Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών

" Διδακτική της Τεχνολογίας & Ψηφιακά Συστήματα "

“A Unified Benchmark Framework for Cloud Deployed Applications Tool in IaaS”

Συγγραφέας: Έλτον Κεβάνη

Διπλωματική Εργασία υποβληθείσα στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στα Δικτυοκεντρικά Συστήματα

Επιβλέπων Καθηγητής: Μαρίνος Θεμιστοκλέους

Πειραιάς, Απρίλιος 2014

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Πανεπιστήμο Πειραιά, στο Τμήμα Ψηφιακών Συστημάτων , κατά το έτος 2013-2014.

Η ολοκλήρωση της μεταπτυχιακής αυτής εργασίας θα ήταν αδύνατη χώρις την πολύτιμη υποστήριξη του καθηγητή μου, Αν. Καθηγητή των Ψηφιακών Συστημάτων Κου Μαρίνου Θεμιστοκλέους. Χρωστάω επίσης, ένα πολύ μεγάλο ευχαριστώ στο Λέκτορα Καθηγητή Κυριαζή Δημοσθένη για την υποστήριξή του, την καθοδήγησή του, την κατανόηση του και την προσφορά των μηχανημάτων βάσει των οποίων αυτή η εργασία ήρθε εις πέρας. Επίσης, ευχαριστώ στον κύριο Γιώργο Κουσιούρη , ο οποίος με κατεύθυνε ορθώς με σκέψη άρτια και επιστημονική.

Επίσης θα ήθελα , από βάθους καρδιάς να ευχαριστήσω το φίλο, συνάδερφο και συνεργάτη Χριστόφορο Σταμπολτά, του οποίου η συνεργασία, για την ολοκλήρωση αυτής της εργασίας, ήταν άριστη.

Τέλος,θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη τους καθόλη τη διάρκεια των σπουδών μου

Πίνακας περιεχομένων

Πίνακας Περιεχομένων Εικόνων

Πίνακας Περιεχομένων Γραφημάτων

1 Περίληψη

Στην παρούσα διπλωματική εργασία επιχειρήθηκε να δοθεί λύση στο πρόβλημα της αξιολόγησης της απόδοσης διαφόρων εφαρμογών που εκτελούνται σε περιβάλλον υπολογιστικού νέφους , στο στρώμα Υποδομή ως Υπηρεσία, και πως η απόδοση τους εξαρτάται από την απόδοση του περιβάλλοντος υπολογιστικού νέφους. Για τον λόγο αυτό αναπτύξαμε μια εφαρμογή που δίνει στους χρηστές τις εξής δυνατότητες :

1. Εκτέλεση δοκιμών αξιολόγησης εφαρμογών που θα εγκατασταθούν σε εικονικές μηχανές οι οποίες θα δημιουργηθούν σε περιβάλλον υπολογιστικού νέφους.
2. Αποθήκευση και Ανάλυση αποτελεσμάτων των δοκιμών και πρόταση βέλτιστων επιλογών βάσει των αποτελεσμάτων.

Η εφαρμογή αποτελείται από δυο μέρη ,το γραφικό περιβάλλον (Web Tool) και το κυρίως μέρος (Tool). Στο Web Tool ο χρήστης μπορεί να επιλέξει ττην δοκιμή που επιθυμεί για να εκτελέσει καταχωρώντας τα κατάλληλα δεδομένα εισόδου τα οποία είναι παράμετροι της δοκιμής, παράμετροι επαναληπτικότητας και παράμετροι επικοινωνίας με το περιβάλλον υπολογιστικού νέφους. Επίσης , από το Web Tool ο χρήστης έχει τη δυνατότητα να αναλύσει τα αποτελέσματα προηγούμενων δοκιμών που έχει εκτελέσει για όλες τις εφαρμογές μέσω εξειδικευμένων συναρτήσεων. Το Tool κάνει δυνατή την υλοποίηση όλων των παραπάνω. Μέσω του κώδικα που αναπτύχτηκε το Tool , βάσει των δεδομένων

εισόδου που έχει δεχτεί από το Web Tool, επικοινωνεί με το περιβάλλον υπολογιστικού νέφους και με αυτοματοποιημένο τρόπο εγκαταστεί και παραμετροποιεί τις δοκιμές και τις εφαρμογές προς αξιολόγηση, εκτελεί τις δοκιμές και επιστρέφει τα αρχεία αποτελεσμάτων πίσω στο Web Tool το οποίο και τα αποθηκεύει στην βάση δεδομένων.

Η εφαρμογή που αναπτύχτηκε, από άποψη προγραμματισμού, στηρίχτηκε σε τεχνολογίας αιχμής χρησιμοποιώντας τεχνολογίες όπως JAVA EE, Spring Project, HTML5, CSS3, Bootstrap Twitter και εργαλεία όπως το Jclouds. Στην παρούσα έκδοση του εργαλείου που αναπτύξαμε εφαρμογές προς αξιολόγηση που υποστηρίζονται είναι Βάσεις Δεδομένων, Web Serving και MapReduce. Το περιβάλλον υπολογιστικού νέφους που υποστηρίζεται είναι το Openstack.

Εκτός από την ανάπτυξη της εφαρμογής στην παρούσα εργασία αναλύουμε την έννοια του υπολογιστικού νέφους και των μοντέλων τους , την ανάγκη αξιολόγησης των υποδομών υπολογιστικού νέφους, τις προσπάθειες που έχουν γίνει σε αυτήν την κατεύθυνση από άλλες ερευνητικές ομάδες και εμπορικές επιχειρήσεις. Επίσης , παρουσιάζουμε την αρχιτεκτονική του εργαλείου που αναπτύξαμε και αναλύουμε τα υποστηρικτικά εργαλεία που χρησιμοποιήσαμε και τις δικιμές που υποστηρίζει το εργαλείο αυτό . Επιπρόσθετα , γίνεται επίδειξη του εργαλείου μας τρέχοντας όλες τις δικιμές των εφαρμογών σε δυο σενάρια και αναλύουμε τα αποτελέσματα που λάβαμε. Τέλος ,κάνουμε αναφορά σε μελλοντική μελέτη και βελτίωση που ενδεχομένως μπορεί να γίνει .

2 ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο θα αναλύσουμε διεξοδικά την έννοια του Υπολογιστικού Νέφους , του βασικού μοντέλου εφαρμοσμένων υπηρεσιοστρεφών υποδομών, καθώς και οι διαφορετικοί ρόλοι των οντοτήτων μέσα σε αυτό και των σχέσεων μεταξύ τους. Εν συνεχεία θα προχωρήσουμε σε μία σύντομη περιγραφή της δομής της διπλωματικής εργασίας , ενώ εν τέλει θα παρουσιάσουμε τη δομή και την οργάνωση της.

2.1 Ορισμός του Υπολογιστικού Νέφους

Σύμφωνα με τον οργανισμό NIST (National Institute of Standards and Technology) το Υπολογιστικό Νέφος (Cloud Computing) ορίζεται ως εξής [1]:

Το Υπολογιστικό Νέφος είναι ένα μοντέλο για τη πραγματοποίηση της συνεχούς, κατάλληλης, και κατά παραγγελίας πρόσβασης μέσω δικτύου σε μια κοινή ομάδα διαμορφώσιμων πόρων υπολογισμού (π.χ., δίκτυα, κεντρικοί υπολογιστές, αποθηκευτικοί χώροι, εφαρμογές, και υπηρεσίες) που μπορούν να είναι διαθέσιμα γρήγορα και με την ελάχιστη διαχειριστική προσπάθεια ή αλληλεπίδραση φορέων παροχής υπηρεσιών. Αυτό το μοντέλο νέφους αποτελείται από πέντε (5) βασικά χαρακτηριστικά, τρία (3) μοντέλα υπηρεσιών και τέσσερα (4) μοντέλα ανάπτυξης.

2.1.1 Βασικά Χαρακτηριστικά

- Αυτο-εξυπηρέτηση: ένας καταναλωτής μπορεί να αυξήσει τους υπολογιστικούς πόρους που χρησιμοποιεί χωρίς ανθρώπινη αλληλεπίδραση
- Πρόσβαση μέσω διαδικτύου: οι δυνατότητες αυτές παρέχονται μέσω δικτύου και μπορούν να προσπελαστούν μέσω εφαρμογών πελάτη
- Διαχείριση πόρων: οι πόροι του παρόχου εξυπηρετούν πολλαπλούς καταναλωτές με διαφορετικές φυσικές και εικονικές ανάγκες και ανατίθενται σε αυτούς δυναμικά. Οι καταναλωτές δεν γνωρίζουν την κατάσταση στο εσωτερικό της υποδομής και δεν μπορούν να ορίσουν τα φυσικά μηχανήματα που θα καταλάβουν οι πόροι τους (εκτός από την γενική γεωγραφική τοποθεσία)
- Ελαστικότητα πόρων: οι πόροι που χρησιμοποιεί ένας καταναλωτής μπορούν να μεταβάλλονται ελαστικά και γρήγορα, ώστε να ακολουθούν τη ζήτηση και τις ανάγκες του τελευταίου
- Καταγραφή χρησιμοποιούμενων πόρων: οι πάροχοι πρέπει να παρέχουν τρόπους καταγραφής των χρησιμοποιούμενων πόρων για λόγους χρέωσης και διαχείρισης.

2.1.2 Μοντέλα Υπηρεσιών

- Λογισμικό ως Υπηρεσία (Software as a Service (SaaS): Η δυνατότητα που παρέχεται στον καταναλωτή είναι η χρήση των εφαρμογών του παρόχου οι οποίες εκτελούνται σε μία υποδομή νέφους υπολογιστών. Οι εφαρμογές είναι προσβάσιμες από διάφορες συσκευής – πελάτες (client devices) , είτε μέσω μιας πολύ ελαφριάς πλατφόρμας τύπου πελάτη, (πχ ενός περιηγητή ιστοσελίδων) , είτε τέλος από μία εφαρμογή που λειτουργεί αποκλειστικά ως πελάτης για τη διασύνδεση με το χρήστη. Ο καταναλωτής δε διαχειρίζεται ή έχει πρόσβαση στην

υποκείμενη τεχνολογία λειτουργίας του νέφους , όσον αφορά το δίκτυο, τους εξυπηρετητές , το λειτουργικό σύστημα , μεθόδους αποθήκευσης ή ακόμα και μεμονωμένες δυνατότητες της εφαρμογής , με την πιθανή εξαίρεση τη δυνατότητα ο χρήστης να είναι σε θέση να διαχειριστεί συγκεκριμένες εξ' ορισμού ρυθμίσεις της εφαρμογής. Εκτός από την ανάπτυξη/προσαρμογή της εφαρμογής, αυτός ο ρόλος προσδιορίζει επίσης ποιοι είναι οι όροι που μπορούν να ρυθμιστούν από τους τελικούς καταναλωτές της εφαρμογής:

Παράμετροι φόρτου εργασίας: διαμορφώσιμες παράμετροι με τις οποίες ένας πελάτης θέλει να εκτελέσει μια υπηρεσία (π.χ. αριθμός χρηστών, ανάλυση μιας εικόνας σε εφαρμογές πολυμέσων κ.λπ.)

Ποιότητα των παραμέτρων υπηρεσίας (Quality of Service-QoS): η αναμενόμενη έξοδος της υπηρεσίας, αυτή που χρησιμοποιείται για να εκφράσει τα επίπεδα QoS (π.χ. χρόνος απόκρισης, πλαίσια ανά δευτερόλεπτο μιας μετάδοσης πολυμέσων κ.λπ.). Αυτό είναι επίσης γνωστό ως Βασικοί Δείκτες Απόδοσης (Key Performance Indicators-KPIs)

- Πλατφόρμα ως Υπηρεσία (Platform as a Service (PaaS)): Η πλατφόρμα ως υπηρεσία ή αλλιώς γνωστό και ως cloudware είναι η συνέχεια του SaaS. Παρέχει μια cloud πλατφόρμα εφαρμογών για εταιρείες ή ιδιώτες που κατασκευάζουν λογισμικό για τους ίδιους είτε για τρίτους. Το PaaS παρέχει όλους τους πόρους που απαιτούνται για να δημιουργηθούν εφαρμογές και υπηρεσίες μέσω του Internet, χωρίς να πρέπει να κατεβάσει ή να εγκαταστήσει λογισμικό (μέσω γλωσσών προγραμματισμού, βιβλιοθήκες, βάσεις δεδομένων, υπηρεσίες και εφαρμογές που υποστηρίζονται από τους παρόχους). Οι υπηρεσίες PaaS περιλαμβάνουν την σχεδίαση εφαρμογών, την ανάπτυξη, τον έλεγχο, την εγκατάσταση και την φιλοξενία εφαρμογών. Έχει σχεδιαστεί για να χρησιμοποιείται από πολλούς

χρήστες ταυτόχρονα και παρέχει αυτόματες διευκολύνσεις για ταυτόχρονη διαχείριση, κλιμάκωση, ανακατεύθυνση και ασφάλεια. Αυτός ο ρόλος ενσωματώνει ενέργειες όπως η μοντελοποίηση της εφαρμογής, η πρόβλεψη απόδοσής της, η παρακολούθηση, αξιολόγηση των γεγονότων και υλοποίηση διορθωτικών ενεργειών (π.χ. αύξηση των πόρων). Σε αυτήν την διαδικασία μπορεί να χρησιμοποιήσει τις πληροφορίες που παρέχονται από τον υπεύθυνο για την ανάπτυξη εφαρμογής μέσω μιας περιγραφής του τμήματος λογισμικού (π.χ. σε γλώσσα XML (W3 Schools)).Ο πελάτης δεν διαχειρίζεται ή ελέγχει τις υποδομές όπως το δίκτυο, τους διακομιστές, τα λειτουργικά συστήματα παρά μόνο την εφαρμογή που θα ανεβάσει στο νέφος [1][2].

- Υποδομή ως Υπηρεσία (Infrastructure As a Service (Iaas)): Η υποδομή ως υπηρεσία ή αλλιώς γνωστή και ως υλικό (Hardware as a Service-HaaS) προσφέρει το υλικό (σε αντίθεση με τα SaaS και PaaS τα οποία παρέχουν εφαρμογές), έτσι ώστε η επιχείρηση να μπορεί να βάζει ότι θέλει σε αυτό. Η εταιρεία ή ο ιδιώτης μπορεί να υπενοικιάσει υποδομή ανάλογα με τις απαιτήσεις εκείνης της χρονικής στιγμής, αντί να προβεί στην αγορά εξοπλισμού (υπολογιστικού, δικτυακού, κλπ). Ο χρήστης δεν διαχειρίζεται/ ελέγχει τη βασική υποδομή cloud, αλλά έχει τον έλεγχο των λειτουργικών συστημάτων, της αποθήκευσης, και τις αναπτυσσόμενες εφαρμογές και ενδεχομένως περιορισμένο έλεγχο της επιλογής εξαρτημάτων δικτύωσης (π.χ. firewalls υποδοχής). Σημαντικό πλεονέκτημα του IaaS είναι η δυνατότητα μεταφοράς εικονικών μηχανών από το ιδιόκτητο περιβάλλον της εταιρείας ή του ιδιώτη στο cloud, με συνοπτικές διαδικασίες καθώς και ότι πολλοί χρήστες μπορούν να χρησιμοποιήσουν τον εξοπλισμό ταυτόχρονα[1][2].
- Καταναλωτής/πελάτης: μια οντότητα που χρησιμοποιεί μια εφαρμογή που προσφέρεται ως υπηρεσία. Ο καταναλωτής έρχεται σε επαφή με τον προμηθευτή PaaS που έχει καταστήσει αυτό το SaaS διαθέσιμο και ζητά αυτό το λογισμικό για

μια συγκεκριμένη διαμόρφωση (παραμέτρους φόρτου εργασίας). Επιπλέον απαιτεί ορισμένα επίπεδα ποιότητας της υπηρεσίας (QoS), όπως αυτή υπολογίζεται από τους συγκεκριμένους βασικούς δείκτες απόδοσης (KPIs) της εφαρμογής. Ο προμηθευτής PaaS έρχεται έπειτα σε επαφή με έναν προμηθευτή IaaS και ζητά τους πόρους υλικού που έχει προβλέψει ότι θα ικανοποιήσουν τις ανάγκες του καταναλωτή. Μετά από μια επιτυχή διαπραγμάτευση, η εφαρμογή (SaaS) εκκινείται (από το PaaS) στις υποδομές που παρέχονται από το IaaS.

Όλες οι προαναφερθείσες ενέργειες επικυρώνονται τυπικά μέσω των Συμφωνιών Επιπέδων Υπηρεσιών (Service Level Agreements-SLAs[6] μεταξύ των συμβαλλόμενων μερών. Το SLA περιγράφει τη συμφωνία για αυτούς τους όρους και διατάξεις και δεσμεύει νομικά τους εμπλεκόμενους παρόχους. Χαρακτηριστικά, υπάρχουν δύο SLAs, ένα μεταξύ του καταναλωτή και του προμηθευτή PaaS (Application SLA, A-SLA), που αναφέρονται στους όρους εφαρμογής (παράμετροι φόρτου εργασίας και επίπεδα τιμών των παραμέτρων υπηρεσίας), και ένα μεταξύ των προμηθευτών PaaS και IaaS (Technical SLA, T-SLA), που εκφράζονται με όρους επιπέδων εικονικών υπολογιστικών πόρων.

2.1.3 Μοντέλα Ανάπτυξης

- Ιδιωτικά Συστήματα Υπολογιστικών Νεφών (Private Cloud). Τα συγκεκριμένα συστήματα υπολογιστικών νεφών , έχουν αναπτυχθεί και χρησιμοποιούνται αποκλειστικά και μόνο από έναν οργανισμό που εξυπηρετεί ή/και περιλαμβάνει πολλαπλούς καταναλωτές (πχ επιχειρήσεις). Μπορεί να ανήκει στον οργανισμό , να τον διαχειρίζεται/χειρίζεται ο συγκεκριμένος οργανισμός ή ένα τρίτο συμβαλλόμενο μέρος ή ακόμα και ένας συνδυασμός των δύο. Δύναται η

συγκεκριμένη υλοποίηση να είναι εντός ή και εκτός γεωγραφικών ορίων του οργανισμού.

- Νέφη Κοινοτήτων (Community cloud). Η συγκεκριμένη υλοποίηση χρησιμοποιείται από μία συγκεκριμένη ομάδα εταιρειών , οργανισμών συγκεκριμένων ή/και παρεμφερών συμφερόντων (αποστολή, απαιτήσεις ασφάλειας, πολιτική, εκτιμήσεις συμμόρφωσης). Δύναται η συγκεκριμένη υλοποίηση να είναι εντός ή και εκτός γεωγραφικών ορίων του οργανισμού.
- Δημόσια Νέφη (Public Cloud). Η υλοποίηση των δημοσίων νεφών τροφοδοτείται για ελεύθερη και δημόσια χρήση από το ευρύ κοινό. Δημόσιοι οργανισμοί μπορούν να έχουν , διαχειρίζονται και να χρησιμοποιούν δημόσια νέφη, όπως επιχειρήσεις, ακαδημαϊκές κοινότητες , δημόσιοι οργανισμοί ή πιθανοί συνδυασμοί αυτών. Η συγκεκριμένη υλοποίηση υπάρχει αποκλειστικά εντός των γεωγραφικών ορίων του οργανισμού.
- Υβριδικά Νέφη (Hybrid Cloud). Η υλοποίηση αυτή είναι το συνονθύλευμα δύο ή και περισσοτέρων διακριτών τύπων νεφών (δημοσίων, κοινοτήτων, ιδιωτικών) που παραμένουν ομαδικές οντότητες, παρόλα αυτά υποχρεωμένες να ακολουθούν τυποποιημένες ή αποκλειστικές τεχνολογίες που επιτρέπουν τη μεταφορά δεδομένων και εφαρμογών.

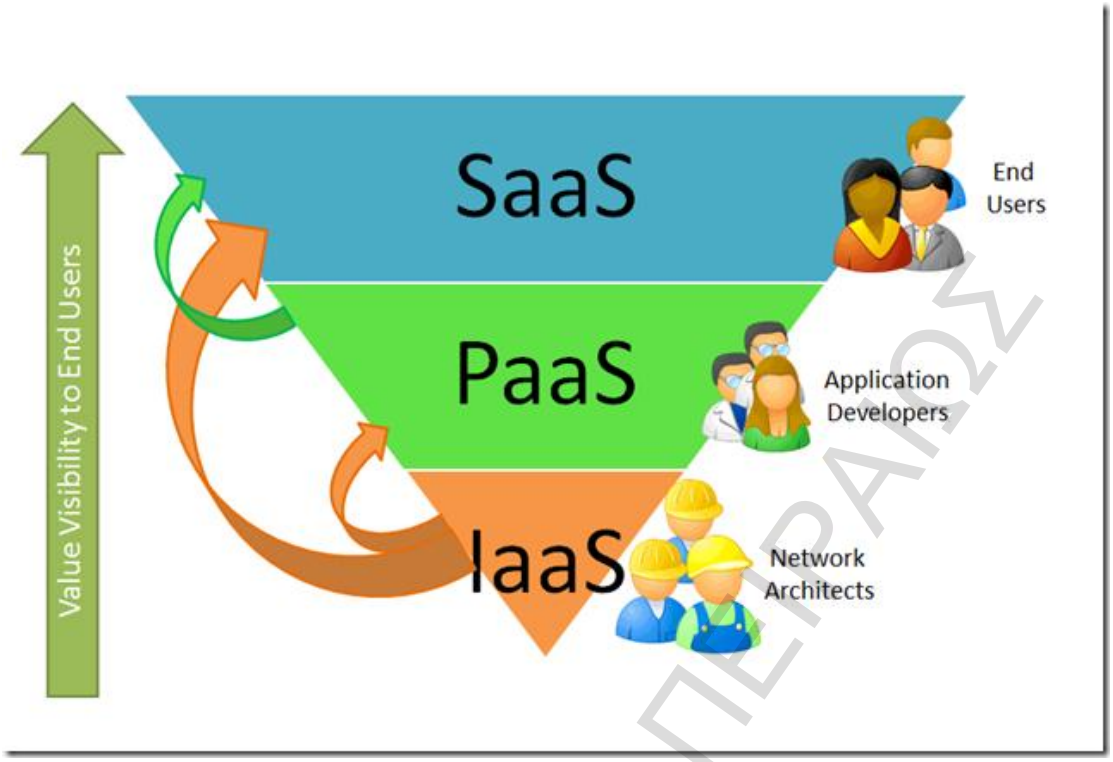
2.2 Σχέση με Περιβάλλοντα Πλέγματος

Τα περιβάλλοντα Πλέγματος[4] ξεκίνησαν σαν μία πολλά υποσχόμενη τεχνολογία η οποία ενσωμάτωνε ετερογενείς υπολογιστικούς πόρους μεγάλης κλίμακας σε ένα κοινό σύστημα διαχείρισης. Οι πόροι αυτοί πιθανόν να ανήκαν στον ίδιο ή διαφορετικούς παρόχους (και αντίστοιχες γεωγραφικές τοποθεσίες) και η διαχείρισή τους

επιτυγχανόταν μέσω των υπηρεσιοστρεφών αρχιτεκτονικών και υλοποιήσεων. Το υπηρεσιοστρεφές πλαίσιο φρόντιζε να παρέχει την απαραίτητη διασύνδεση και κοινή διαχείριση.

Παρόλο που η τεχνολογία αυτή ξεκίνησε πολύ δυναμικά, περιορίστηκε τελικά σε ακαδημαϊκούς και ερευνητικούς ρόλους ή σε πολύ συγκεκριμένες εφαρμογές που μπορούσαν να εκμεταλλευτούν τις δυνατότητές τους αλλά και να περιορίσουν τα μειονεκτήματά τους. Το βασικό μειονέκτημα ήταν η έλλειψη απομόνωσης μεταξύ των χρηστών, που δημιουργούσε προβλήματα σχετικά με την εγγύηση της παρεχόμενης υπηρεσίας, την ασφάλεια αλλά κυρίως με τις διαφορετικές απαιτήσεις περιβάλλοντος κάθε εφαρμογής. Έτσι, δύο εφαρμογές με διαφορετικές ανάγκες π.χ. σε λειτουργικό σύστημα, βιβλιοθήκες ή εκδόσεις λογισμικού δεν μπορούσαν να εκτελούνται στον ίδιο φυσικό πόρο.

Η βασική διαφορά των Υπολογιστικών Νεφών έγκειται στο γεγονός ότι χρησιμοποιούν την τεχνολογία της εικονικοποίησης (virtualization). Μέσω αυτής της τεχνικής, κάθε χρήστης είναι εντελώς απομονωμένος από τους υπόλοιπους και μπορεί να χειρίζεται κατά το δοκούν την διαμόρφωση του περιβάλλοντος εκτέλεσης των εφαρμογών του. Το γεγονός αυτό επέτρεψε στα Υπολογιστικά Νέφη να γίνουν σχεδόν αμέσως εμπορικά εκμεταλλεύσιμα και να μπορούν να χρησιμοποιηθούν για υπολογιστική ισχύ γενικού σκοπού. Μέσω αυτής της τεχνολογίας άτομα ή επιχειρήσεις έχουν πρόσβαση σε ελαστικούς υπολογιστικούς πόρους ή λογισμικό με πληρωμή ανάλογα με τη χρήση, χωρίς την ανάγκη κεφαλαιακής επένδυσης εξαρχής ή μετέπειτα συντήρησης. Έτσι μπορούν να τα προσαρμόζουν ανάλογα με τις ανάγκες τους ή τη ζήτηση, μειώνοντας το κόστος.



