πιο κάτω :



Σχήμα 4.9 Upload Loading Animation

Παράλληλα γίνονται όλοι οι απαραίτητοι έλεγχοι και αν χρειαστεί γίνεται εμφάνιση κατάλληλων ενημερωτικών μηνυμάτων και μηνυμάτων λάθους . Όταν τελειώσει η διαδικασία του Upload τότε ο χρήστης θα μεταφερθεί αυτόματα στο Main Dashboard view (4.3.2.2.3)

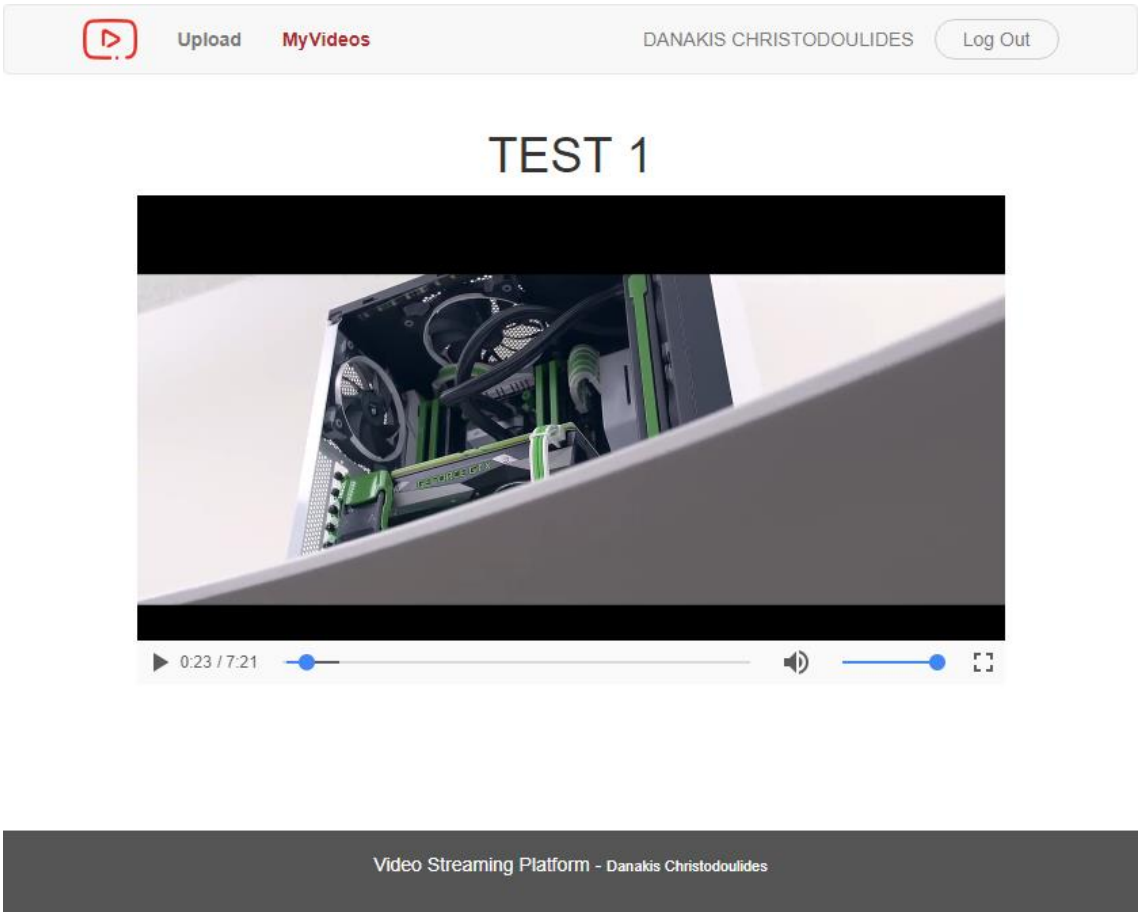
4.3.2.2.5 MyVideos View



Σχήμα 4.10 MyVideos view

Σε αυτό το view θα μεταφερθεί ο χρήστης πατώντας το κεντρικό κουμπί “Go To MyVideos” από το Main Dashboard view (4.2.3.2.3) ή επιλέγοντας το “MyVideos” από την μπάρα πλοήγησης. Εδώ εμφανίζεται μια λίστα με όλα τα βίντεο που ανέβασε ο χρήστης και για το κάθε ένα έχει την επιλογή “Play” για την αναπαραγωγή του βίντεο και την επιλογή “Delete” με την οποία μπορεί να διαγράψει το βίντεο.

4.3.2.2.6 Video Player View



Σχήμα 4.11 Video Player View

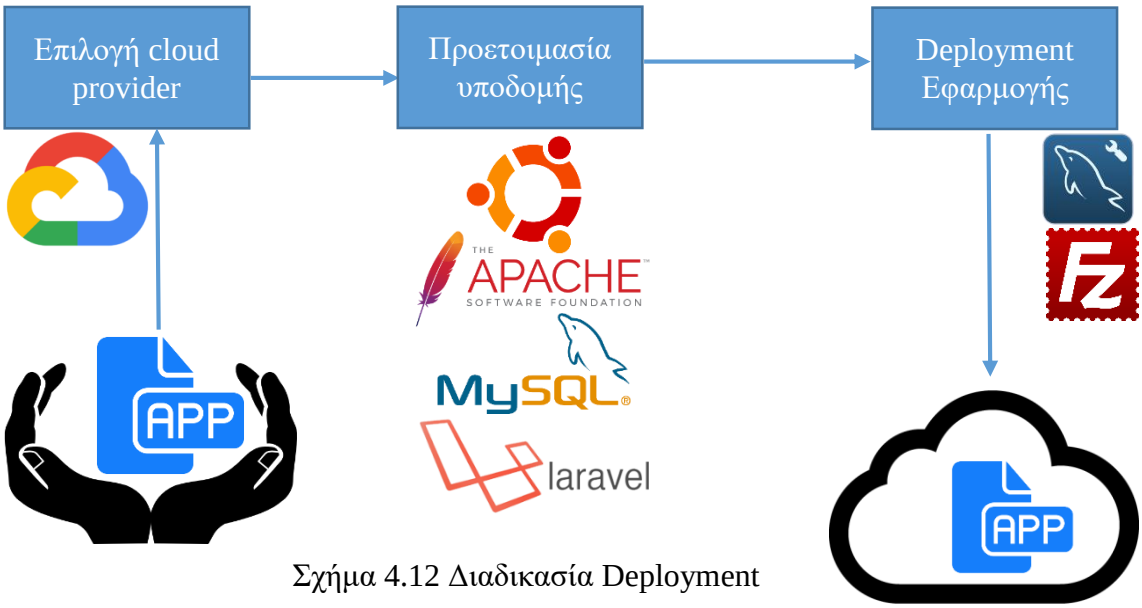
Σε αυτό το view θα μεταφερθεί ο χρήστης πατώντας κάποιο κουμπί “Play” από την λίστα με τα δικά του βίντεο στο MyVideos View. Σε αυτό το view γίνετε η αναπαραγωγή του βίντεο που επέλεξε ο χρήστης.

4.4 Deployment

Εφόσον έχει γίνει η υλοποίηση της εφαρμογής ακολουθήθηκε μια διαδικασία για να γίνει deploy στο cloud και στην συνέχεια έγιναν κάποιες παρατηρήσεις που αφορούν την εφαρμογή

4.4.1 Διαδικασία Deployment

Αρχικά η υποδομή που χρησιμοποιήθηκε είναι το Google Cloud Platform (GCP) όπου δημιουργήθηκε μια εικονική μηχανή (VM) με ένα εικονικό κεντρικό επεξεργαστή (vCPU) και 4 (3.75) GB μνήμης λειτουργικό σύστημα Ubuntu 16.04 LTS . Στην συνέχεια έγινε εγκατάσταση της MySQL και του Apache HTTP server[40] και έγιναν οι κατάλληλες διαδικασίες και ρυθμίσεις έτσι ώστε να μπορεί να δουλέψει ένα Laravel project σε Apache server. Για την μεταφορά των αρχείων στην εικονική μηχανή χρησιμοποιήθηκε το εργαλείο FileZilla[41] και για την διαχείριση της βάσης δεδομένων χρησιμοποιήθηκε το εργαλείο MySQL Workbench[42]. Πλέον εισάγοντας το IP address της εικονικής μηχανής σε ένα οποιοδήποτε φυλλομετρητή θα παρουσιαστεί η αρχική διεπαφή της εφαρμογής που είναι το Login View. Με το πέρας αυτής της φάσης υπάρχει η εφαρμογή στο google cloud και δουλεύει απροβλημάτιστα ,και στην συνέχεια γίνετε η μελέτη της συμπεριφοράς της εφαρμογής και να γίνουν κάποιές παρατηρήσεις που θα είναι καθοριστικές στο επόμενο κομμάτι της έρευνας που είναι η μοντελοποίηση. Η διαδικασία αύτη φενεται στο ποιο κάτω σχήμα 4.12



Σχήμα 4.12 Διαδικασία Deployment

4.4.2 Παρατηρήσεις

Δεδομένου ότι η εφαρμογή δουλεύει όπως αναμένεται στο cloud έγιναν κάποιες παρατηρήσεις που αφορούν το μέγεθος των πακέτων του βίντεο που λαμβάνει ο χρήστης για τις αναλύσεις βίντεο (1080p , 720p , 360p) κατά την διάρκεια του streaming ενός βίντεο . Παρατηρώντας οποιαδήποτε τυχαία χρονικά διαστήματα κατά την διάρκεια του streaming παρατηρηθήκαν τα εξής :

Video Resolution	Packets/sec	Average size
360p	2	42KB
720p	2	115KB
1080p	2	396KB

Πίνακας 4.1 Παρατηρήσεις μεγέθους πακέτων

Όπως φαίνεται στον πίνακα 4.1

- Για 360p βίντεο ένας χρήστης λαμβάνει 2 πακέτα το δευτερόλεπτο με μέσο μέγεθος 42KB το κάθε πακέτο.
- Για 720p βίντεο ένας χρήστης λαμβάνει 2 πακέτα το δευτερόλεπτο με μέσο μέγεθος 115KB το κάθε πακέτο.
- Για 1080p βίντεο ένας χρήστης λαμβάνει 2 πακέτο το δευτερόλεπτο με μέσο μέγεθος 396KB το κάθε πακέτο.

4.5 Μοντελοποίηση

Σε αυτή την ενότητα περιγράφονται οι μετρικές που θα συλλέγονται κατά την διάρκεια της εκτέλεσης του πειράματος αλλά και η μοντελοποίηση της video streaming εφαρμογής η οποία γίνετε για να είναι εφικτή η συλλογή των μετρικών.

4.5.1 Προσδιορισμός μετρικών

Οι μετρικές οι οποίες θα συλλέγονται κατά την διάρκεια της εκτέλεσης του πειράματος είναι συμπαντικές για να γίνει κατανοητό το τι γίνεται στην εικονική μηχανή και στο

δίκτυο κατά την διάρκεια του πειράματος. Οι μετρικές αυτές αφορούν τον Κεντρικό επεξεργαστή (CPU) , την Μνήμη (RAM) , το δίκτυο (Network) και των σκληρό δίσκο.

Βασικές μετρικές :

- CPU Utilization: Το ποσοστό της συνολικής υπολογιστικής ισχύος που χρησιμοποιείτε. Η υπερβολική χρήση της CPU μπορεί να αποτελέσει αιτία ανησυχίας στην περίπτωση μας και είναι μια ένδειξη για να γίνει κάποιο είδος scale up.
- Memory Utilization: Το ποσοστό της συνολικής μνήμης που χρησιμοποιείτε. Σε περίπτωση πού δεν υπάρχει διαθέσιμη μνήμη μπορεί να επηρεαστεί και να αυξηθεί και το CPU utilization .
- Network Send Bytes: Ο αριθμός τον bytes που στέλνονται στο δίκτυο από την εικονική μηχανή (VM). Είναι μια ένδειξη ότι η εφαρμογή του video streaming δουλεύει σωστά και ότι αποστέλλονται τα πακέτα των βίντεο.
- Concurrent Users (Ταυτόχρονοι Χρήστες): Πολύ σημαντική μετρική για τα πειράματα αυτής της έρευνας διότι είναι μια ένδειξη ποιότητάς της υποδομής που θα βοηθήσει στον εντοπισμό του μέγιστου αριθμού ταυτόχρονων χρηστών μπορεί να εξυπηρετήσει η υποδομή αυτή.
- Requests Per Second RPS (Αιτήματα ανά δευτερόλεπτο): Αντιστοιχεί στον αριθμό των αιτημάτων που δέχεται η εφαρμογή ανά μονάδα χρόνου . Είναι η μετρική η οποία επηρεάζει όλες τις υπόλοιπες μετρικές
- Throughput Per Second TPS: Πόσες μονάδες πληροφοριών μπορεί να επεξεργαστεί/εξυπηρετήσει ένα σύστημα σε μια καθορισμένη χρονική περίοδο. Είναι η ένδειξη η οποία δείχνει πόσα από τα εισερχόμενα request εξυπηρετούνται σε περίπτωση που δεν εξυπηρετούνται όλα τα requests πρέπει να γίνει κάποια ενέργεια .
- Latency: Ο χρόνος ο οποίος περνά από την στιγμή που γίνετε ένα request στον server μέχρι να έρθει πίσω η απάντηση στο request αυτό. Είναι επιθυμητό αυτό το νούμερο να είναι όσο ποιο μικρό γίνετε διότι στην περίπτωση του streaming όταν καθυστερούν τα πακέτα δεν έχει νόημα να κάνει κάποιος streaming .

Άλλες μετρικές :

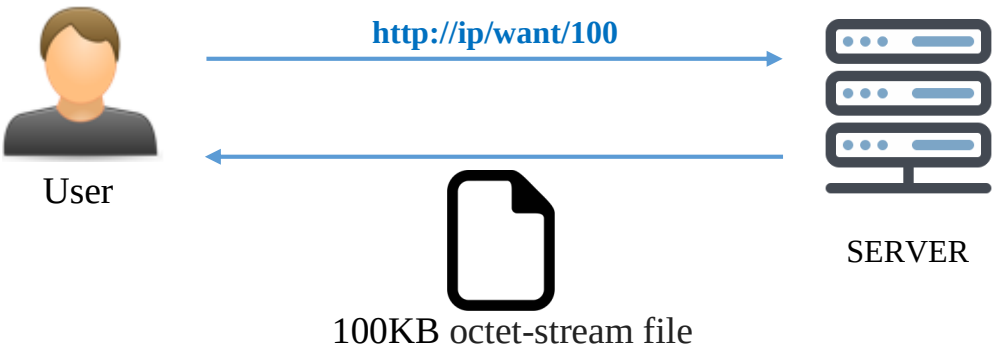
- Available Memory: η μνήμη που μπορεί να δοθεί άμεσα σε Διεργασίες χωρίς να γίνει εναλλαγή του συστήματος
- Free Memory: η μνήμη η οποία δεν χρησιμοποιείται καθόλου (μηδέν) που είναι άμεσα διαθέσιμη.
- USED Memory: η μνήμη η οποία χρησιμοποιείται.
- Disk Read Bytes: Ο αριθμός τον bytes που διαβάζονται από το Σκληρό δίσκο.
- Disk Write Bytes: Ο αριθμός τον bytes που γράφονται στο Σκληρό δίσκο.
- Network Receive Bytes: Ο αριθμός τον bytes που λαμβάνονται στην εικονική μηχανή (VM) από το δίκτυο.
- Network Packets Bytes: Ο αριθμός τον πακέτων που στέλνονται στο δίκτυο από την εικονική μηχανή (VM).
- Network Packets Bytes: Ο αριθμός τον πακέτων που λαμβάνονται στην εικονική μηχανή (VM) από το δίκτυο.

4.5.2 Μοντελοποίηση video streaming εφαρμογής

Λόγω του γεγονότος ότι η video streaming εφαρμογή που αναπτύχτηκε κατά την φάση της υλοποίησης όσο αφορά το video streaming (4.3.2.1) γίνεται με “κλειστό” τρόπο, δηλαδή όταν ο χρήστης ξεκινήσει την αναπαραγωγή ενός βίντεο γίνετε request στον server και λαμβάνετε το .MPD manifest αρχείο , από εκείνη την στιγμή αναλαμβάνει το DASH JS τα request για τα πακέτα του βίντεο από την μεριά του client με αυτόματο τρόπο . Με αποτέλεσμα να μην είναι εφικτό να γίνει η προσομοίωση πολλών χρηστών μεσώ ενός workload generator κατά την διάρκεια του πειράματος ούτε να είναι εφικτή η συλλογή κάποιων μετρικών όπως latency και Throughput Per Second (TPS). Έτσι με βάση των παρατηρήσεων που περιγράφονται στην υποενότητα (4.4.2) αναπτύχθηκε ουσιαστικά μια νέα εφαρμογή η οποία έχει σκοπό να δέχεται ένα HTTP request το οποίο θα ζητά ένα αριθμό σε KB και η εφαρμογή να απαντά με ένα HTTP μήνυμα όπου το περιεχόμενο θα ένα πακέτο με μέγεθος ίσο με αυτό που ζητήθηκε αντίστοιχο με τα πακέτα που στέλνει η εφαρμογή video streaming.

π.χ.

http://ip/want/42 αν τοποθετήσει κάποιος το link αυτό σε ένα φυλλομετρητή όπου ip είναι το IP της εφαρμογής αυτής, θα πάρει σαν απάντηση ένα πακέτο μεγέθους 42 KB με το Content-Type της απάντησης να είναι application/octet-stream.



Σχήμα 4.13 Τρόπος λειτουργίας εφαρμογής η οποία μοντελοποιεί την video streaming εφαρμογή

4.6 Σχεδιασμός Περάματος

4.6.1 Work Load Generator

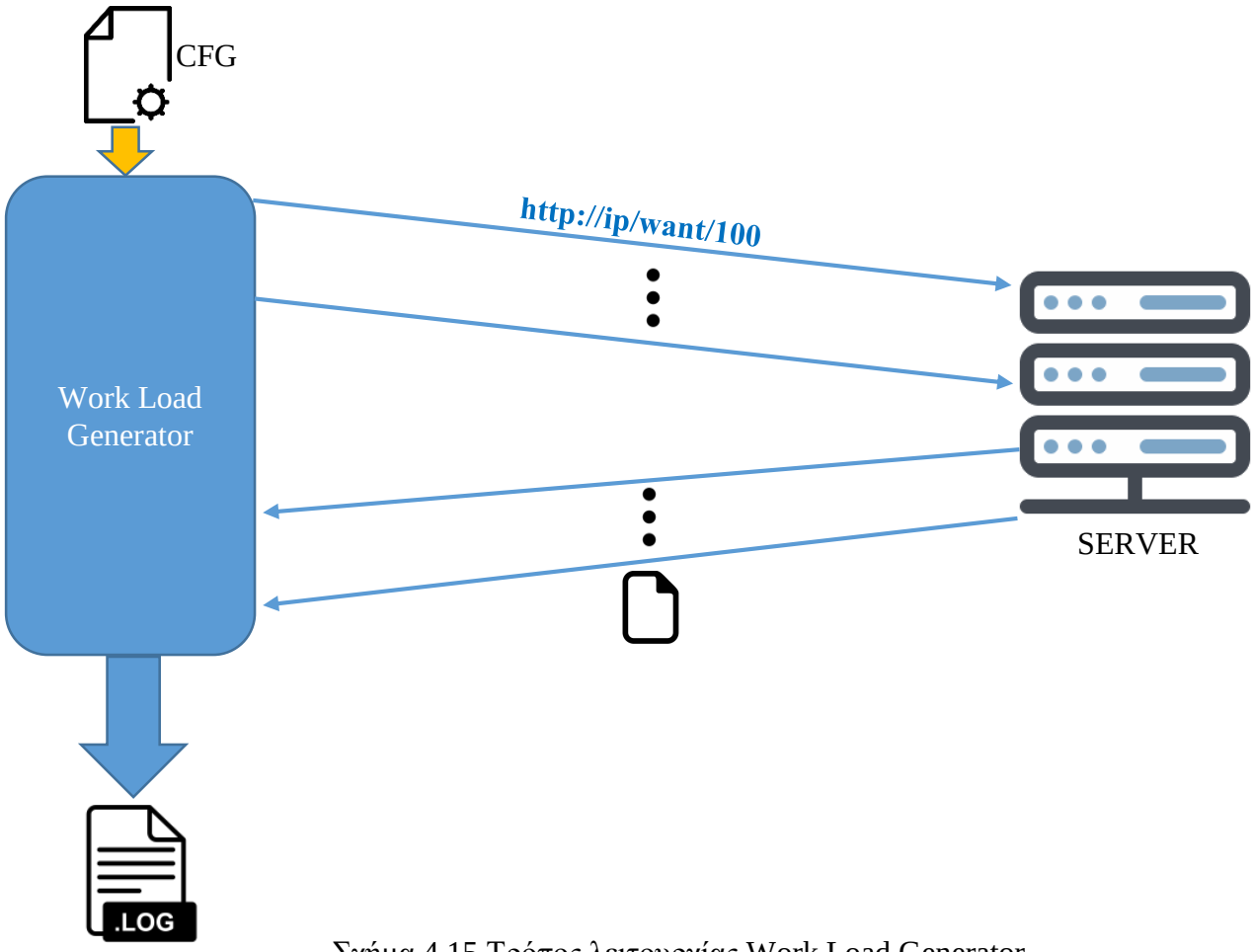
Ένα βασικό συστατικό στοιχείο του πειράματος είναι ο Work load Generator που ουσιαστικά είναι ένα πρόγραμμα το οποίο στέλνει HTTP requests σε κάποιο link που ορίζεται στις ρυθμίσεις του . Είναι απαραίτητος για να γίνει εφικτή η προσομοίωση πολλών χρηστών της εφαρμογής και της ποσότητας εργασίας σε πραγματικά σενάρια . Έτσι χρησιμοποιήθηκε ένα έτοιμος Workload Generator [43] ο οποίος για το συγκεκριμένο πείραμα χρησιμοποιείτε ως εξής. Εάν θέλουμε να προσομοιώσουμε 10 ταυτόχρονους χρήστες στο σύστημα μας οι οποίοι βλέπουν 720p videos. Τότε πάμε στο configuration file του Work Load generator όπως φαίνεται στο σχήμα 4.14 και το τροποποιούμε έτσι ώστε να δημιουργήσει 10 threads (αριθμός χρηστών) το οποίο κάθε thread θα στέλνει 2 request/sec (minOperations , maxOperations = 20) και να ζητά πακέτο με μέγεθος 115KB (url) από την εφαρμογή που μοντελοποιεί την video streaming εφαρμογή (4.5.2) . Έτσι αλλάζοντας τις πιο πάνω παραμέτρους κάθε φορά πετυχαίνουμε την προσομοίωση των χρηστών αλλά και των αριθμό των request (φόρτος εργασίας) για κάθε πείραμα . Επιπρόσθετα ο Work Load Generator υπολογίζει το Latency (min , max , average) , Throughput Per Second (TPS) και Requests Per Second RPS (Configure RPS , Real RPS) και τα αποτελέσματα τα τοποθετεί σε ένα αρχείο log κάθε συγκεκριμένο

χρονικό διάστημα που του ορίζεις εσύ(output time : 5) . Ο τρόπος λειτουργείας του Workload Generator φαίνεται στο σχήμα 4.15 και παράδειγμα από το log αρχείο που περιέχει τις μετρικές φαίνεται στο σχήμα 4.16.