Εικόνα 1: Αρχιτεκτονική Cloud[20]

2.3 Σκοπός της Παρούσας Διπλωματικής Εργασίας

Η απόδοση σε περιβάλλοντα Υπολογιστικών Νεφών άρχισε να κερδίζει μεγάλη προσοχή τα τελευταία χρόνια (Hauck,2010)[11]. Έπειτα από υποσχέσεις για απεριόριστους πόρους και επεκτασιμότητα κατά το δοκούν (on-demand scalability) , άρχισαν να προκύπτουν θέματα σε σχέση με την αστάθεια λειτουργίας περιβαλλόντων υπολογιστικών νεφών , ιδιαίτερα δε όσον αφορά ζητήματα απόδοσης των διατιθέμενων πόρων (Kousiouris et al., 2012)[12]. Έτσι, προκειμένου μία διαδικασία μετάβασης σε περιβάλλον υπολογιστικού νέφους να θεωρηθεί επιτυχής , πρέπει να τεθεί υπόψη το ζήτημα της απόδοσης του παρόχου , τόσο για να εξοικονομηθούν χρήματα όσο και για να είναι εγγυημένη η σταθερότητα της μετεγκατεστημένης εφαρμογής . Όμως υπάρχουν διάφοροι και διαφορετικοί πάροχοι υπηρεσιών υπολογιστικών νεφών , έχοντας ο καθένας τις δικές

του μετρικές και στρατηγικές μετρήσεων προκειμένου να εξασφαλίζουν τη λεγόμενη ποιότητα υπηρεσίας (QoS) , χωρίς να υπάρχει ουσιαστικά μία ενοποιημένη και αντικειμενική πλατφόρμα μετρήσεων και εξαγωγής ασφαλών συμπερασμάτων.

Οι κύριες πτυχές των επιδόσεων των υπολογιστικών νεφών μπορούν να συνοψιστούν ως εξής:

- Ετερογενείς και άγνωστοι πόροι υλικού. Οι υπολογιστικοί πόροι που προσφέρουν οι υπηρεσίες νεφών είναι παντελώς άγνωστοι στους χρήστες. Κάποιες πληροφορίες μπορεί να είναι διαθέσιμες , όπως για παράδειγμα οι διαθέσιμοι πυρήνες , η συνολική διαθέσιμη μνήμη ή η χωρητικότητα των δίσκων. Αυτές όμως δεν είναι αρκετά επαρκείς ώστε ο χρήστης να εξάγει ασφαλή συμπεράσματα για τις δυνατότητες του υλικού του παρόχου , δυνατότητες που έχουν άρρηκτη σχέση με την αρχιτεκτονική , συνδεσιμότητα , ταχύτητα μνήμης κλπ. Σύμφωνα με μία μελέτη που πραγματοποιήθηκε από το Πανεπιστήμιο Aalto (Zhonghong Ou et al., 2012)[13], οι διαφορές που παρουσιάστηκαν μεταξύ των «γρήγορων» στιγμιότυπων (instances)[5] και «αργών» δύναται να φτάσουν το 40%. Σε κάποιες ειδικά περιπτώσεις , όπου πραγματοποιήθηκαν δοκιμές επί εφαρμογών , η διακύμανση στην απόδοση έφτασε και στο 60%.
- Διαφορετικές Ρυθμίσεις. Ακόμα και σε περιπτώσεις πανομοιότυπου υλικού , ο τρόπος με τον οποίο το υλικό διατίθεται ή έχει εξαρχής ρυθμισθεί η λειτουργία του υλικού σε κάθε instance, διαδραματίζει σημαντικό ρόλο στην απόδοση. Φυσικά το ίδιο συμβαίνει και σε περιπτώσεις διαμόρφωσης λογισμικού. (πχ instance Βάσης Δεδομένων σε εικονικό σύμπλεγμα).
- Ταυτόχρονη Μίσθωση και Χρήση του ιδίου Instance: Οι υποδομές Cloud μπορούν ταυτόχρονα να εξυπηρετήσουν μια πληθώρα από χρήστες, χρήστες οι οποίοι μπορούν να ξεκινήσουν να εκμεταλλεύονται τους διαθέσιμους πόρους που έχουν προμηθευτεί από τον πάροχο. Όμως στην περίπτωση της ταυτόχρονης χρήσης των

εικονικών μηχανών , υποβαθμίζεται σημαντικά η πραγματική απόδοση της εικονικής μηχανής. Επιπλέον , αποφάσεις βελτιστοποίησης των πόρων , που έχουν παρθεί από τον πάροχο , εν αγνοία του πελάτη, μπορεί να προχωρήσουν σε ομαδοποίηση των συγκεκριμένων πόρων ,γεγονός που οδηγεί σε εξίσου υποβάθμιση των πραγματικών διαθέσιμων πόρων.

- Φαινόμενα παρεμβολής των επιδόσεων των εικονικών μηχανών. Κατά το Koh et al., 2007[14] πραγματοποιήθηκε μία μελέτη κατά την οποία ερευνάται η παρεμβολή στην απόδοση ενός αριθμού εφαρμογών που εκτελούνται πάνω σε ένα πειραματικό εικονικό περιβάλλον το οποίο επιλέχθηκε ειδικά για να γίνει κατηγοριοποίηση βάσει ανεξάρτητων και διαφορετικών μετρήσεων. Έπειτα από τη μελέτη των αποτελεσμάτων προκύπτει ότι η συνδυασμένη απόδοση ποικίλλει σημαντικά με διαφορετικούς συνδυασμούς των εφαρμογών. Εφαρμογές που σπάνια αλληλεπιδρούσαν μεταξύ τους δεν επηρέαζαν την απόδοση , εν αντιθέσει με εφαρμογές που όταν αλληλεπιδρούσαν μεταξύ τους έριχναν σημαντικά την απόδοση του εικονικού μηχανήματος. Επιπλέον, το virtualization είναι μια τεχνολογία που χρησιμοποιείται σε όλα τα κέντρα δεδομένων Συστημάτων Νεφών για να εξασφαλιστεί υψηλό ποσοστό χρησιμοποίησης των πόρων του υλικού και της καλύτερης διαχείρισης των εικονικών μηχανών. Παρά τα πλεονεκτήματα που παρέχονται από το virtualization δεν παρέχουν αποτελεσματική απομόνωση των επιδόσεων.

Όλες αυτές οι πτυχές συν το γεγονός ότι οι πάροχοι Υπηρεσιών Υπολογιστικών Νεφών (Cloud Providers) είναι ξεχωριστές οντότητες και δεν υπάρχουν διαθέσιμες πληροφορίες για την εσωτερική δομή και τη λειτουργία τους , καθιστά αναγκαίο να εξεταστεί μακροσκοπικά η συμπεριφορά ενός παρόχου όσον αφορά τους προσφερόμενους πόρους μέσα από μία σειρά από μετρήσεις . Η διαδικασία αυτή θα πρέπει να πραγματοποιηθεί μέσω της συγκριτικής αξιολόγησης , χρησιμοποιώντας τα κατάλληλα

εργαλεία και δοκιμές . Μία από τις βασικές πτυχές είναι ότι λόγω αυτής της δυναμικότητας στη διαχείριση των πόρων , η διαδικασία της συγκριτικής αξιολόγησης πρέπει να επαναλαμβάνεται την πάροδο του χρόνου , έτσι ώστε να μπορούμε να εξασφαλίσουμε όσο το δυνατόν διαφορετικά υλικά , διαφορετικές αποφάσεις διαχείρισης (όπως π.χ. ενημέρωση / αναδιάρθρωση / βελτίωση της υποδομής) αλλά και να τηρούνται τα βασικά χαρακτηριστικά , όπως η διακύμανση των επιδόσεων , η τυπική απόκλιση , κλπ. Τέλος , η αποκτηθείσα πληροφορία θα πρέπει να εκπροσωπείται με ένα κατανοητό τρόπο , ώστε να χρησιμοποιηθεί σε συστήματα λήψης αποφάσεων .

2.4 Δομή της Παρούσας Εργασίας

Στην παρούσα διπλωματική εργασία θα παρουσιάσουμε αρχικά τους λόγους που μας οδήγησαν να υλοποιήσουμε αυτή την εφαρμογή καθώς επίσης και τι έχει πραγματοποιηθεί/υλοποιηθεί μέχρι τώρα από την επιστημονική και εμπορική κοινότητα. Εν συνεχεία θα παρουσιάσουμε την λύση ολοκληρωμένη, τόσο το εργαλείο (tool) όσο και το GUI.Έπειτα θα παρουσιάσουμε την αρχιτεκτονική που ακολουθήθηκε για την υλοποίηση αυτής της εφαρμογής , καθώς επίσης και τη λειτουργικότητα της . Θα συνεχίσουμε με την αναλυτική περιγραφή της υλοποίησης της χρησιμοποιώντας class diagrams για την ευκολότερη κατανόηση της λογικής όσον αφορά το GUI.Έπειτα θα παρουσιάσουμε αναλυτικά γραφήματα με τα αποτελέσματα που εξήχθεισαν με μετρήσεις που πραγματοποιήσαμε, όπως και screenshots της εφαρμογής σε όλα τα βήματα της. Στο τέλος θα αναφερθούμε σε πιθανά σενάρια βελτιστοποίησης της εφαρμογής (future work).

3 ΑΝΑΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Στο κεφάλαιο αυτό θα ασχοληθούμε με τη θεωρητική προσέγγιση του προβλήματος , την ανάγκη δηλαδή, που οδήγησε στην εκπόνηση της παρούσας διπλωματικής εργασίας. Επίσης θα κάνουμε αναφορά στη βιβλιογραφία που αφορά αυτό το θέμα , όπου και έπειτα θα πραγματοποιηθεί μία σύντομη κριτική ανάλυσή της. Τέλος , θα αναφερθούμε στις μελέτες και εργασίες που έχουν πραγματοποιηθεί από άλλους ερευνητές (State of the Art) .

3.1 Ανάγκη (Problem Description)

Η διαδικασία εφαρμογής μετρήσεων υπολογιστικών συστημάτων , αποτελεί πρωταρχικό στόχο μέτρησης της απόδοσής τους αλλά και άλλων μη μετρήσιμων χαρακτηριστικών , ώστε να είναι δυνατή η σύγκριση των ιδίων με άλλα συστήματα ή πολλές φορές και με συστήματα που ακολουθούν βιομηχανικά προσυμφωνηθέντα πρότυπα.

Τα εργαλεία μετρήσεων υπολογιστικών επιδόσεων (benchmarks), πλέον είναι ένα καθιερωμένο πρότυπο στο ερευνητικό τοπίο της υψηλής υπολογιστικής απόδοσης[7]. Οι υπηρεσίες των Υπολογιστικών Νεφών , τείνουν να έχουν αναπτυχθεί και να είναι προσανατολισμένες , σε ένα ευρύ φάσμα πιθανών εφαρμογών, πολλές από τις οποίες

προσβλέπουν προς τη φιλοξενία επιχειρηματικών εφαρμογών. Καθώς η τεχνολογία των ΥΝ , γίνεται όλο και περισσότερο αποδεκτή , τόσο από επιχειρήσεις , όσο και από ιδιώτες, τόσο γίνεται και πιο επιτακτική η ανάγκη , για την δημιουργία σωστά-προσανατολισμένων εφαρμογών μετρήσεων (benchmark tools) , εργαλεία που θα προσφέρουν , σωστή και δίκαια αξιολόγηση τέτοιων συστημάτων , αλλά θα δημιουργούν και το κατώφλι από το οποίο μία τέτοια υπηρεσία θα είναι αποδεκτή και θα δύναται να βελτιστοποιηθεί. Η ποικιλία των επιλογών και των ρυθμίσεων των ΥΝ , και οι προσπάθειες που απαιτούνται για να φτάσουμε στο σημείο στο οποίο, παραδοσιακές εφαρμογές μετρήσεων μπορούν να εκτελεστούν , έχει διάφορες επιπτώσεις σχετικά με την ορθότητα της σύγκρισης - και , μάλιστα , για το ζήτημα της αναλογίας ποιότητα/κόστος (value for money). Τα εργαλεία συγκριτικών μετρήσεων των ΥΝ πρέπει να προσφέρουν δυνατότητες σύγκρισης σε όλα τα επίπεδα IaaS , καθώς επίσης πρέπει να προσφέρουν λεπτομερείς αναφορές για τον κύκλο ζωής του , εν χρήσει και προς εξέταση , ΥΝ σύστημα. Τα υπάρχοντα εργαλεία μετρήσεων (benchmark tools) δεν προσφέρουν αυτήν τη δυνατότητα σύγκρισης. Πρέπει , έστω και στο ελάχιστο , να είμαστε σε θέση να κατανοήσουμε , κατά πόσο ένα «μέτριο» σύστημα ΥΝ , αποδίδει στις απαιτήσεις του χρήστη η/και της εταιρείας , σε από την άποψη του πραγματικού bandwidth , ώστε ο τελικός χρήστης να γνωρίζει με μία ελάχιστη απόκλιση , το χρόνο μεταφόρτωσης και μετεγκατάστασης δεδομένων προς/από ένα ΥΝ σύστημα , αλλά και το χρόνο επικοινωνίας μεταξύ ξεχωριστών παρόχων ΥΝ. Εν συνεχεία θα ήταν απαραίτητο να γνωρίζουμε κατά πόσο η διαθέσιμη μνήμη, επεξεργαστική ισχύς , ταχύτητα και χωρητικότητα δίσκου , είναι αυτές τις οποίες ο πάροχος δηλώνει πως παρέχει ή όχι , και αν όχι κατά πόσο αποκλίνουν από την πραγματικότητα ,γεγονός το οποίο θα μπορούσε να οδηγήσει σε συμφόρηση των εικονικοποιημένων πόρων και κατ' επέκταση εκτέλεσης των εφαρμογών που βασίζονται σε αυτούς τους πόρους. Κατά τη σύναψη συμφωνίας μεταξύ παρόχου και πελάτη (ιδιώτης ή/και εταιρείας) , υπογράφεται ένα συμφωνητικό επιπέδου υπηρεσιών (Service Level Agreement – SLA). Η ανάγκη για την ύπαρξη αυτών των

ανεξάρτητων εφαρμογών μετρήσεων προέρχεται από την τήρηση των προσυμφωνηθέντων υπηρεσιών , από μέρους του παρόχου , ώστε ο πελάτης να χρησιμοποιεί τις υπηρεσίες με την ελάχιστη δυνατή απόκλιση από το SLA.

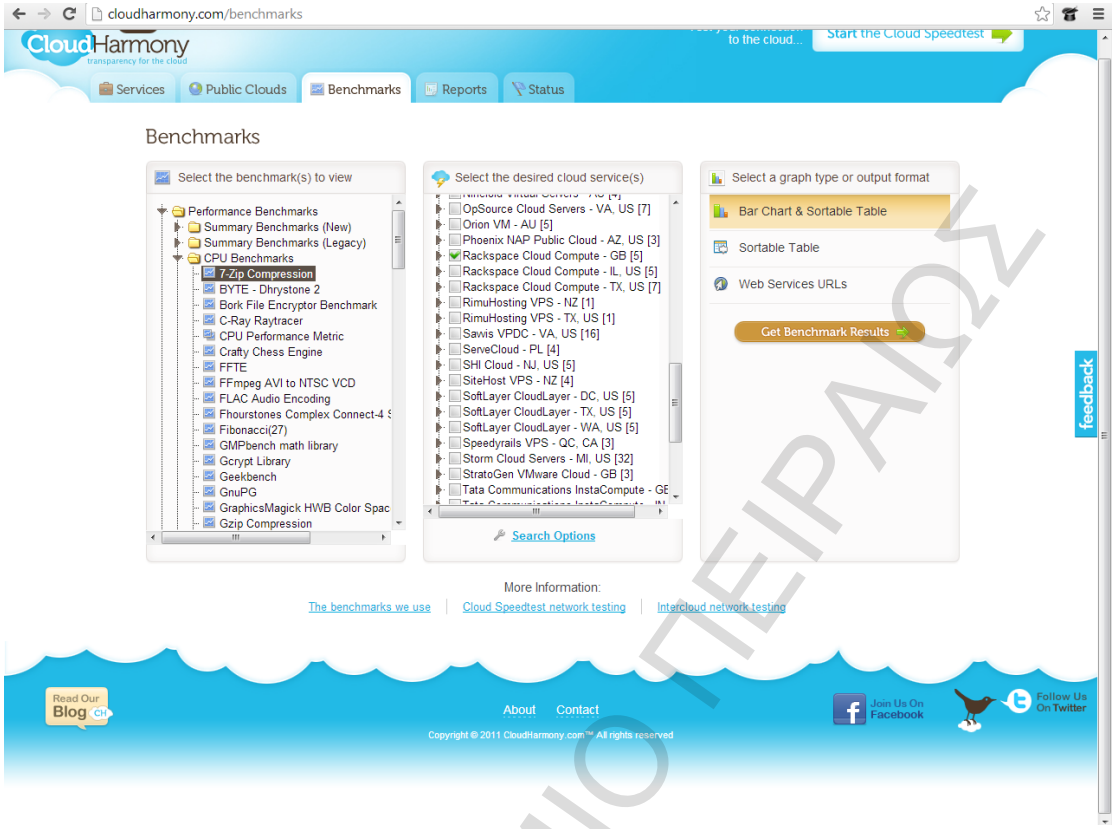
3.2Ερευνητικές Διεργασίες (State of the Art)

Στην πάροδο των ετών , έχουν πραγματοποιηθεί μία σειρά από επιστημονικές προσεγγίσεις στο θέμα των μετρήσεων συστημάτων ΥΝ , αλλά και εμπορικές εφαρμογές έχουν αναπτυχθεί για το σκοπό αυτό (συνήθως εμπορικοί ιστότοποι). Συγκεκριμένα όσον αφορά τα εμπορικά εργαλεία μετρήσεων , υπάρχουν και πάροχοι ΥΝ , που παρουσιάζουν τα συστήματά τους ταχύτερα/φθηνότερα/πιο αξιόπιστα σε σχέση αυτών των ανταγωνιστών τους.

Σε αυτήν την παράγραφο θα αναφερθούμε όσο δύναται πιο αναλυτικά στην προσπάθεια αυτή , με σκοπό εν τέλει να θεμελιώσουμε την αναγκαιότητα της παρούσας διπλωματικής εργασίας.

3.2.1 CLOUDHARMONY

Η εταιρεία CloudHarmony ασχολείται με την παρακολούθηση της διαθεσιμότητας υπηρεσιών ΥΝ από το 2009. Πραγματοποιείται παρακολούθηση όσον αφορά το IaaS για τις εταιρείες GoGrid , Rackspace, Amazon Web Services ,DynDns κ.α ενώ για το θέμα των PaaS υπηρεσιών παρακολουθεί τα Microsoft Azure και Google App Engine, AppFog, Stackable,Force.com Platform, Gandi Simple Hosting ,Google Sites[15].



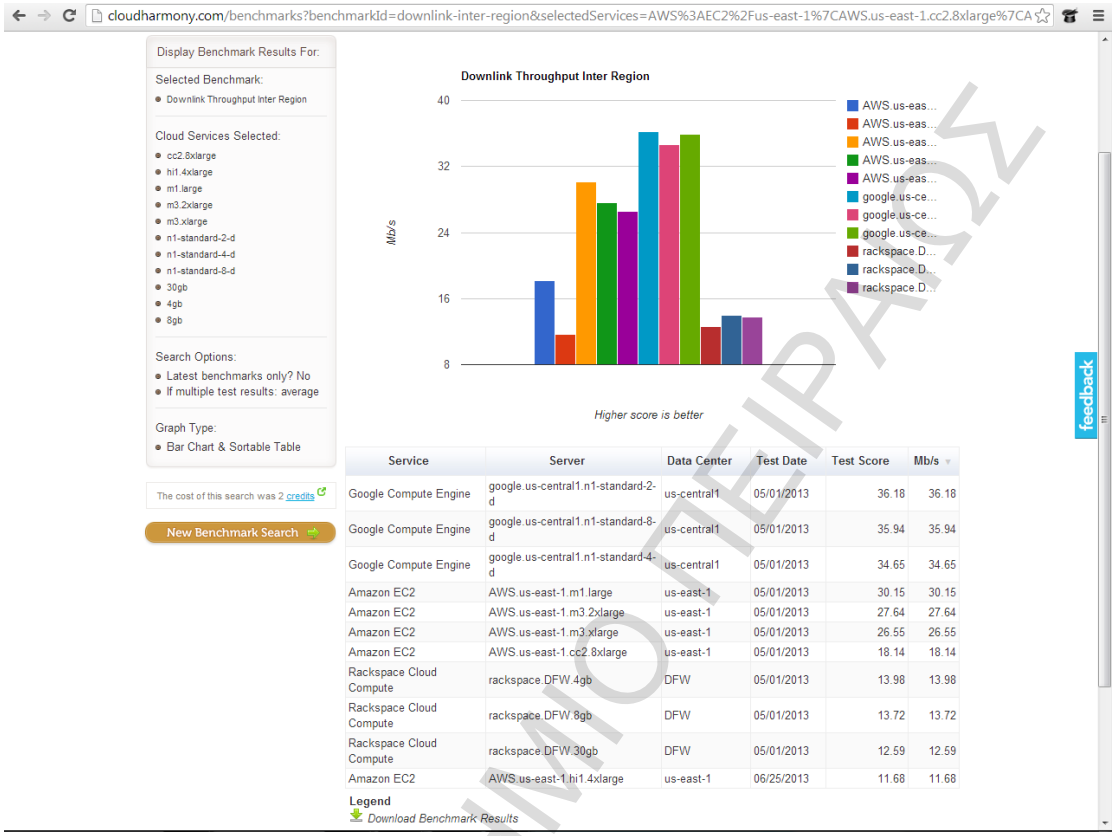
Εικόνα 2: CLOUDHARMONY[15]

Η ιστοσελίδα προσφέρει συγκριτικές δικιμές για τις παρακάτω κατηγορίες:

- Performance Benchmarks
  - Summary Benchmarks
  - CPU Benchmarks
  - Multi-Threaded CPU Benchmarks
  - IO Benchmarks
    - Disk IO
    - Memory IO
  - Application Benchmarks
    - Compilation
    - Compression
    - Database Server
    - Encoding
    - Interpreted Languages
    - Web/App Server
- Network Benchmarks
    - Throughput
    - Object Storage

Η απόδοση κάθε δικιμής που εκτελείται όποτε αυτό ζητηθεί, αποθηκεύεται με τη μορφή ιστορικού σε κάθε μια από τις προαναφερθείσες κατηγορίες , σε μία Βάση

Δεδομένων. Έτσι , όταν ένας χρήστης ζητήσει αναφορά κάποιων προηγουμένων δοκιμων, η πληροφορία ανακαλείται και παρουσιάζεται στο χρήστη με τη μορφή γραφήματος.

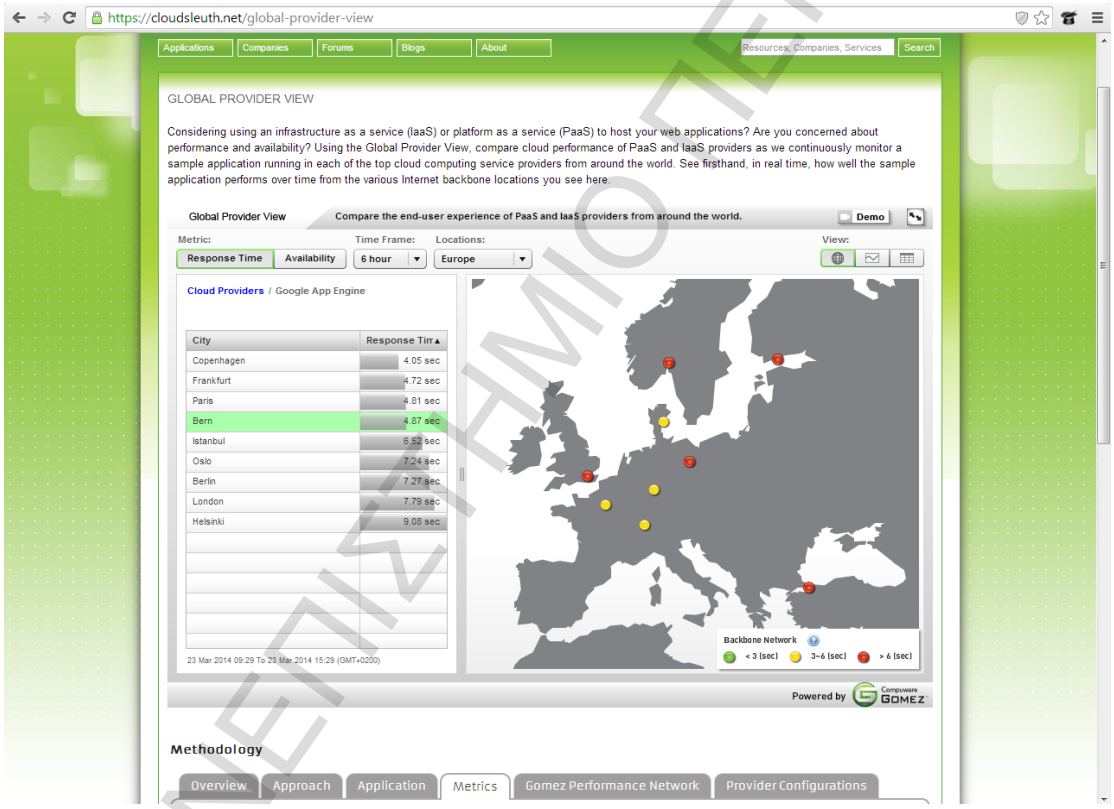


Εικόνα 3: Αποτελέσματα CLOUDHARMONY[15]

Δυστυχώς όμως όπως είναι σαφές και από την Εικ.3 δεν είναι δυνατόν να εξαχθούν ασφαλή και σωστά συμπεράσματα , για την αξιοπιστία των μετρήσεων μιας και αυτά έχουν εκτελεστεί πριν από τουλάχιστον 12 μήνες και δε γνωρίζουμε αν οι εταιρείες τις οποίες ο χρήστης έχει επιλέξει έχουν αναβαθμίσει τόσο το υλικό τους όσο και το λογισμικό τους για καλύτερες αποδόσεις. Επιπλέον δεν υπάρχει ένδειξη σχετικά για το αν κάθε σημείο που έχει επιλεγεί για κάθε μέτρηση , έχει επιλεγεί ως μέσο σημείο ή ως καλύτερο ή χειρότερο για το συγκεκριμένο πάροχο[15].