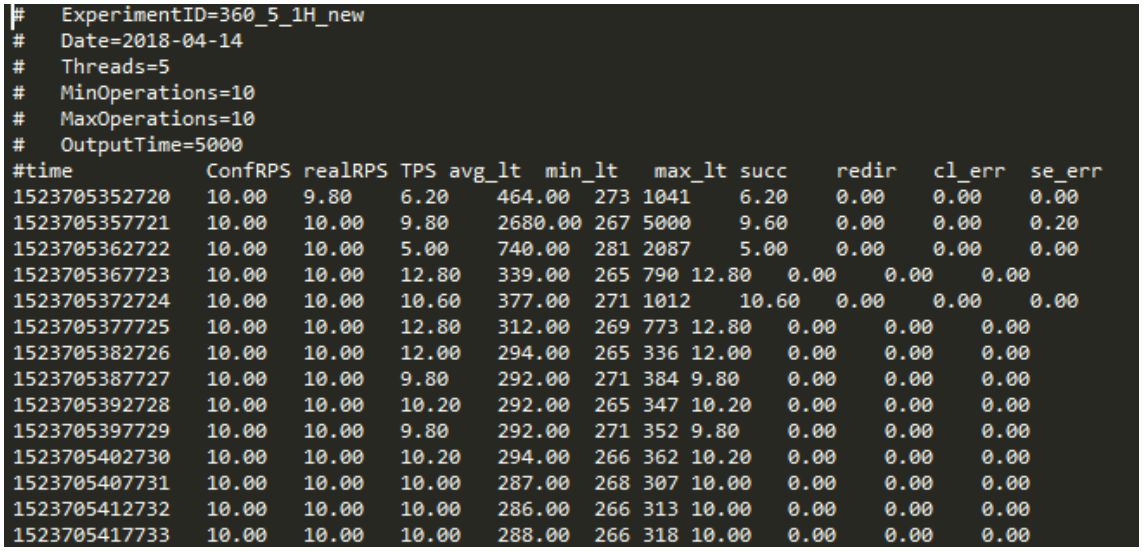
```
    "experiment":"720p 1 Hour",
    "maxThreads" : 10,
    "minOperations" :20 ,
    "maxOperations" :20 ,
    "outputTime" : 5

{
  "operationId": "720p",
  "weight":100,
  "url": "http://35.187.87.183/want/115",
  "method": "GET"
},
```

Σχήμα 4.14 Configuration Workload Generator για 10 ταυτόχρονους χρήστες που βλέπουν 720p βίντεο για μια ώρα



Σχήμα 4.15 Τρόπος λειτουργίας Work Load Generator



```
#   ExperimentID=360_5_1H_new
#   Date=2018-04-14
#   Threads=5
#   MinOperations=10
#   MaxOperations=10
#   OutputTime=5000
#time          ConfRPS realRPS TPS avg_lt  min_lt   max_lt succ    redir   cl_err  se_err
1523705352720   10.00   9.80    6.20    464.00  273 1041    6.20    0.00    0.00    0.00
1523705357721   10.00   10.00   9.80    2680.00 267 5000    9.60    0.00    0.00    0.20
1523705362722   10.00   10.00   5.00    740.00  281 2087    5.00    0.00    0.00    0.00
1523705367723   10.00   10.00   12.80   339.00  265 790 12.80   0.00    0.00    0.00
1523705372724   10.00   10.00   10.60   377.00  271 1012    10.60   0.00    0.00    0.00
1523705377725   10.00   10.00   12.80   312.00  269 773 12.80   0.00    0.00    0.00
1523705382726   10.00   10.00   12.00   294.00  265 336 12.00   0.00    0.00    0.00
1523705387727   10.00   10.00   9.80    292.00  271 384 9.80    0.00    0.00    0.00
1523705392728   10.00   10.00   10.20   292.00  265 347 10.20   0.00    0.00    0.00
1523705397729   10.00   10.00   9.80    292.00  271 352 9.80    0.00    0.00    0.00
1523705402730   10.00   10.00   10.20   294.00  266 362 10.20   0.00    0.00    0.00
1523705407731   10.00   10.00   10.00   287.00  268 307 10.00   0.00    0.00    0.00
1523705412732   10.00   10.00   10.00   286.00  266 313 10.00   0.00    0.00    0.00
1523705417733   10.00   10.00   10.00   288.00  266 318 10.00   0.00    0.00    0.00
```

Σχήμα 4.16 Παράδειγμα του output log file του Workload Generator

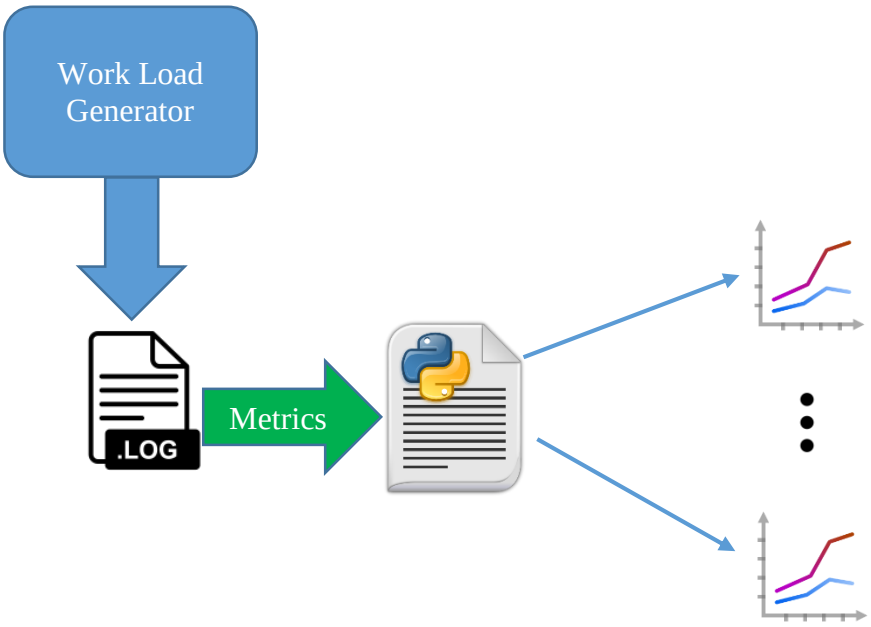
4.6.2 Εργαλείο Συλλογής Μετρικών

Για την συλλογή των μετρικών που περιγράφονται στην υποενότητα Προσδιορισμός μετρικών (4.5.1) και την εξαγωγή των γραφικών παραστάσεων αναπτύχθηκε ένα εργαλείο στην γλώσσα προγραμματισμού Python και αποτελείτε από 2 τμήματα (1) βρίσκετε στην μεριά του client (Work Load Generator) , (2) βρίσκεται στην μεριά του Server (εφαρμογή μοντελοποίησης εφαρμογής Video Streaming) .

Όσο αφορά την αλληλεπίδραση με τη βάση δεδομένων χρησιμοποιήθηκε το SQLAlchemy που είναι ένα ολοκληρωμένο σύνολο εργαλείων για βάσεις δεδομένων και Python.

Για την δημιουργία και την εξαγωγή των γραφικών παρατάσεων χρησιμοποιήθηκε η βιβλιοθήκη Matplotlib.

Για την εξαγωγή των γραφικών παραστάσεων από την μεριά του client δηλαδή από την μεριά του Work Load Generator υπάρχει ένα python script . Με το πέρας της εκτέλεσης ενός σεναρίου-πειράματος όπως εξηγήθηκε ποιο πριν ο Work Load Generator δημιουργεί ένα log αρχείο με της μετρήσεις. Η δουλειά του script είναι να ανοίξει το log αρχείο, να διαβάσει τις μετρικές και να δημιουργήσει και να αποθηκεύσει τις επιθυμητές γραφικές παρατάσεις στον ίδιο φάκελο με το log file του Workload Gernerator.



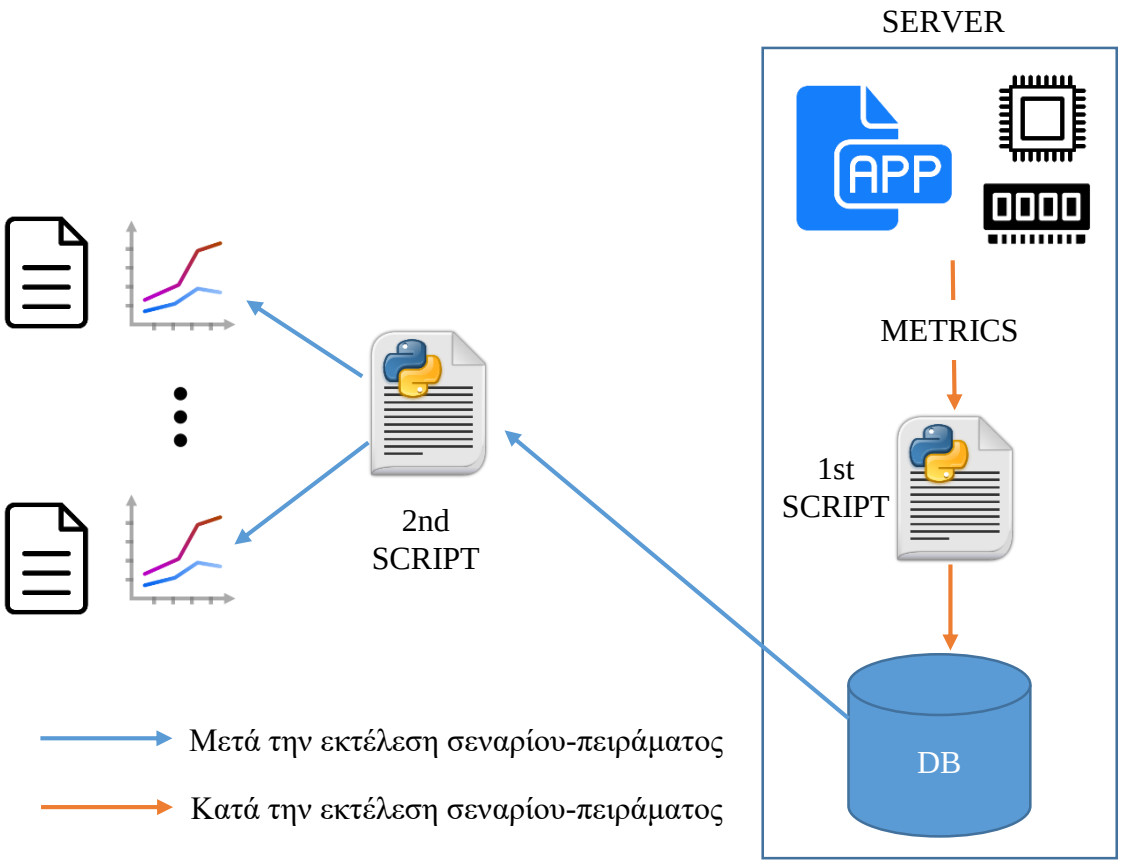
Σχήμα 4.17 Τρόπος λειτουργίας Monitoring Tool στην μεριά του Client.

Οι γραφικές παραστάσεις που εξάγονται είναι :

- Latency (MIN , MAX , AVERAGE) / Time
- REAL RPS / Time
- TPS / Time
- RPS(Configure RPS , Real RPS , TPS) / Time

Για την συλλογή των μετρικών και την εξαγωγή των γραφικών παραστάσεων από την μεριά του Server δηλαδή από την μεριά του VM που τρέχει η εφαρμογή μοντελοποίησης της video streaming εφαρμογής υπάρχουν 2 python scripts . Ένα για την συλλογή των μετρικών και δεύτερο για την δημιουργία και την εξαγωγή των γραφικών παραστάσεων. Το πρώτο script τρέχει κατά την διάρκεια της εκτέλεσης ενός σεναρίου-πειράματος συλλέγει / υπολογίζει τις μετρικές και τις αποθηκεύει στην βάση δεδομένων του Server (Mysql). Οι μετρικές αυτές με τις κατάλληλες διαδικασίες συλλέγονται / υπολογίζονται κάθε 5 δευτερόλεπτα. Το δεύτερο τρέχει με το πέρας της εκτέλεσης ενός σεναρίου-πειράματος και η δουλεία του είναι να διαβάσει τις μετρικές από την βάση δεδομένων να δημιουργήσει και να αποθηκεύσει τις επιθυμητές γραφικές παρατάσεις και επίσης να δημιουργήσει αρχεία με τις τιμές των μετρικών όπως ήταν στην βάση δεδομένων διότι

με το επόμενη εκτέλεση σεναρίου – πειράματος ότι υπάρχει στην βάση δεδομένων πριν σβήνετε για να μπουν οι νέες μετρικές .



Σχήμα 4.18 Τρόπος λειτουργίας Monitoring Tool στην μεριά του Server.

Οι μετρικές που συλλέγονται / υπολογίζονται είναι :

- CPU Utilization
- Memory Utilization
- Available Memory
- Free Memory
- USED Memory
- Disk Read Bytes
- Disk Write Bytes
- Network Send Bytes
- Network Receive Bytes
- Network Packets Bytes

- Network Packets Bytes

μέσω της βιβλιοθήκης psutil (python system and utility utilities) που χρησιμοποιείτε για την ανάκτηση πληροφοριών σχετικά με τις τρέχουσες διαδικασίες και τη χρήση του συστήματος (CPU, μνήμη, δίσκους, δίκτυο, αισθητήρες) στην Python και αποθηκεύονται σε βάση δεδομένων mysql με τρόπο που φενεται στο σχήμα 4.19.

CPU Metrics	
Table name	cpu_utils
fields	
name	type
id	INTEGER
value	DOUBLE
created_at	DATETIME

MEMORY Metrics	
Table name	mem_metrics
fields	
name	type
id	INTEGER
mem_util	DOUBLE
mem_total	BIGINT
mem_ave	BIGINT
mem_used	BIGINT
mem_free	BIGINT
created_at	DATETIME

DISK Metrics	
Table name	disk_metrics
fields	
name	type
id	INTEGER
read_bytes	BIGINT
write_bytes	BIGINT
created_at	DATETIME

NETWORK Metrics	
Table name	network_metrics
fields	
name	type
id	INTEGER
bytes_sent	BIGINT
bytes_receve	BIGINT
packets_sent	BIGINT
packets_receve	BIGINT
created_at	DATETIME

Σχήμα 4.19 Πίνακες μετρικών στην βάση δεδομένων .

Οι γραφικές παραστάσεις που εξάγονται και αποθηκεύονται στον apache server είναι :

- CPU Utilization / Time
- CPU Utilization Moving Average/ Time
- Memory Utilization / Time
- Available Memory / Time
- Free Memory / Time
- USED Memory / Time
- Disk Read Bytes / Time
- Disk Write Bytes / Time
- Network Send Bytes / Time
- Network Receive Bytes / Time
- Network Packets Bytes / Time
- Network Packets Bytes / Time

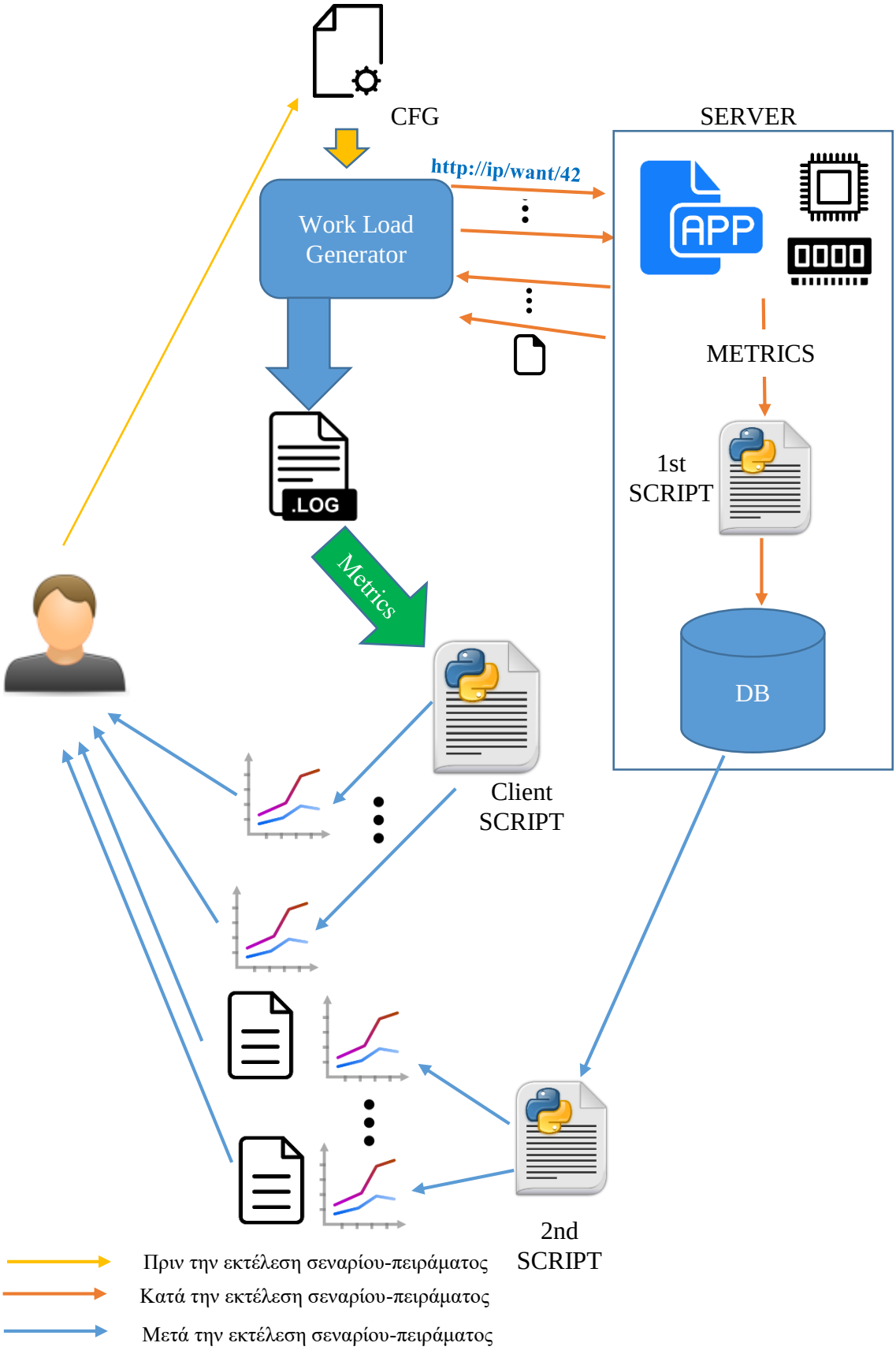
4.6.3 Πειραματικό Framework

Το πειραματικό Framework αποτελείτε από 3 βασικά συστατικά στοιχεία:

1. Work Load Generator (4.6.1) όπου χρησιμοποιείτε για την προσομοίωση πολλών χρηστών και του φόρτου εργασίας αλλά είναι υπεύθυνος για την συλλογή κάποιων μετρικών όπως (TPS , LATENCY, RPS)
2. Εφαρμογή μοντελοποίησης εφαρμογής video streaming (4.5.2) η οποία μοντελοποιεί την εφαρμογή video streaming για να είναι εφικτή η προσομοίωση των χρηστών και σωστή συλλογή των μετρικών .
3. Εργαλείο συλλογής μετρικών (4.6.2) το οποίο είναι υπεύθυνο για την εξαγωγή των γραφικών παραστάσεων από την μεριά του client-Workload Generator , για την συλλογή / υπολογισμό των μετρικών από την μεριά του server ,την αποθήκευση τους στην βάση δεδομένων και εξαγωγή των γραφικών παραστάσεων

4.6.3.1 Τρόπος λειτουργίας – λογική Πειράματος Α

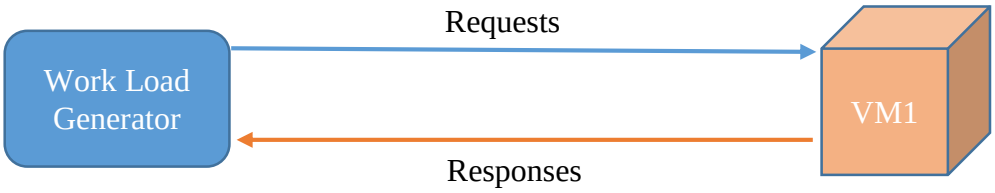
Η λογική για την εκτέλεση ενός σεναρίου είναι η εξής :
Έστω ότι θέλουμε να τρέξουμε το πείραμα 10 ταυτόχρονων χρηστών που βλέπουν βίντεο ανάλυσης 360p για διάρκεια μια ώρα .Αφού ρυθμίσουμε τον Workload Generator με τον τρόπο που περιγράφετε στην υποενότητα 4.6.1 και δεδομένου ότι η εφαρμογή μοντελοποίησης της εφαρμογής video streaming "τρέχει" σε εικονική μηχανή στο Google Cloud Platform , τότε τεθούμε σε λειτουργία τον Workload generator και παράλληλα το πρώτο python script του monitoring tool στην πλευρά του server για να συλλέγει / υπολογίζει τις μετρικές και να τις τοποθετεί στην βάση δεδομένων και περιμένουμε μια ώρα (χρόνος πειράματος) Με το πέρας της μιας ώρας σταματούμε την λειτουργεία του Workload Generator και του 1ου python script του monitoring tool στον Server.Στην συνέχεια τρέχομε το python script του monitoring tool στην πλευρά του client-Workload Generator και το 2ο python script του monitoring tool στον Server για την εξαγωγή των μετρικών και γραφικών παραστάσεων. Η διαδικασία αυτή φαίνεται στο σχήμα 4.20 που ακολουθεί.



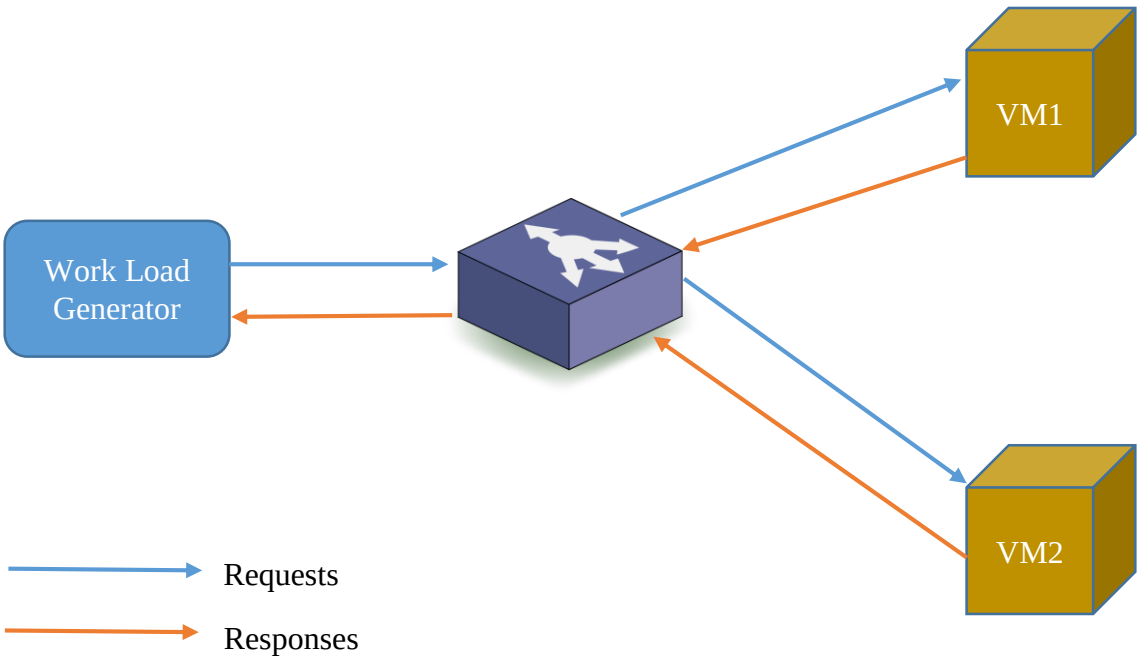
Σχήμα 4.20 Τρόπος λειτουργίας Πειραματικού Framework.

4.6.3.2 Τρόπος λειτουργίας – λογική Πειράματος Β

Ο τρόπος λειτουργίας και η λογική του πειράματος Β είναι ο ίδιος με αυτόν που περιγράφεται στο Πείραμα Α με την διαφορά εδώ ότι εισάγονται έννοιες της ελαστικότητας – επεκτασιμότητας. Ουσιαστικά εδώ προστίθεται ακόμα μια εικονική μηχανή (VM) στο πειραματικό Framework η οποία είναι ακριβός η ίδια με την αρχική και επίσης τοποθετείτε ένας Load Balancer μεταξύ του Workload Generator και των δυο εικονικών μηχανών (VMs). Αυτό επιτυγχάνεται με την χρήση του Google Cloud Platform - HTTP Load Balancing με τον αλγόριθμο κατανομής φορτίου του load balancer στις δύο εικονικές μηχανές (VMs) να είναι ο αλγόριθμος round-robin . Οι διαφορές στην βασική δομή του πειραματικού Framework στις δύο φάσεις φαίνονται στα ποιο κάτω σήματα.



Σχήμα 4.21 Βασική Δομή πειραματικού Framework κατά το πείραμα Α.



Σχήμα 4.22 Βασική Δομή πειραματικού Framework κατά το πείραμα Β.

4.6.4 Σενάρια εκτέλεσης πειραμάτων

Τα σενάρια εκτέλεσης της φάσης Α είναι :

Πείραμα Α – Μια εικονική μηχανή (VM)		
Σενάρια	Διάρκεια	Διαμόρφωση
Σενάριο 1	1 Ώρα	10 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p
Σενάριο 2	1 Ώρα	20 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p
Σενάριο 3	1 Ώρα	40 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p

Πίνακας 4.2 σενάρια εκτέλεσης της φάσης Α

Τα σενάρια εκτέλεσης της φάσης Β είναι :

Πείραμα Β – Δύο εικονικές μηχανές (VMs) και ένας Load Balancer		
Σενάρια	Διάρκεια	Διαμόρφωση
Σενάριο 1	1 Ώρα	10 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p
Σενάριο 2	1 Ώρα	20 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p
Σενάριο 3	1 Ώρα	40 Ταυτόχρονοι χρήστες Ανάλυση Βίντεο 360p

Πίνακας 4.3 σενάρια εκτέλεσης της φάσης Β

Κεφάλαιο 5

Εκτέλεση του Πειράματος και αποτελέσματα

5.1 Εκτέλεση και αποτελέσματα πειράματος Α

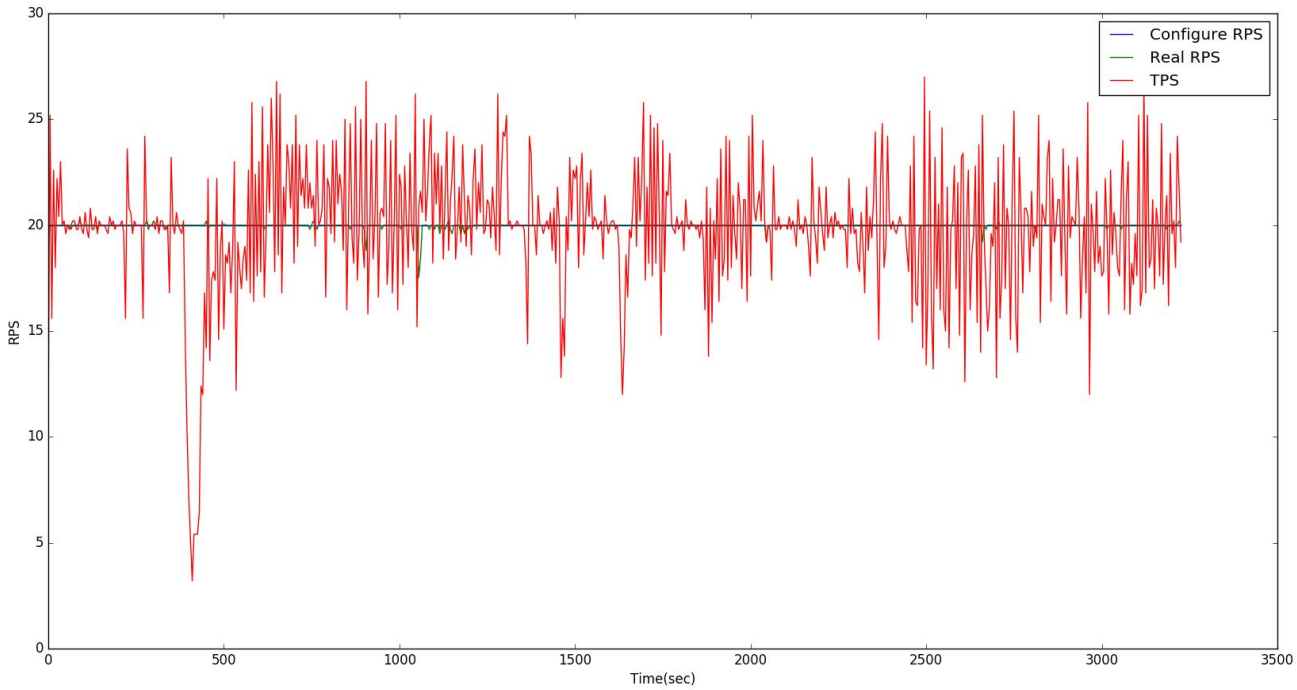
Ο τρόπος λειτουργείας και η λογική του πειράματος Α περιγράφονται στην υποενότητα 4.6.3.1 (Τρόπος λειτουργίας – λογική πειράματος Α) και η βασική δομή πειραματικού Framework Πειράματος Α φαίνεται στο σχήμα 4.21

5.1.1 Σενάριο 1

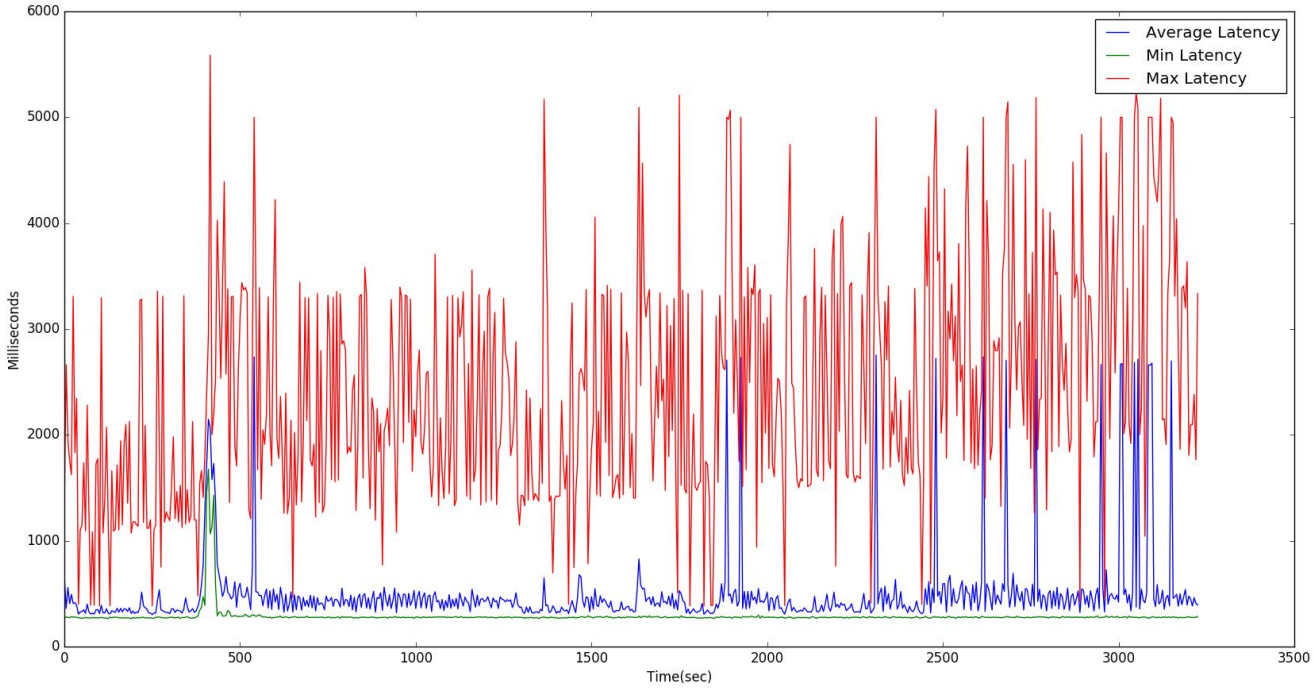
Σενάριο 1	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	10
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	10
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	20

Πίνακας 5.1 διαμόρφωση Σενάριού 1

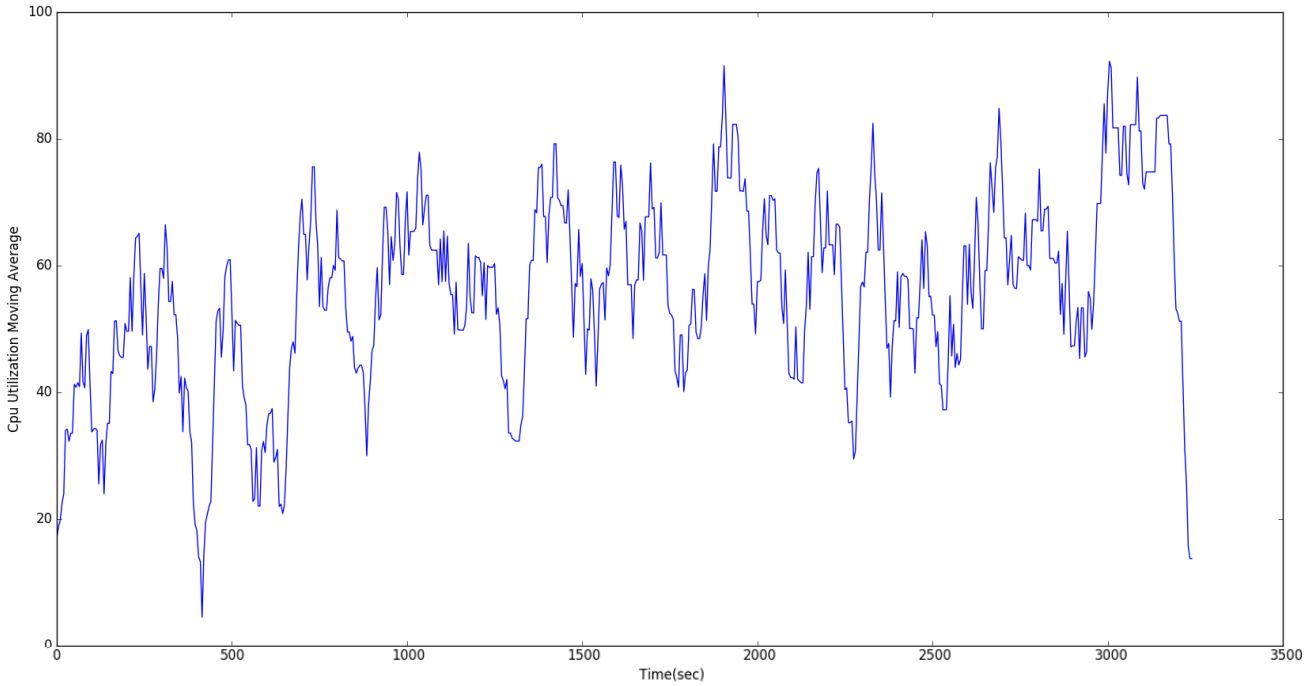
5.1.1.1 Γραφικές παραστάσεις – αποτελέσματα



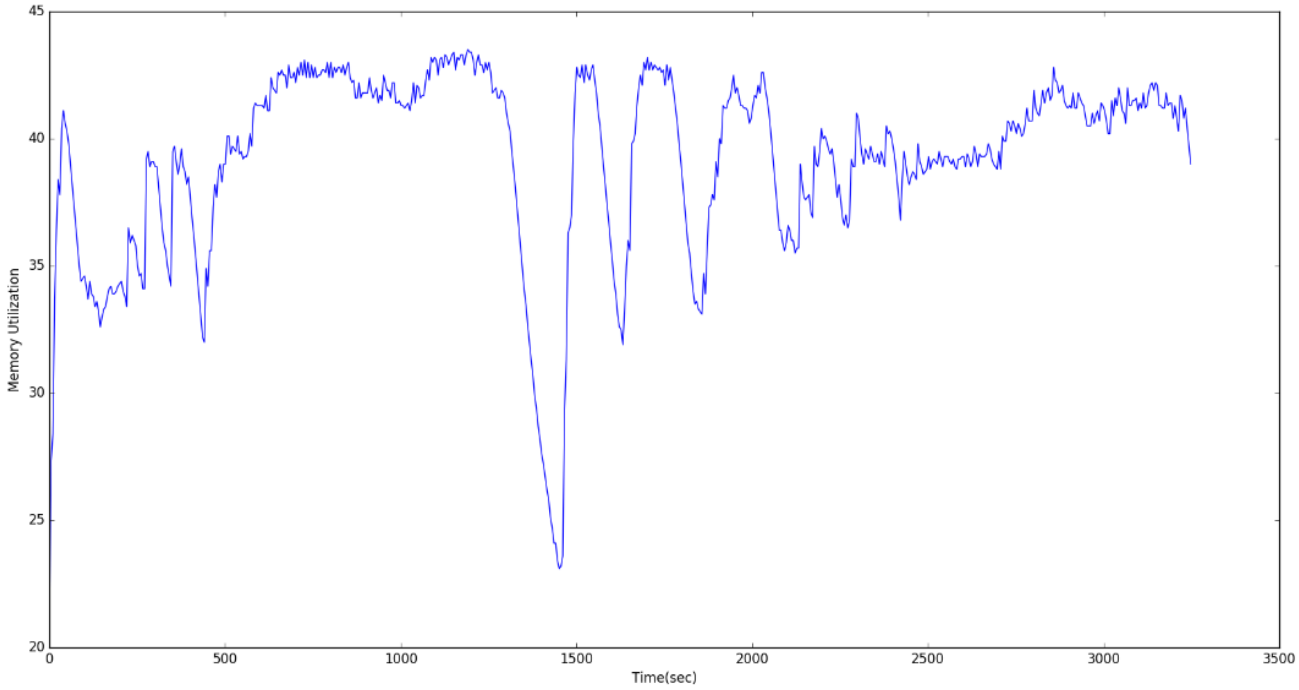
Σχήμα 5.1 RPS (Configure , Real RPS, Throughput) σενάριο 1 πειράματος Α



Σχήμα 5.2 Latency (Min , Max, Average) σενάριο 1 πειράματος Α



Σχήμα 5.3 CPU Moving Average σενάριο 1 πειράματος Α



Σχήμα 5.4 Memory Utilization σενάριο 1 πειράματος Α

5.1.1.2 Παρατηρήσεις

Στο Σχήμα 5.1 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και παρατηρούμε ότι αυτά τα 2 σε γενικές γραμμές είναι ίσα δηλαδή εξυπηρετούνται όλα τα αιτήματα .

Στο Σχήμα 5.2 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος είναι κάτω από μισό (0.5) δευτερόλεπτο .

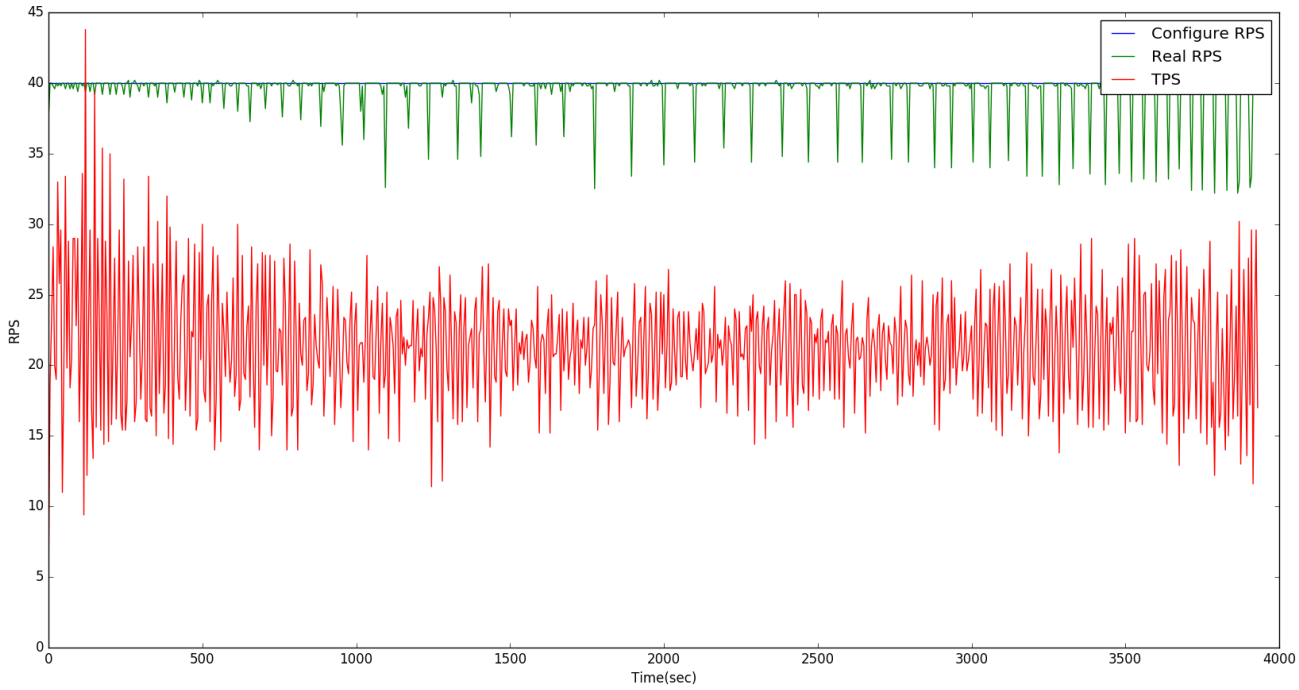
Στο Σχήμα 5.3 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) κυμαίνεται μεταξύ 20% και 90% , ένδειξη ότι ο επεξεργαστής δουλεύει στα όρια του .