3.2.2 CLOUDSLEUTH

Η εταιρεία Cloudsleuth προσφέρει τη δυνατότητα απεικόνισης των χρόνων απόκρισης και διαθεσιμότητας για τους παρόχους υπηρεσιών ΥΝ , αποτυπώνοντας την πληροφορία γραφικά στον παγκόσμιο χάρτη. Η παρακάτω εικόνα δείχνει ένα παράδειγμα του χρόνου απόκρισης υπηρεσιών PaaS και IaaS για την Ευρώπη τις τελευταίες έξι(6)ώρες για τον πάροχο Google App Engine[15].



Εικόνα 4:CLOUDSLEUTH[15]

Ωστόσο δεν αναφέρεται πουθενά κάποια συγκεκριμένες δοκιμες, ή κάποια επίπεδα αναφοράς προκειμένου να πραγματοποιηθεί ορθή σύγκριση και να εξαχθούν ασφαλή συμπεράσματα για το χρόνο απόκρισης.

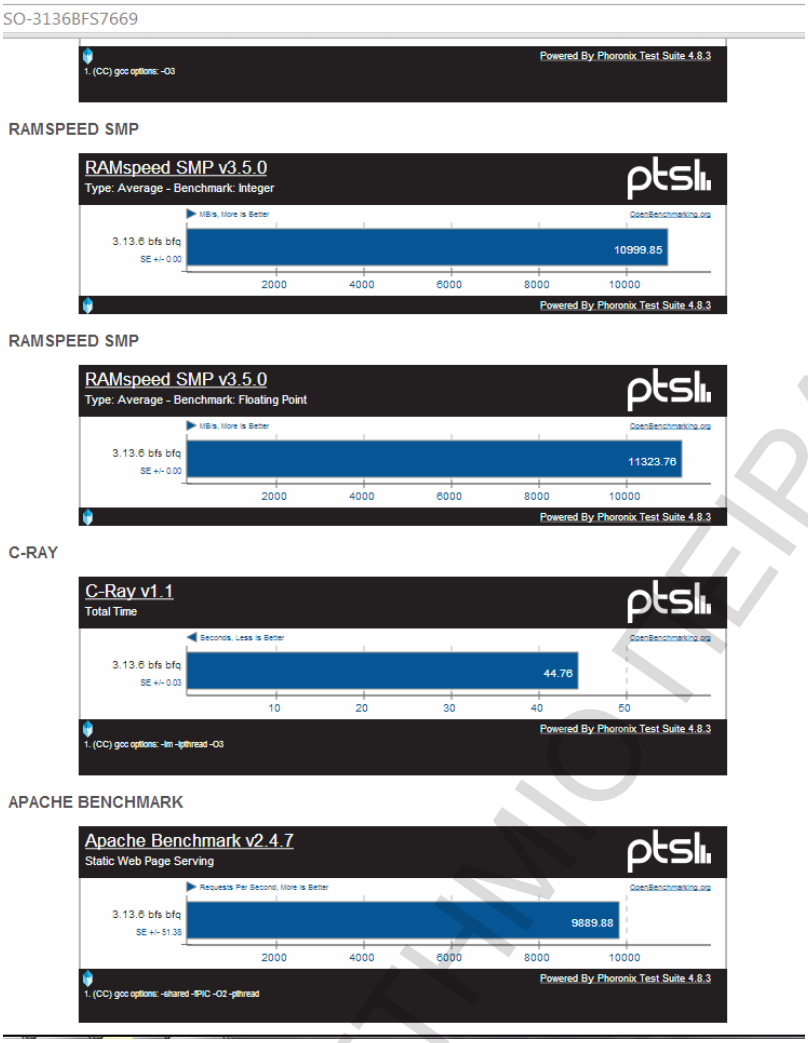
3.2.3 OpenBenchmarking.org

Η ιστοσελίδα openbenchmarking.org προσφέρει πληροφορίες από διάφορες συγκριτικές δοκιμές που έχουν πραγματοποιηθεί σε παρόχους ΥΝ με βάση το Phoronix Test Suite. Τα αποτελέσματα αυτών των δοκιμών είναι διαθέσιμα και αφορούν κυρίως τους παρόχους AWS και Rackspace. Ωστόσο οι πληροφοριες που παρέχονται είναι ελλιπείς. Στην παρακάτω εικόνα παρουσιάζουμε τα αποτελέσματα σύγκρισης για Apache όπως χαρακτηριστικά αναγράφεται στην ιστοσελίδα[16].

Χαρακτηριστικά της δοκιμής:

Εικόνα 5: BFS[16]

Αποτελέσματα:



Εικόνα 6: Αποτελέσματα BFS[16]

Δυστυχώς όμως, όπως φαίνεται ενώ τα αντίστοιχα αποτελέσματα είναι επαρκή , παρουσιάζονται με έναν απλό τρόπο, χωρίς να φαίνονται ιδιαίτερες λεπτομέρειες των συγκριτικών δοκιμών που πραγματοποιήθηκαν.

3.2.4 Ερευνητικές Εργασίες

Συσχετιζόμενες εργασίες , με την διπλωματική που παρουσιάζουμε έχουν πραγματοποιηθεί στα πλαίσια των μελετών επιδόσεων ΥΝ. Στο άρθρο (Garg, 2012)[21] , παρουσιάζεται ένα πολύ-επίπεδο framework συγκριτικών δοκιμών, κατά το οποίο μελετώνται η ευκινησία , η διαθεσιμότητα , η απόδοση , η ασφάλεια και το κόστος , παρόχων (agility, availability, accountability, performance, security and cost).Το SkyMark είναι ένα Framework το οποίο έχει αναπτυχθεί για την ανάλυση των επιδόσεων συστημάτων ΥΝ στο επίπεδο IaaS. Αρχικά το SkyMark παρέχει υπηρεσίες για τη διαχείριση των εισερχόμενων αιτήσεων και των μισθωμένων πόρων από τον πάροχο ΥΝ. Για το πρώτο υο SkyMark παρέχει απλές ή πολλαπλές ουρές εργασίας , ανάλογα με τις ρυθμίσεις της συγκριτικής δοκιμής και κάθε ουρά εργασίας υποστηρίζει μία ποικιλία από απλές πολιτικές προγραμματισμού (πχ FCFS [9]). Για το τελευταίο το εργαλείο SkyMark υποστηρίζει μία σειρά από δυναμικές και στατικές πολιτικές διαχείρισης των υπολογιστικών πόρων. Το SkyMark υποστηρίζει πολύπλοκα φορτία εργασιών (Δομικό στοιχείο Grenchmark) (complex workloads) . Κάθε εργασία χωρίζεται σε ξεχωριστές οντότητες και κάθε οντότητα ορίζεται από τον πόρο στο οποίο πρόκειται να κατανεμηθεί (Δομικό στοιχείο C-Meter).

Δεύτερο εργαλείο που έχει αναπτυχθεί για το σκοπό των συγκριτικών μετρήσεων είναι το CloudCmp (Li, 2010)[17]. Παρέχει μία μεθοδολογία και αποσκοπεί στο να επιτευχθεί η καλύτερη αναλογία απόδοσης/κόστους για μία εφαρμογή που δύναται να τρέχει σε ένα σύστημα ΥΝ. Ένας πιθανός πελάτης ΥΝ μπορεί να χρησιμοποιήσει το αποτελέσματα των δοκιμών για να συγκρίνει διάφορους παρόχους και τελικά να κρίνει αν θα πρέπει να μεταβεί στο νέφος και ποιος πάροχος είναι ο ιδανικός για τις απαιτήσεις των εφαρμογών που ο πελάτης εκτελεί. Οι δοκιμές που εκτελούνται στο CloudCmp framework έχουν σχέση με θέματα ταχύτητας εκτέλεσης υπολογισμών (compute) , αποθήκευσης (persistent storage) , καθώς με intra-cloud και widearea networking. Με το προαναφερθέν

εργαλείο , μπορεί να δοθεί σαφής επεξήγηση που οφείλεται η διαφορά στην απόδοση μεταξύ παρόχων , έτσι ώστε οι τελευταίοι να προχωρούν σε στοχευμένες κινήσεις προώθησης των υπηρεσιών τους.

Εν γένει η πλατφόρμα CloudCmp εστιάζει στα παρακάτω:

1. *Επιλογή του σωστού παρόχου (Guide a customer’s choice of provider)*
2. *Συσχετισμός της συγκεριμένης υπηρεσίας με τις απαιτήσεις (Relevant to cloud providers)*
3. *Δίκαιες μετρικές και εξορθολογισμένα αποτελέσματα μεταξύ των παρόχων (Fair)*
4. *Πληρότητα στη συχνότητα των μετρήσεων (Thoroughness vs. measurement cost)*
5. *Μέγιστη δυνατή κάλυψη των υπαρχόντων παρόχων υπηρεσιών YN (Coverage vs. development cost)*
6. *Συμμόρφωση με τους κανόνες χρήσης του εκάστοτε παρόχου (Compliant with acceptable use policies)*

Η HIBENCH SUITE [10] είναι μία εφαρμογή εκτέλεσης συγκριτικών δοκιμών σε συστήματα YN που περιλαμβάνουν αποκλειστικά και μόνο το Hadoop. Είναι μία νέα , ρεαλιστική και ολοκληρωμένη σουίτα μετρήσεων για το Hadoop. Αποτελείται από ένα σύνολο προγραμμάτων για το Hadoop , συμπεριλαμβανομένων τεχνητών μικρο-συγκρίσεων και real-time μετρήσεων. Οι μετρήσεις που γίνονται αφορούν την ταχύτητα , το χρόνο εκτέλεσης των διεργασιών, το throughput , τον αριθμό των ολοκληρωμένων δειεργασιών, τη χωρητικότητα του δικτύου , τη χρησιμοποίηση του συστήματος πόρων καθώς και τους τρόπους πρόσβασης στα δεδομένα (speed , job running time, throughput ,number of tasks completed per minute,bandwidth, system resource utilizations, data access patterns).

3.3 Κύρια Συνεισφορά Εργασίας

Στα πλαίσια της διπλωματικής εργασίας , θεωρούμε πως με την δημιουργία του συγκεκριμένου benchmark framework, είμαστε σε θέση να προσφέρουμε μία ολοκληρωμένη λύση , στο θέμα των συγκριτικών δοκιμών, για συστήματα YN. Ο σκοπός του παρόντος εγγράφου είναι η παροχή μηχανισμών για την αντιμετώπιση των προαναφερόμενων θεμάτων (παρ. 1.2) . Ένα πλαίσιο συγκριτικής αξιολόγησης παρουσιάζεται προκειμένου να μετρηθεί η ικανότητα των διαφόρων παρόχων YN σε ένα φάσμα εφαρμογών , ειδικότερα δε σε βάσεις δεδομένων, web serving υπηρεσίες και τεχνολογίες map – reduce . Η συγκεκριμένη εφαρμογή Benchmark είναι προσανατολισμένη στην εφαρμογή και όχι στον πάροχο. Επιπρόσθετα έχουν εφαρμοσθεί τεχνολογίες υλοποίησης και προγραμματισμού up to date , προσφέροντας στο χρήστη βήμα- προς βήμα καθοδήγηση για την εκτέλεση της δοκιμής , ενώ με την ολοκλήρωση του, εμφανίζεται γραφική απεικόνιση των αποτελεσμάτων , ενώ όλα τα benchmark tests που έχουν πραγματοποιηθεί και τα αποτελέσματα αυτών , φυλάσσονται σε μία Sql DB , προκειμένου ο χρήστης να ανατρέξει σε αυτά κατά το δοκούν.

4 ΤΟ ΕΡΓΑΛΕΙΟ

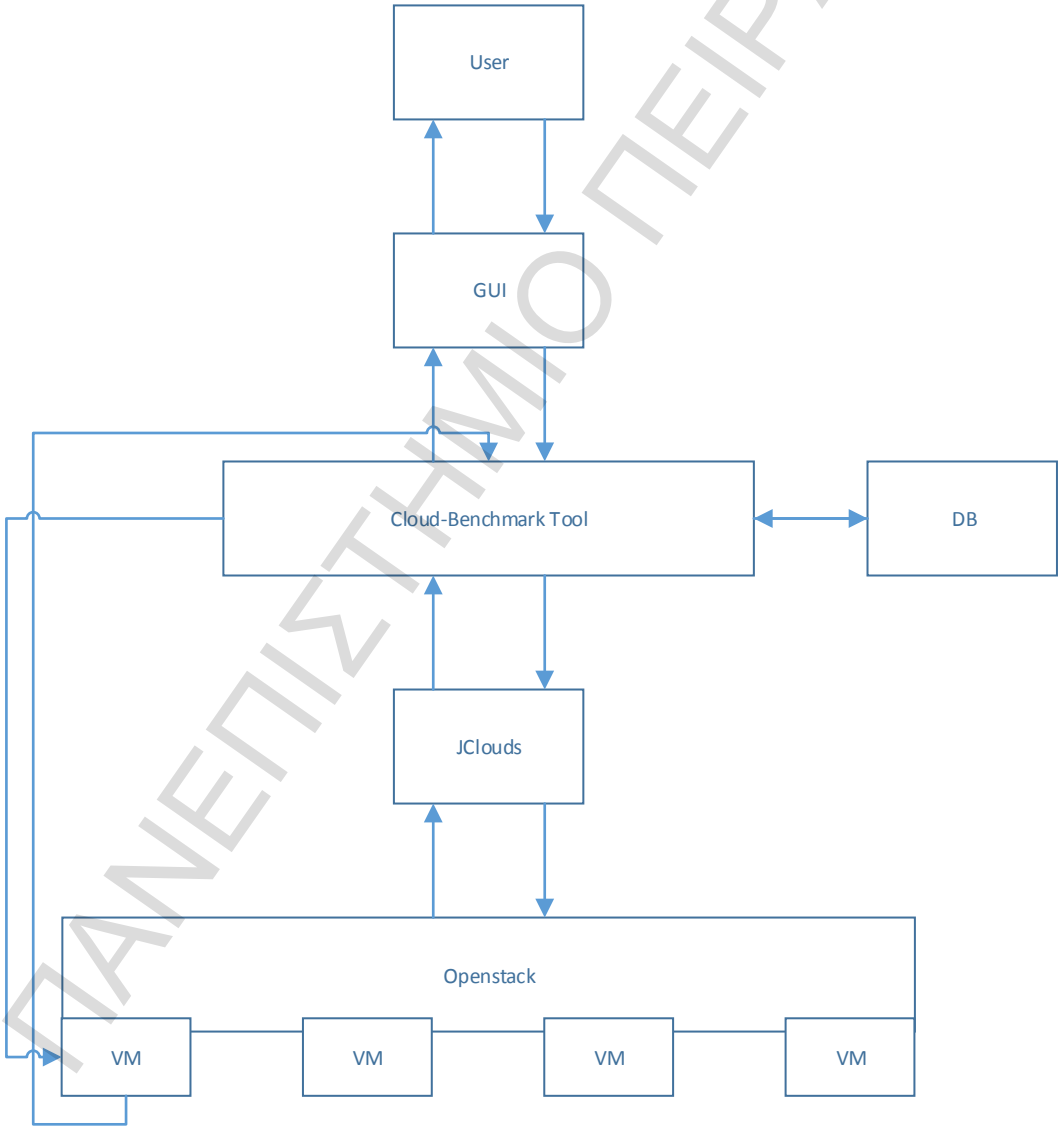
Στο κεφάλαιο αυτό θα παρουσιάσουμε την εφαρμογή που έχουμε υλοποιήσει. Συγκεκριμένα. Θα αναλύσουμε την αρχιτεκτονική και τις τεχνολογίες που ακολουθήσαμε και χρησιμοποιήσαμε , έπειτα θα περάσουμε σε μία λεπτομερή περιγραφή της λειτουργικότητας της εφαρμογής , ενώ τέλος θα επεξηγήσουμε αναλυτικά πως υλοποιήθηκε η εν λόγω εφαρμογή.

4.1 Ολοκληρωμένη Λύση Εφαρμογής

Στο παρακάτω σχεδιάγραμμα παρουσιάζουμε με ποιον τρόπο λειτουργεί η εφαρμογή που έχουμε δημιουργήσει. Πιο συγκεκριμένα ο χρήστης με την είσοδό του στο Γραφικό περιβάλλον της εφαρμογής (GUI) , μπορεί να εκτελέσει κάποιο από τις συνολικές τρία (3) δοκιμές. Με την επιλογή κάποιου , το εργαλείο που βρίσκεται στο backend (Cloud Benchmark Tool) , χρησιμοποιεί σαν ενδιάμεσο το open-source πρόγραμμα JClouds , που είναι υπεύθυνο για να επικοινωνεί με τον αντίστοιχο cloud provider.

Όταν δημιουργηθεί ένα εικονικό μηχάνημα(VM) στον provider(στην υλοποίησή μας Openstack) , πλέον η επικοινωνία γίνεται μόνο μεταξύ VM και Tool. Με την επιτυχή ολοκλήρωση ενός test , τα αποτελέσματα παρουσιάζονται στην οθόνη του χρήστη , παράλληλα όμως αποθηκεύονται και σε μία βάση δεδομένων , όπου και διατηρούνται για

περαιτέρω πιθανή μελέτη. Εδώ θα πρέπει να κάνουμε σαφές , πως το Tool είναι σε θέση να εκτελεί πολλαπλά test , τόσο με συγκεκριμένη συχνότητα αλλά και με επαναλήψεις κάθε φορά (πχ ανά τρεις ώρες να εκτελούνται 5 δοκιμές για DB). Στο χρήστη όμως εμφανίζονται τα αποτελέσματα κάθε φορά της τελευταίας δοκιμής. Επίσης, παρέχεται στον χρήστη η δυνατότητα να δει αποτελέσματα προηγουμένων δοκιμών που έχει εκτελέσει ώστε να μπορεί να αναλύσει τα αποτελέσματα αυτά σε καλύτερο βάθος συγκρίνοντας τα με διάφορες παραμέτρους.



Εικόνα 7 Ολοκληρωμένη Λύση

4.2 Εργαλεία και Αρχιτεκτονική Υλοποίησης των δοκιμών(Benchmarks)

Για την υλοποίησης της παρούσας εφαρμογής εκτός από το κώδικα που γράφτηκε για την κυρία λειτουργικότητα χρησιμοποιήθηκαν και αλλά εργαλεία (framework) για να είναι δυνατόν η εφαρμογή να λειτούργει σαν ένα σύνολο. Τα εργαλεία αυτά είναι ανεξάρτητα του κώδικα που γράφτηκε και μπορούν να χρησιμοποιηθούν καλώντας το απλό API ή REST API τους . Τα εργαλεία αυτά είναι τα έξης:

- Οι δοκιμές (benchmarks) που χρησιμοποιούμε για να τρέξουμε τις δόκιμες και να εξετάσουμε την υποδομή του περιβάλλοντος υπολογιστικού νέφους ανά είδος εφαρμογής (Βάσεις δεδομένων, Web Serving, MapReduce).
- Ενδιάμεσο framework για επικοινωνία με περιβάλλον υπολογιστικού νέφους .
- Περιβάλλον υπολογιστικού νέφους που θα υπόκεινται σε δόκιμες.

Τα εργαλεία αυτά περιγράφοντα παρακάτω διεξοδικά καθώς και η αρχιτεκτονική και υλοποίηση τους μέσα στην εφαρμογή.

4.2.1 BENCHMARKS

Τα είδη των εφαρμογών που επιλεχτήκαν για να δοκιμαστούν στο περιβάλλον υπολογιστικού νέφους είναι Βάσεις δεδομένων, Web Serving και MapReduce και τα αντίστοιχα Benchmarks με τα οποία θα εξετάσουμε την υποδομή του περιβάλλοντος υπολογιστικού νέφους είναι YCSB, Weighttp για εφαρμογές MapReduce TESTDFSIO, MRBench, NNBench και Teragen-Terasort.

4.2.1.1 YCSB

Το εργαλείο YCSB (Yahoo Cloud Serving Benchmark) αναπτύχτηκε από την εταιρεία Yahoo και είναι ένα framework για τη συγκριτική αξιολόγηση των συστημάτων αποθήκευσης δεδομένων (data store systems) . Το framework αυτό περιέχει διασυνδέσεις (interfaces) για να ανεβάσει δεδομένα και να εξετάσει την αποδοτικότητα πολλών δημοφιλών data store systems.

Ο στόχος του YCSB είναι να αναπτύξει ένα framework και ένα κοινό σύνολο φόρτων εργασίας(workload) για την αξιολόγηση της απόδοσης των διαφόρων NoSQL και "cloud" data store systems καθώς και παραδοσιακών Βάσεων Δεδομένων[18]. Το framework αυτό αποτελείται από :

- Τον YCSB Πελάτη (Client), μια επεκτάσιμη γεννήτρια workload
- Τον Πυρήνα του workload (Core workloads), μια σειρά από σενάρια workload που μπορούν να εκτελεστούν από τη γεννήτρια

Παρά το γεγονός ότι τα core workloads μπορούν να παρέχουν μια ολοκληρωμένη εικόνα της απόδοσης ενός συστήματος , ο Client είναι επεκτάσιμος , ώστε να υπάρχει η δυνατότητα να ορίσουμε ένα νέο και διαφορετικό workload για να εξετάσει τις πτυχές του συστήματος , ή τα σενάρια μιας εφαρμογής τα οποία δεν καλύπτονται επαρκώς από τα core workloads . Ομοίως , ο Πελάτης είναι επεκτάσιμος ώστε να υπάρχει η δυνατότητα υποστήριξης συγκριτικής αξιολόγησης διαφορετικών βάσεων δεδομένων . Με αλλά λόγια αν δεν υπάρχει Client για μια βάση δεδομένων είναι πολύ εύκολο να δημιουργήσεις ένα νέο Client για αυτήν την βάση δεδομένων [18].

Υπάρχουν πολλές βάσεις δεδομένων που το YCSB υποστηρίζει , όπως οι παρακάτω:

PNUTS	CouchDB	GemFire
BigTable	Voldemort	GigaSpaces XAP
HBase	MongoDb	DynamoDB
Hypertable	Infinispan	MySql
Azure	Dynomite	Postgre
Cassandra	Redis	

Υπάρχουν 6 βήματα για να εκτελεστεί ένα workload:

1) Ρύθμιση συστήματος βάσης δεδομένων που θα ελεγχθεί

2) Επιλογή του κατάλληλου interface Βάσης Δεδομένων

3) Επιλογή του κατάλληλου workload

4) Επιλογή των κατάλληλων παραμέτρων εκτέλεσης (αριθμός threads , στόχος, μέγεθος δεδομένων κλπ.)

5) Φόρτωση δεδομένων (load phase)

6) Εκτέλεση του workload (transaction phase)

Βήμα 1. Ρύθμιση συστήματος βάσης δεδομένων που θα ελεγχθεί

Το πρώτο βήμα είναι να ρυθμιστεί το σύστημα βάσης δεδομένων που θέλουμε να αξιολογήσουμε. Αυτό μπορεί να γίνει σε μια μοναδική βάση δεδομένων ή ένα σύμπλεγμα (cluster) βάσεων δεδομένων , ανάλογα με τη διαμόρφωση που θέλουμε να συγκρίνουμε .

Θα πρέπει επίσης να δημιουργήσουμε ή να ρυθμίσουμε τα πινάκες ή τα keyspaces ή τα storage buckets για την αποθήκευση εγγραφών. Οι λεπτομέρειες διαφέρουν ανάλογα με το κάθε σύστημα βάσης δεδομένων, και εξαρτάται από το workload που θέλουμε να εκτελέσουμε . Πριν τρέξει ο YCSB Client, πρέπει να δημιουργηθούν οι πίνακες , αφού ο ίδιος ο Client δεν θα ζητήσει τη δημιουργία των πινάκων. Αυτό οφείλεται στο γεγονός ότι για κάποια συστήματα , υπάρχει ένα χειροκίνητο στάδιο δημιουργίας πινάκων, και για άλλα

συστήματα , ο πίνακας θα πρέπει να δημιουργηθεί πριν από την εκκίνηση του συμπλέγματος βάσεων δεδομένων [18].

Βήμα 2. Επιλογή του κατάλληλου interface Βάσης Δεδομένων

Το στρώμα διεπαφής (interface layer) της Βάσης Δεδομένων, είναι μια κλάση Java που εκτελεί ανάγνωση, εισαγωγή, ενημέρωση και διαγραφή και μετατρέπει τις κλήσεις που δημιουργούνται από τον YCSB Client σε κλήσεις για το API της Βάσης Δεδομένων. Θα πρέπει να καθορίσουμε το όνομα της κλάσης του interface layer στη γραμμή εντολών όταν τρέχουμε τον YCSB Client και ο Client θα φορτώσει δυναμικά την κλάση του interface. Οποιεσδήποτε ιδιότητες που ορίζονται στη γραμμή εντολών , ή σε αρχεία παραμέτρων που έχουν καθοριστεί στη γραμμή εντολών , θα να περάσουν στο interface instance της Βάσης Δεδομένων (για παράδειγμα, για να δηλωθεί το hostname ή την πόρτα που ακούει η βάση δεδομένων που βρίσκεται υπό αξιολόγηση). Επίσης πρέπει να φορτωθούν στο YCSB οι σωστοί connectors της κάθε Βάσης[18].

Βήμα 3. Επιλογή του κατάλληλου workload

Το workload καθορίζει τα στοιχεία που θα πρέπει να φορτωθούν στη Βάση Δεδομένων κατά τη φάση του load, καθώς και τις εργασίες που θα εκτελεστούν στα σύνολα των δεδομένων στη φάση του transaction.

Τυπικά, ένας φόρτος εργασίας είναι ένας συνδυασμός από:

- Μία κλάση Java workload
- Αρχείο παραμέτρων (σε μορφή επιλογών Java)

Επειδή οι ιδιότητες του συνόλου δεδομένων πρέπει να είναι γνωστές κατά τη διάρκεια της φάσης φόρτωσης (έτσι ώστε να μπορεί να κατασκευαστεί και να τοποθετηθεί

το κατάλληλο είδος εγγραφής) και κατά τη φάση της συναλλαγής (έτσι ώστε να μπορούν να αναφέρονται οι σωστές ταυτότητες εγγραφών και τα πεδία) ένα ενιαίο σύνολο ιδιοτήτων μοιράζεται μεταξύ των δύο φάσεων . Έτσι το αρχείο παραμέτρων χρησιμοποιείται και στις δύο φάσεις . Η κλάση Java του workload χρησιμοποιεί αυτές τις ιδιότητες είτε για την εισαγωγή εγγραφών (φάση load) είτε για την εκτέλεση συναλλαγών (φάση transaction) . Η επιλογή της φάσης που θα τρέξει βασίζεται σε μια παράμετρο που καθορίζετε κατά την εκτέλεση της εντολής ycsb [18].

Καθορίζουμε την κλάση Java workload και το αρχείο παραμέτρων στη γραμμή εντολών όταν τρέψουμε τον YCSB Client . Ο Client θα φορτώσει δυναμικά την κλάση του workload, θα του περάσει τις ιδιότητες από το αρχείο παραμέτρων (και τυχόν πρόσθετες ιδιότητες που έχουν καθοριστεί στη γραμμή εντολών) και στη συνέχεια θα εκτελέσει το workload . Αυτό συμβαίνει τόσο στη φάση load όσο και στη φάση transaction, καθώς ισχύουν και για τις δύο οι ίδιες ιδιότητες και η ίδια λογική του workload. Για παράδειγμα , αν η φάση φόρτωσης δημιουργεί εγγραφές με 10 πεδία , τότε η φάση της συναλλαγής πρέπει να γνωρίζει ότι υπάρχουν 10 τομείς τους οποίους μπορεί να χρησιμοποιήσει για ερωτήματα και επεξεργασία.

Βήμα 4. Επιλογή των κατάλληλων παραμέτρων εκτέλεσης

Παρά το γεγονός ότι η κλάση workload και το αρχείο παραμέτρων καθορίζουν ένα συγκεκριμένο workload , υπάρχουν πρόσθετες ρυθμίσεις που μπορεί να θελήσουμε να ορίσουμε για μια συγκεκριμένη εκτέλεση του benchmark . Αυτές οι ρυθμίσεις παρέχονται στη γραμμή εντολών όταν τρεχουμε τον YCSB Client. Αυτές οι ρυθμίσεις είναι :

- threads : ο αριθμός των νημάτων του Client. Από προεπιλογή , ο YCSB Client χρησιμοποιεί ένα νήμα, αλλά μπορούν να καθοριστούν πρόσθετα θέματα . Αυτό

γίνεται συχνά για να αυξηθεί η ποσότητα του φορτίου που προσφέρεται στη βάση δεδομένων .

- target : ο αριθμός -στόχος των πράξεων ανά δευτερόλεπτο . Από προεπιλογή , ο YCSB Client θα προσπαθήσει να κάνει όσες εργασίες μπορεί. Για παράδειγμα , αν κάθε λειτουργία χρειάζεται 100 χιλιοστά του δευτερολέπτου , κατά μέσο όρο , ο πελάτης θα κάνει περίπου 10 πράξεις ανά δευτερόλεπτο ανά εργαζόμενο νήμα . Ωστόσο , μπορουμε να αυξήσουμε τον αριθμό-στόχο των πράξεων ανά δευτερόλεπτο.
- s : Κατάσταση. Για μεγάλο χρονικά workload , μπορεί να είναι χρήσιμο να υπάρχει η αναφορά κατάστασης του Πελάτη , απλά για να μας διαβεβαιώσει ότι δεν έχει καταρρεύσει και να μας δώσει μια ιδέα της προόδου του . Με την ένδειξη " - s" στη γραμμή εντολών , ο Client θα αναφέρει την κατάστασή του κάθε 10 δευτερόλεπτα στο.

Βήμα 5. Φόρτωση δεδομένων (load phase)

Για να φορτώσετε τα δεδομένα , πρέπει να τρέχουμε το YCSB Client και να επιλέγουμε να εκτελέσει το τμήμα φόρτωσης .

Για παράδειγμα για να φορτώσουμε το προκαθορισμένο σύνολο δεδομένων εκτελουμε την έξης εντολη:

$ ./bin/ycsb load basic -P workloads/workloada

- Η παράμετρος load λέει στο Client να εκτελέσει το τμήμα φόρτωσης του workload .
- Η basic παράμετρος λέει στο Client να χρησιμοποιήσει το ψεύτικο BasicDB interface layer.
- Η παράμετρος "- P " χρησιμοποιείται για να φορτώσουμε τα αρχεία ιδιοτήτων . Σε αυτή την περίπτωση, το χρησιμοποιήσαμε για να φορτωθεί το αρχείο παραμέτρων του workloada.

Τα βασικά αρχεία παραμέτρων των workloads δημιουργούν πολύ μικρές βάσεις δεδομένων. Για παράδειγμα, το workloada δημιουργεί μόνο 1.000 εγγραφές . Ωστόσο, για να εκτελέσουμε μια πραγματική συγκριτική δοκιμή θα πρέπει να δημιουργήσουμε μια πολύ μεγαλύτερη βάση δεδομένων . Για παράδειγμα , εάν θέλουμε να φορτώσουμε 100 εκατομμύρια εγγραφές θα χρειαστεί να παρακάμψουμε την προεπιλεγμένη " recordcount " παράμετρο στο αρχείο workloada [18]. Αυτό μπορεί να γίνει με έναν από δύο τρόπους:

- -Καθορισμός ενός νέου αρχείου ιδιοτήτων που περιέχει μια νέα τιμή της recordcount . Εάν αυτό το αρχείο έχει καθοριστεί στη γραμμή εντολών μετά το αρχείο workloada , θα υπερισχύει κάθε ιδιότητας του workloada αρχείου.
- Καθορισμός μια νέα τιμή της ιδιότητας recordcount στη γραμμή εντολών . Οποιεσδήποτε ιδιότητες που καθορίζονται στη γραμμή εντολών παρακάμπτουν αυτές που ορίζονται στα αρχεία ιδιοτήτων.

Επειδή η φόρτωση μεγάλου αριθμού δεδομένων θα πάρει πολύ χρόνο μπορούμε να ζητήσουμε από τον Client να αποθηκεύσει τα αποτελέσματα της δοκιμής σε ένα αρχείο. Για παράδειγμα αποθηκεύουμε τα αποτελέσματα στο αρχείο load.dat:

```
$ ./bin/ycsb load basic -P workloads/workloada -P large.dat -s > load.dat
```

Βήμα 6. Εκτέλεση του workload (transaction phase)

Όταν φορτωθούν τα δεδομένα από το προηγούμενο βήμα , μπορούμε να εκτελέσουμε το workload . Αυτό γίνεται λέγοντας στο Client να τρέξει το τμήμα transaction του workload . Αυτό το κάνουμε με την ακόλουθη εντολή :

```
$ ./bin/ycsb run basic -P workloads/workloada -P large.dat -s > transactions.dat
```

Η κύρια διαφορά σε σχέση με τη φάση load είναι ότι χρησιμοποιήσαμε την παράμετρο run για να πούμε στον Client να χρησιμοποιήσει το τμήμα του transaction αντί του τμήματος load ..

Συνήθως θέλουμε να χρησιμοποιήσουμε τις παραμέτρους -threads και -target για να ελέγξουμε την ποσότητα του προσφερόμενου load. Για παράδειγμα, θα μπορούσαμε να θέλουμε 10 threads να επιχειρήσουν συνολικά 100 πράξεις ανά δευτερόλεπτο (π.χ. 10 πράξεις ανά sec ανά thread.) Από τη στιγμή που η μέση καθυστέρηση δεν είναι πάνω από 100 ms , κάθε thread θα είναι σε θέση να εκτελεί τις προβλεπόμενες του 10 πράξεις ανά δευτερόλεπτο . Σε γενικές γραμμές , θα πρέπει να έχουμε αρκετά threads, έτσι ώστε κανένα threadsνα μην προσπαθεί να κάνει περισσότερες πράξεις ανά δευτερόλεπτο από ότι είναι δυνατόν, αλλιώς το throughput, θα είναι μικρότερο από την καθορισμένο στόχο. Για αυτό το παράδειγμα μπορούμε να εκτελέσουμε τον εξής εντολή :

$ ./bin/ycsb run basic -P workloads/workloada -P large.dat -s -threads 10 -target 100 > transactions.dat

Εναλλακτικά , οι τιμές -threads και -target μπορούν να οριστούν στο αρχείο παραμέτρων χρησιμοποιώντας τις επιλογές threadcount και target αντίστοιχα . Για παράδειγμα :

threadcount=10

target=100

Στο τέλος των φάσεων load και transaction, ο Client θα δώσει αναφορά σχετικά με τα στατιστικά στοιχεία της απόδοσης της δοκιμής. Στο βήματα 5 και 6 , τα στατιστικά αυτά θα γραφτούν στο αρχείο load.dat και transactions.dat αντίστοιχα . Η προεπιλογή είναι να παράγει τα εξής: overall runtime, throughput, average, min, max, 95th και 99th percentile latency για κάθε τύπο λειτουργίας (εισαγωγή, ανάγνωση, ενημέρωση , κλπ.) , και την καταμέτρηση των κωδικών επιστροφής (return codes)για κάθε πράξη . Οι κωδικοί επιστροφής έχουν οριστεί από το interface layer της βάσης δεδομένων, και μας επιτρέπουν να δουμε αν υπήρχαν λάθη κατά τη εκτέλεση του workload . Ένα αρχείο αποτύπωσης αποτελεσμάτων θα ήταν το παρακάτω:

[OVERALL],RunTime(ms), 10110

[OVERALL],Throughput(ops/sec), 98.91196834817013

[UPDATE], Operations, 491

[UPDATE], AverageLatency(ms), 0.054989816700611

[UPDATE], MinLatency(ms), 0

[UPDATE], MaxLatency(ms), 1

[UPDATE], 95thPercentileLatency(ms), 1

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 491

[UPDATE], 0, 464

[UPDATE], 1, 27

[UPDATE], 2, 0

[UPDATE], 3, 0

[UPDATE], 4, 0

...

Αυτά τα αποτελέσματα μας δείχνουν[18]:

- Ο συνολικός χρόνος εκτέλεσης ήταν 10,11 δευτερόλεπτα
- Η μέση απόδοση ήταν 98,9 πράξεις / sec (σε όλα τα threads)
- Υπήρχαν 491 πράξεις ενημέρωσης (update) , με τα αντίστοιχες μέσες, ελάχιστες, μέγιστες, 95στη και 99στη ποσοστιαίες καθυστερήσεις .
- Όλες οι 491 πράξεις ενημέρωσης είχαν κωδικό επιστροφής μηδέν (επιτυχία σε αυτή την περίπτωση)
- 464 εργασίες ολοκληρώθηκαν σε λιγότερο από 1 ms , ενώ 27 ολοκληρώθηκαν μεταξύ 1 και 2 ms .

Παρόμοια στατιστικά στοιχεία είναι διαθέσιμα για τις λειτουργίες ανάγνωσης και εισαγωγής.

Οι Βάσεις Δεδομένων που επιλεχτήκαν προς αξιολόγηση είναι οι MySQL, PostgreSQL, Cassandra και MongoDb. Ο λόγος που επιλέχτηκαν αυτές οι βάσεις είναι επειδή όλες τους είναι ανοιχτού κώδικα ,εύκολα παραμετροποίησιμες ,υποστηρίζονται από

YCSB και είναι δημοφιλή στην αγορά για λύσεις σχεσιακών (MySQL και PostgreSQL)και μη-σχεσιακών (Cassandra και MongoDb)βάσεων δεδομένων .

4.2.1.2 Weighttp

Σε συγκριτική αξιολόγηση ενός web server , ο στόχος είναι συνήθως να προσδιοριστεί ο αριθμός των αιτήσεων ανά δευτερόλεπτο τις οποίες ο server είναι σε θέση να υποστηρίξει, ενώ ικανοποιεί ορισμένους περιορισμούς. Τυπικοί περιορισμοί περιλαμβάνουν[22] :

- Η απαίτηση ότι ένας Client δεν θα πρέπει να περιμένει περισσότερο από «n» δευτερόλεπτα για μια απάντηση . (Το «n» είναι συνήθως χαμηλό , περίπου 5-8 δευτερόλεπτα) .
- Μη έγκυρη αίτηση θα πρέπει να αποστέλλει ένα μήνυμα λάθους ως απάντηση .
- Οι αιτήσεις θα πρέπει να επιλέγονται τυχαία από μια μεγάλη δεξαμενή , για να διασφαλιστεί ότι ο μηχανισμός caching του web server εκτελείται πλήρως .

Για τη διεξαγωγή της συγκριτικής αξιολόγησης του web server, απαιτείται ένα σύστημα με τουλάχιστον τα ακόλουθα στοιχεία[22] :

- Ένας Server ο οποίος τρέχει το λογισμικό του web server υπό δοκιμή.
- Έναν ή περισσότερους Clients που εκτελούν το φορτίο παραγωγής λογισμικού .
- Ένα δίκτυο που συνδέει τους Clients με το Server και είναι απαλλαγμένος από άλλη κυκλοφορία , και δεν θα κορεστεί από τις προγραμματισμένες δοκιμές.

Εργαλεία συγκριτική αξιολόγηση ενός web server:

- Apache Bench
- http_load
- weighttp
- httperf
- Tsung

Για την παρούσα εργασία επιλέχτηκε το weighttp.

Το weighttp (προφέρεται ως weighty) είναι ένα ελαφρύ και μικρό εργαλείο συγκριτικής αξιολόγησης για τους webservers. Σχεδιάστηκε για να είναι πολύ γρήγορο και εύκολο στη χρήση και υποστηρίζει μόνο ένα μικρό τμήμα του πρωτοκόλλου HTTP , ώστε να

είναι απλό και γρήγορο. Το weighttp υποστηρίζει multithreading για καλή χρήση των σύγχρονων επεξεργαστών με πολλαπλούς πυρήνες , καθώς και ασύγχρονη είσοδο – έξοδο για ταυτόχρονες αιτήσεις σε ένα thread[19].

Για το χειρισμό συμβάντων, το weighttp βασίζεται στο στην βιβλιοθήκη libev που ταιριάζει απόλυτα στο σχεδιασμό του weighttp, όντας ελαφρύ και γρήγορο . Χάρη σε αυτό, το weighty υποστηρίζει όλες τα σύγχρονα event interface υψηλής απόδοσης , όπως το Epoll ή το kqueue.Επιπρόσθετα το εργαλείο αυτό υποστηρίζει όλους τους γνωστούς web server[19].

Το weighttp χρησιμοποίει πανόμοιες εντολές στην γραμμή εντολών (commandline arguments) με το apache bench (ab):

weighttp <options> <url>

- n num πλήθος requests (υποχρεωτικό)
- t num αριθμός thread (προεπιλογή: 1)
- c num ταυτοχρονες αιτήσεις(concurrent requests) (προεπιλογή: 1)
- k keep alive (προεπιλογή: όχι)

Μια τυπική εντολή weighttp θα μπορούσε να είναι:

weighttp -n 1000 -c 20 -t 1 http://127.0.0.1:80/index.html

Όταν εκτελεστεί η εντολή θα παραγάγει ένα αρχείο αποτελεσμάτων όπως το παρακάτω

starting benchmark...

spawning thread #1: 20 concurrent requests, 1000 total requests

progress: 10% done

progress: 20% done

progress: 30% done

progress: 40% done

progress: 50% done

progress: 60% done

progress: 70% done

progress: 80% done

progress: 90% done

progress: 100% done

finished in 0 sec, 106 millisec and 760 microsec, 9366 req/s, 2634 kbyte/s

requests: 1000 total, 1000 started, 1000 done, 1000 succeeded, 0 failed, 0 errored

status codes: 1000 2xx, 0 3xx, 0 4xx

traffic: 288000 bytes total, 243000 bytes http, 45000 bytes data

Αυτά τα αποτελέσματα μας δείχνουν[19]:

- Σε ποσό χρόνο ολοκληρώθηκε η δοκιμή(0 sec, 106 millisec and 760 microsec)
- Πόσες αιτήσεις εξυπηρετήθηκαν ανά δευτερόλεπτο (9366 req/s)
- Πόσα KB καταναλωθήκαν ανά δευτερόλεπτο (2634 kbyte/s)
- Πόσες αιτήσεις έγιναν συνολικά, πόσες ξεκίνησαν, έγιναν, επέτυχαν, απέτυχαν και πόσες περιείχαν λάθη (1000 total, 1000 started, 1000 done, 1000 succeeded, 0 failed, 0 errored)
- Ποια και πόσες ήταν οι έξοδοι της εκτέλεσης των αιτήσεων (1000 2xx, 0 3xx, 0 4xx). Η έξοδος με κωδικό 2xx αναφέρετε στις αιτήσεις που επέτυχαν, 3xx σε αυτές που απέτυχαν και η 4xx σε αυτές που περιείχαν λάθη.
- Πόσα bytes κυκλοφόρησαν μεταξύ του Client και Web Server και πόσα από τα συνολικά bytes αφορούν καθαρά δεδομένα και πόσα αφορούν δεδομένα του http πρωτοκόλλου

Οι Web Servers που επιλεχτήκαν προς αξιολόγηση στην παρούσα εργασία είναι οι Apache2.2, Apache2.4, Nginx και Lighttpd καθώς είναι οι πιο ευρέως χρησιμοποιημένοι σήμερα για δοκιμαστικό περιβάλλον αλλά και περιβάλλοντα παράγωγης.

4.2.1.3 Mapreduce Benchmarks

Το framework Hadoop περιλαμβάνει μια σειρά από συγκριτικές δοκιμές, τα οποία περιέχονται στις βιβλιοθήκες hadoop-*test*.jar, hadoop-mapreduce-client-jobclient-*-tests.jar και hadoop-*examples*.jar. Οι τέσσερις συγκριτικές δοκιμές που θα εξετάσουμε με περισσότερες λεπτομέρειες είναι τα TestDFSIO , NNBench , MRBenchκαι TeraGen - TeraSort [23]. Να σημειωθεί ότι οποιοδήποτε πρόγραμμα MapReduce θα μπορούσε θα μπορούσε να χρησιμοποιηθεί για να εξεταστεί η υποδομή του περιβάλλοντος υπολογιστικού νέφους πάνω στο οποίο είναι εγκατεστημένο το Hadoop. Αυτό θα μπορούσε να γίνει με άπλα μετρώντας τους χρόνους εκτελέσεις των εφαρμογών MapReduce. Παρόλα αυτά οι δοκιμές που επιλέχτηκαν είναι τα πιο γνωστά ανάμεσα στην κοινότητα του Hadoop και προσφέρουν πιο εξειδικευμένα αποτελέσματα που αφορούν σχεδόν όλα τα στοιχειά (components) του Hadoop και που πιθανόν να ενδιαφέρουν πολλούς χρήστες. Για το πάρων έργο χρησιμοποιήθηκε η έκδοση 2.2.0 του Hadoop σε Standalone λειτουργία με όλα τα components να τρέχουν σε έναν κόμβο.

4.2.1.3.1 TestDFSIO

Το TestDFSIO είναι μια συγκριτική δοκιμή για ανάγνωση και εγγραφή του αρχείων στο σύστημα αποθήκευσης δεδομένων του Hadoop (HDFS -Hadoop Distributed File System). Είναι χρήσιμο για εργασίες όπως, προσομοίωση ακραίων καταστάσεων (stress testing) στο HDFS, για να ανακαλύψουμε σημεία συμφόρησης στο δίκτυό, για να ελέγξει το υλικό , το λειτουργικό σύστημα και την εγκατάσταση Hadoop στο cluster (ιδιαίτερα τον διαχειριστικό κόμβο ή αλλιώς NameNode και κόμβους αποθήκευσης ή αλλιώς DataNodes) και να μας δώσει μια πρώτη εντύπωση για το πόσο γρήγορο είναι το σύστημα μας από την άποψη εισόδου εξόδου (I/O)[24].

Το TestDFSIO μπορεί να τρέξει με τις εξής παραμέτρους[25]:

- *read (random | -backward):* Για ανάγνωση αρχείων από το HDFS. Η ανάγνωση μπορεί να είναι τυχαία ή να αρχίσει από το τέλος.
- *write:* Για εισαγωγή αρχείων στο HDFS.
- *Append:* Για εισαγωγή αρχείων σε σημείο του HDFS ως συνέχεια άλλων αρχείων.
- *clean:* Για διαγραφή όλων των αρχείων που δημιουργήθηκαν μέσω του TestDFSIO.
- *nrFiles:* Αριθμός αρχείων
- *size (B|KB|MB|GB|TB):* Μέγεθος αρχείων. Η προεπιλογή είναι σε MB
- baseDir = /benchmarks/TestDFSIO
- *resFile resultFileNam:* Όνομα αρχείου αποθήκευσης των αποτελεσμάτων . Όχι απαραίτητο
- *bufferSize Bytes:* Μέγεθος του Buffer. Όχι απαραίτητο. Η προεπιλογή είναι 1000000
- *rootDir:* Όνομα φακέλου που θα αποθηκεύσουν τα αρχεία στο HDFS . Όχι απαραίτητο. Η προεπιλογή είναι baseDir = /benchmarks/TestDFSIO

Η δοκιμή TestDFSIO που αφορά την ανάγνωση δεν παράγει τα δικά του αρχεία εισόδου . Για το λόγο αυτό, πρώτα πρέπει να τρέξει η δοκιμή TestDFSIO που αφορά εγγραφή αρχείων μέσω του -write και στη συνέχεια μια δοκιμή ανάγνωσης μέσω του -read (με τις ίδιες παραμέτρους όπως και κατά την εκτέλεση της εγγραφής) . Και σαν τελικό βήμα αδειάζουμε το HDFS με δοκιμή TestDFSIO που αφορά την διαγραφή των αρχείων μέσω του –clean. Είναι σημαντικό κατά το –write να μην παραγάγουμε μέγεθος αρχείων μεγαλύτερο από το διαθέσιμο χώρο στο HDFS,αλλιώς η δοκιμή θα αποτύχει. Μια τυπική ακολουθία εντολών για να τρέξουμε το TestDFSIO είναι :

```
hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar TestDFSIO -write -nrFiles 10 -fileSize 100

hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar TestDFSIO -read -nrFiles 10 -fileSize 100

hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar TestDFSIO –clean
```

Η εντολές αυτές θα παραγάγουν δυο αρχεία αποτελεσμάτων για την φάση της εγράφης . Και τα δυο αρχεία έχουν την παρακάτω μορφή:

```
fs.TestDFSIO: ----- TestDFSIO ----- : write/read

fs.TestDFSIO:           Date & time: Wed Mar 05 22:12:01 UTC 2014

fs.TestDFSIO:       Number of files: 2
```

fs.TestDFSIO: Total MBytes processed: 2.0

fs.TestDFSIO: Throughput mb/sec: 2.6143790849673203

fs.TestDFSIO: Average IO rate mb/sec: 3.3959550857543945

fs.TestDFSIO: IO rate std deviation: 1.6291705104690328

fs.TestDFSIO: Test exec time sec: 28.226

Για την αξιολόγηση της απόδοσης του περιβάλλοντος υπολογιστικού νέφους τα πιο αξιοσημείωτα αποτελέσματα είναι ο χρόνος εκτέλεσης της δοκιμής (Test exec time sec: 28.226), η απόδοση (Throughput mb/sec: 2.6143790849673203) και η μέση τιμή εισόδου-εξόδου δεδομένων (Average IO rate mb/sec: 3.3959550857543945). Τόσο το Throughput όσο και Average IO rate βασίζονται στο μέγεθος του αρχείου που εγγράφετε ή διαβάζετε στο HDFS από τις διεργασίες Map και στο χρόνο που χρειάστηκε για να γίνει αυτό.

Ο τύπος για τον υπολογισμό του Throughput για μια εργασία TestDFSIO που χρησιμοποίει N διεργασίες Map είναι ο παρακάτω[25]:

$$Throughput(N) = \frac{\sum_{i=0}^{N} filesize_i}{\sum_{i=0}^{N} time_i}$$

Εικόνα 8:Συνάρτηση Throughput[25]

Παρομοίως ο τύπος για το Average IO rate ορίζεται όπως παρακάτω:

$$Average\ IO\ rate(N) = \frac{\sum_{i=0}^{N} rate_i}{N} = \frac{\sum_{i=0}^{N} \frac{filesize_i}{time_i}}{N}$$

Εικόνα 9: Συνάρτηση Avreage IO rate[25]

Άλλη μια σημαντική παράμετρο που πρέπει να λάβουμε υπόψη μας όταν αναλύουμε τα αποτελέσματα μιας TestDFSIO δοκιμής είναι η τιμή του παραμέτρου αντιγραφής των δεδομένων στο HDFS (replication factor). Εάν για παράδειγμα συγκρίνουμε δυο TestDFSIO δοκιμές με replication factor 1 και 2 αντίστοιχα θα δούμε ότι η δοκιμή με το μικρότερο replication factor θα έχει καλύτερες τιμές για το Throughput και Average IO rate.Οι κλάσεις της TestDFSIO δοκιμής ωστόσο μπορούν να παραμετροποιημένου και ο χρήστης μπορεί να ορίσει μέσα από την κλάση παραμέτρους όπως replication factor,διεργασίες Map και Reduce που από προεπιλογή έχουν τιμή 1.