5.1.2 Σενάριο 2

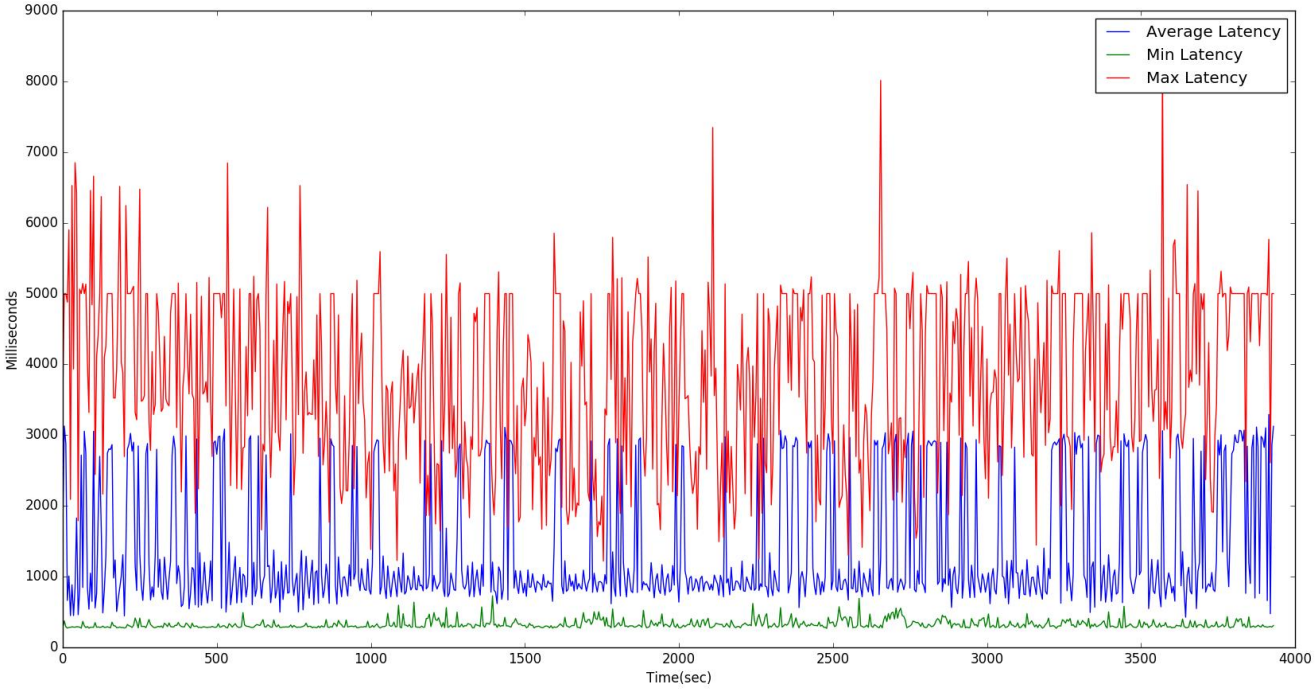
Σενάριο 2	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	20
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	20
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	40

Πίνακας 5.2 διαμόρφωση Σενάριού 2

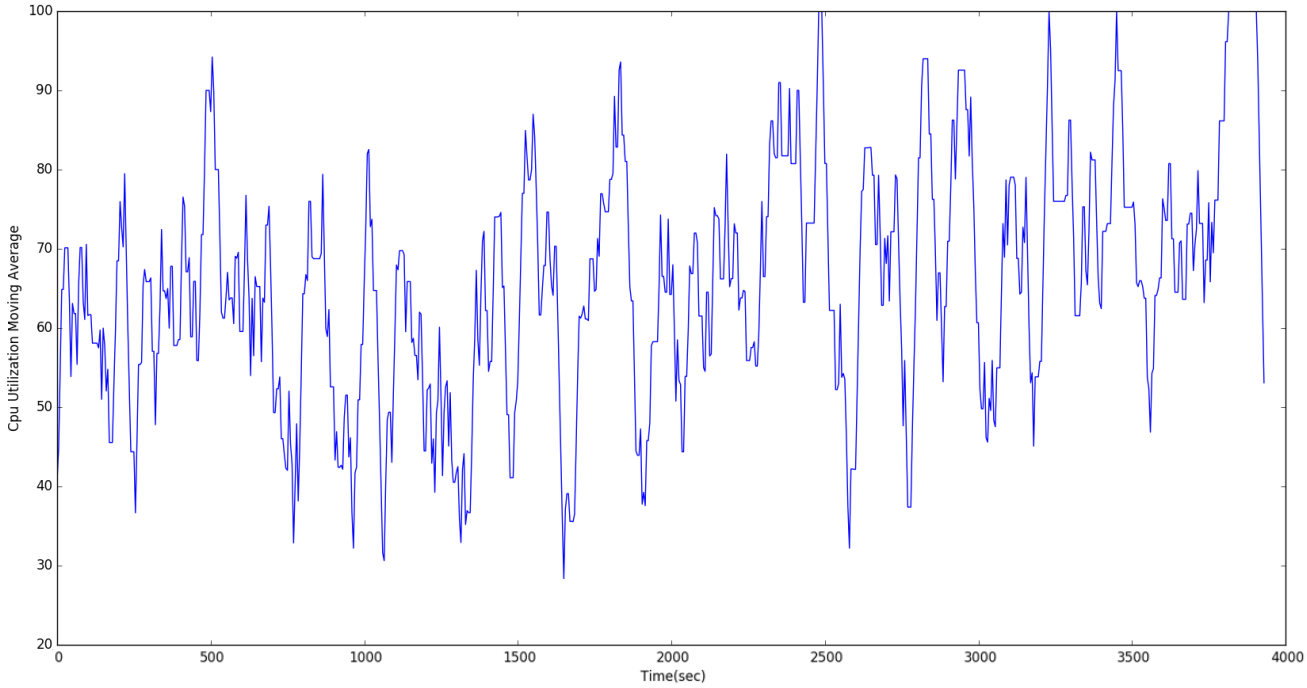
5.1.2.1 Γραφικές παραστάσεις – αποτελέσματα



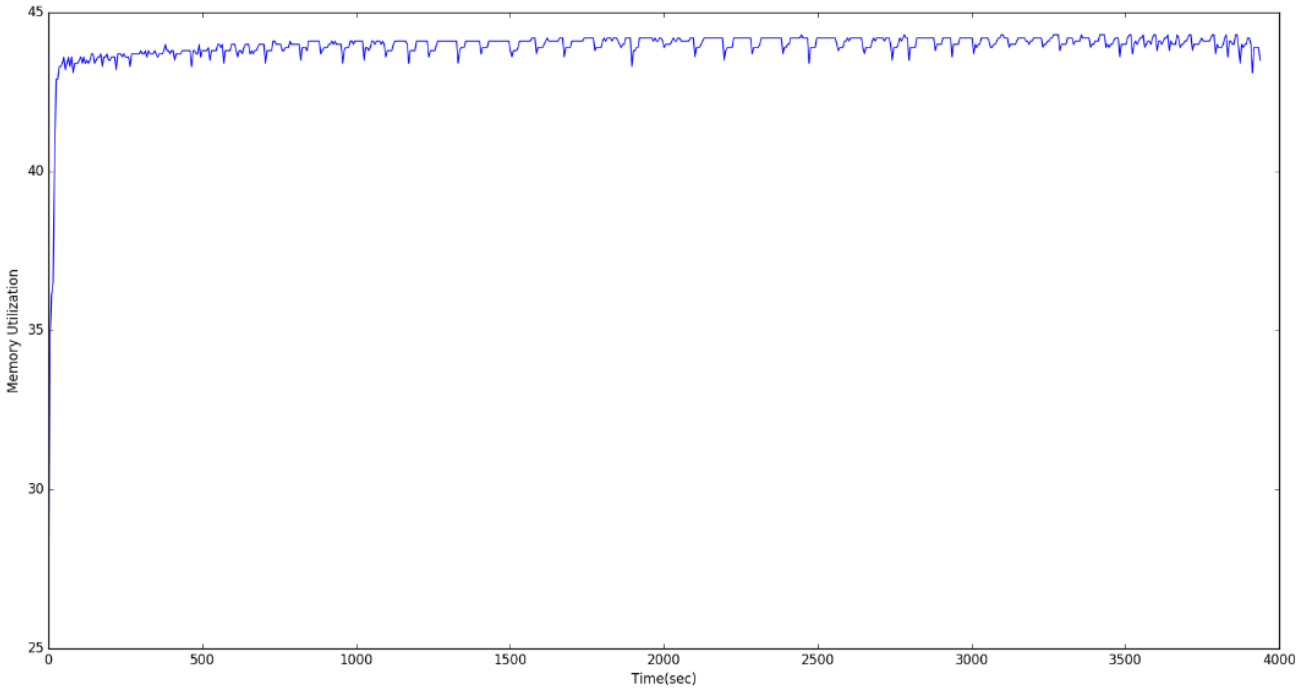
Σχήμα 5.5 RPS (Configure , Real RPS, Throughput) σενάριο 2 πειράματος Α



Σχήμα 5.6 Latency (Min , Max, Average) σενάριο 2 πειράματος Α



Σχήμα 5.7 CPU Moving Average σενάριο 2 πειράματος Α



Σχήμα 5.8 Memory Utilization σενάριο 2 πειράματος Α

5.1.2.2 Παρατηρήσεις

Στο Σχήμα 5.5 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και παρατηρούμε ότι από τα 40 αιτήματα ανά δευτερόλεπτο εξυπηρετούνε περίπου τα 20.

Στο Σχήμα 5.6 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος περίπου 1 δευτερόλεπτο .

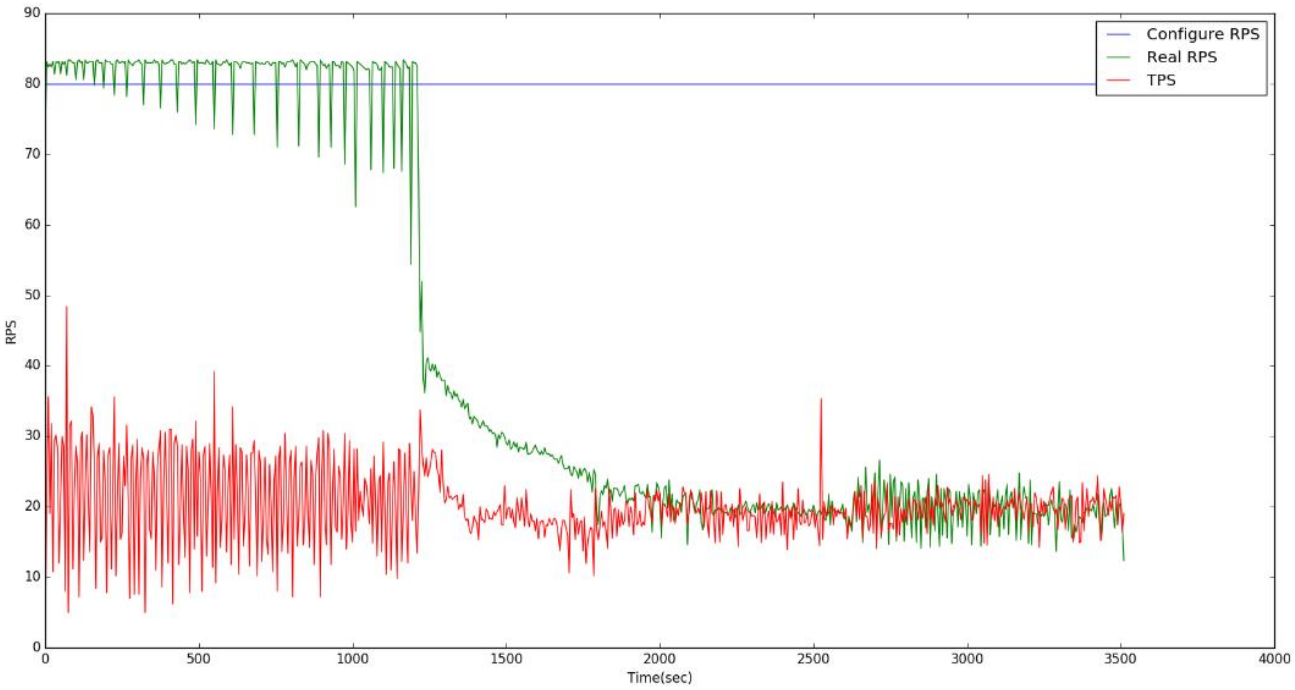
Στο Σχήμα 5.7 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) κυμαίνεται μεταξύ 40% και 100% , δηλαδή υπάρχουν περιπτώσεις που ο επεξεργαστής δουλεύει στο μέγιστο .

5.1.3 Σενάριο 3

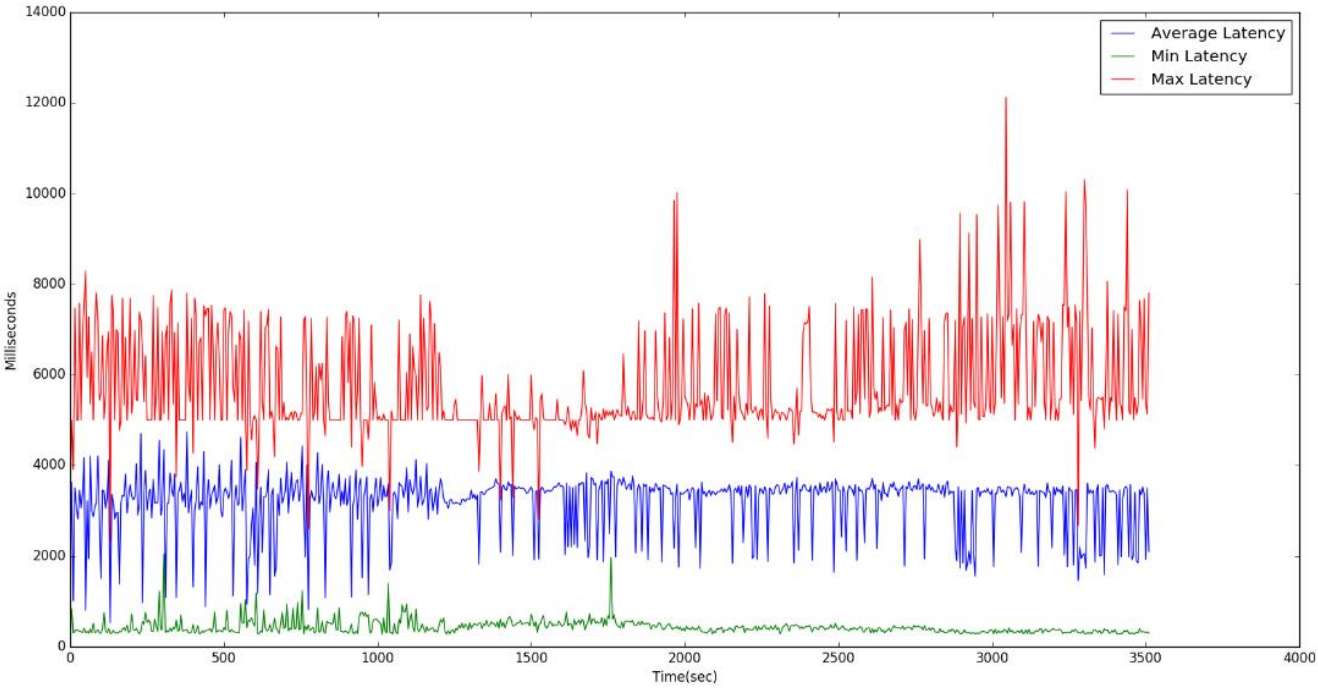
Σενάριο 3	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	40
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	40
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	80

Πίνακας 5.3 διαμόρφωση Σενάριού 3

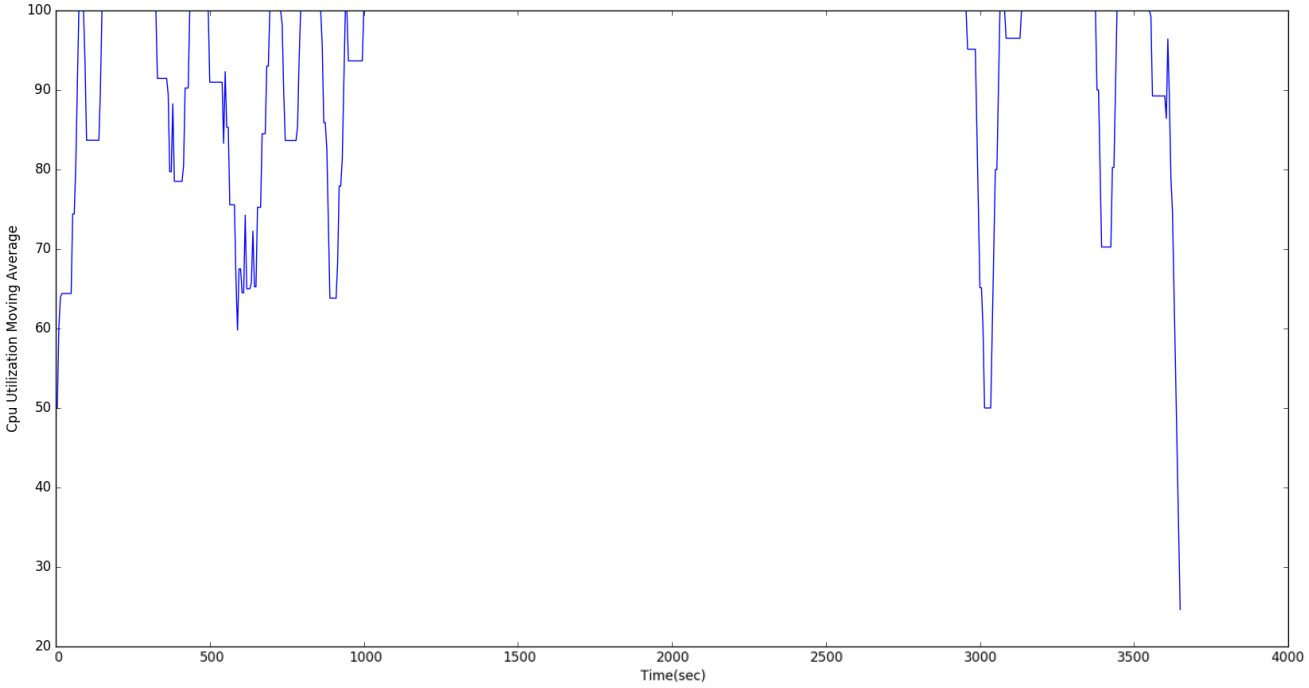
5.1.3.1 Γραφικές παραστάσεις - αποτελέσματα



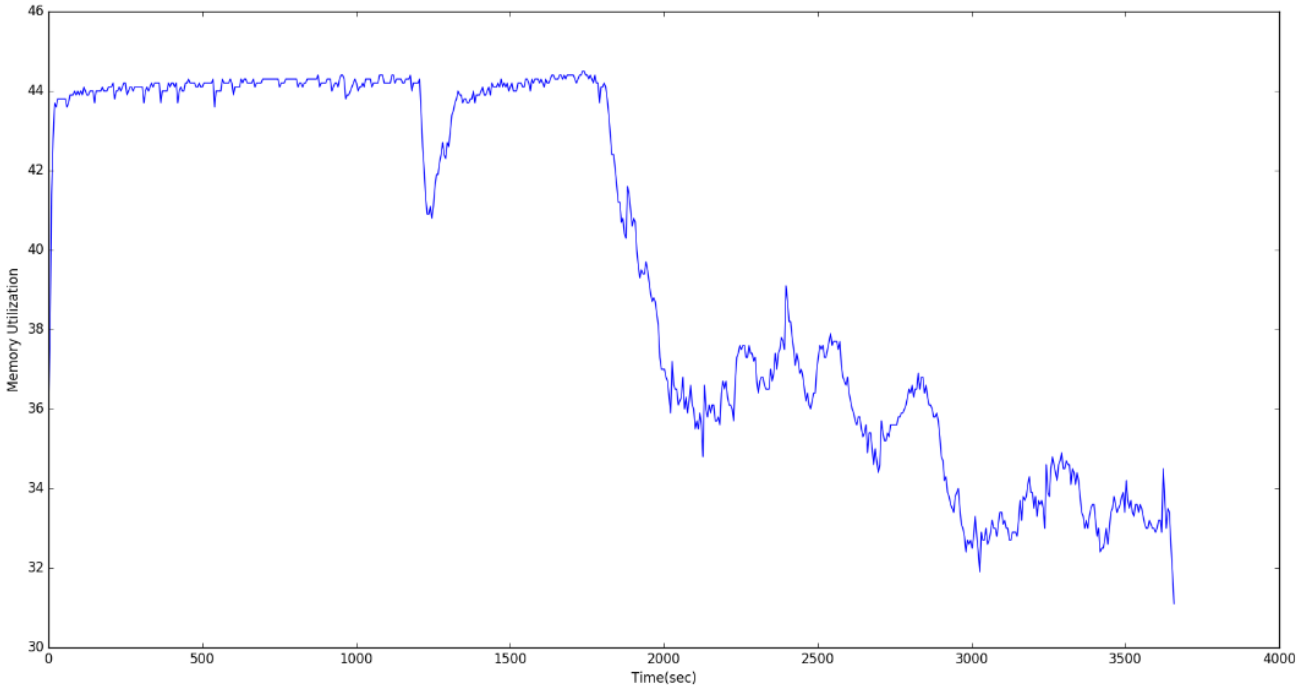
Σχήμα 5.9 RPS (Configure , Real RPS, Throughput) σενάριο 3 πειράματος Α



Σχήμα 5.10 Latency (Min , Max, Average) σενάριο 3 πειράματος Α



Σχήμα 5.11 CPU Moving Average σενάριο 3 πειράματος Α



Σχήμα 5.12 Memory Utilization σενάριο 3 πειράματος Α

5.1.3.2 Παρατηρήσεις

Στο Σχήμα 5.9 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και παρατηρούμε ότι από τα 80 αιτήματα ανά δευτερόλεπτο εξυπηρετούνε περίπου τα 20 και από ένα διάστημα και μετά ο workload δεν στέλνει καν τα request που είναι ρυθμισμένος να στέλνει.

Στο Σχήμα 5.10 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος περίπου 3.5 δευτερόλεπτα .

Στο Σχήμα 5.11 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) σχεδόν καθ' όλη την διάρκεια του πειράματος ήταν 100%

5.1.4 Παρατηρήσεις Πειράματος Α

Παρατηρώντας και τα 3 σενάρια του πειράματος Α παρατηρούμε ότι όσο αυξάνονται οι ταυτόχρονοι χρήστες (δηλαδή τα αιτήματα ανά δευτερόλεπτο) αυξάνετε το latency και το CPU utilization . Επίσης παρατηρούμε ότι ο μέγιστος αριθμός ταυτόχρονων χρηστών που μπορεί να εξυπηρετήσει η εφαρμογή μας στο πείραμα Α είναι 10 ταυτόχρονοι χρήστες δηλαδή 20 Request Per Second (αιτήματα ανά δευτερόλεπτο) . Όσο αφορά το latency όταν το average latency είναι κάτω από μισό δευτερόλεπτο εξυπηρετούνται όλα τα αιτήματα όταν ανεβεί πάνω από το 0.5 τότε δεν εξυπηρετούνται όλα τα αιτήματα . Τα ο ίδιο ισχύει και για το κεντρικό επεξεργαστή όταν η χρήση του (utilization) είναι 100% για συνεχόμενο χρονικό διάστημα τότε δεν μπορούν να εξυπηρετηθούν όλα τα αιτήματα .

5.2 Εκτέλεση και αποτελέσματα Φάσης Β

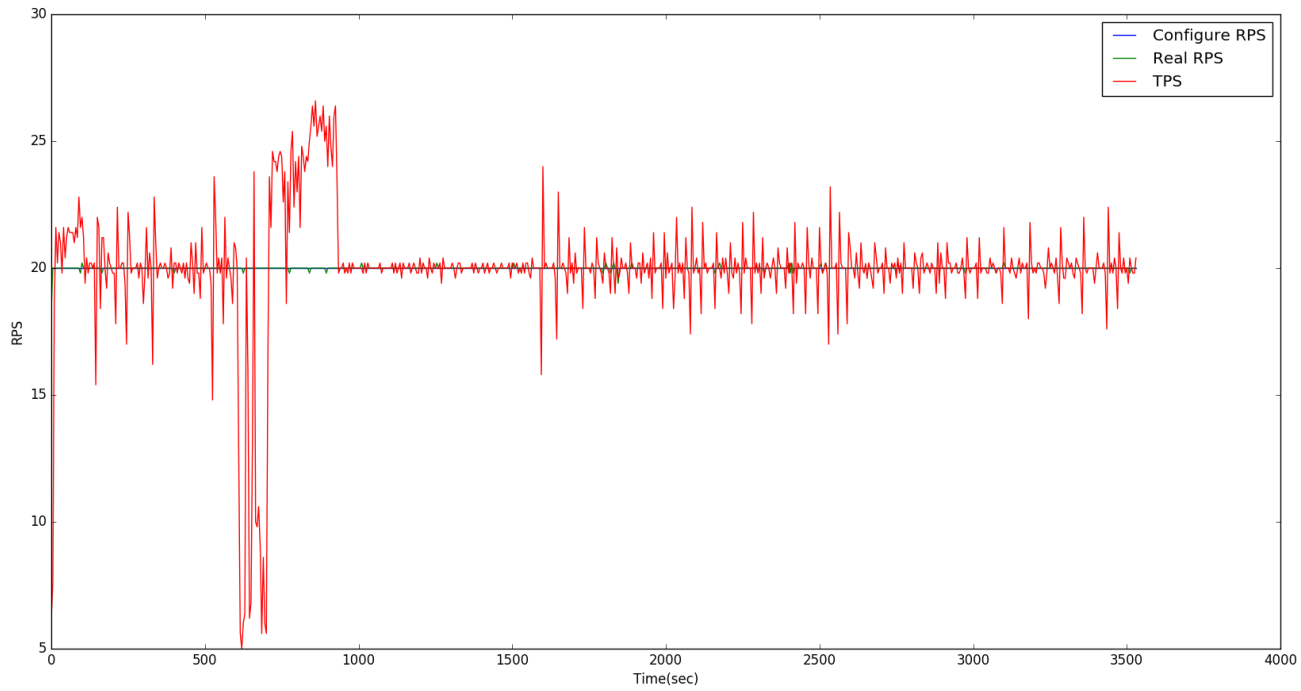
Ο τρόπος λειτουργείας και η λογική του πειράματος Β περιγράφονται στην υποενότητα 4.6.3.2 (Τρόπος λειτουργίας – λογική πειράματος Β) και η βασική δομή πειραματικού Framework Πειράματος Β φαίνεται στο σχήμα 4.22

5.2.1 Σενάριο 1

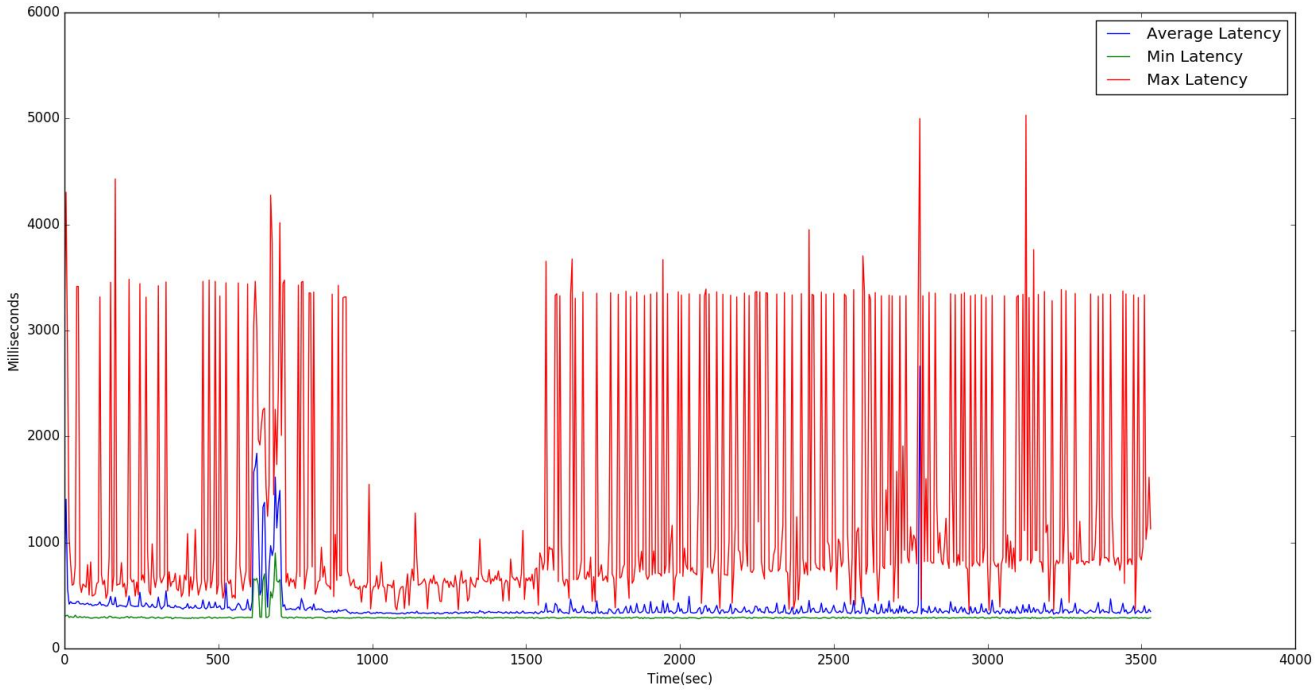
Σενάριο 1	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	10
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	10
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	20

Πίνακας 5.4 διαμόρφωση Σενάριού 1

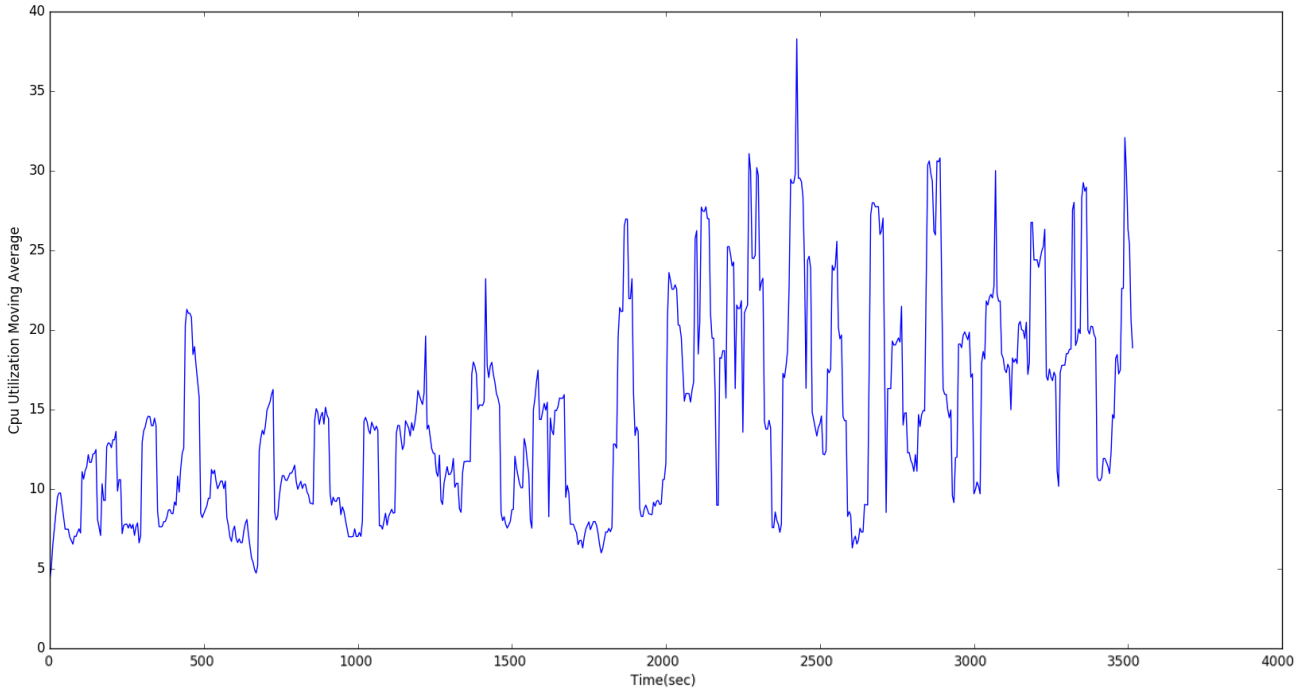
5.2.1.1 Γραφικές παραστάσεις - αποτελέσματα



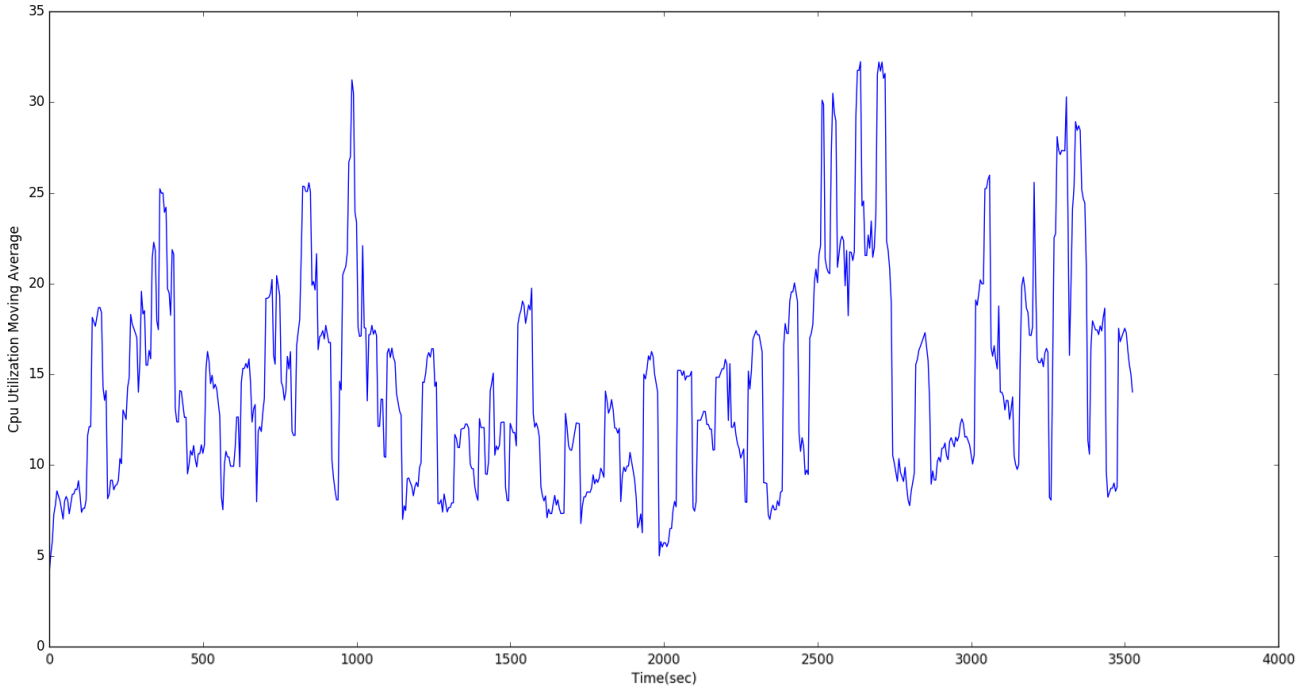
Σχήμα 5.13 RPS (Configure, Real RPS, Throughput) σενάριο 1 πειράματος Β



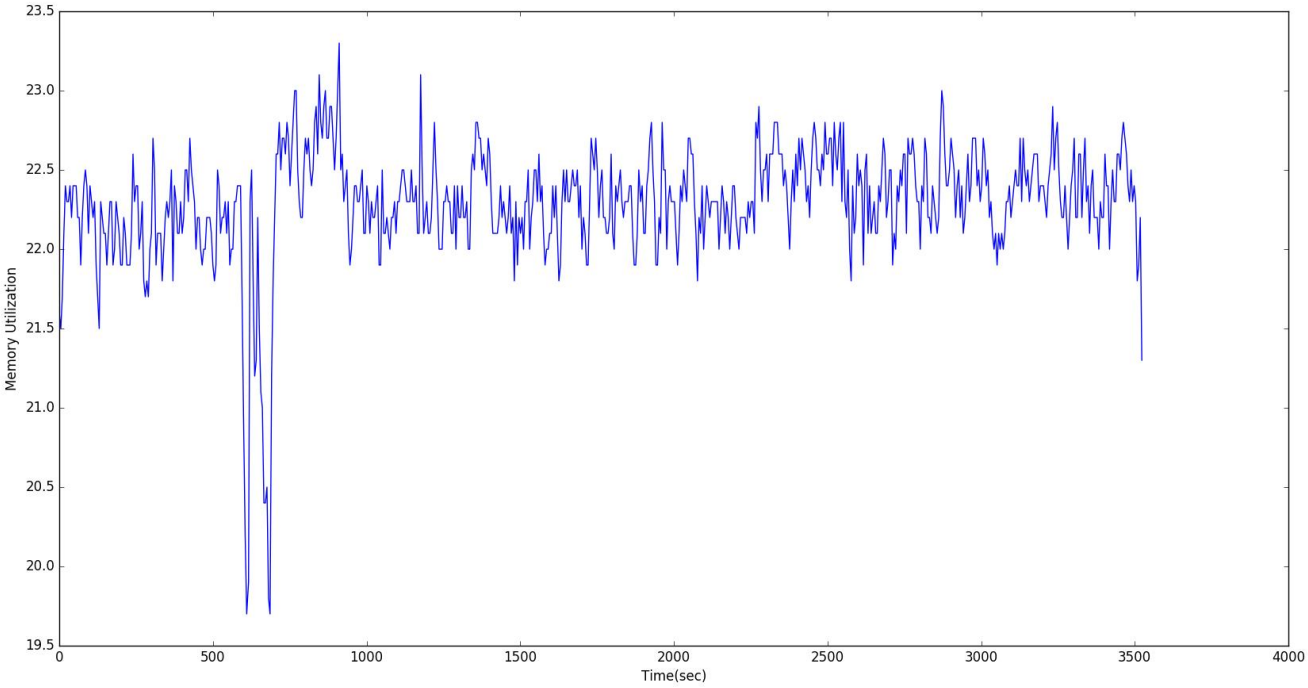
Σχήμα 5.14 Latency (Min , Max, Average) σενάριο 1 πειράματος Β



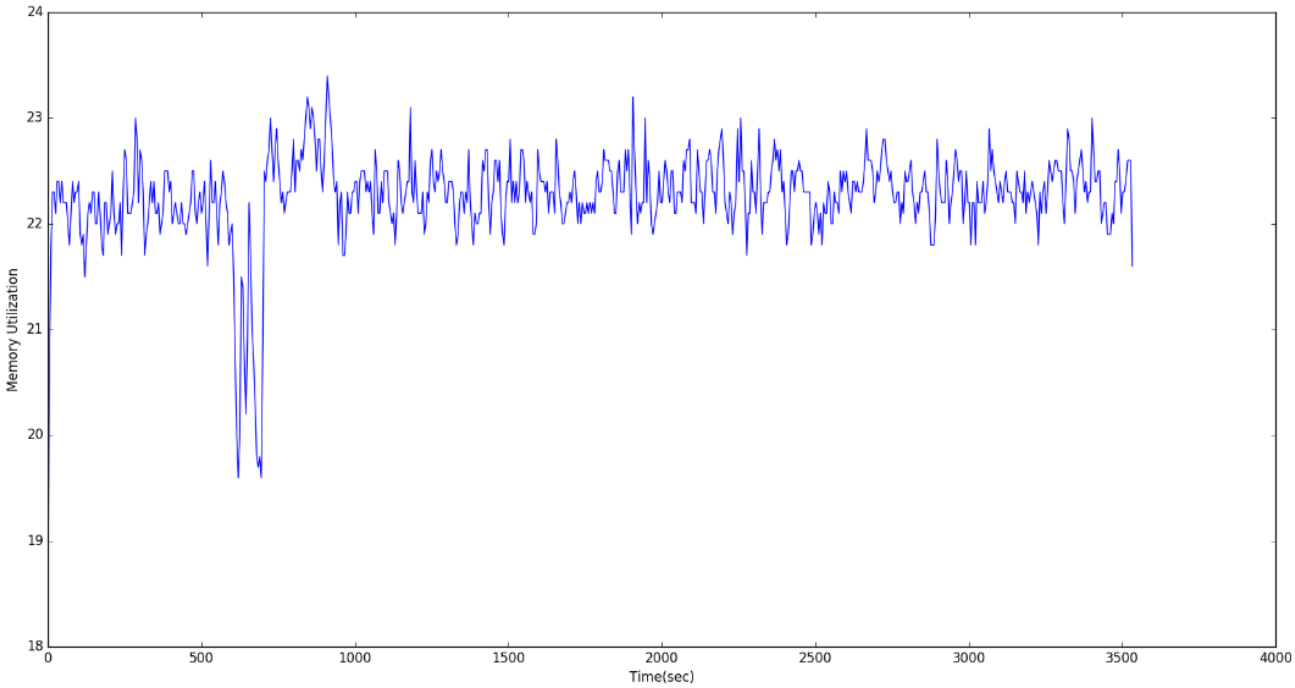
Σχήμα 5.15 CPU Moving VM 1 Average σενάριο 1 πειράματος Β



Σχήμα 5.16 CPU Moving VM 2 Average σενάριο 1 πειράματος Β



Σχήμα 5.17 Memory Utilization VM1 σενάριο 1 πειράματος B



Σχήμα 5.18 Memory Utilization VM2 σενάριο 1 πειράματος B

5.2.1.2 Παρατηρήσεις

Στο Σχήμα 5.13 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και παρατηρούμε ότι αυτά τα 2 σε γενικές γραμμές είναι ίσα δηλαδή εξυπηρετούνται όλα τα αιτήματα .

Στο Σχήμα 5.14 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος είναι κάτω από μισό (0.5) δευτερόλεπτο .

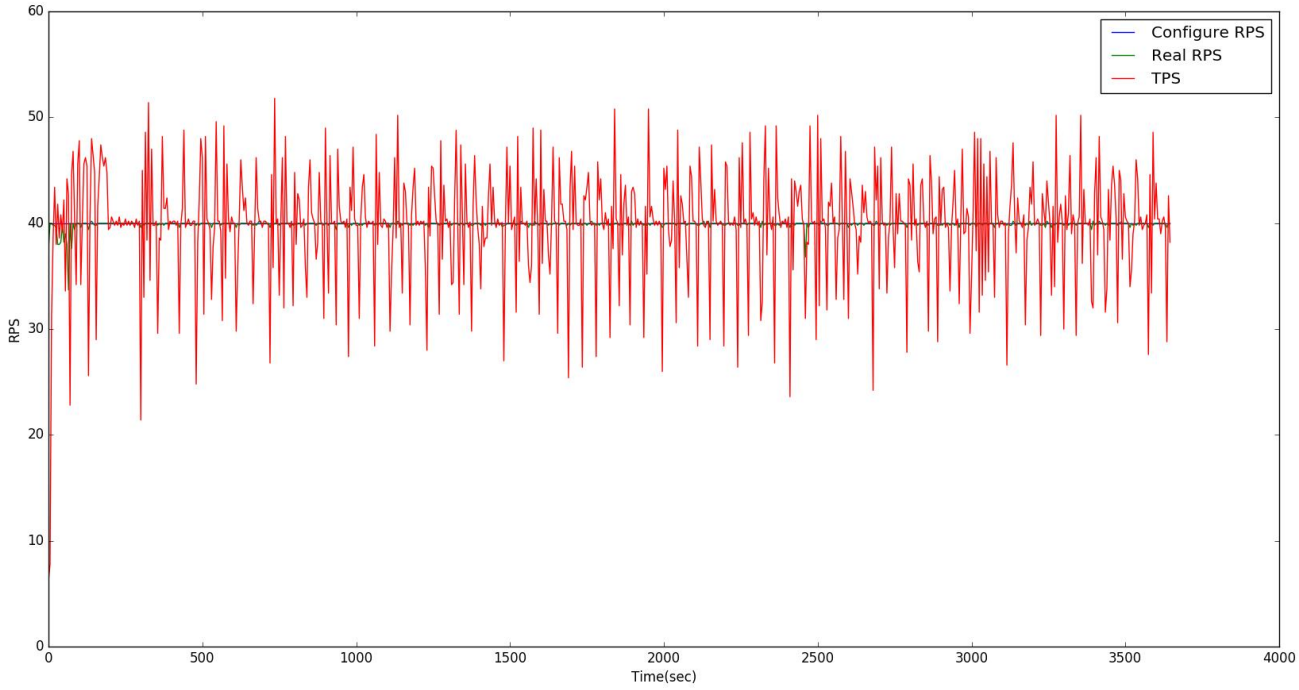
Λαμβάνοντας υπόψη το Σχήμα 5.15 και το Σχήμα 5.16 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) και των 2 εικονικών μηχανών VM κυμαίνεται μεταξύ 5% και 30% .