4.2.1.3.2 TeraSort-Teragen

Η δοκιμή TeraSort είναι ίσως η πιο γνωστή δοκιμή αξιολόγησης του Hadoop. Στόχος του TeraSort είναι να ταξινομήσει (sort) δεδομένα μεγάλου μεγέθους όσο το δυνατόν γρηγορότερα . Η δοκιμή αυτή συνδυάζει αξιολόγηση του HDFS και των στρωμάτων του MapReduce σε ένα Hadoop cluster[25].

Τυπικά πεδίο εφαρμογής του TeraSort είναι η εξέταση εάν η διεργασίες Map και Reduce είναι υγιή (καθώς εξαρτώνται από τις μεταβλητές όπως ο αριθμός των πυρήνων ανά TaskTracker και τη διαθέσιμη μνήμη RAM)[25][26] .

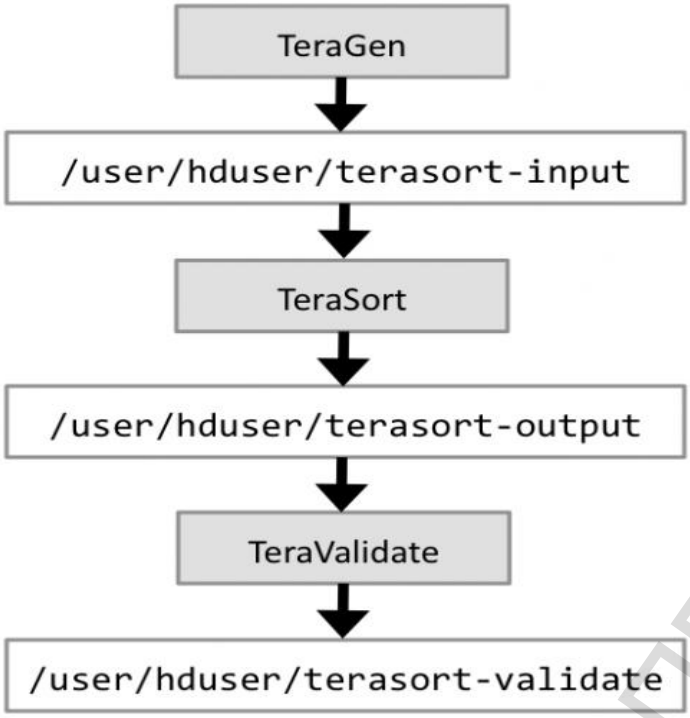
Μια τυπική δοκιμή TeraSort περιλαμβάνει αυτά τα 3 βήματα:

1. Παραγωγή των δεδομένων εισόδου μέσω TeraGen .

2. Ταξινόμηση δεδομένων εισόδου μέσω του TeraSort.

3. Επικύρωση των ταξινόμηση δεδομένων εξόδου μέσω TeraValidate .

Η παρακάτω εικόνα περιγράφει την όλη διαδικασία



Εικόνα 10: Λειτουργία Terasort[25]

TeraGen παράγει τυχαία δεδομένα, που το Terasort θα χρησιμοποίηση ύστερα, μέσω της παρακάτω εντολής:

hadoop jar hadoop-mapreduce-examples-2.2.0.jar teragen 1000000 /terasort-in

Έτσι έχουμε παραγάγει αρχείο μεγέθους 1000000 bytes που αποθηκεύονται στο φάκελο terasort-in στο HDFS.Επειδή κατά την εκτέλεση τις εντολής αυτή δημιουργούμε ουσιαστικά τον φάκελο terasort-in ,εάν θέλουμε να χρησιμοποιήσουμε το ίδιο φάκελο σε επομένη εντολή TeraGen πρώτα πρέπει να τον διαγράψουμε από το HDFS[25][26].

Το TeraSort ,που έχει υλοποιηθεί σαν μια προσαρμοσμένη εργασία MapReduce, ταξινομεί τα αρχεία που βρίσκονται φάκελο terasort-in και τα αποθηκεύει στον φάκελο terasort-out μέσω της παρακάτω εντολής:

hadoop hadoop-mapreduce-examples-2.2.0.jar terasort /terasort-in /terasort-out

Το Teravalidate , που επίσης έχει υλοποιηθεί σαν μια προσαρμοσμένη εργασία MapReduce ,με τη σειρά τους επικυρώνει τα αποτελέσματα από τον φάκελο /terasort-out και αποθηκεύει τα δικά του αποτελέσματα στον φάκελο /terasort-validate. Η εντολή για να εκτελεστεί το Teravalidate αυτό είναι η παρακάτω:

hadoop jar hadoop-mapreduce-examples-2.2.0.jar teravalidate /terasort-out /terasort-validate

Επειδή κατά την εκτέλεση τις εντολής αυτή δημιουργούμε ουσιαστικά τον φάκελο terasort- validate,εάν θέλουμε να χρησιμοποιήσουμε το ίδιο φάκελο σε επομένη εντολή Teravalidate πρώτα πρέπει να τον διαγράψουμε από το HDFS.

Οι εφαρμογές TeraGen , Terasort καιTeravalidate παράγουν αρκετά χρήσιμα αποτελέσματα για τον χρήστη όπως ο χρόνος που χρειάστηκε για να γίνει η βασική διαίρεση του αρχείου εισόδου, ο χρόνος που χρειάστηκε για τις διαιρέσεις τον αρχείων εισόδου στις διεργασίες Map και Reduce και συνολικός χρόνος για να υπολογιστούν οι διαιρέσεις. Ωστόσο, στα πλαίσια αυτής τις εργασίας τα αποτελέσματα που μας ενδιαφέρουν είναι οι συνολικοί χρόνοι που χρειαστήκαν οι διεργασίες Map και Reduce και στις 3 εργασίας TeraGen , Terasort καιTeravalidate

3.2.1.3.3 NNbench

Η δοκιμή NNBench είναι χρήσιμο για τον αξιολόγηση του υλικού (hardware)και τη παραμετροποίηση του κόμβου που είναι εγκατεστημένο το διαχειριστής (NameNode) ενός Hadoop cluster. Η δοκιμή αυτό παράγει πολλά αιτήματα σε αρχεία που είναι αποθηκευμένα στο HDFS και συνήθως έχουν μικρό φορτίο με στόχο αξιολογήσουν την διαχειριστική ικανότητα του NameNode σε μεγάλο φόρτο εργασίας. Το NNBench

προσομοιωνει αιτιματα που αφορούν την δημιουργία , την ανάγνωση, τη μετονομασία και διαγραφή αρχείων στο HDFS[27].

Η παράμετροι που το NNBench μπορεί να λάβει είναι οι παρακάτω[27]:

- -operation: Η διαθέσιμες λειτουργίες είναι δημιουργία (create_write) ,ανάγνωση (open_read) , μετονομασία (rename) και διαγράφη (delete) αρχείων. Η επιλογή αυτή είναι υποχρεωτική Η open_read , rename και delete εργασίες προϋποθέτουν ότι τα αρχεία πάνω στα οποία θα εκτελέσουν τις διεργασίες τους είναι ήδη διαθέσιμα . Η εργασία create_write πρέπει να εκτελεστεί πριν από την εκτέλεση των άλλων λειτουργιών .
- Maps: Ο αριθμός των maps διεργασιών . Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- reduces: Ο αριθμός των reduces διεργασιών . Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- StartTime Ο χρόνο για να ξεκινήσει η δοκιμή. 2 λεπτά . Όχι υποχρεωτικό.
- Blocksize: Μέγεθος των μπλοκ σε bytes. Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- bytesToWrite: Αριθμός Bytes προς εγράφη . Προεπιλεγμένη τιμή είναι 0 . Όχι υποχρεωτικό.
- bytesPerChecksum: Bytes ανά έλεγχο (checksum) για τα αρχεία. Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- numberOfFiles: Ο αριθμός των αρχείων που θα δημιουργηθούν . Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- replicationFactorPerFile: Ο παράγοντα αναπαραγωγής των αρχείων στο HDFS . Προεπιλεγμένη τιμή είναι 1 . Όχι υποχρεωτικό.
- baseDir: Ο φάκελος στο HDFS στον οποίο θα αποθηκευτούν τα αρχεία . Προεπιλογή είναι ο φάκελος / becnhmarks / NNBench . Όχι υποχρεωτικό.
- readFileAfterOpen: Παράμετρο που ορίζει αν το αρχείο που θα ανοιχτεί θα πρέπει ταυτόχρονα να αναγνωστεί και παίρνει τιμές αληθής ή ψευδής . Αυτό ισχύει με τη λειτουργία open_read . Προεπιλογή είναι ψευδής . Όχι υποχρεωτικό.

Οι τυπικές εντολές και η σειρά για να τρέξουμε μια NNBench δοκιμή είναι οι παρακάτω:

hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar nnbench -operation create_write -maps 1 -reduces 1 -blockSize 1 -bytesToWrite 2000 -bytesPerChecksum 1 -numberOfFiles 20 -replicationFactorPerFile 1

hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar nnbench -operation open_read -maps 1 -reduces 1 -blockSize 1 -bytesToWrite 2000 -bytesPerChecksum 1 -numberOfFiles 20 -replicationFactorPerFile 1

hadoop jar hadoop-mapreduce-client-jobclient-2.2.0-tests.jar nnbench -operation delete -maps 1 -reduces 1 -blockSize 1 -bytesToWrite 2000 -bytesPerChecksum 1 -numberOfFiles 20 -replicationFactorPerFile 1

Τα αποτελέσματα που παράγει το NNBench κατά την φάση write , read και delete είναι στην παρακάτω μορφή :

INFO hdfs.NNBench: ---------- NNBench ---------- :
INFO hdfs.NNBench: Test Operation: write/read/delete
INFO hdfs.NNBench: Successful file operations: 20
INFO hdfs.NNBench: # maps that missed the barrier: 0
INFO hdfs.NNBench: # exceptions: 0
INFO hdfs.NNBench: TPS:69
INFO hdfs.NNBench: Avg Exec time (ms):14.4
INFO hdfs.NNBench: Avg Lat (ms): Delete: 14.0
INFO hdfs.NNBench: RAW DATA: AL Total #1: 280
INFO hdfs.NNBench: RAW DATA: AL Total #2: 0
INFO hdfs.NNBench: RAW DATA: TPS Total (ms): 288
INFO hdfs.NNBench: RAW DATA: Longest Map Time (ms): 288.0
INFO hdfs.NNBench: RAW DATA: Late maps: 0
INFO hdfs.NNBench: RAW DATA: # of exceptions: 0

Τα αποτελέσματα που μας είναι χρήσιμα για την αξιολόγηση της υποδομής του υπολογιστικού νέφους είναι ο ρυθμός των συναλλαγών (TPS) καθώς και μέσος χρόνος των εκτέλεσης αυτών .

3.2.1.3.4 MRBench

Το MRBench εκτελεί πολλές φόρες μια μικρή εργασία MapReduce. Η δοκιμή αυτή ελέγχει κατά πόσο οι μικρές εργασίας ανταποκρίνονται σωστά και εκτελούνται αποτελεσματικά στο Hadoop cluster . Επίσης ,δίνει έμφαση στο στρώμα του MapReduce και όχι τόσο στο HDFS[25].

Οι παράμετροι της δοκιμής αυτής είναι οι παρακάτω

- maps: Ο αριθμός των Map διεργασιών . Η προεπιλογή είναι 2.
- Reduces: Ο αριθμός των Reduce διεργασιών . Η προεπιλογή είναι 1.
- inputLines: Ο αριθμός των γραμμών εισόδου που θα παραχθούν. Η προεπιλογή είναι 1.

- inputType: Ο τύπος της γεννήτριας των δεδομένων εισόδου. Τιμές που μπορεί να λάβει είναι αύξουσα (default), φθίνουσα και τυχαία.

Τα αποτελέσματα όταν εκτελούμε το MRBench είναι τα έξης :

DataLines	Maps	Reduces	AvgTime (milliseconds)
1	2	1	31414

Ο μέσος χρόνος εκτέλεσης των MapReduce διεργασιών είναι ένα χρήσιμο στοιχείο για την αξιολόγηση της υποδομής του υπολογιστικού νέφους εάν εκτελέσουμε την δοκιμή αυτή αρκετές φόρες.

4.2.2 Jclouds

Για να είναι δυναμική μια εφαρμογή δεν πρέπει να περιορίζετε μονό σε στατικές μεταβλητές. Στην περίπτωση της εφαρμογής αυτής πρέπει να δίνετε η δυνατότητα στον χριστή να επιλεγεί μεταξύ παρόχων δημόσιου ή ιδιωτικού περιβάλλοντος υπολογιστικού νέφους Υποδομής σαν Υπηρεσία . Επειδή δεν είναι εύκολο και αποδοτικό να γραφτεί κώδικας από την αρχή για την χρησιμοποίηση των πόρων όλων των παρόχων έπρεπε να χρησιμοποιήσουμε εργαλεία ανοιχτού κώδικα όπως Jclouds kai Libcloud . Τα εργαλεία αυτά χρησιμοποιώντας το REST API των παρόχων δίνουν την δυνατότητα να δημιουργήσεις και χρησιμοποιήσεις προγραμματιστικά πόρους του παρόχου. Επίσης παρέχουν δικό τους απλό API για το πώς να δημιουργείς και χρησιμοποιείς πόρους του παρόχου χωρίς να χρειάζεται να γνωρίζεις την πολύπλοκη και συνήθως μεγάλη αρχιτεκτονική του κώδικα του παρόχου[28]. Για την παρόν έργο χρησιμοποιήθηκε το Jclouds. Παρόλο που το Jcloud είναι λίγο πιο φτωχό στην λειτουργικότητα σε σχέση με το Libcloud , προτιμήθηκε γιατί είναι γραμμένο μονό σε JAVA και αυτό βοηθά στην ενσωμάτωση του στον δικό μας κώδικα που είναι επίσης γραμμένο σε JAVA, ενώ το Libcloud είναι γραμμένο σε Python.

Το Jclouds επιτρέπει τον έλεγχο και δημιουργία πόρων-υπηρεσίες σε παρόχους υπολογιστικού νέφους χρησιμοποιώντας τις γλώσσες προγραμματισμού Java and Clojure. Οι ποιο ώριμες υπηρεσίες του παρόχου στις οποίες το Jclouds δίνει πρόσβαση είναι η υπολογιστική υπηρεσία (ComputeService) και υπηρεσία αποθήκευσης δεδομένων (BlobStore). Το Jclouds δίνει πρόσβαση σε παρόχους όπως Amazon, Azure, GoGrid, Ninefold, OpenStack, Rackspace, vCloud και σε όλους τους παρόχους που βασίζονται στο Openstack. Το μεγαλύτερο πλεονέκτημα του Jclouds είναι ότι μέσω του απλού API που διαθέτει δίνει την δυνατότητα στους προγραμματιστές να κώδικα που με μικρές αλλαγές μπορεί να χρησιμοποιηθεί σε όλους τους παρόχους που υποστηρίζει[28]. Το εργαλείο αυτό με αλλά λόγια μετατρέπει σε πραγματικότητα το multi-cloud, δηλαδή μια πλατφόρμα την δυνατότητα έλεγχου πολλών παρόχων υπολογιστικού νέφους μόνο από ένα σημείο.

4.2.3 Openstack

Για την υλοποίηση του παρόντος έργου είναι αναγκαίο η ύπαρξη ενός περιβάλλοντος υπολογιστικού νέφους που θα προσφέρει Υποδομή σαν Υπηρεσία. Το περιβάλλον υπολογιστικού νέφους που χρησιμοποιήθηκε είναι ιδιωτικό υπολογιστικό νέφος βασισμένο στο ανοιχτό λογισμικό υπολογιστικού νέφους Openstack (έκδοση Grizzly).

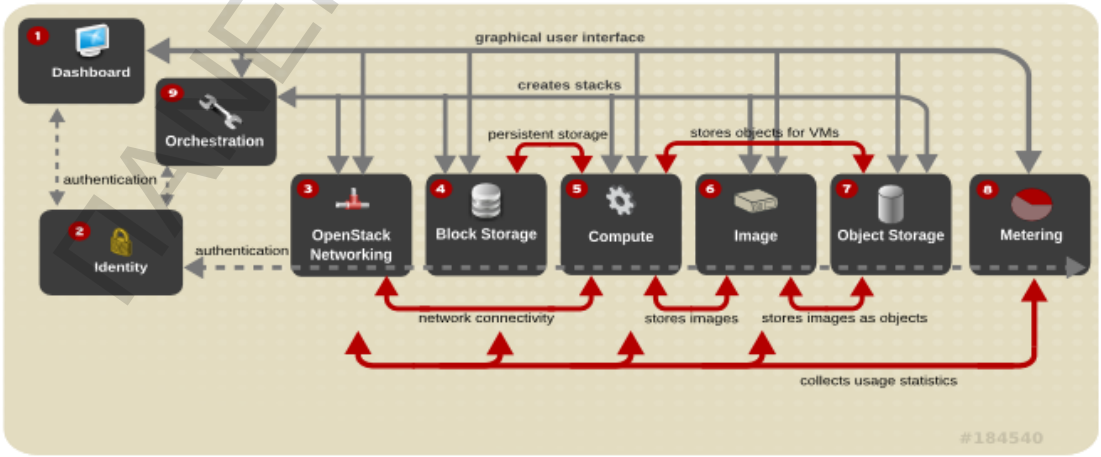
Το Openstack είναι μία δωρεάν και open-source πλατφόρμα λογισμικού, πάνω στην οποία ο καθένας μπορεί να αναπτύξει, να δοκιμάσει και να τρέξει εφαρμογές και υπηρεσίες cloud. Το Openstack έγινε πολύ γρήγορα το πρότυπο για open cloud infrastructure. Ο κώδικας του Openstack είναι ελεύθερα διαθέσιμος υπό την άδεια Apache 2.0, ώστε ο καθένας να μπορεί να το εγκαταστήσει , να το τρέξει και να συμβάλει στο έργο. Είναι ένα μοντέλο ανάπτυξης που έχει καλλιεργήσει μια τεράστια κοινότητα, που υποστηρίζεται από ένα μεγάλο και συνεχώς αυξανόμενο οικοσύστημα εργαλείων, λύσεων, και παρόχων υπηρεσιών [29].

Το Openstack είναι ένα ανοιχτό και επεκτάσιμο λειτουργικό σύστημα για την κατασκευή δημόσιων και ιδιωτικών clouds. Παρέχει τόσο σε μεγάλους, όσο και σε μικρούς οργανισμούς μια εναλλακτική λύση για τα κλειστά περιβάλλοντα cloud, μειώνοντας τους κινδύνους της δέσμευσης με ιδιόκτητες πλατφόρμες [30].

Το Openstack είναι μια παγκόσμια συνεργασία από developers και cloud computing technologists που παράγουν μια ανοιχτού προτύπου cloud computing πλατφόρμα για δημόσια και ιδιωτικά clouds. Το έργο αυτό έχει σαν στόχο να προσφέρει λύσεις για όλους τους τύπους cloud, όντας πολύ απλό στην εφαρμογή του, μαζικά επεκτάσιμο και με πλούσια χαρακτηριστικά. Η τεχνολογία αποτελείται από μια σειρά αλληλοσυνδεόμενων projects που προσφέρουν διάφορα συστατικά για μια ολοκληρωμένη cloud infrastructure λύση [31].

Το Openstack αποτελείται από 6 κύρια projects (service families) :

- Nova – Compute Service
- Swift + Cinder – Storage Service
- Glance – Imaging Service
- Keystone – Identity Service
- Neutron – Network Service
- Ceilometer – Monitoring Service
- Horizon – User Interface Service



Εικόνα 11: *Αρχιτεκτονική OpenStack [32]*

Nova – Compute Service

Το Openstack Compute έχει σχεδιαστεί για να παρέχει και να διαχειρίζεται μεγάλα δίκτυα εικονικών μηχανών (virtual machines), δημιουργώντας μια επεκτάσιμη πλατφόρμα cloud computing. Παρέχει το λογισμικό, τα APIs, και τα control panels που απαιτούνται για την ενορχήστρωση ενός περιβάλλοντος cloud, το “τρέξιμο” των instances, την διαχείριση του δικτύου και του ελέγχου πρόσβασης των χρηστών και των έργων. Το Openstack Compute μπορεί να θεωρηθεί παράλληλα hardware και hypervisor-agnostic, και υποστηρίζει ποικιλία από hardware configurations και hypervisors [30].

Swift + Cinder – Storage Service

Το Openstack Object Storage ή αλλιώς Swift παρέχει επεκτάσιμη αποθήκευση αντικειμένων χρησιμοποιώντας τυποποιημένα συμπλέγματα διακομιστών (standardized servers). Το Openstack Storage επεκτείνεται ώστε να αποθηκεύει petabytes δεδομένων, κάτι που το καθιστά κατάλληλο να χειρίζεται virtual machine images, αποθήκευση φωτογραφιών, αποθήκευση email καθώς και αποθήκευση αντιγράφων ασφαλείας, ανέξοδα μέσω της αντιγραφής των δεδομένων και της διανομής τους σε commodity hard drives[33].

Το Openstack Block Storage ή αλλιώς Cinder διαχειρίζεται τη δημιουργία, σύνδεση και αποσύνδεσης των block devices σε servers. Τα block storage volumes είναι πλήρης ενσωματωμένα στο OpenStack Compute επιτρέποντας στους χρηστές να διαχειριστούν οι ίδιοι τις ανάγκες αποθήκευσης δεδομένων.Η block αποθήκευση είναι κατάλληλη για καλή απόδοση σε αποθήκευση βάσεων δεδομένων, επεκτάσιμα file systems ή σε περιπτώσεις που παρέχεις έναν server με πρόσβαση σε raw block level αποθήκευση[34].

Glance – Imaging Service

Το Glance παρέχει υπηρεσίες για την ανακάλυψη, την εγγραφή, την ανάκτηση VM. Το Glance έχει RESTful API που επιτρέπει την επερώτηση μεταδεδομένων των VM εικόνας καθώς και την ανάκτηση VM. VM images που διατίθενται μέσω του Glance μπορούν να αποθηκευτούν είτε σαν απλά filesystems είτε σαν object-storage systems (OpenStack Swift)[35].

Keystone – Identity Service

Το Keystone είναι ένα έργο που παρέχει Ταυτοποίηση, Token, Κατάλογο και Πολιτικές για να χρησιμοποιηθούν από τα έργα στην οικογένεια OpenStack. Η γενική ιδέα βασίζεται σε έναν κεντρικό κατάλογο χρηστών που αντιστοιχίζονται με τις υπηρεσίες OpenStack στις οποίες μπορούν να έχουν πρόσβαση. Επιπλέον, ο κατάλογος παρέχει μια λίστα όλων των υπηρεσιών που αναπτύσσονται σε ένα περιβάλλον OpenStack σε ένα ενιαίο μητρώο [36].

Neutron – Network Service

Το Neutron παρέχει μοντέλα δικτύωσης για διαφορετικές εφαρμογές ή ομάδες χρηστών. Είναι ένα επεκτάσιμο σύστημα για τη διαχείριση των δικτύων και διευθύνσεων IP. Το Neutron (πρώην Quantum) εξασφαλίζει ότι το δίκτυο δεν θα είναι το σημείο συμφόρησης ή περιοριστικός παράγοντας σε περιβάλλον cloud και δίνει στους χρήστες μια πραγματικη αυτό-διαχειριζόμενη υπηρεσία , ακόμα και στις ρυθμίσεις του δικτύου τους [37].

Ceilometer – Monitoring Service

Το Ceilometer είναι μια υπηρεσία παρακολούθησης και μέτρησης. Πρόκειται ουσιαστικά για την υποδομή στη συλλογή μετρήσεων στο OpenStack . Επίσης, προσφέρει ένα μοναδικό σημείο επαφής για τα συστήματα τιμολόγησης για την απόκτηση του συνόλου των μετρήσεων που χρειάζονται για τη δημιουργία τιμολογίων [38].

Horizon – User Interface Service

Το Horizon είναι το γραφικό περιβάλλον του Openstack μέσω του οποίου ο χρήστης έχει πρόσβαση σε όλες την υπηρεσίες[39].