5.2.2 Σενάριο 2

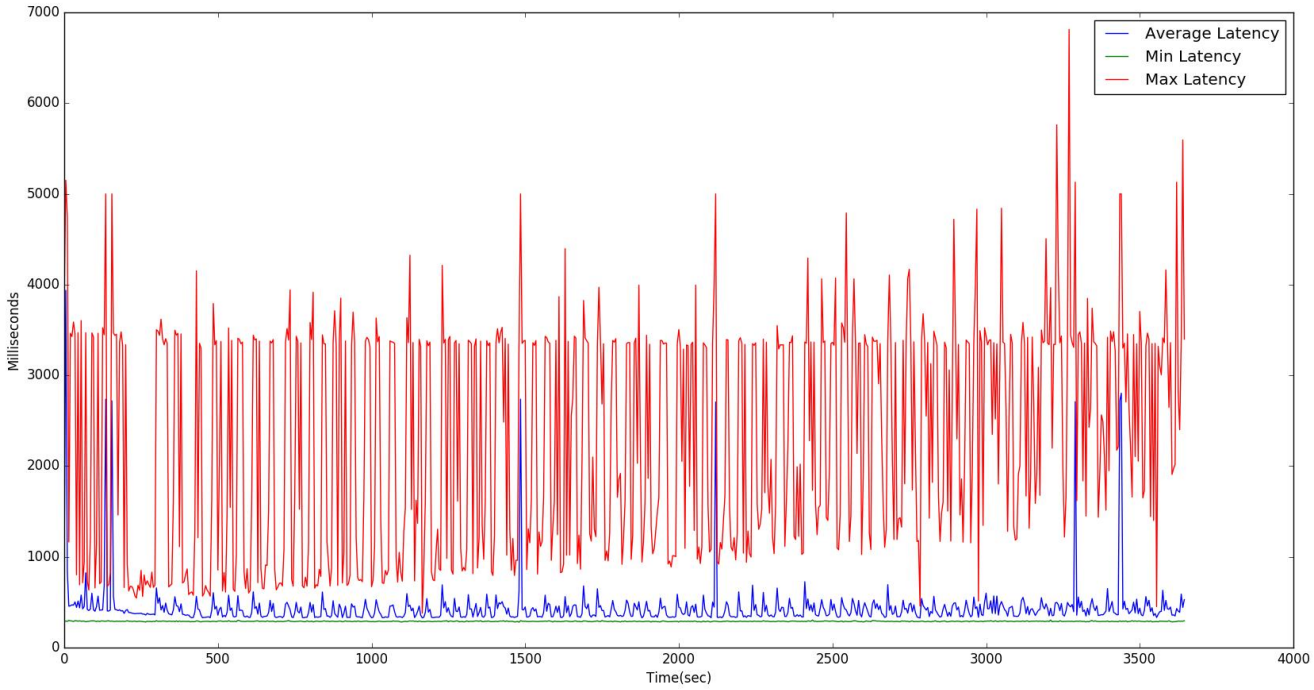
Σενάριο 2	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	20
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	20
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	40

Πίνακας 5.5 διαμόρφωση Σενάριού 2

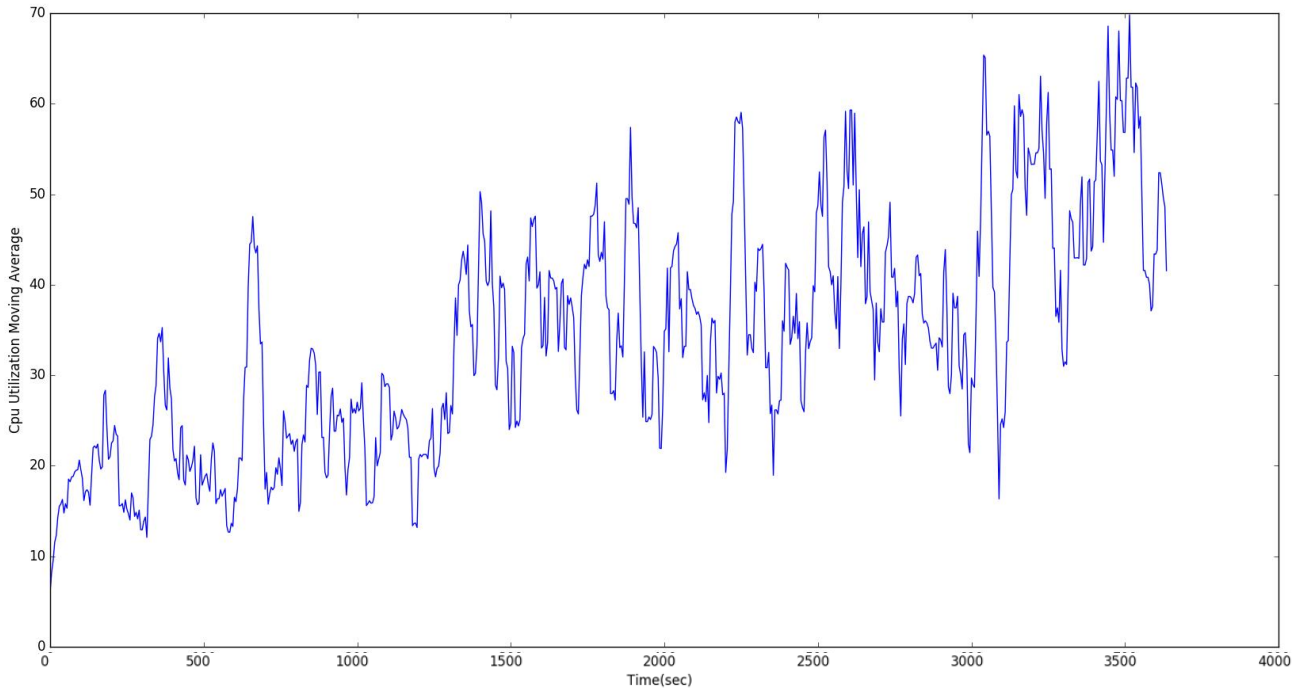
5.2.2.1 Γραφικές παραστάσεις - αποτελέσματα



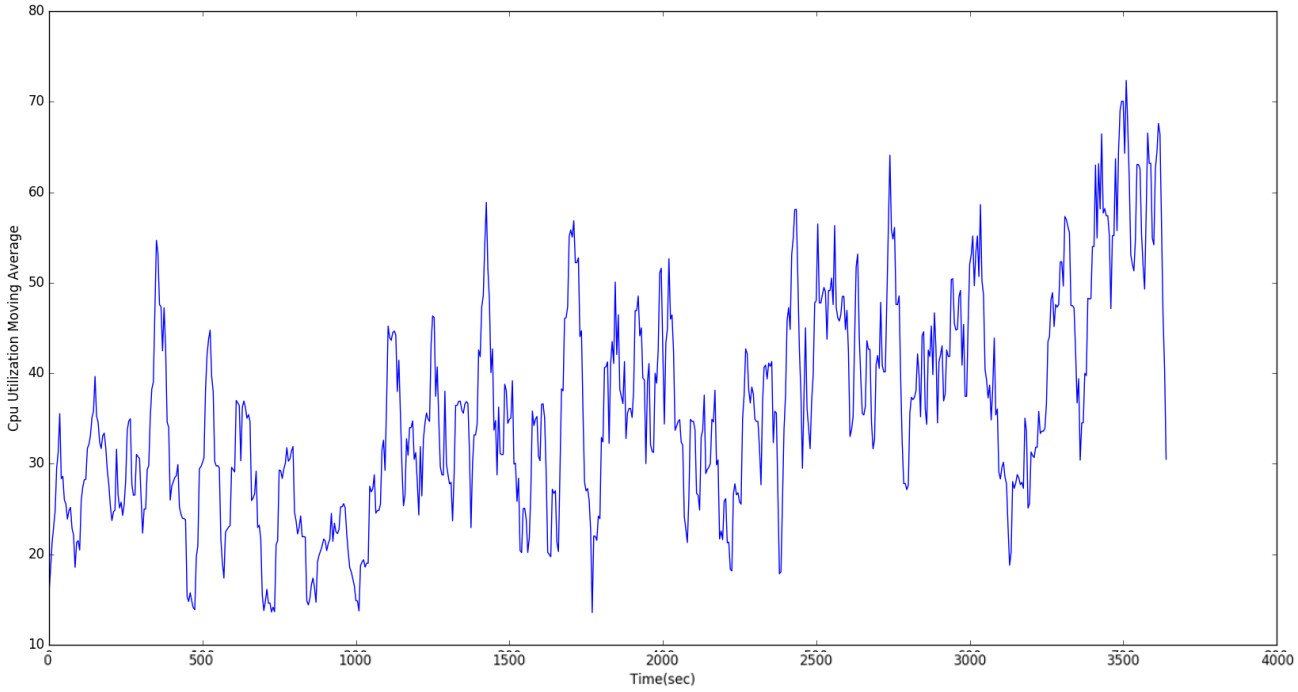
Σχήμα 5.19 RPS (Configure , Real RPS, Throughput) σενάριο 2 πειράματος Β



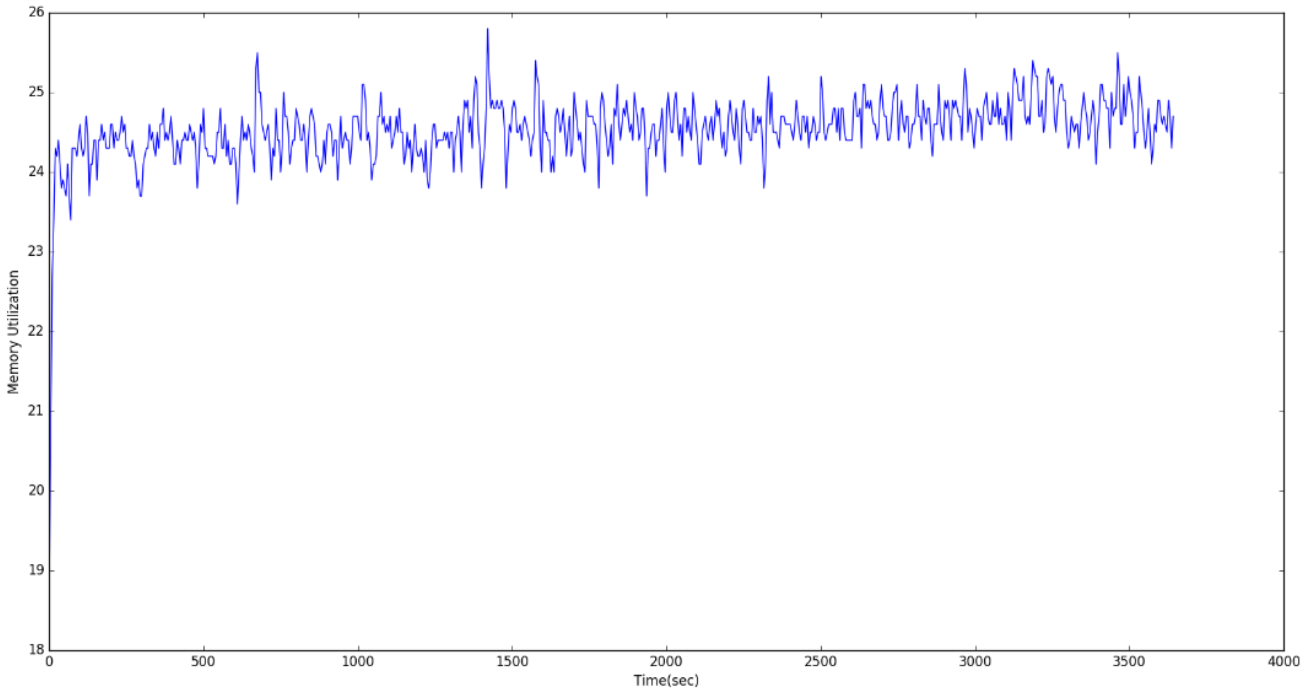
Σχήμα 5.20 Latency (Min , Max, Average) πειράματος 2 Φάση Β



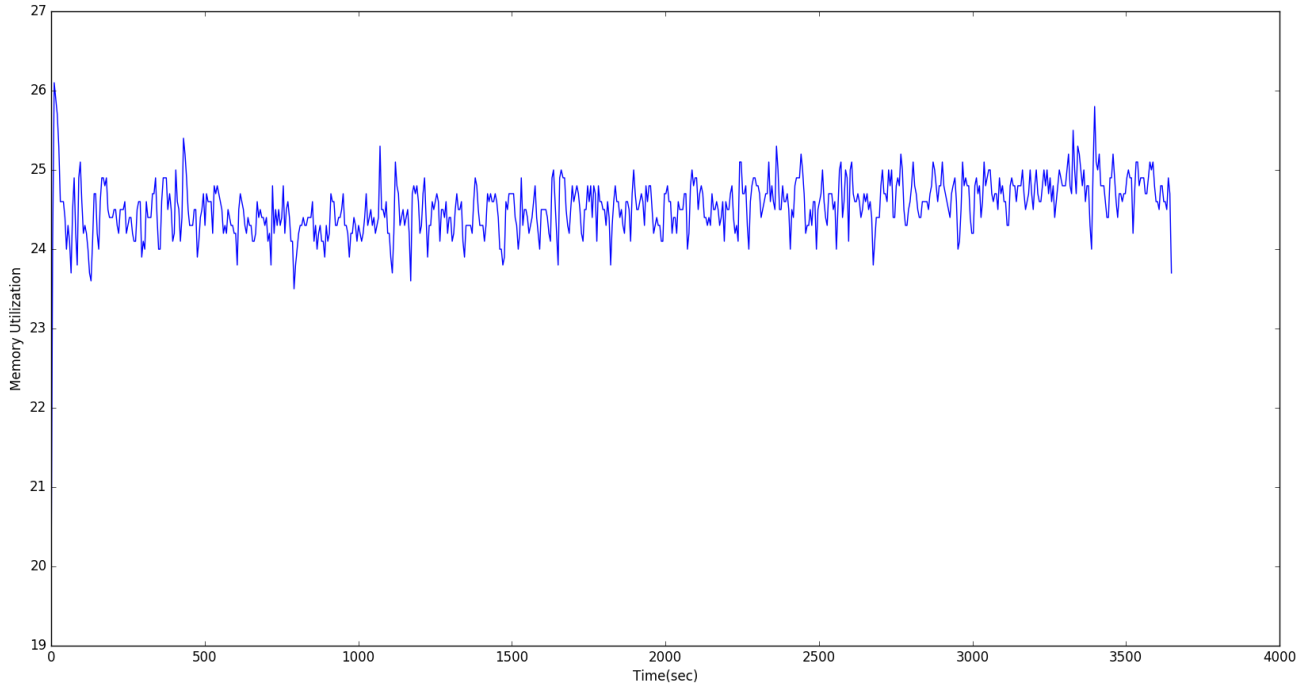
Σχήμα 5.21 CPU Moving VM 1 Average σενάριο 2 πειράματος Β



Σχήμα 5.22 CPU Moving VM 2 Average σενάριο 2 πειράματος Β



Σχήμα 5.23 Memory Utilization VM1 σενάριο 2 πειράματος B



Σχήμα 5.24 Memory Utilization VM2 σενάριο 2 πειράματος B

5.2.2.2 Παρατηρήσεις

Στο Σχήμα 5.19 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και παρατηρούμε ότι αυτά τα 2 σε γενικές γραμμές είναι ίσα δηλαδή εξυπηρετούνται όλα τα αιτήματα .

Στο Σχήμα 5.20 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος είναι κάτω από μισό (0.5) δευτερόλεπτο .

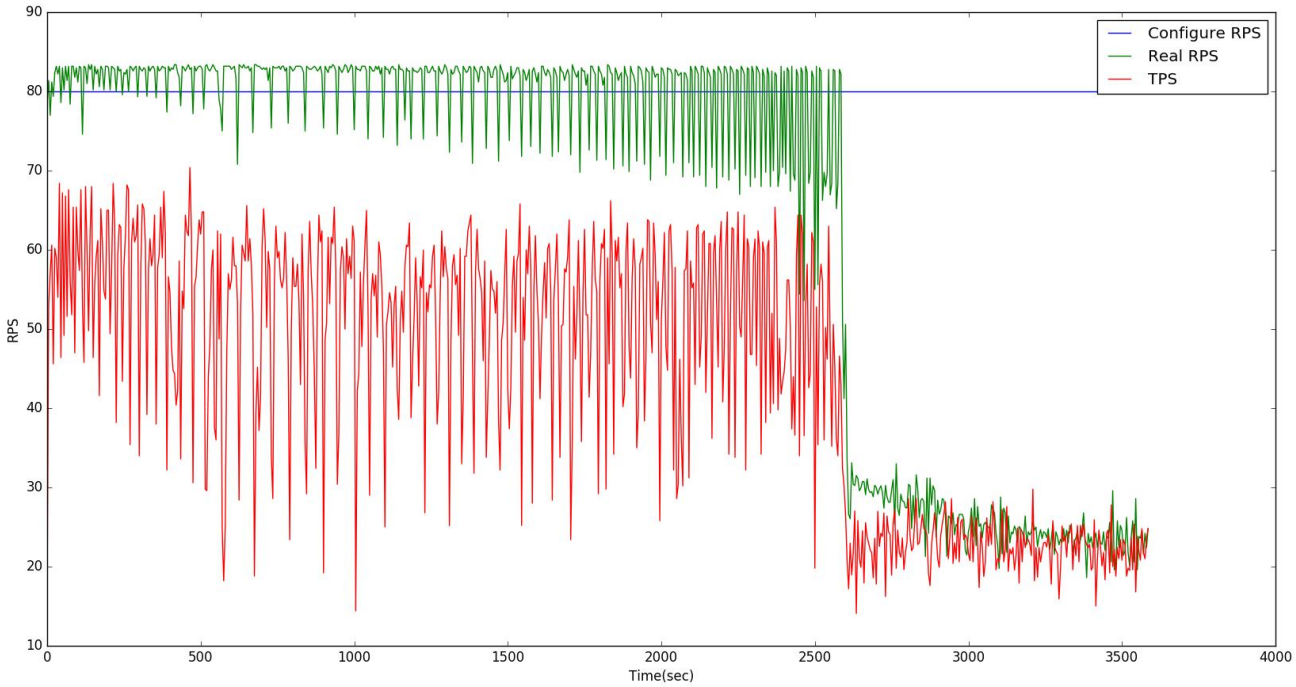
Λαμβάνοντας υπόψη το Σχήμα 5.21 και το Σχήμα 5.22 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) και των δύο εικονικών μηχανών (VM) κυμαίνεται μεταξύ 20% και 70% .

5.2.3 Σενάριο 3

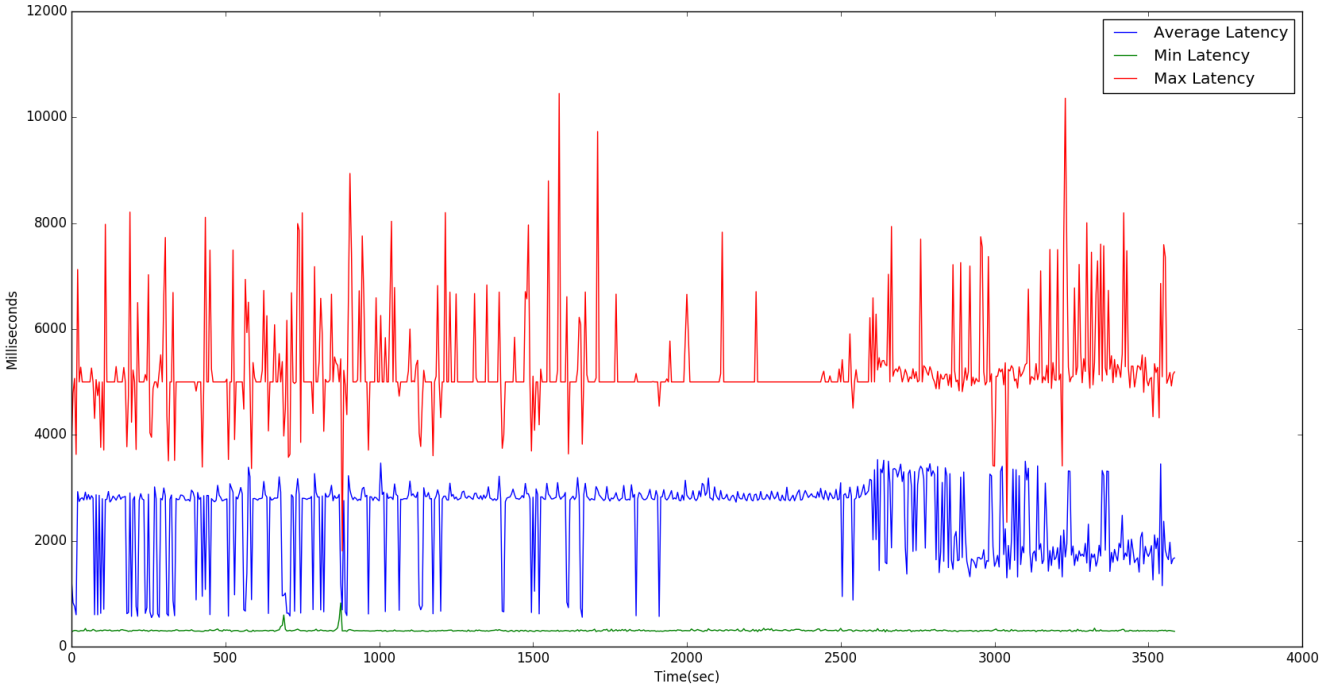
Σενάριο 3	
Διάρκεια	1 Ώρα
Ταυτόχρονοι χρήστες	40
Ανάλυση Βίντεο	360p
Ρυθμίσεις Workload Generator	
Αριθμός νημάτων	40
Αριθμός αιτήσεων ανά δευτερόλεπτο για κάθε νήμα	2
Συνολικός αριθμός αιτήσεων ανά δευτερόλεπτο	80

Πίνακας 5.6 διαμόρφωση Σενάριού 3

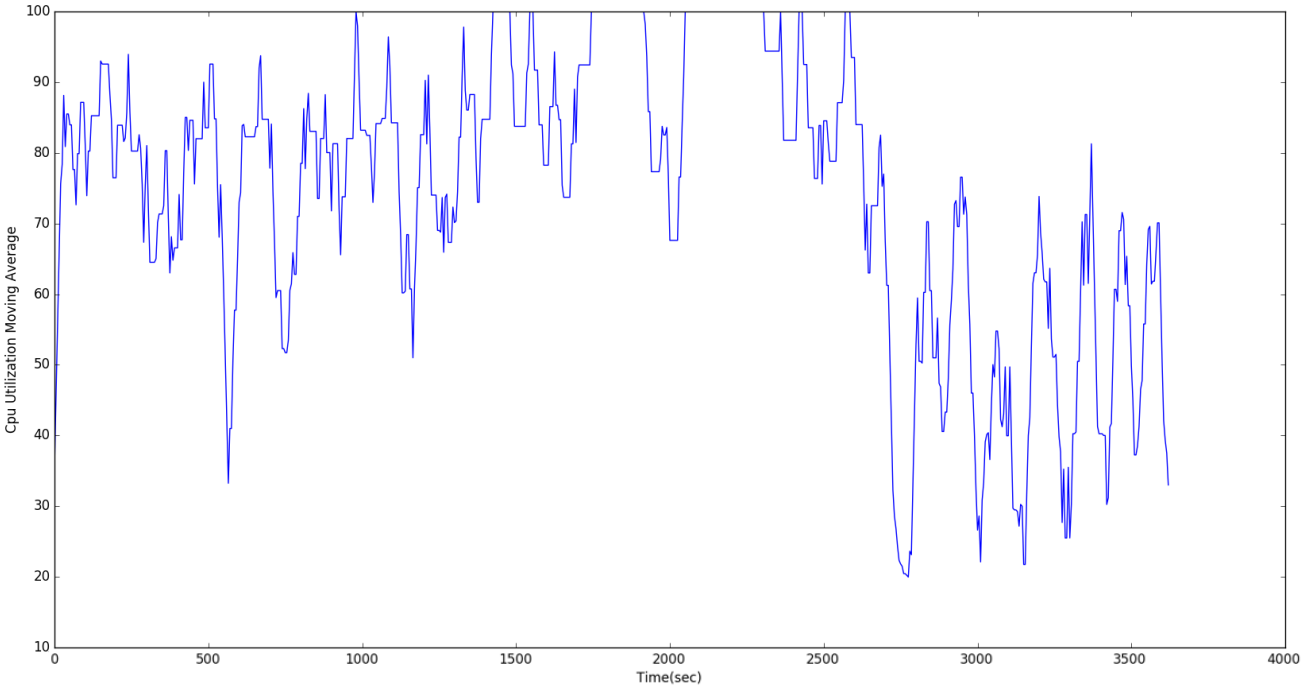
5.2.3.1 Γραφικές παραστάσεις - αποτελέσματα