Για την εγκατάσταση του Openstack χρησιμοποιήθηκαν δυο υπολογιστές με τα παρακάτω χαρακτηριστικά:

Λειτουργικό Σύστημα	Επεξεργαστής	Μνήμη	Σκληρός Δίσκος
Ubuntu Server v12.04 LTS	AMD FX 6300(AM3+,3,5 GHz,14 Mb)	8GB 1600 MHz	750 GB

Στον ένα υπολογιστή εγκατασταθήκαν οι υπηρεσίες Cinder , Glance, Keystone, Horizon και Ceilometer και στον δεύτερο Nova, Neutron και Swift

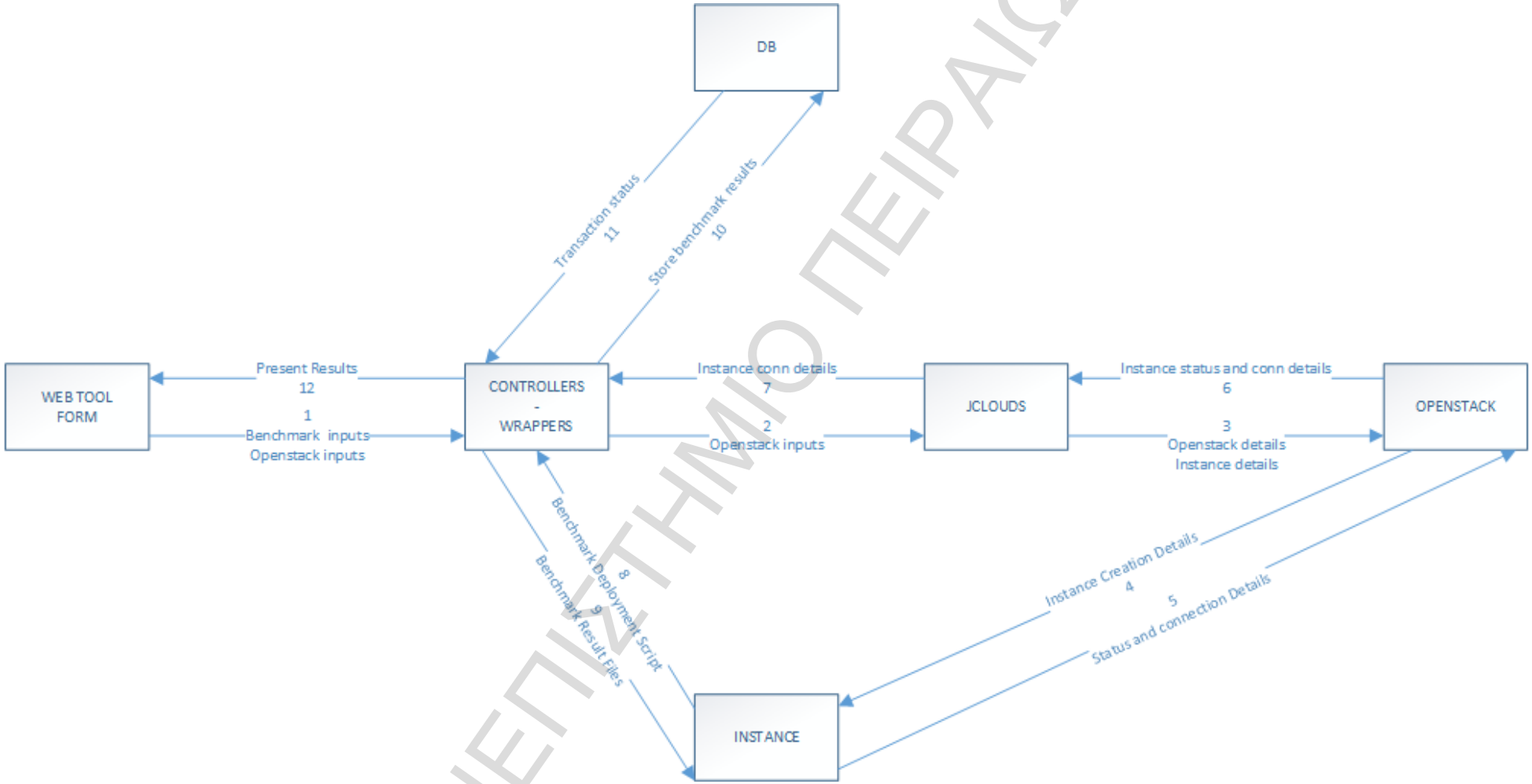
4.2.4 Αρχιτεκτονική και Υλοποίηση

4.2.4.1.1 Αρχιτεκτονική

Στα πλαίσια αυτής της εργασίας δεν ήταν δυνατή η πρόσβαση σε δημοσίους παρόχους Υποδομής σαν Υπηρεσία (IaaS public cloud providers), όπως Amazon, Rackspace ή Azure. Το περιβάλλον υπολογιστικού νέφους που χρησιμοποιήθηκε είναι ιδιωτικό υπολογιστικό νέφος βασισμένο στο ανοιχτό λογισμικό υπολογιστικού νέφους Openstack (έκδοση Grizzly) . Ωστόσο, επειδή το Openstack αποτελεί την βάση πολλών δημοσίων παρόχων υπολογιστικού νέφους, ο κώδικας του παρόντος έργου μπορεί να χρησιμοποιηθεί, με ελάχιστες μετατροπές , σε παρόχους όπως π.χ. η Rackspace.

Η επικοινωνία μεταξύ εφαρμογής και του περιβάλλοντος υπολογιστικού νέφους θα γίνετε μέσω του εργαλείου Jclouds, το οποίο έχει την δυνατότητα να επικοινωνεί με το Openstack και να διαχειρίζεται πόρους του.

Στην παρακάτω εικόνα παρουσιάζετε η αρχιτεκτονική και ο τρόπος που τα δεδομένα ρέουν στο κομμάτι του έργου που αφορά την ανάπτυξη των εικονικών μηχανών ,την εγκατάσταση σε αυτά των απαραίτητων λογισμικών και πακέτων ,την εκτέλεση των δοκιμών και την επιστροφή των αποτελεσμάτων για παρουσίαση στο γραφικό περιβάλλον και αποθήκευση στην βάση δεδομένων.



Εικόνα 12: Αρχιτεκτονική του Tool

Πιο αναλυτικά η διαδικασία που ακολουθείτε είναι η έξης :

1) Ο Controller δέχεται από το γραφικό περιβάλλον δεδομένα που έχει εισάγει ο χρήστης και αυτά χωρίζονται σε δύο κατηγορίες. Δεδομένα που χρειάζονται για την επικοινωνία με τον περιβάλλον υπολογιστικού νέφους για να δημιουργηθούνε εικονικές μηχανές και δεδομένα που αφορούν την δοκιμή.

2) Ο Controller μεταβιβάζει τα πληροφορίες που αφορούν το Openstack και την δημιουργία εικονικών μηχανών στον ενδιάμεσο εργαλείο Jclouds

3) Το Jclouds με την σειρά του μεταφέρει αυτά τα δεδομένα στο Openstack.

4) Εφόσον το Jclouds ταυτοποίηση τον εαυτό του στο Openstack ,αυτό αναλαμβάνει σύμφωνα με τα δεδομένα που έχει λάβει να δημιουργήσει την εικονική μηχανή.

5) Εφόσον η εικονική μηχανή δημιουργηθεί ενημερώνει τις εσωτερικές οντότητες του Openstack για την κατάσταση του

6) Με την σειρά του το Openstack ενημερώνει το Jclouds για την κατάσταση της εικονικής μηχανής. Η ενημέρωση αυτή περιλαμβάνει και τα στοιχειά της εικονικής μηχανής.

7) Το Jclouds με την σειρά του μεταφέρει αυτές της πληροφορίες στον Controller .

8) Ο Controller ελέγχει τις πληροφορίες που έλαβε. Εάν η εικονική μηχανή δημιουργήθηκε επιτυχώς επικοινωνεί απευθείας με αυτή και βάσει των δεδομένων που έλαβε στο πρώτο βήμα και αφορούσαν την δοκιμή, αποφασίζει ποια δοκιμή θα εγκατασταθεί, ποια επιπλέον λογισμικά θα εγκατασταθούν και με ποιες παραμέτρους θα τρέξει η δοκιμή. Εάν εικονική μηχανή δεν δημιουργήθηκε επιτυχώς επιστρέφεται στο γραφικό περιβάλλον ανάλογο μήνυμα.

9) Η εικονική μηχανή ελέγχει όλα τα δεδομένα που έλαβε και τα εκτελεί. Εάν η εκτέλεση είναι επιτυχής επιστέφει στον Controller τα αρχεία με τα αποτελέσματα της δοκιμής,εάν όχι επιστρέφει αντίστοιχο μήνυμα.

10) Μόλις λάβει τα αρχεία με τα αποτελέσματα ο Controller τα μεταβιβάζει σε άλλες κλάσεις τις εφαρμογής για να σαρωθούν τα αποτελέσματα και να αποθηκευτούν στην βάση δεδομένων.

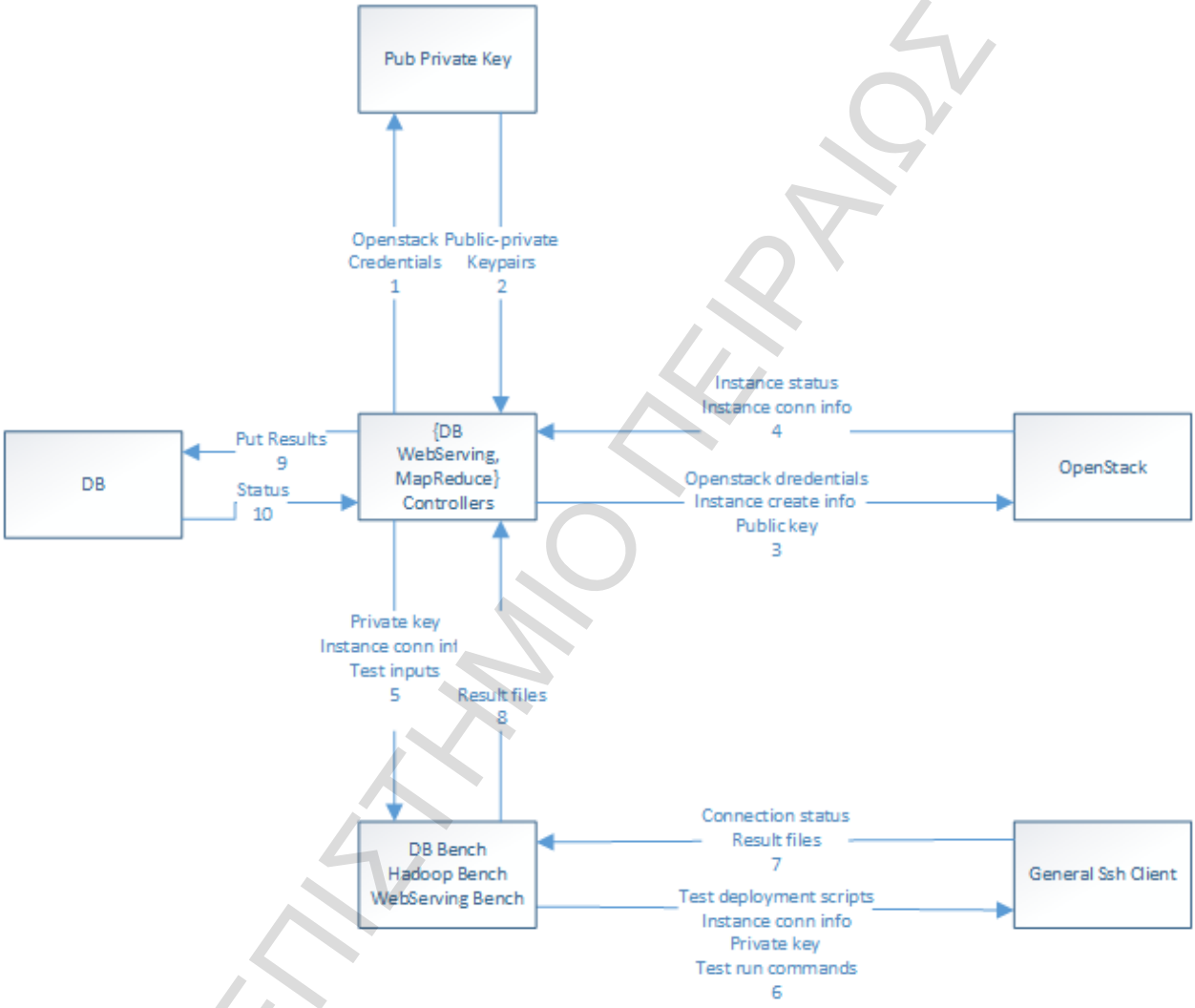
11) Η βάση δεδομένων ενημερώνει τον Controller για την κατάσταση της επερωτήσεις.

12) Ο Controller προωθεί τα αποτελέσματα στο γραφικό περιβάλλον για παρουσίαση στον χρηστή

Στην συγκεκριμένη αρχιτεκτονική η επικοινωνία με την εικονική μηχανή για την εγκατάσταση και εκτέλεση των δοκιμών θα μπορούσε να γίνει απευθείας από το Jclouds. Ωστόσο, το Jclouds δεν είναι πολύ ευέλικτο στην μεταφορά πολύ μεγάλου όγκου δεδομένα και αρχείων στο Openstack ,όπως πολύπλοκων εντολών εκτέλεσης και μεγάλα αρχεία ρυθμίσεων . Αυτός είναι και ο λόγος που προτιμήθηκε απευθείας επικοινωνία με την εικονική μηχανή μέσω του πρωτοκόλλου SSH που προσφέρει αρκετά μεγάλη ευελιξία στο συγκεκριμένο θέμα. Επιπλέον, επιλέξαμε οι δοκιμές να εγκαθιστούν και να εκτελεστούν στην εικονική μηχανή και όχι στον σερβερ που τρέχει η εφαρμογή. Αυτό όμως επηρεάζει σε ένα σημείο την επίδοση την εικονικής μηχανής και ακολούθως την απόδοση των δοκιμών που τρέχουν σε αυτήν αλλά από την στιγμή που όλα οι δοκιμές για άλλους του παρόχους εκτελούνται με τις ίδιες ακριβώς προδιαγραφές αυτό δεν επηρεάζει την σύγκριση μεταξύ τους. Επίσης, έτσι καταφέρνουμε να σχεδόν εξαλείψουμε τον επηρεασμό τον συγκρίσεων από την καθυστέρηση του δικτύου.

4.2.4.1.2 Υλοποίηση

Στην παρακάτω εικόνα παρουσίαζε η υλοποίηση των δοκιμών



Εικόνα 13: : Υλοποίηση των δοκιμών

Στο παραπάνω σχήμα η πληροφορία ακολουθεί αυτή την ροή:

1) Ο Controller (DB, WebServing, MapRedude) εφόσον έχει λάβει τα απαραίτητα δεδομένα από το γραφικό περιβάλλον επικοινωνεί με την κλάση PubPrivateKeyUtils και του

προωθεί τα στοιχεία ταυτοποίησης με το περιβάλλον υπολογιστικού νέφους για να ζητήσει ένα ζεύγος ιδιωτικών-δημοσιών κλειδιών

2) Η κλάση PubPrivateKeyUtils με την σειρά της δημιουργεί το ζεύγος κλειδιών και επιστρέφει στον Controller το ιδιωτικό κλειδί και το όνομα με το οποίο έχει αποθηκευτεί στο περιβάλλον υπολογιστικού νέφους

3) Ο Controller (DB, WebServing, MapRedude) επικοινωνεί με την κλάση Openstack και του προωθεί τα στοιχεία ταυτοποίησης με το περιβάλλον υπολογιστικού νέφους, τα στοιχεία που πρέπει να έχει η εικονική μηχανή καθώς και το δημόσιο κλειδί που θα πρέπει να έχει η εικονική μηχανή από την δημιουργία της .

4) Η κλάση Openstack επικοινωνεί με το περιβάλλον υπολογιστικού νέφους για να δημιουργήσει την εικονική μηχανή και εφόσον αυτή δημιουργηθεί επιστρέφει στο Controller τα στοιχεία επικοινωνίας της εικονικής μηχανής καθώς και την κατάσταση της .

5) Ο Controller ξέροντας στοιχεία επικοινωνίας της εικονικής μηχανής τα προωθεί αυτά στην αντίστοιχη Bench κλάση (DbBench, WebServingBench, HadoopBench) , μαζί με το ιδιωτικό κλειδί και τις παραμέτρους της αντιστοίχης δοκιμής.

6) Η Bench κλάση αφού ελέγξει τους παραμέτρους μεταφέρει σ την κλάση GeneralSshClient τα στοιχεία επικοινωνίας της εικονικής , το ιδιωτικό κλειδί, τα αρχεία παραμετροποίησης και τις εντολές εγκατάστασης πακέτων και εκτέλεσης της δοκιμής.

7) Η κλάση GeneralSshClient μεταφέρει και εκτελεί ότι έλαβε από την Bench κλάση και περιμένει από την εικονική μηχανή τα αρχεία με τα αποτελέσματα. Όταν τα λάβει τα επιστρέφει στην κλάση Bench μαζί με την κατάσταση της σύνδεσης τις εικονικής μηχανής.

8) Με την σειρά της η κλάση Bench επιστρέφει τα αρχεία αποτελεσμάτων στην αντίστοιχη κλάση Controller.

9) Η κλάση Controller με την βοήθεια άλλων κλάσεων της εφαρμογής μεταφέρει τα αποτελέσματα στην βάση δεδομένων

10) Και τέλος η κλάσεις αρμόδιες για σάρωση και επικοινωνία με την βάση δεδομένων, επιστρέφουν στον Controller την κατάσταση του αιτήματος του αποθήκευση τον αποτελεσμάτων.

Ο χρήστης μπορεί, εφόσον το επιλέξει από το γραφικό περιβάλλον, να χρησιμοποιήσει τα δικά του ζεύγη ιδιωτικών-ξένων κλειδιών που έχει ήδη αποθηκευμένα στο Openstack.

Η λογική εκτέλεσης κάθε δοκιμής είναι ενσωματωμένη στον αντίστοιχο Controller και Bench κλάση.

Βάσεις Δεδομένων -> DbController, DbBench

Web Serving -> WebServingController, WebServingBench

MapReduce-> MapReduceController, HadoopBench

Όλες οι υλοποιήσεις των δοκιμών χρησιμοποιούν από κοινού τις κλάσεις Openstack , GeneralSshClient και PubPrivateKeyUtils.

Openstack

Η κλάση Openstack είναι υπεύθυνη για την επικοινωνία με το περιβάλλον υπολογιστικού νέφους Openstack. Χρησιμοποιώντας το API του Jclouds η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Ταυτοποίηση με το Openstack

2 Δημιουργία εικονικών μηχανών

3 Διαγραφή εικονικών μηχανών

4 Επιστροφή στοιχείων εικονικών μηχανών

PubPrivateKeyUtils

Η κλάση PubPrivateKeyUtils είναι υπεύθυνη για την δημιουργία ζευγών ιδιωτικών-ξένων κλειδιών μέσα στο Openstack . Τα κλειδιά είναι απαραίτητα για την επικοινωνία των Controllers με τις εικονικές μηχανές. Χρησιμοποιώντας το API του Jclouds η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Ταυτοποίηση με το Openstack

2 Δημιουργία ζευγών ιδιωτικών-ξένων κλειδιών

3 Αποθήκευση ζευγών ιδιωτικών-ξένων κλειδιών τοπικά ή στον σερβερ που τρέχει η εφαρμογή

GeneralSshClient

Η κλάση GeneralSshClient είναι υπεύθυνη για την δημιουργία καναλιών επικοινωνίας με τις εικονικές μηχανές . Η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Άνοιγμα καναλιών επικοινωνίας για την μεταφορά εντολών εκτέλεσης.

2 Άνοιγμα καναλιών επικοινωνίας για την μεταφορά αρχείων στο εικονικό μηχάνημα

3 Άνοιγμα καναλιών επικοινωνίας για το κατέβασμα αρχείων από εικονικό μηχάνημα

Υλοποίηση δοκιμής για βάσεις δεδομένων :

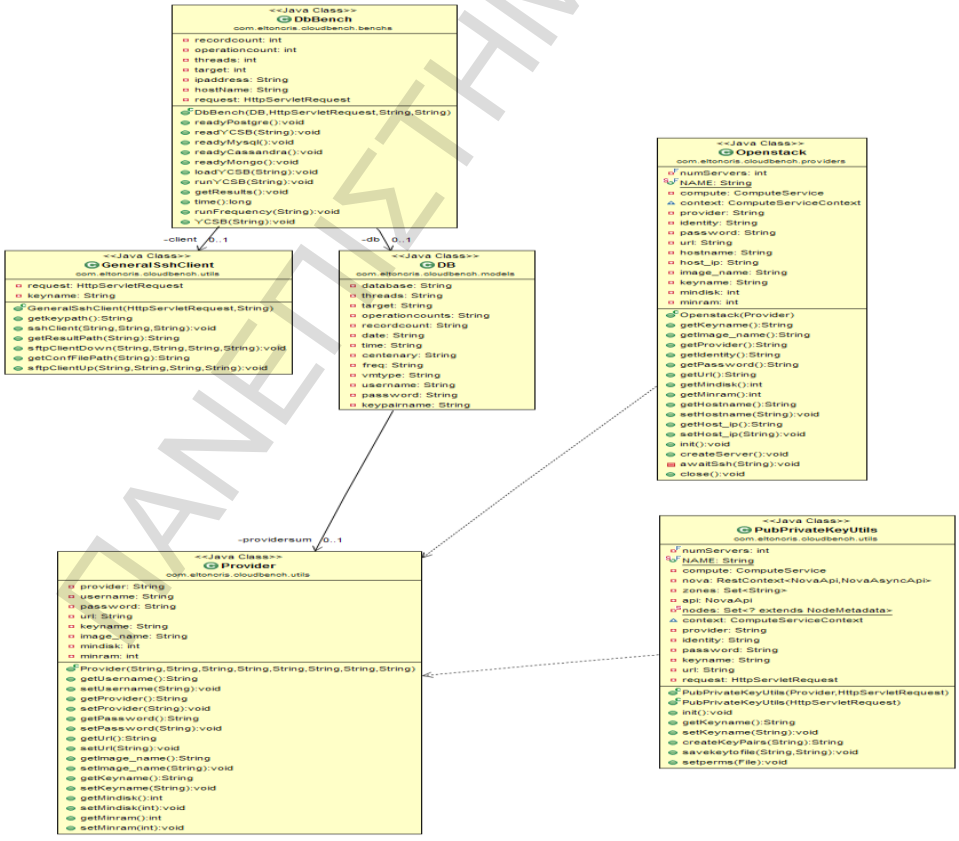
Εφόσον η εικονική μηχανή έχει δημιουργηθεί επιτυχώς από τα προηγούμενα βήματα της εφαρμογής η κλάση που αναλαμβάνει εγκατάσταση των απαραίτητων πακέτων και την ρύθμιση αυτών για την υλοποίηση της δοκιμής Βάσεων Δεδομένων, χρησιμοποιώντας την κλάση GeneralSshClient, είναι η κλάση DbBench. Η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Αυτόματη εγκατάσταση, παραμετροποίηση και εκκίνηση των βάσεων δεδομένων (MySQL, PostgreSQL, Cassandra, MongoDb)

2 Αυτόματη εγκατάσταση και παραμετροποίηση του εργαλείου YCSB

3 Εκτέλεση των load και run δοκιμών

Παρακάτω παρουσιάζεται και δίγραμμα της κλάσης αυτής και πως αλληλεπιδρά με άλλες κλάσεις



Εικόνα 14: Διάγραμμα κλάσης DBBench

Υλοποίηση δοκιμής για Web Serving:

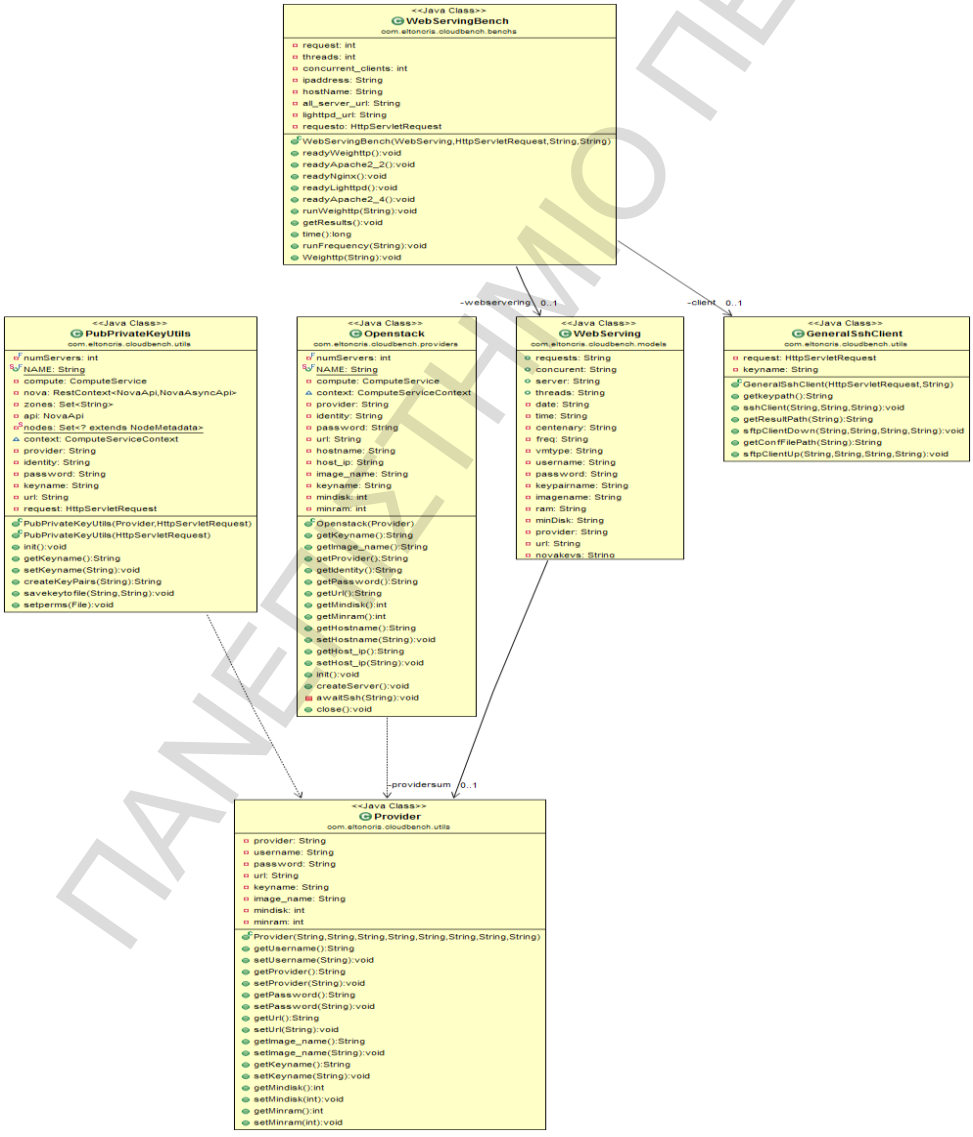
Εφόσον η εικονική μηχανή έχει δημιουργηθεί επιτυχώς από τα προηγούμενα βήματα της εφαρμογής η κλάση που αναλαμβάνει εγκατάσταση των απαραίτητων πακέτων και την ρύθμιση αυτών για την υλοποίηση της δοκιμής Web Serving, χρησιμοποιώντας την κλάση GeneralSshClient, είναι η κλάση WebServingBench. Η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Αυτόματη εγκατάσταση, παραμετροποίηση και εκκίνηση των Web Servers (Apache2.2, Apache2.4, Nginx, Lighttpd)