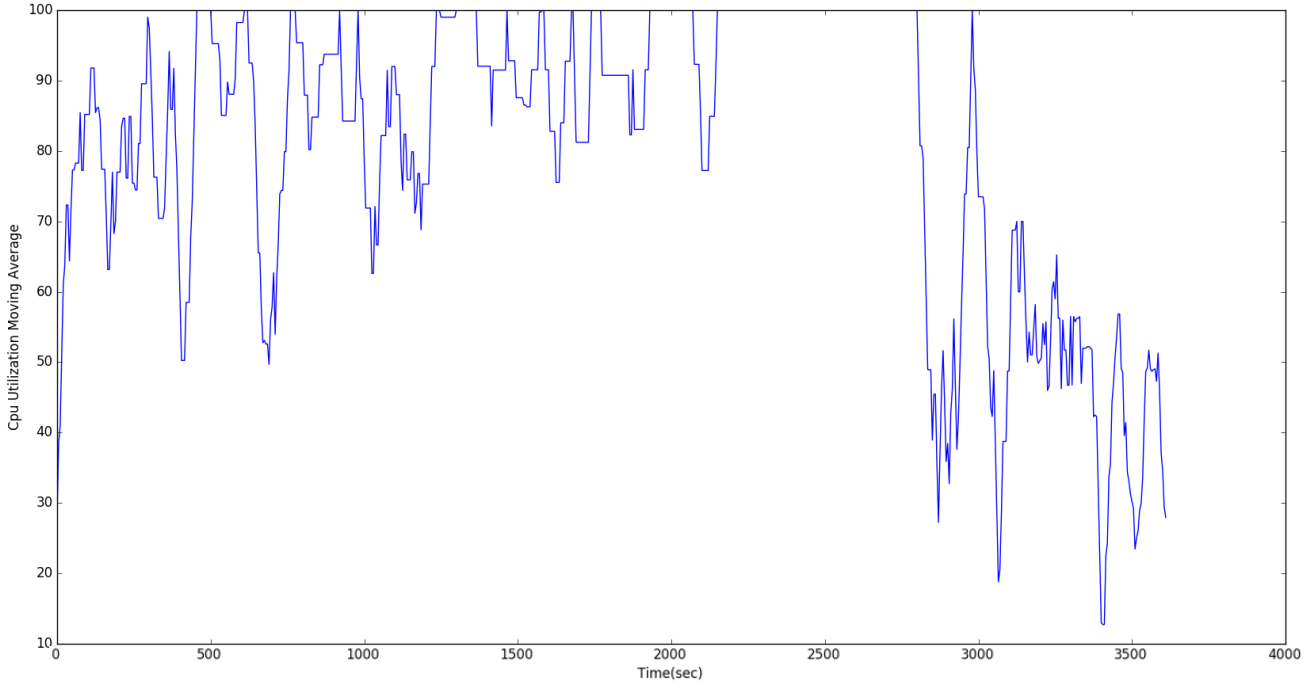
Σχήμα 5.25 RPS (Configure , Real RPS, Throughput) σενάριο 3 πειράματος B



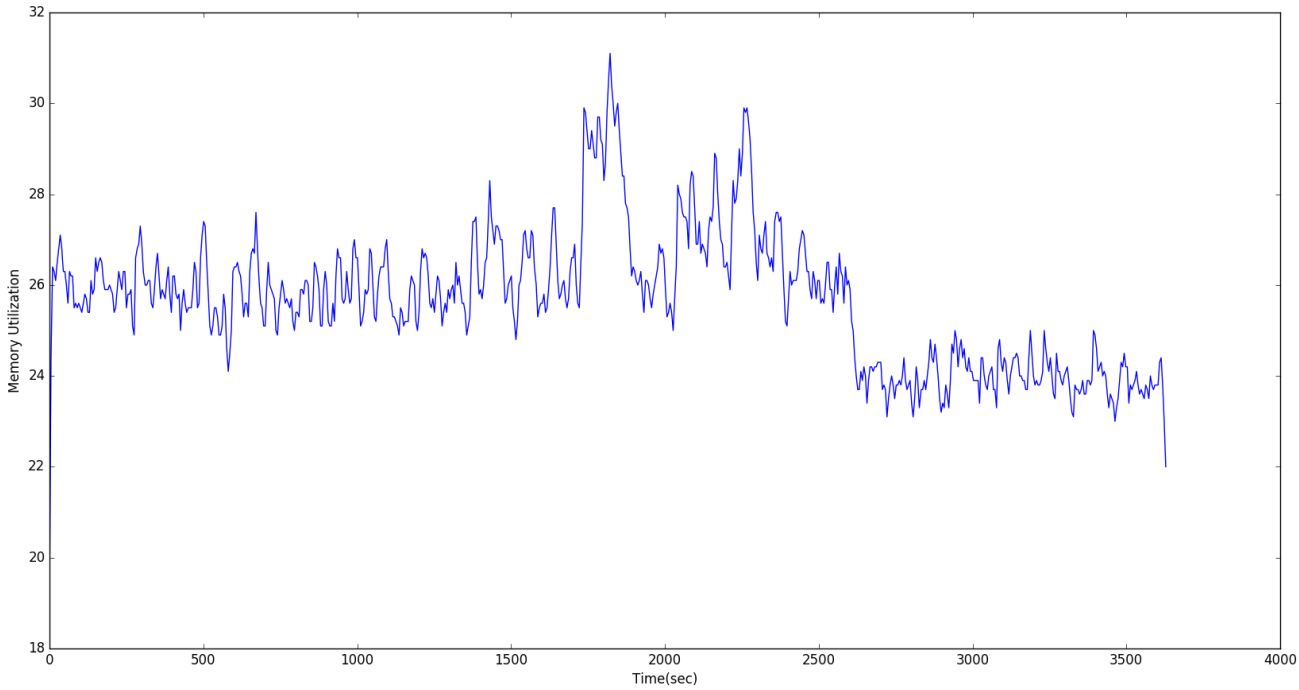
Σχήμα 5.26 Latency (Min , Max, Average) σενάριο 3 πειράματος B



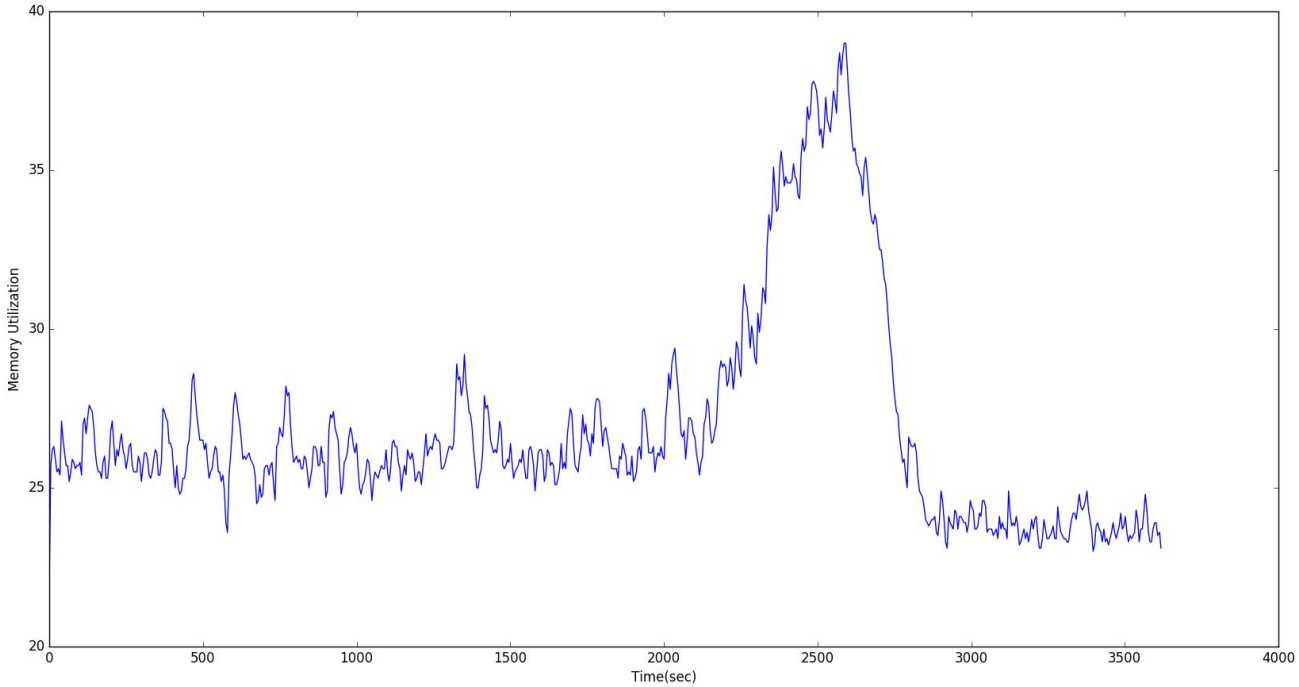
Σχήμα 5.27 CPU Moving VM 1 Average σενάριο 3 πειράματος Β



Σχήμα 5.28 CPU Moving VM 2 Average σενάριο 3 πειράματος Β



Σχήμα 5.29 Memory Utilization VM1 σενάριο 3 πειράματος B



Σχήμα 5.30 Memory Utilization VM2 σενάριο 3 πειράματος B

5.2.3.2 Παρατηρήσεις

Στο Σχήμα 5.25 βλέπουμε τα Request Per Second (αιτήματα ανά δευτερόλεπτο) και το throughput Per Second και και παρατηρούμε ότι από τα 80 αιτήματα ανά δευτερόλεπτο εξυπηρετούνε περίπου τα 50 και από ένα διάστημα και μετά ο workload δεν στέλνει καν τα request που είναι ρυθμισμένος να στέλνει με αποτέλεσμα να μειώνετε και throughput Per Second .

Στο Σχήμα 5.26 βλέπουμε το Min , Max, Average Latency ανά δευτερόλεπτο και παρατηρούμε ότι το average latency κατά διάρκεια του πειράματος περίπου 2.7 δευτερόλεπτα .

Λαμβάνοντας υπόψη το Σχήμα 5.27 και το Σχήμα 5.28 παρατηρούμε ότι η χρήση της κεντρικής μονάδας του επεξεργαστή (CPU) και των δύο εικονικών μηχανών (VM) για μεγάλα χρονικά διαστήματα είναι στο 100%.

5.2.4 Παρατηρήσεις Πειράματος Β

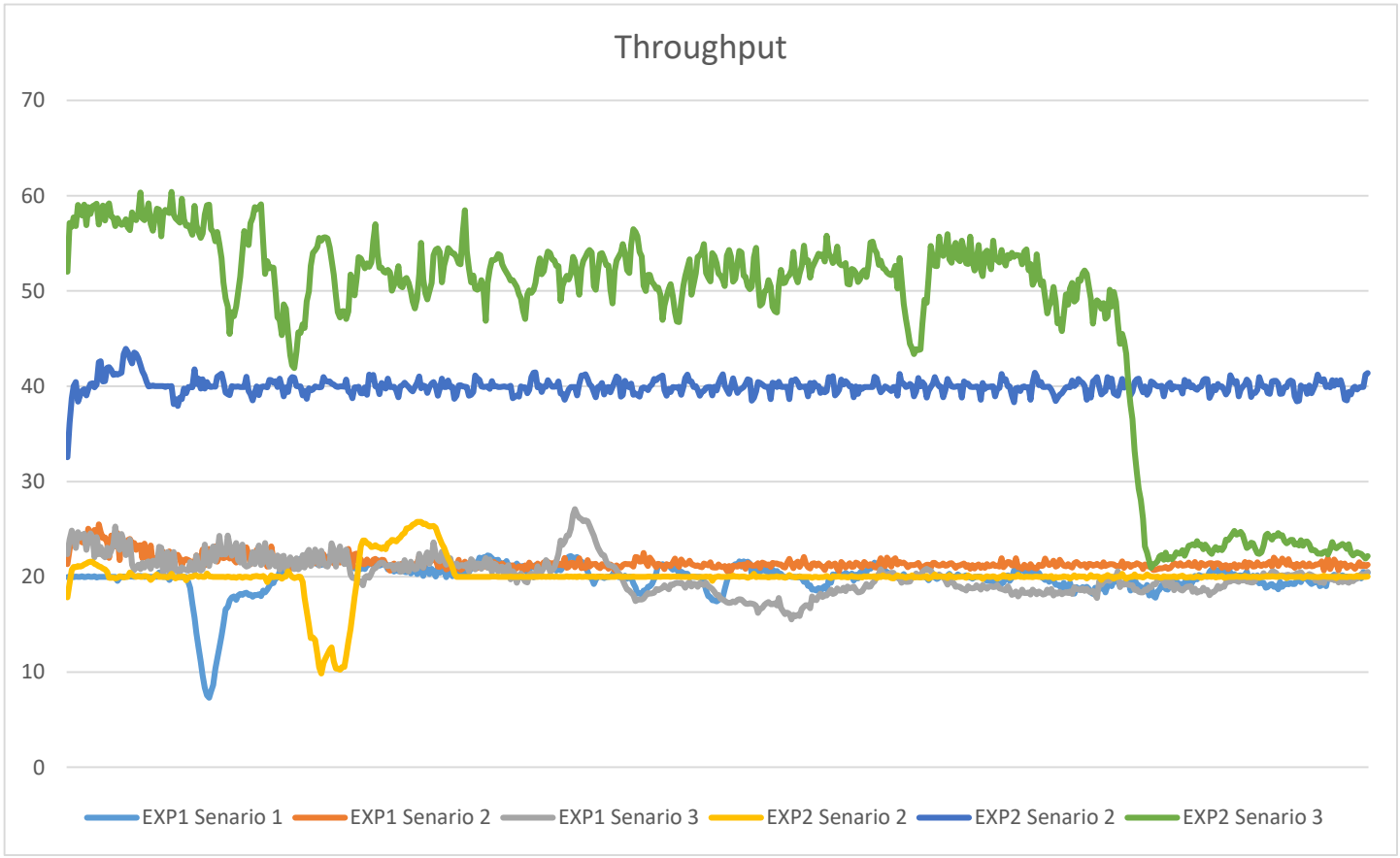
Παρατηρώντας και τα 3 σενάρια του πειράματος Α παρατηρούμε ότι όσο αυξάνονται οι ταυτόχρονοι χρήστες (δηλαδή τα αιτήματα ανά δευτερόλεπτο) αυξάνετε το latency και το CPU utilization . Επίσης παρατηρούμε ότι ο μέγιστος αριθμός ταυτόχρονων χρηστών που μπορεί να εξυπηρετήσει η εφαρμογή μας στο πείραμα Α είναι περίπου 25 ταυτόχρονοι χρήστες δηλαδή 50 Request Per Second (αιτήματα ανά δευτερόλεπτο) .

Όσο αφορά το latency όταν το average latency είναι κάτω από μισό δευτερόλεπτο εξυπηρετούνται όλα τα αιτήματα όταν ανεβεί πάνω από το 0.5 τότε δεν εξυπηρετούνται όλα τα αιτήματα . Τα ο ίδιο ισχύει και για το κεντρικό επεξεργαστή όταν η χρήση του (utilization) είναι 100% για συνεχόμενο χρονικό διάστημα τότε δεν μπορούν να εξυπηρετηθούν όλα τα αιτήματα . Επιπλέων παρατηρούμε ότι τα αποτελέσματα που αφορούν τις 2 εικονικές μηχανές είναι ανάλογα δηλαδή μοιράζονται το φόρτο εργασίας.

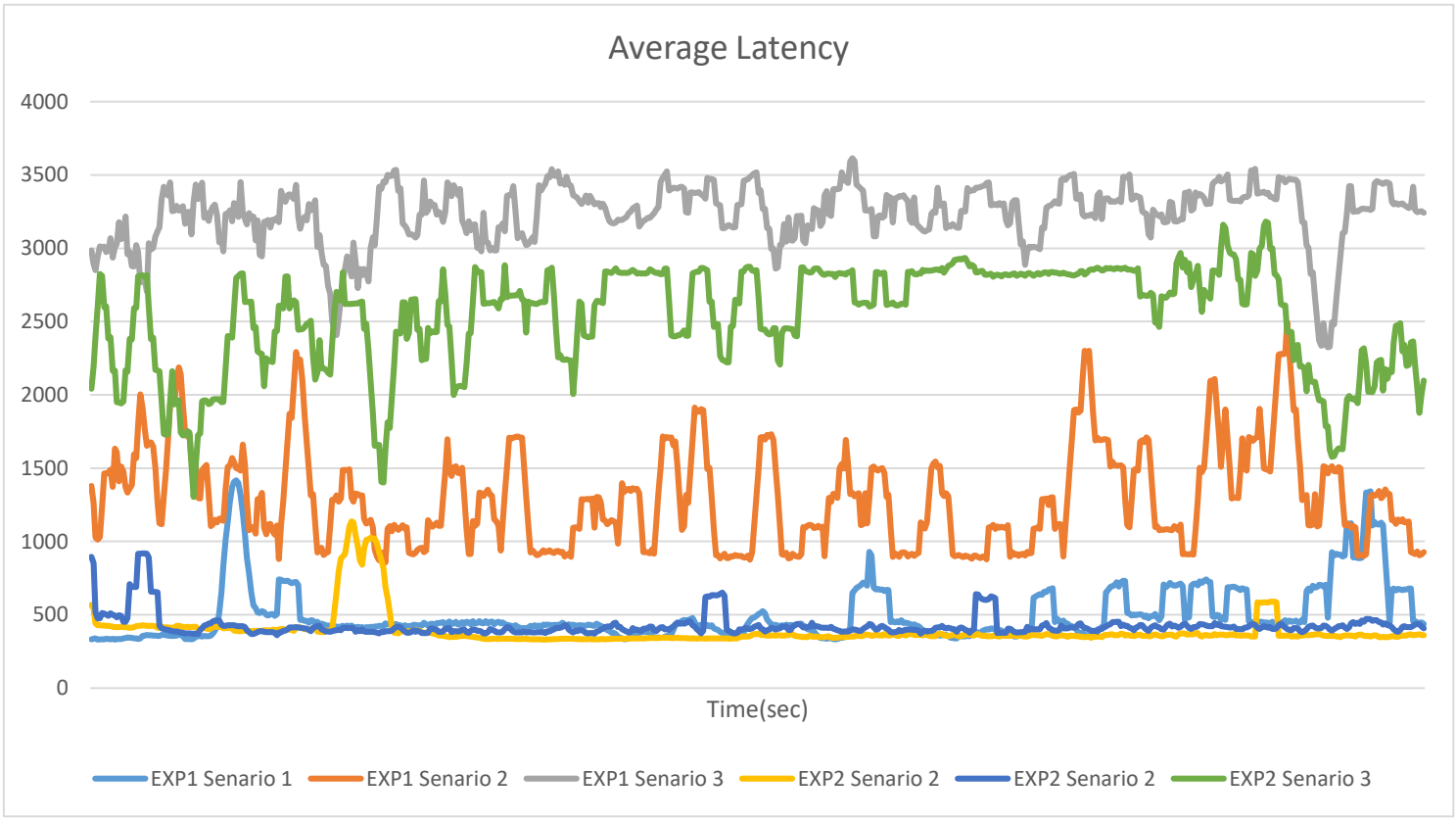
Κεφάλαιο 6

Συμπεράσματα και Μελλοντική Εργασία

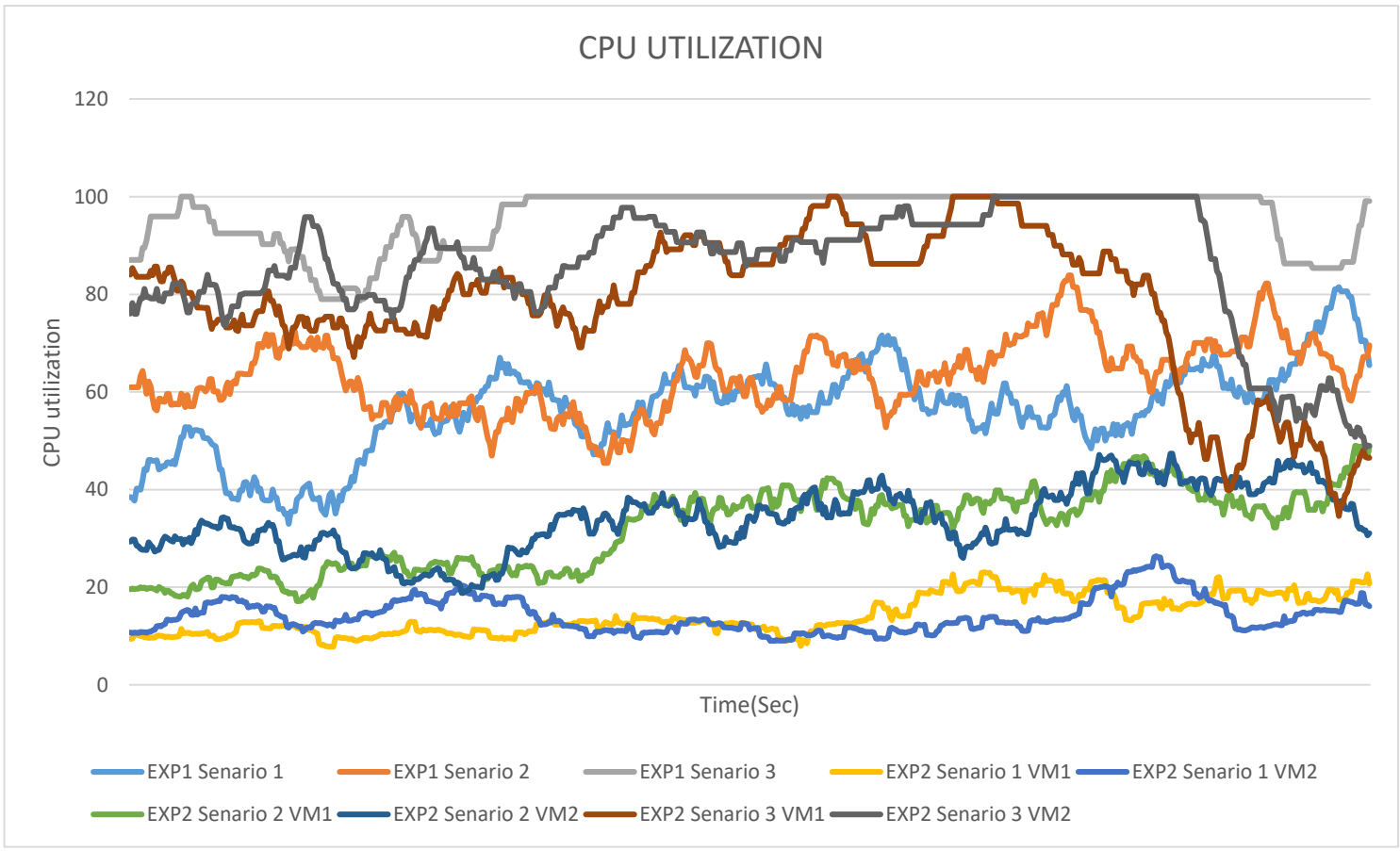
6.1 Συμπεράσματα



Σχήμα 6.1 Throughput όλων των σεναρίων



Σχήμα 6.2 Average latency όλων των σεναρίων



Σχήμα 6.3 CPU utilization όλων των σεναρίων

Διανύουμε μία εποχή όπου οι τεχνολογίες και οι υπηρεσίες cloud χρησιμοποιούνται ολοένα και περισσότερο και βλέπουμε την εννιά του cloud streaming να εμφανίζεται στην καθημερινότητα μας όπως το audio και video streaming , άλλα και σε καινούριους τομείς στο προσεχές μέλλον όπως είναι το streaming ηλεκτρονικών παιχνιδιών, πράγμα που σημαίνει ότι οι απαιτήσεις των τελικών χρηστών αυξάνονται ,και επιπρόσθετα αυξάνονται και οι υποχρεώσεις των παρόχων τέτοιων υπηρεσιών. Είναι πολύ σημαντική η παρακολούθηση των υποδομών των υπηρεσιών διότι υπάρχουν ενδείξεις που απαιτούν ενέργειες . Όσο αυξάνονται οι χρήστες μιας τέτοιας υπηρεσίας παρατηρείται αύξηση στις τιμές κάποιον μετρικών κλειδιών (Average Latency, CPU Utilization , Throughput) όπως φαίνονται στα σχήματα 6.1 6.2 6.3 και όταν οι τιμές αυτές υπερβούν ένα κατώφλι που όρισε ο πάροχος τότε για να ικανοποιηθούν οι απατήσεις ποιότητας της υπηρεσίας , έτσι ώστε οι χρήστες να απολαμβάνουν την καλύτερη δυνατή ποιότητα υπηρεσιών πρέπει να γίνει κάποια ενέργεια όπως μία ενέργεια οριζόντιας επεκτασιμότητας με την πρόσθεση εικονικών μηχανών . Σε αντίθετη περίπτωση οι χρήστες δεν θα είναι ικανοποιημένοι διότι δεν θα λαμβάνουν αυτό που πληρώνουν και αυτό θα έχει αντίκτυπο στη φήμη της υπηρεσίας αυτής αλλά και στα κέρδη του παρόχου .