2 Αυτόματη εγκατάσταση και παραμετροποίηση του εργαλείου Weighttp

3 Εκτέλεση των load και run δοκιμών

Παρακάτω παρουσιάζεται και δίγραμμα της κλάσης αυτής και πως αλληλεπιδρά με άλλες κλάσεις



Εικόνα 15: Διάγραμμα κλάσης WebServingBench

Υλοποίηση δοκιμής για Hadoop:

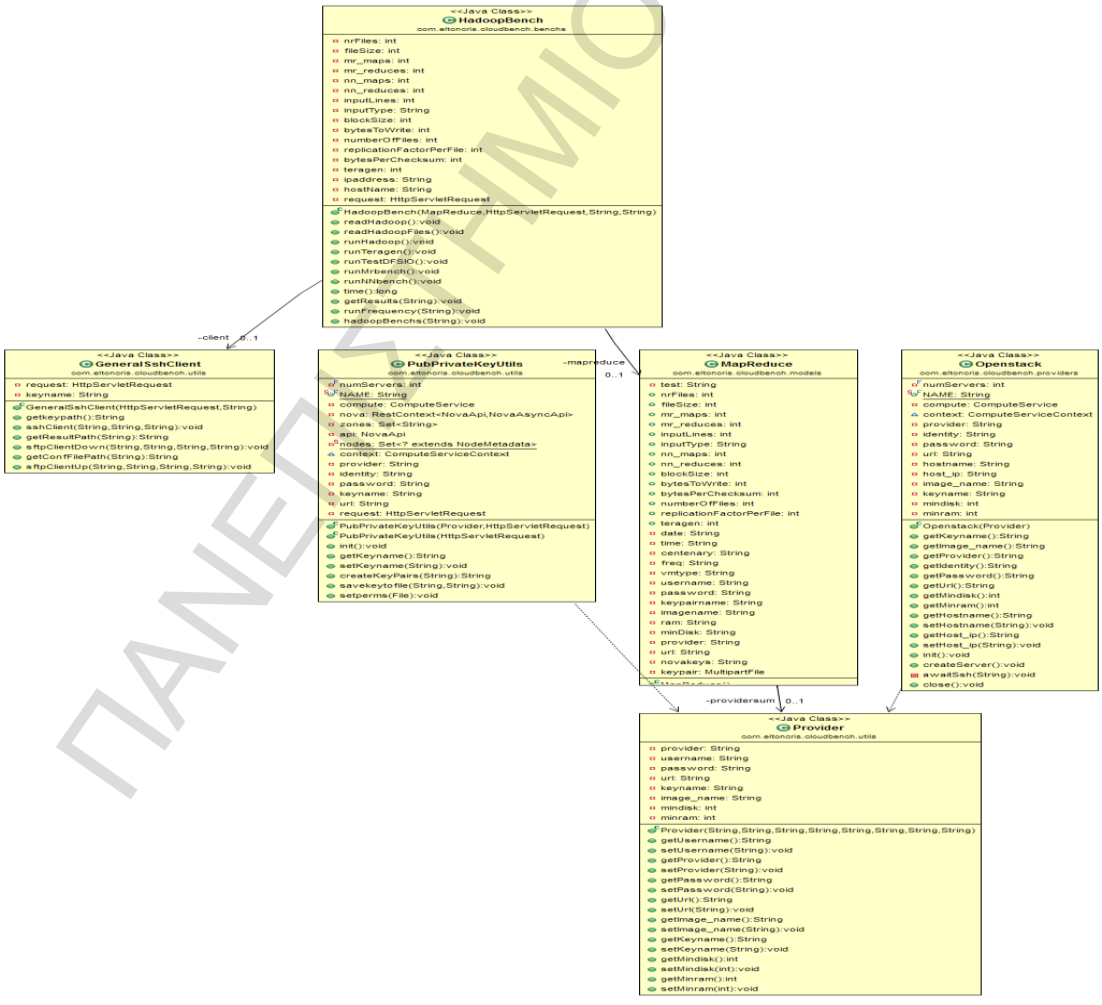
Εφόσον η εικονική μηχανή έχει δημιουργηθεί επιτυχώς από τα προηγούμενα βήματα της εφαρμογής η κλάση που αναλαμβάνει εγκατάσταση των απαραίτητων πακέτων και την ρύθμιση αυτών για την υλοποίηση της δοκιμής MaReduce, χρησιμοποιώντας την κλάση GeneralSshClient, είναι η κλάση HadoopBench. Η κλάση αυτή μέσω των μεθόδων της επιτρέπει :

1 Αυτόματη εγκατάσταση ,παραμετροποίηση και εκκίνηση του περιβάλλοντος Hadoop

2 Εγκατάσταση και παραμετροποίηση των εργαλείων TestDFSIO, MRBench, NNBench, Teragen-Terasort

3 Εκτέλεση των load και run δοκιμών

Παρακάτω παρουσιάζεται και δίγραμμα της κλάσης αυτής και πως αλληλεπιδρά με άλλες κλάσεις



Εικόνα 16: : Διάγραμμα κλάσης HadoobBench

5 ΑΠΟΤΕΛΕΣΜΑΤΑ

Στην παρούσα εργασία όπως αναφέραμε στο κεφάλαιο 4 χρησιμοποιήθηκε μόνο ένα ιδιωτικό περιβάλλον υπολογιστικού νέφους(private cloud) Openstack καθώς δεν υπήρχε η δυνατότητα να χρησιμοποιήσουμε παρόχους δημοσίου υπολογιστικού νέφους (public cloud providers) όπως Amazon, Rackspace ή Azure.Υπό κανονικές συνθήκες θα έπρεπε να εκτελέσουμε όλα της δοκιμές με τις ίδιες μεταβλητές εισόδου σε όλους τους παρόχους για να λάβουμε αποτελέσματα που θα εξηγούσαν και αξιολογούσαν ποιος παρόχος είναι καλύτερος για την κάθε είδους εφαρμογής. Επίσης, θα έπρεπε να εκτελέσουμε τις δοκιμές σε όλους τους παρόχους σε διαφορετικές χρονικές περιόδους μέσα σε μια μέρα για να δούμε πως αποδίδει κάθε παρόχος στην διάρκεια μια μέρας. Αυτό είναι σημαντικό καθώς είναι γνωστό ότι σε ένα public cloud υπάρχουν στιγμές μέσα στην ημέρα που το φόρτο εργασίας που δέχεται η υποδομή είναι μεγαλύτερη από άλλες στιγμές. Για παράδειγμα είναι λογικό κατά την διάρκεια της ημέρας να υπάρχει πιο πολύ φόρτο εργασίας στην υποδομή παρά το βραδύ. Αυτό είναι πολύ σημαντικό στοιχείο για έναν χρηστή γιατί του δίνει την πληροφορία όχι μόνο για το ποιος παρόχος είναι καλύτερος γενικά αλλά και ποιος παρόχος αποδίδει καλύτερα σε συγκεκριμένες ώρες της ημέρας.

Επειδή, όπως προαναφέραμε, έχουμε πρόσβαση μόνο σε private cloud Openstack δεν μπορούμε εκτελέσουμε δοκιμές για άλλους παρόχους θα προσπαθήσουμε να προσομοιώσουμε σενάρια με διαφορετικά φόρτοι εργασίας στην υποδομή του private cloud για να αξιολογήσουμε την απόδοση των εφαρμογών σε διαφορετικά σενάρια . Για να το επιτύχουμε αυτό δημιουργήσαμε δυο σενάρια.

- Στο πρώτο σενάριο στην υποδομή του Openstack υπάρχει μόνο μια εικονική μηχανή στην οποία εγκαθιστάτε η εφαρμογή την οποία θέλουμε να αξιολογήσουμε και την αντίστοιχη δοκιμή.
- Στο δεύτερο σενάριο στην υποδομή του Openstack, εκτός από την εικονική μηχανή στην οποία είναι εγκατεστημένη η εφαρμογή προς αξιολόγηση και τη αντίστοιχη δοκιμή, δημιουργούμε άλλες 5 εικονικές μηχανές οι οποίες τρέχουν διαφορές εφαρμογές με σκοπό να καταναλώσουν πόρους από την υποδομή του Openstack (επεξεργαστική ισχύ, μνήμη, χώρος αποθήκευσης και δίκτυο). Σκοπός είναι να

φέρουμε την υποδομή στα άκρα της και επειδή το μικρό ιδιωτικό μας cloud μπορεί να ανταπεξέλθει μέχρι και σε 6 εικονικές μηχανές, με χαρακτηριστικά που περιγράφονται παρακάτω, δημιουργώντας 5 εικονικές μηχανές μαζί με την μια εικονική μηχανή για τις δοκιμές φέρνουμε την υποδομή μας στα άκρα.

Οι εικονικές μηχανές που δημιουργήθηκαν και στα δυο σενάρια έχουν τα έξης χαρακτηριστικά :

Εικόνα (Image)	Εικονικούς επεξεργαστές (VCPU)	Μνήμη (RAM)	Εφήμερος χώρος αποθήκευσης (Ephemeral Disk)	Μόνιμος χώρος αποθήκευσης (Root Disk)
Ubuntu Server 12.04 LTS	2	2 GB	0 GB	10 GB

5.1 Αποτελέσματα για Βάσεις Δεδομένων

Οι δοκιμές εκτελεστήκαν για τις βάσεις δεδομένων MySQL, PostgreSQL, Cassandra και MongoDb. Οι δοκιμές σε όλες τις υπό εξέταση βάσεις δεδομένων εκτελέστηκαν και στα δυο σενάρια με τις ίδιες ακριβώς παραμέτρους έτσι ώστε να δούμε πώς επηρεάζετε η απόδοση τους από το φόρτο εργασίας της υποδομής Cloud . Τα αποτελέσματα που επιλέξαμε προς σύγκριση είναι ο συνολικός χρόνος εκτέλεσης μίας δοκιμ΄ς και ο ρυθμός εκτέλεσης συναλλαγών ανά δευτερόλεπτο(Throughput) . Όπως αναφέραμε στο κεφάλαιο 3 το εργαλείο YCSB έχει δυο φάσεις , load και run(transaction), περνούμε αποτελέσματα και για τις δυο φάσεις.

Η φάση load και transaction σε όλες τις βάσεις δεδομένων και στα δυο σενάρια εκτελέστηκε με τις παρακάτω παραμέτρους :

Στόχος (Target) = 100 (Φάση transaction)

Νήματα (Threads) = 10 (Φάση transaction και load)

Εισαγωγές στην βάση (recordcound) = 10000 (Φάση load)

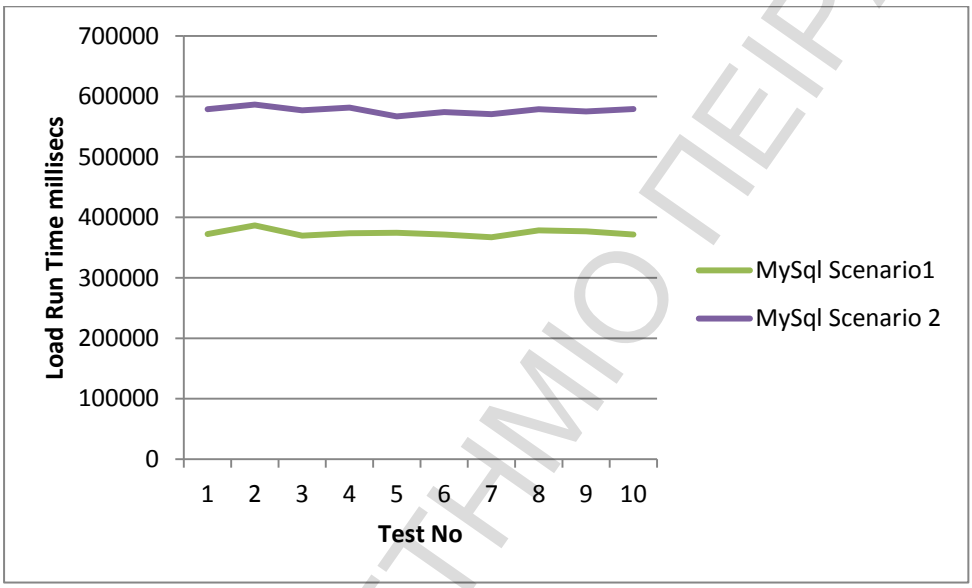
Συναλλαγές ανάγνωσης και ενημέρωσης με την βάση (Operationcount)= 10000 (Φάση transaction)

Η δοκιμή επαναλήφτηκε 10 φόρες για την κάθε βάση δεδομένων και στα δυο σενάρια ώστε να αναλύσουμε καλύτερα τα αποτελέσματα.

5.1.1 MySql

Φάση Load

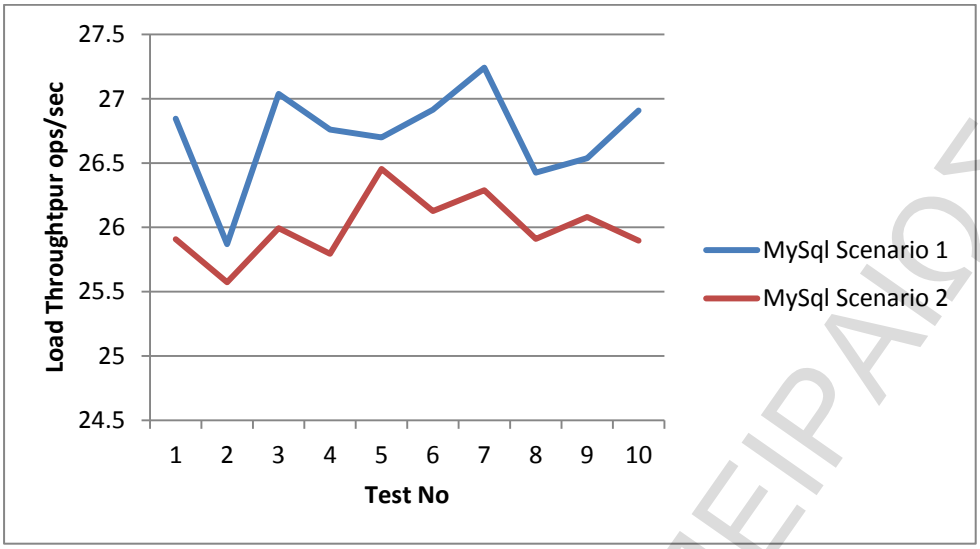
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 1: Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση load για MySQL

Στο γράφημα παρατηρούμε ότι ο συνολικός χρόνος εκτέλεσης της δοκιμής στην φάση load και στα δυο σενάρια κυμαίνεται σταθερός ωστόσο η διάφορα του χρόνου μεταξύ των δυο σεναρίων είναι αρκετά μεγάλος.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:

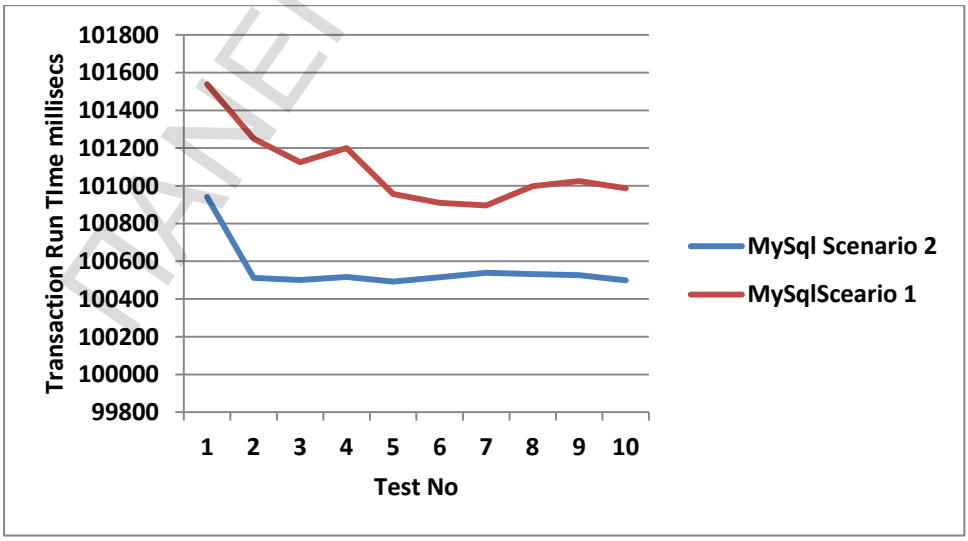


Γράφημα 2:Throughput στην φάση load για MySQL

Στο γραφήματα αυτό παρατηρούμε ότι στο δεύτερο σενάριο το throughput μειώνετε αρκετά καθώς η υποδομή cloud βρίσκεται στα όρια της και η απόδοση της εικονικής μηχανής, πάνω στην οποία η MySQL είναι εγκατεστημένη, αρχίζει και πέφτει και αυτό έχει αντίκτυπο στο ρυθμό των συναλλαγών που μπορεί να κάνει η MySQL. Γενικά όμως το throughput και στα δυο σενάρια είναι αρκετό χαμηλό για την φάση αυτή.

Φάση Transaction

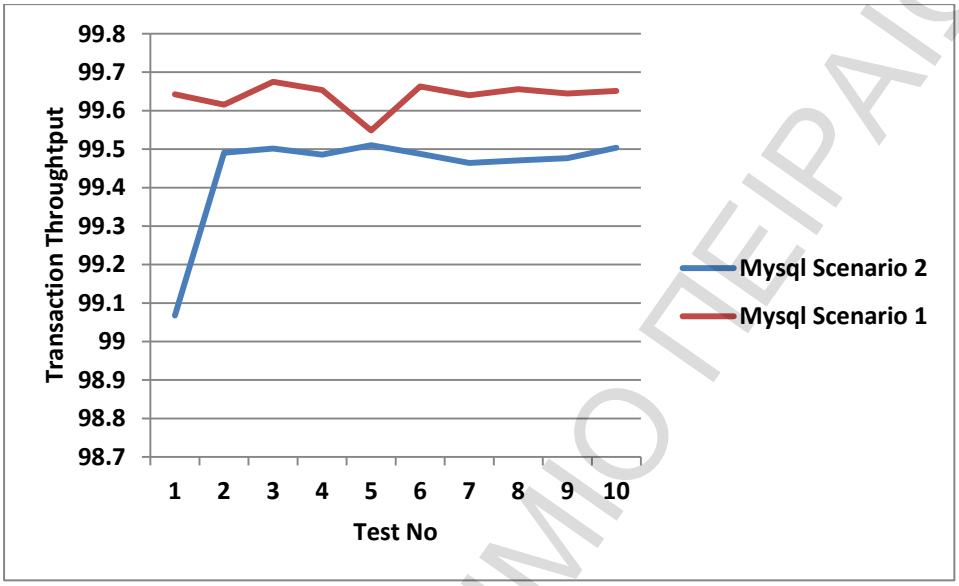
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 3:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση transaction για MySQL

Όπως και στην φάση load έτσι και στην φάση transaction ο χρόνος εκτέλεσης της δοκιμής για το δεύτερο σενάριο είναι μεγαλύτερος του προτού σεναρίου σε όλες τις επαναλήψεις . Ωστόσο σε αυτήν την φάση η χρόνοι είναι καλύτεροι σε σχέση με την φάση load και στο δυο σενάρια και οι δόκιμες εκτελούνται πιο γρήγορα

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 4:Throughput στην φάση transaction για MySQL

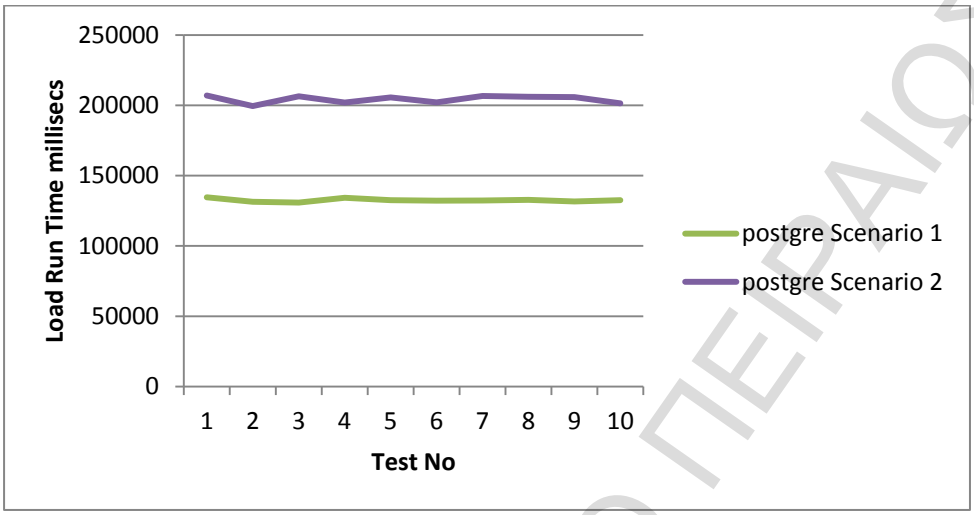
Το Throughput ,στην φάση transaction, έχει βελτιωθεί αισθητά και στα δυο σενάρια, και στα δυο σενάρια πλησιάζουν τον στόχο των 100 συναλλαγών ανά δευτερόλεπτο που είχαμε θέσει όταν εκτελέσαμε την δόκιμη, με την δεύτερο σενάριο όπως είναι φυσικό να υπολείπετε του πρώτου.

Γενικά παρατήσουμε στην MySQL ότι είναι αργή κατά την φάση load, οπού και εισάγει τα δεδομένα στην βάση δεδομένων, όποτε σε αυτήν την περίπτωση ένα χρήστης πρέπει να γνωρίζει ποιος πάροχος είναι καλύτερος όταν η υποδομή του δέχεται πολύ φόρτο έτσι ώστε ο χρόνος μεταφόρτωσης δεδομένων στην βάση να επηρεάζετε λιγότερο

5.1.2 PostgreSQL

Φάση Load

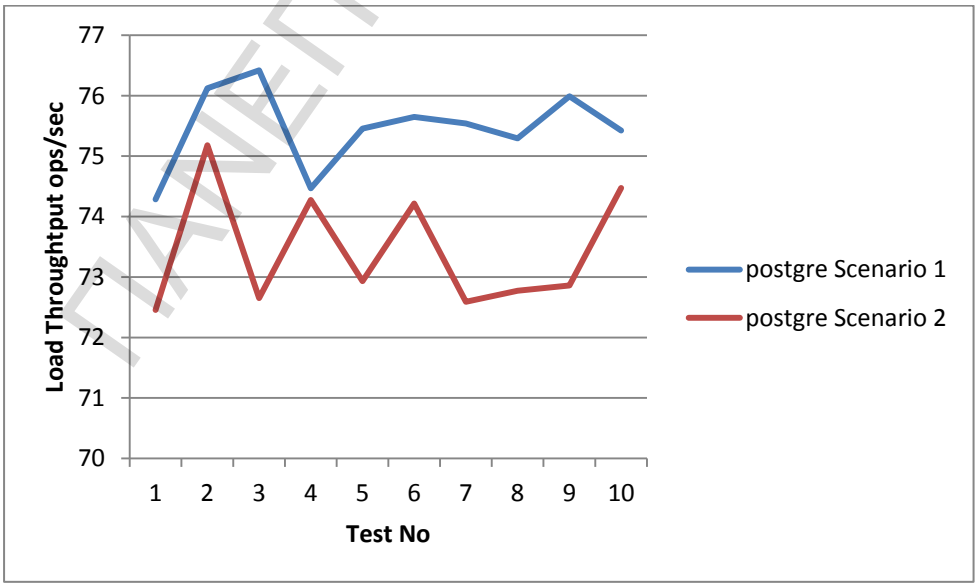
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 5:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση load για PostgreSQL

Ο συνολικός χρόνος εκτέλεσης των δοκίμων ακολουθεί το ίδιο μοτίβο με την MySQL αλλά με καλυτέρους χρόνους και αυτό οφείλεται κυρίως ότι η Postgre έχει γενικότερα καλύτερη απόδοση από την MySQL ,ασχέτως υποδομής που είναι εγκατεστημένη

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:

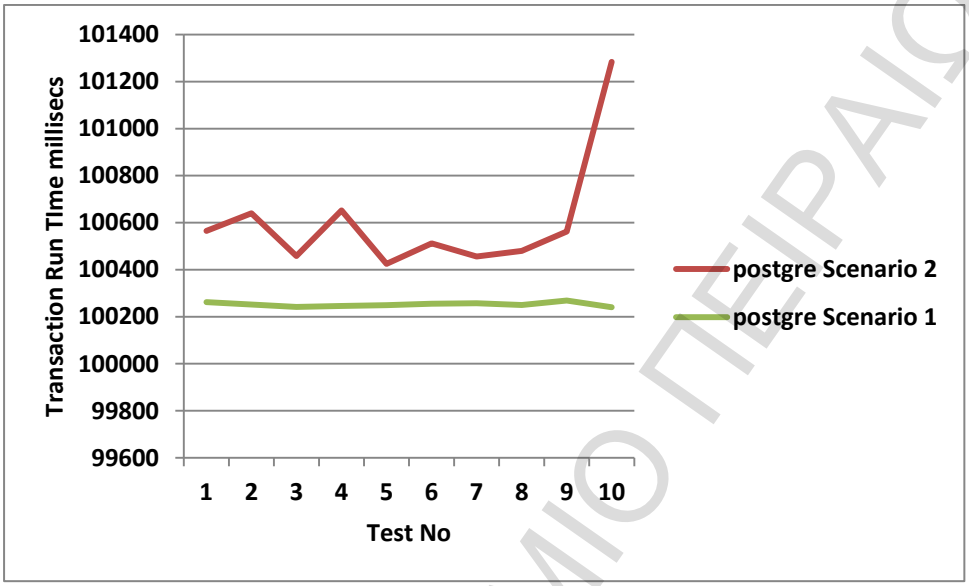


Γράφημα 6:Throughput στην φάση load για PostgreSQL

Το ίδιο παρατηρούμε και σε αυτό το γράφημα. Το throughput έχει βελτιωθεί σε σχέση με την MySQL για τον ίδιο λόγο που προαναφέραμε και όπως αναμέναμε το δεύτερο σενάριο έχει μικρότερο throughput.