6.2 Μελλοντική Εργασία

Μπορεί να γίνει η εφαρμογή Deploy σε διαφορετικούς παρόχους cloud για συγκριτικούς λόγους .

Επίσης αντί να γίνει οριζόντια επεκτασιμότητα όπως περιγράφεται στο κεφάλαιο 4 να γίνει κάθετη επεκτασιμότητα επίσης για συγκριτικούς λόγους .

Και επιπρόσθετα μπορεί να γίνει διασπαστή της video streaming εφαρμογής σε Microservices και deployment σε Docker Container[48]. Έτσι υπολογίζεται ότι θα μειωθούν περισσότερο τα κόστη λειτουργίας του παρόχου , η εφαρμογή θα είναι ποιο διαχειρίσιμη όταν υπάρχει πρόβλημα σε κάποιο κομμάτι να μην σταματά η εφαρμογή παρά μόνο το κομμάτι αυτό . Ποιο εύκολη η επεκτασιμότητα διότι επεκτείνουμε μόνο τα κομμάτια τα οποία χρειάζεται και όχι ολόκληρη την εφαρμογή . Όμως εδώ υπάρχουν κάποιες προκλήσεις στην προσέγγιση αυτή όπως για παράδειγμα με την αρχιτεκτονική των containers απαιτούνται διαφορετικά εργαλεία παρακολούθησης από αυτά των μονολιθικών εφαρμογών .

Βιβλιογραφία

[1] Amazon EC2, https://aws.amazon.com/ec2/

[2] Office365, https://docs.microsoft.com/en-us/microsoft-365/

[3] Eclipse Che, https://www.eclipse.org/che/

[4] Spotify, https://www.spotify.com/

[5] Netflix, https://www.netflix.com/

[6] Microsoft Azure, https://azure.microsoft.com/en-us/?v=18.20

[7] Google Cloud Compute Engine, https://cloud.google.com/compute/

[8] AWS Elastic Beanstalk, https://aws.amazon.com/elasticbeanstalk/

[9] Azure Cloud Services, https://azure.microsoft.com/en-us/services/cloud-services/

[10] Google Cloud App Engine, https://cloud.google.com/appengine/

[11] Dropbox, https://www.dropbox.com/

[12] Amazon Web Services (AWS), https://aws.amazon.com/

[13] Amazon S3, https://aws.amazon.com/s3/

[14] Amazon VPC, https://aws.amazon.com/vpc/

[15] Amazon CloudWarch https://aws.amazon.com/cloudwatch/

[16] Microsoft Windows https://www.microsoft.com/en-us/windows

[17] Sql Azure, https://azure.microsoft.com/en-us/services/sql-database/

[18] ASP.NET, https://www.asp.net/

[19] .NET FRAMEWORK, https://docs.microsoft.com/en-us/dotnet/

[20] C#, https://docs.microsoft.com/en-us/dotnet/csharp/

[21] Visual Basic, https://docs.microsoft.com/en-us/dotnet/visual-basic/

[22] C++, http://devdocs.io/cpp/

[23] Google Cloud Platform, https://cloud.google.com/

[24] Nagios, https://www.nagios.org/

[25] StackDriven, https://cloud.google.com/stackdriver/

[26] Demetris Trihinas, George Pallis, Marios D. Dikaiakos," JCatascopia: Monitoring Elastically Adaptive Applications in the Cloud"

[27] Streama, https://github.com/streamaserver/streama.git

[28] Laravel, https://laravel.com/docs/5.6

[29] PHP, http://php.net/

[30] HTML, https://www.w3.org/TR/html52/

[31] CSS3, https://www.w3.org/TR/2001/WD-css3-roadmap-20010523/

[32] JavaScript, https://www.w3.org/standards/webdesign/script

[33] Bootstrap, https://getbootstrap.com/

[34] MySQL, https://www.mysql.com/

[35] MPEG DASH, https://mpeg.chiariglione.org/standards/mpeg-dash

[36] FFmpeg, https://www.ffmpeg.org/

[37] MPAC, https://gpac.wp.imt.fr/home/

[38] MP4BOX, https://gpac.wp.imt.fr/mp4box/

[39] DASH.JS, https://github.com/Dash-Industry-Forum/dash.js/wiki

[40] Apache HTTP Server, https://httpd.apache.org/

[41] FileZilla, https://filezilla-project.org/

[42] MySql Workbench, https://www.mysql.com/products/workbench/

[43] WorkLoad Generator, https://github.com/zgeorg03/Workload-Generator.git

[44] SQL Alchemy, https://www.sqlalchemy.org/

[45] Python, https://www.python.org/

[46] Matplotlib, https://matplotlib.org/

[47] Demetris Trihinas, Zacharias Georgiou, George Pallis, Marios D. Dikaiakos, "Improving Rule-Based Elasticity Control by Adapting the Sensitivity of the Auto-Scaling Decision Timefram

[48] Docker Container, https://www.docker.com/what-container

Παράρτημα Α

Ακολουθεί ο κώδικας του εργαλείου για την συλλογή των μετρικών.

Plots.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqla import cpu_u,mem_metrics,disk_metrics,network_metrics, Base
import psutil
import time
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import datetime
import numpy as np
engine = create_engine('mysql+pymysql://root:1234@localhost/video_streaming')
# create a configured "Session" class
Session = sessionmaker(bind=engine)
session = Session()
cpu_mets = session.query(cpu_u).all()
cpu_util_values = []
cpu_times = []
xaxis = []
mem_xaxis=[]




file = open("CPU_UTIL.txt","w")


for x in cpu_mets:
    cpu_util_values.append(x.value)

    cpu_times.append(x.created_at)
    file.write(str(x.value)+"\t"+str(x.created_at)+"\n")

file.close()
now1=cpu_times[0]
#print(now1)

count = 0
trs=0
xaxis.append(trs)
for x in range(0,len(cpu_times)-1):
    time1=cpu_times[x]
    time2=cpu_times[x+1]
    time3 =(time2 - time1).total_seconds()
    count = count + time3
    xaxis.append(count)


#print(xaxis)
plt.figure(figsize=(20,10))
plt.plot(xaxis,cpu_util_values)
plt.ylabel('Cpu Utilization')
plt.xlabel('Time(sec)')
plt.savefig('CPU_Util')
plt.close()

mem_mets = session.query(mem_metrics).all()
mem_util_values = []
mem_ave_values = []
mem_used_values = []
```

```
61  mem_free_values = []
62  mem_times = []
63
64
65  file = open("MEMORY_UTIL.txt","w")
66  file1 = open("AVE_MEM.txt","w")
67  file2 = open("USED_MEM.txt","w")
68  file3 = open("FREE_MEM.txt","w")
69
70  for x in mem_mets:
71      mem_util_values.append(x.mem_util)
72      mem_times.append(x.created_at)
73      mem_ave_values.append(x.mem_ave)
74      mem_used_values.append(x.mem_used)
75      mem_free_values.append(x.mem_free)
76      file.write(str(x.mem_util)+"\t"+str(x.created_at)+"\n")
77      file1.write(str(x.mem_ave)+"\t"+str(x.created_at)+"\n")
78      file2.write(str(x.mem_used)+"\t"+str(x.created_at)+"\n")
79      file3.write(str(x.mem_free)+"\t"+str(x.created_at)+"\n")
80  file.close()
81  file1.close()
82  file2.close()
83  file3.close()
84
85  count = 0
86  trs=0
87  mem_xaxis.append(trs)
88  for x in range(0,len(mem_times)-1):
89      time1=mem_times[x]
90      time2=mem_times[x+1]
91      time3 =(time2 - time1).total_seconds()
92      count = count + time3
93      mem_xaxis.append(count)
94  plt.figure(figsize=(20,10))
95  plt.plot(mem_xaxis,mem_util_values)
96  plt.ylabel('Memory Utilization')
97  plt.xlabel('Time(sec)')
98  plt.savefig('MEM_Util')
99  plt.close()
100 plt.figure(figsize=(20,10))
101 plt.plot(mem_xaxis,mem_ave_values)
102 plt.ylabel('Available Memory')
103 plt.xlabel('Time(sec)')
104 plt.savefig('AVE_MEM')
105 plt.close()
106 plt.figure(figsize=(20,10))
107 plt.plot(mem_xaxis,mem_used_values)
108 plt.ylabel('Used Memory')
109 plt.xlabel('Time(sec)')
110 plt.savefig('USED_MEM')
111 plt.close()
112 plt.figure(figsize=(20,10))
113 plt.plot(mem_xaxis,mem_free_values)
114 plt.ylabel('Free Memory')
115 plt.xlabel('Time(sec)')
```

```
116  plt.savefig('FREE_MEM')
117  plt.close()
118
119  disk_mets = session.query(disk_metrics).all()
120  disk_bytes_read_values = []
121  disk_bytes_write_values = []
122  disk_times = []
123  disk_xaxis = []
124
125
126  file = open("DISK_READ.txt","w")
127  file1 = open("DISK_WRITE.txt","w")
128  for x in disk_mets:
129      disk_times.append(x.created_at)
130      disk_bytes_read_values.append(x.read_bytes)
131      disk_bytes_write_values.append(x.write_bytes)
132      file.write(str(x.read_bytes)+"\t"+str(x.created_at)+"\n")
133      file1.write(str(x.write_bytes)+"\t"+str(x.created_at)+"\n")
134  file.close()
135  file1.close()
136  count = 0
137  trs=0
138  disk_xaxis.append(trs)
139  for x in range(0,len(disk_times)-1):
140      time1=disk_times[x]
141      time2=disk_times[x+1]
142      time3 =(time2 - time1).total_seconds()
143      count = count + time3
144      disk_xaxis.append(count)
145  plt.figure(figsize=(20,10))
146  plt.plot(disk_xaxis,disk_bytes_read_values)
147  plt.ylabel('Disk Bytes Read')
148  plt.xlabel('Time(sec)')
149  plt.savefig('Disk Bytes Read')
150  plt.close()
151  plt.figure(figsize=(20,10))
152  plt.plot(disk_xaxis,disk_bytes_write_values)
153  plt.ylabel('Disk Bytes Write')
154  plt.xlabel('Time(sec)')
155  plt.savefig('Disk Bytes Write')
156  plt.close()
157
158  network_mets = session.query(network_metrics).all()
159  network_bytes_sent_values = []
160  network_bytes_receve_values = []
161  network_packets_sent_values = []
162  network_packets_receve_values = []
163  network_times = []
164  network_xaxis = []
165
166  file = open("NETWORK_BYTES_SENT.txt","w")
167  file1 = open("NETWORK_BYTES_RECEVE.txt","w")
168  file2 = open("NETWORK_PACKETS_SENT.txt","w")
169  file3 = open("NETWORK_PACKETS_RECEVE.txt","w")
```

```
for x in network_mets:
    network_times.append(x.created_at)
    network_bytes_sent_values.append(x.bytes_sent)
    network_bytes_receve_values.append(x.bytes_receve)
    network_packets_sent_values.append(x.packets_sent)
    network_packets_receve_values.append(x.packets_receve)
    file.write(str(x.bytes_sent)+"\t"+str(x.created_at)+"\n")
    file1.write(str(x.bytes_receve)+"\t"+str(x.created_at)+"\n")
    file2.write(str(x.packets_sent)+"\t"+str(x.created_at)+"\n")
    file3.write(str(x.packets_receve)+"\t"+str(x.created_at)+"\n")
file.close()
file1.close()
file2.close()
file3.close()
count = 0
trs=0
network_xaxis.append(trs)
for x in range(0,len(network_times)-1):
    time1=network_times[x]
    time2=network_times[x+1]
    time3 =(time2 - time1).total_seconds()
    count = count + time3
    network_xaxis.append(count)
plt.figure(figsize=(20,10))
plt.plot(network_xaxis,network_bytes_sent_values)
plt.ylabel('Grafs/Network Bytes Sent')
plt.xlabel('Time(sec)')
plt.savefig('Network Bytes Sent')
plt.close()
plt.figure(figsize=(20,10))
plt.plot(network_xaxis,network_bytes_receve_values)
plt.ylabel('Network Bytes Receve')
plt.xlabel('Time(sec)')
plt.savefig('Network Bytes Receve')
plt.close()
plt.figure(figsize=(20,10))
plt.plot(network_xaxis,network_packets_sent_values)
plt.ylabel('Network Packets Sent')
plt.xlabel('Time(sec)')
plt.savefig('Network Packets Sent')
plt.close()
plt.figure(figsize=(20,10))
plt.plot(network_xaxis,network_packets_receve_values)
plt.ylabel('Network Packets Receve')
plt.xlabel('Time(sec)')
```

```
215   plt.savefig('Network Packets Receve')
216   plt.close()
217
218   newval = []
219   for x in range(0,len(cpu_util_values)-1):
220       newval.append (float(cpu_util_values[x]))
221
222   #print (newval)
223
224   def movingaverage(interval, window_size):
225       window= np.ones(int(window_size))/float(window_size)
226       return np.convolve(interval, window, 'same')
227
228   test = movingaverage(newval, 10)
229   print(test[4])
230
231   plt.figure(figsize=(20,10))
232   plt.plot(test)
233   plt.ylabel('Cpu Utilization Moving Average')
234   plt.xlabel('Time(sec)')
235   plt.savefig('CPU_Util_MOV_AVE')
236   plt.close()
```

Sqla.py

```python
# coding: utf-8
from sqlalchemy import Column, ForeignKey, Integer, String, LargeBinary
from sqlalchemy.dialects.mysql import BIGINT, BINARY, BIT, BLOB, BOOLEAN, CHAR,
DATE,DATETIME, DECIMAL, DECIMAL, DOUBLE, ENUM, FLOAT, INTEGER, LONGBLOB, LONGTEXT,MEDIUMBLOB,
 MEDIUMINT, MEDIUMTEXT, NCHAR, NUMERIC, NVARCHAR, REAL, SET, SMALLINT,TEXT, TIME, TIMESTAMP,
 TINYBLOB, TINYINT, TINYTEXT, VARBINARY, VARCHAR, YEAR
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
import  datetime

Base = declarative_base()
metadata = Base.metadata

class cpu_u(Base):
    __tablename__ = 'cpu_utils'
    id = Column(INTEGER, primary_key=True)
    value = Column(DOUBLE(8,2), nullable=False)
    created_at = Column(DATETIME, default =datetime.datetime.now)

class mem_metrics(Base):
    __tablename__ = 'mem_metrics'
    id = Column(INTEGER, primary_key=True)
    mem_util = Column(DOUBLE(4,2), nullable=False)
    mem_total= Column(BIGINT, nullable=False)
    mem_ave= Column(BIGINT, nullable=False)
    mem_used= Column(BIGINT, nullable=False)
    mem_free= Column(BIGINT, nullable=False)
    created_at = Column(DATETIME, default =datetime.datetime.now)

class disk_metrics(Base):
    __tablename__ = 'disk_metrics'
    id = Column(INTEGER, primary_key=True)
    read_bytes= Column(BIGINT, nullable=False)
    write_bytes= Column(BIGINT, nullable=False)
    created_at = Column(DATETIME, default =datetime.datetime.now)
class network_metrics(Base):
    __tablename__ = 'network_metrics'
    id = Column(INTEGER, primary_key=True)
    bytes_sent= Column(BIGINT, nullable=False)
    bytes_receve= Column(BIGINT, nullable=False)
    packets_sent= Column(BIGINT, nullable=False)
    packets_receve= Column(BIGINT, nullable=False)
    created_at = Column(DATETIME, default =datetime.datetime.now)

#sqlacodegen mysql+pymysql://root:1234@localhost/video_streaming --outfile models.py
engine = create_engine('mysql+pymysql://root:1234@localhost/video_streaming')
Base.metadata.create_all(engine)


```

Metrics.py

```python
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqla import cpu_u,mem_metrics,disk_metrics,network_metrics, Base
import psutil
import time

engine = create_engine('mysql+pymysql://root:1234@localhost/video_streaming')
# create a configured "Session" class
Session = sessionmaker(bind=engine)
# create a Session
session = Session()
todelete= session.query(cpu_u).delete()
todelete1= session.query(mem_metrics).delete()
todelete2= session.query(network_metrics).delete()
todelete3= session.query(disk_metrics).delete()
session.commit()

diskmet= psutil.disk_io_counters()
pre_read= diskmet[2]
pre_write= diskmet[3]

netmet= psutil.net_io_counters()
pre_bytes_sent= netmet[0]
pre_bytes_recv= netmet[1]
pre_packets_sent= netmet[2]
pre_packets_recv= netmet[3]

while True :
    ts = time.time()
    cpu_util = psutil.cpu_percent(interval=0.4, percpu=False)
    meme= psutil.virtual_memory()
    mem_util = meme[2]
    mem_total = meme[0]
    mem_ave = meme[1]
    mem_used = meme[3]
    mem_free = meme[4]
    diskmet2 = psutil.disk_io_counters()
    now_read= diskmet2[2]
    now_write= diskmet2[3]
    netmet2= psutil.net_io_counters()
    now_bytes_sent= netmet2[0]
    now_bytes_recv= netmet2[1]
    now_packets_sent= netmet2[2]
    now_packets_recv= netmet2[3]

    print("------------------------------------------------------------------------")
    print("********************* CPU METRICS ************************\n")
    print ("CPU Utilization :" ,  cpu_util,"\n")
    newcpu_u = cpu_u(value=cpu_util)
    session.add(newcpu_u)
    session.commit()
    print("********************* MEMORY METRICS *********************\n")
    print (meme,"\n")
    print ("Memory Utilization :" ,  mem_util,"\n")
    print ("Total Memory :" ,  mem_total,"\n")
    print ("Available Memory :" ,  mem_ave,"\n")
    print ("Used Memory :" ,  mem_used,"\n")
    print ("Free Memory :" ,  mem_free,"\n")
    newmem_m = mem_metrics(mem_util=mem_util,mem_total=mem_total,mem_ave=mem_ave,mem_used=mem_used,mem_free=mem_free)
    session.add(newmem_m)
```

```
    session.commit()
    print("********** DISK METRICS **********\n")
    print(diskmet2,"\n")
    read_bytes = now_read - pre_read
    write_byte = now_write- pre_write

    newdisk_m = disk_metrics(read_bytes=read_bytes/5,write_bytes=write_byte/5)
    session.add(newdisk_m)
    session.commit()


    print ("Read bytes :" ,  read_bytes,"\n")

    print ("Write bytes :" ,  write_byte,"\n")
    pre_read=now_read
    pre_write=now_write
    print("********** NETWORK METRICS **********")
    print(netmet2,"\n")
    bytes_sent = now_bytes_sent - pre_bytes_sent
    bytes_recv = now_bytes_recv- pre_bytes_recv
    packets_sent = now_packets_sent - pre_packets_sent
    packets_recv = now_packets_recv- pre_packets_recv

    newnet_m = network_metrics(bytes_sent=bytes_sent/5,bytes_receve=bytes_recv/5,packets_sent=packets_sent/5,
        packets_receve=packets_recv/5)
    session.add(newnet_m)
    session.commit()

    print ("Bytes Sent :" ,  bytes_sent,"\n")
    print ("Bytes Receve :" ,  bytes_recv,"\n")
    print ("Packets Sent :" ,  packets_sent,"\n")
    print ("Packets Receve :" ,  packets_recv,"\n")
    pre_bytes_recv=now_bytes_recv
    pre_bytes_sent=now_bytes_sent
    pre_packets_sent=now_packets_sent
    pre_packets_recv=now_packets_recv
    print("************************************************************")


    ts2 = time.time()
    ts3 = ts2-ts
    ts4 = 5-ts3
    if ts4>0:
        time.sleep(ts4)

```

Workload Generator Side

```python
import matplotlib.pyplot as plt
import re
import numpy as np



f = open("2018-04-19_LB_360_40_1H_NEW.log","r") #opens file with name of "test.txt"
line = f.readline()
line = f.readline()
line = f.readline()
line = f.readline()
line = f.readline()
line = f.readline()
line = f.readline()
line = f.readline()


values =[]
tps_values = []
xaxis = []
confrps = []
avelt = []
minlt = []
maxlt = []
while (line):

    wordList = re.sub(" ", " ",  line).split()
    confrps.append(wordList[1])
    values.append(wordList[2])
    tps_values.append(wordList[3])
    avelt.append(wordList[4])
    minlt.append(wordList[5])
    maxlt.append(wordList[6])
    line = f.readline()

# print ("********************************************")

# print(avelt)
# print("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx")
# print(minlt)
# print("------------------------------------------------------------")
# print(maxlt)

count = 0
for x in range(0,len(values)):
     xaxis.append(count)
     count = count + 5
#print(xaxis)

plt.figure(figsize=(20,10))
plt.plot(xaxis,values)

plt.ylabel('Real Rps')
plt.xlabel('Time(sec)')
plt.savefig('REAL.png')
plt.close()

plt.figure(figsize=(20,10))
plt.plot(xaxis,tps_values)
plt.ylabel('TPS')
```

```
61    plt.xlabel('Time(sec)')
62    plt.savefig('tps.png')
63    plt.close()
64
65    plt.figure(figsize=(20,10))
66    plt.plot(xaxis,avelt,label='Average Latency')
67    plt.plot(xaxis,minlt,label='Min Latency')
68    plt.plot(xaxis,maxlt,label='Max Latency')
69    plt.legend()
70    plt.ylabel('Milliseconds')
71    plt.xlabel('Time(sec)')
72    plt.savefig('lat.png')
73    plt.close()
74
75    plt.figure(figsize=(20,10))
76    plt.plot(xaxis,confrps,label='Configure RPS')
77    plt.plot(xaxis,values,label='Real RPS')
78    plt.plot(xaxis,tps_values,label='TPS')
79    plt.legend()
80    plt.ylabel('RPS')
81    plt.xlabel('Time(sec)')
82    plt.savefig('3rps.png')
83    plt.close()
84
85
```