Φάση Transaction

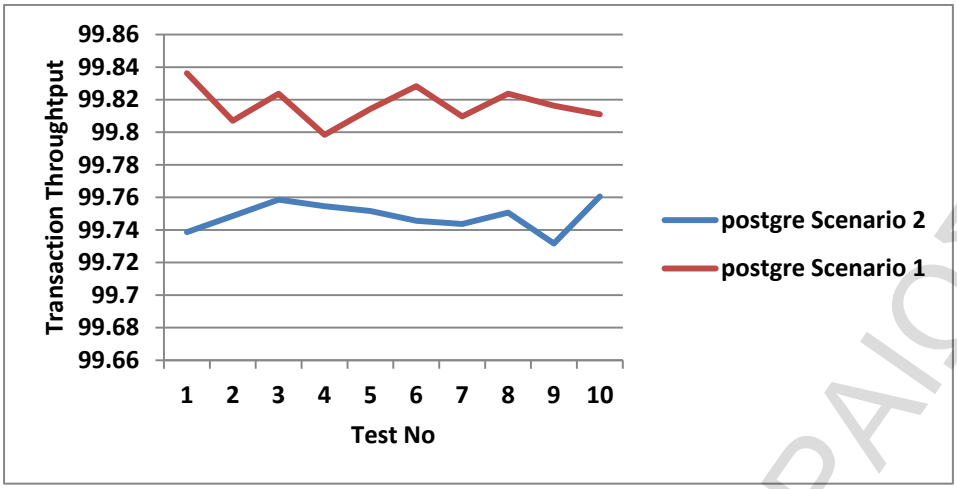
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 7:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση transaction για PostgreSQL

Στην φάση του transaction οι χρόνοι εκτέλεσης της δοκιμής είναι μικρότεροι σε σχέση με την φάση load ωστόσο παρατηρούμε ότι στο δεύτερο σενάριο ότι οι χρόνοι έχουν διακυμάνσεις με μεγαλύτερο αυτό της τελευταίας επανάληψης της δοκιμής. Αυτό πολύ πιθανών να οφείλετε ότι εκείνη την χρονική στιγμή, ενώ η υποδομή cloud βρίσκεται στα όρια του , η εικονική μηχανή οπού η Postgre είναι εγκατεστημένη δυσκολεύετε να βρει αποδοτικούς πόρους για να κάνει τις απαραίτητες συναλλαγές.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:



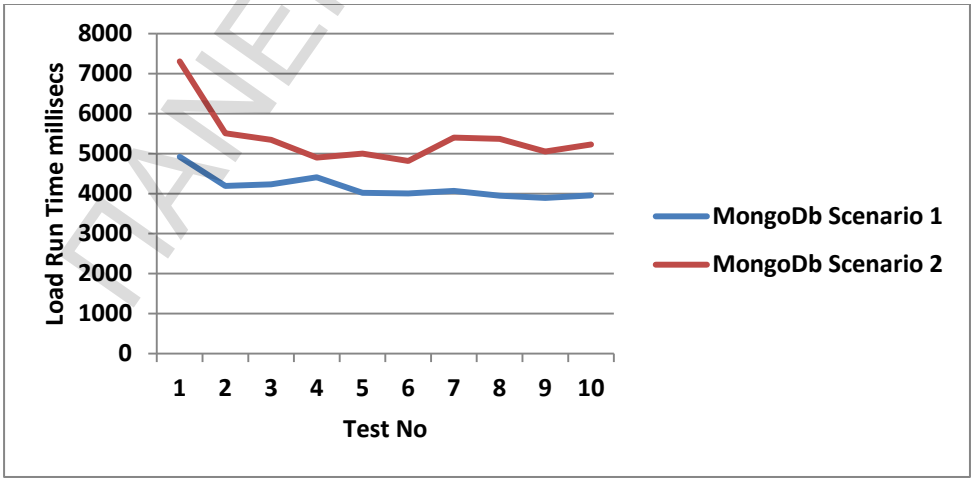
Γράφημα 8:Throughput στην φάση transaction για PostgreSQL

Στο γράφημα αυτό παρατηρούμε ότι ο ρυθμός συναλλαγών ανά δευτερόλεπτο έχει φτάσει σχεδόν τον στόχο μας που είναι 100. Αυτό ισχύει και για το δεύτερο σενάριο με την διάφορα να είναι της τάξης του 0,1 στα 100. Αυτό γενικά μας δείχνει ότι ηPostgre στην φάση transaction συνεχίζει και αποδίδει καλά ακόμα και όταν η υποδομή cloud μας βρίσκεται στα όρια του αλλά επηρεάζετε από αυτό ,όπως και ηMySQL, στην φάση του load.

5.1.3 MongoDb

Φάση Load

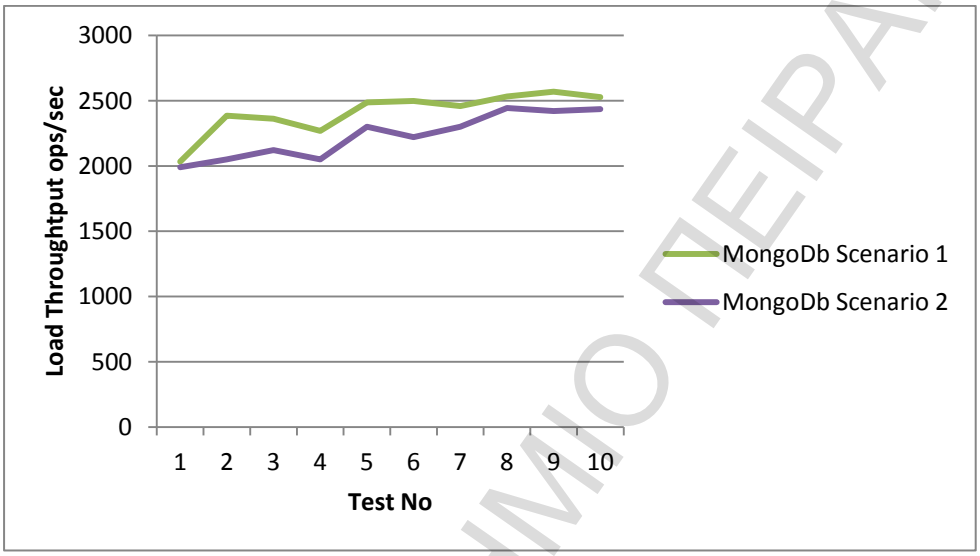
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 9:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση load για MongoDb

Αυτό που παρατηρούμαι αμέσως στο γράφημα αυτό είναι η σημαντική διαφορά στους χρόνους εκτέλεσης των δοκιμών σε σχέση με τις σχεσιακές βάσεις δεδομένων MySQL και PostGre . Η MongoDb, όντας μη-σχεσιακή βάση δεδομένων, έχει χρόνους αρκετά μικρότερους σε σχέση με τις προηγούμενες βάσεις δεδομένων. Παρόλα αυτά όπως και εδώ αναμέναμε οι χρόνοι του δεύτερου σεναρίου υπολείπονται του πρώτου, με πιο ακραίο παράδειγμα την πρώτη επανάληψη που πιθανών οφείλετε στην φθορά της απόδοσης των πόρων της υποδομής cloud λόγο υπερφόρτωσης αυτής.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:

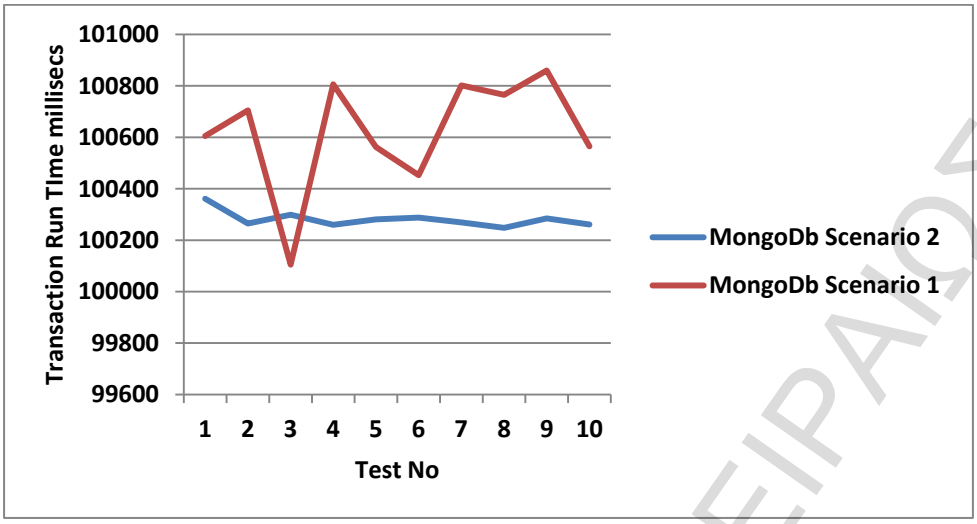


Γράφημα 10:Throughput στην φάση load για MongoDb

Επίσης και σε αυτό το γράφημα στην φάση του load παρατηρούμε ότι το throughput είναι σημαντικό καλύτερο από τις σχεσιακές βάσεις δεδομένων. Ακόμα, και στο δεύτερο σενάριο οι διάφορες του ρυθμού συναλλαγών δεν είναι αρκετές μεγάλες σε σχέση με το πρώτο σενάριο.

Φάση Transaction

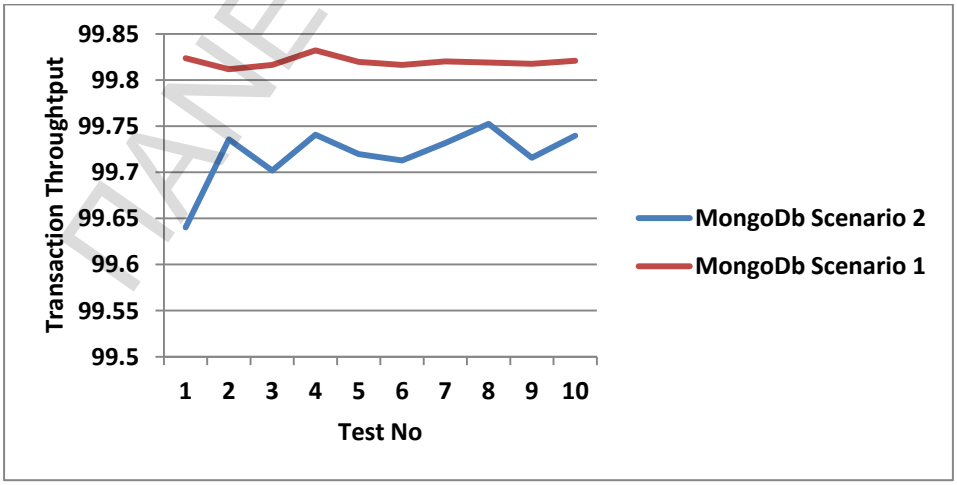
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 11:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση transaction για MongoDb

Στην φάση του transaction οι συνολικοί χρόνοι δεν διαφέρουν παρά πολύ σε σχέση με τις σχεσιακές βάσεις δεδομένων και ακλουθούν σχεδόν το ίδιο μοτίβο. Αξίζει να σημειωθεί ότι ο χρόνος της 3ης επανάληψης της δοκιμής για το δεύτερο σενάριο ακόμα πιο μικρός τις αντίστοιχης επανάληψης για το πρώτο σενάριο. Αυτό συνέβη γιατί πολύ πιθανών εκείνη την χρονική στιγμή που άρχισε η δόκιμη αυτή η υποδομή του cloud συνερχόταν από κάποιο mass failure και η 5 υπόλοιπες εικονικές μηχανές του δεύτερου σεναρίου δεν είχαν ανακτηθεί ακόμα. Αυτό θα είχε σαν αποτέλεσμα όλοι οι πόροι της υποδομής να ήταν διαθέσιμοι μονό στην εικονική μηχανή που έτρεχε η δόκιμη και αυτό με την σειρά του επηρέασε τον χρόνο εκτέλεσης και είναι ο λόγος που βλέπουμε τόσο γρήγορο χρόνο εκτέλεσης.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:



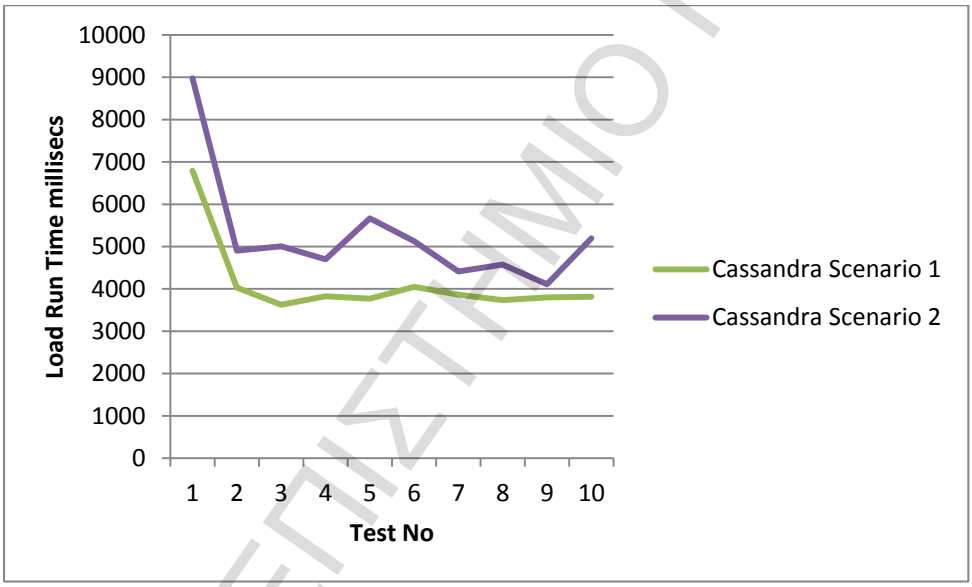
Γράφημα 12:Throughput στην φάση transaction για MongoDb

Το throughput και στα δυο σενάρια σχεδόν πλησιάζει τον στόχο που θέσαμε (100 συναλλαγές ανά δευτερόλεπτο). Το δεύτερο σενάριο όπως είναι λογικό έχει χειρότεροι απόδοση από το πρώτο σενάριο και ξεκάνει από σχετικά χαμηλά σε σχέση με τις υπόλοιπες μετρήσεις αυτού του γραφήματος. Αυτό είναι σημάδι ότι οι πόροι της υποδομής cloud έχουν χάσει την απόδοση στους αλλά το ότι στην συνεχεία το throughput αυξάνεται σημαίνει ότι οι υπόλοιπες 5 εικονικές μηχανές του δεύτερου σεναρίου δεν χρησιμοποιούν τόσο πολύ τους πόρους και έτσι δίνετε η δυνατότητα στην Mongo να αποδώσει καλυτέρα

5.1.4 Cassandra

Φάση Load

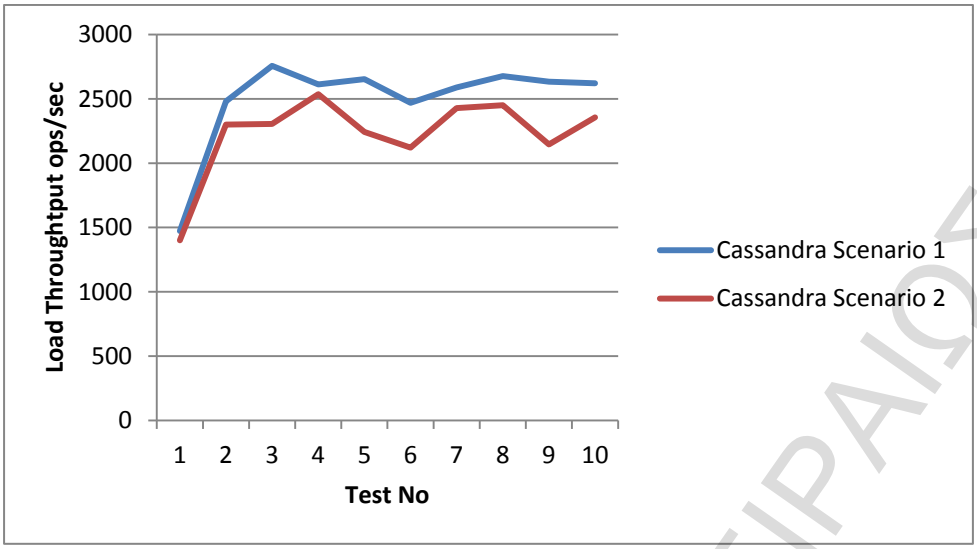
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 13:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση load για Cassandra

Στο γράφημα αυτό παρατηρούμαι το ίδιο μοτίβο στις διαφορές τον χρονών όπως και στην Mongo με τους χρόνους συνολικής εκτέλεσης των δοκιμών να είναι ελαφρώς υψηλότεροι μονό στην 2 πρώτες επαναλήψεις αλλά αυτό μπορεί να οφείλεται στις συνθήκες της υποδομής εκείνες τις χρονικές περιόδους. Οι υπόλοιποι χρόνοι δεν διαφέρουν και τόσο πολύ καθώς η Cassandra όπως και η Mongo είναι μη-σχεσιακές βάσεις δεδομένων και γενικότερα πολύ καλή και πανόμοια απόδοση.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:

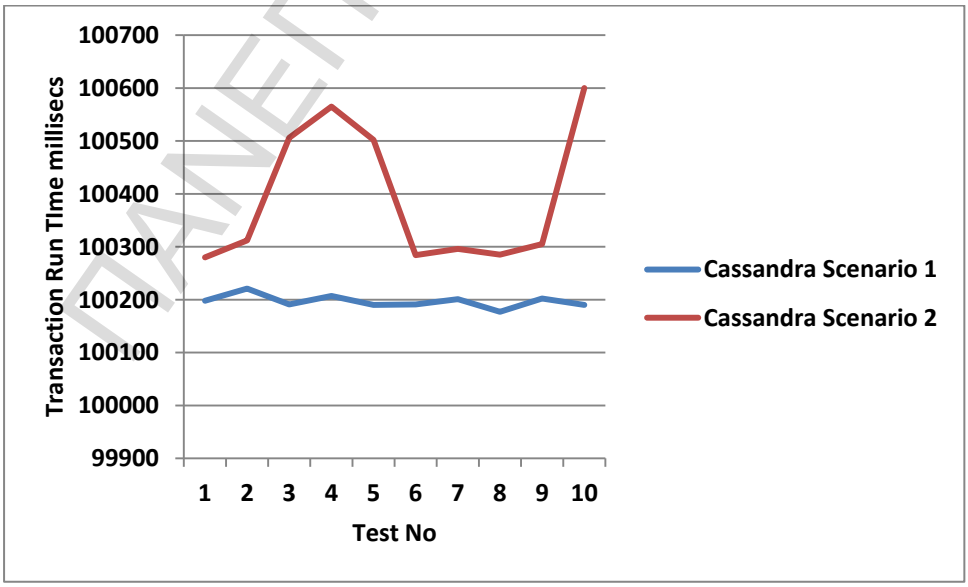


Γράφημα 14:Throughput στην φάση load για Cassandra

Στο γράφημα αυτό παρατηρούμαι ότι το throughput φάση load είναι αρκετά υψηλό και στα δυο σενάρια και το δεύτερο σενάριο δεν φαίνεται να επηρεάζετε πολύ από την υπερφόρτωση της υποδομής cloud . Αυτό που παρατηρούμε επίσης είναι η χαμηλή τιμή του throughput στην πρώτη επανάληψη για το πρώτο σενάριο αλλά πολύ πιθανό αυτό να οφείλετε σε κάποιο αστοχία στην υποδομή cloud εκείνη την χρονική στιγμή που έριξε την απόδοση των πόρων.

Φάση Transaction

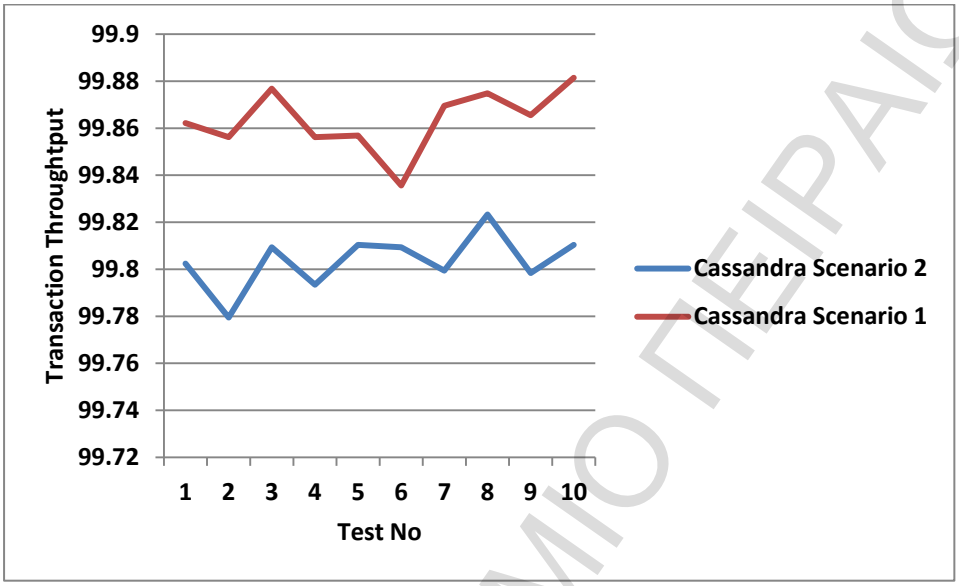
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 15:Συνολικός χρόνος εκτέλεσης δοκιμής στην φάση transaction για Cassandra

Όπως και στην Mongo έτσι και στην Cassandra οι συνολικοί χρόνοι εκτέλεσης στην φάση transaction είναι υψηλού με τους χρόνους του δεύτερου σεναρίου ακόμα πιο υψηλοί. Στο δεύτερο σενάριο παρατηρούμε ότι οι πόροι της υποδομής cloud δεν αποδίδουν καλά στις επαναλήψεις 3, 4, 5 και 10 με τους χρόνους εκτέλεσης των δοκιμών να αυξάνονται σημαντικά.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 16:Throughput στην φάση transaction για Cassandra

Στο γράφημα αυτό παρατηρούμε ότι το throughput είναι αρκετά υψηλό και στα δυο σενάρια με αυτό του δεύτερου να υπολείπετε αλλά οι διαφορές είναι πολύ μικρές.

Γενικά στην μη-σχεσιακές βάσεις παρατηρούμαι ότι αυτό που επηρεάζετε πιο πολύ από την υπερφόρτωση της υποδομής cloud είναι ο συνολικός χρόνος εκτέλεσης συναλλαγών ενημέρωσης και ανάγνωσης (φάση transaction).

5.1.5 Συμπεράσματα για Βάσεις Δεδομένων

Παρατηρούμε σε όλα τα γραφήματα και για όλες τις βάσεις δεδομένων το ίδιο μοτίβο αποτελεσμάτων. Ο χρόνος εκτέλεσης των δοκιμών είναι μεγαλύτερος στο δεύτερο σενάριο και επίσης ο ρυθμός συναλλαγών ανά δευτερόλεπτο είναι μικρότερος στο δεύτερο σενάριο. Αυτό ήταν αναμενόμενο καθώς από την στιγμή που η υποδομή cloud βρίσκετε στα όρια της οι εικονικές μηχανές ανταγωνίζονται η μια την άλλη για επεξεργαστική ισχύ και μνήμη. Από την στιγμή που μια εφαρμογή όπως η βάση δεδομένων χρειάζεται πολλούς πόρους όπως επεξεργαστική ισχύ αλλά ιδιαίτερα μνήμη και μέσο αποθήξευσης ,όταν η υποδομή cloud βρίσκεται στα όρια της, η απόδοση των πόρων πέφτει και αυτό έχει σημαντικό αντίκτυπο στην απόδοση των βάσεων δεδομένων. Ο αντίκτυπος είναι μεγαλύτερος για τις παραδοσιακές σχεσιακές βάσεις δεδομένων, δηλαδή MySQL και PostgreSQL, που από μονές τους απαιτούν υπολογιστικούς πόρους σε σχέση με τις μη-σχεσιακές βάσεις δεδομένων[40].

5.2 Αποτελέσματα για Web Serving

Οι δοκιμές εκτελεστήκαν για τους web servers Apache2.2, Apache2.4, Nginx και Lighttpd. Οι δοκιμές σε όλες τους υπό εξέταση web servers εκτελέστηκαν και στα δυο σενάρια με τις ίδιες ακριβώς παραμέτρους έτσι ώστε να δούμε πώς επηρεάζετε η απόδοση τους από το φόρτο εργασίας της υποδομής Cloud . Τα αποτελέσματα που επιλέξαμε προς σύγκριση είναι ο συνολικός χρόνος εκτέλεσης μίας δοκιμής και ο μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο.

Η δοκιμή Weighthttp εκτελέστηκε σε όλους τους web servers και στα δυο σενάρια με τις παρακάτω παραμέτρους :

Νήματα (Threads) = 2

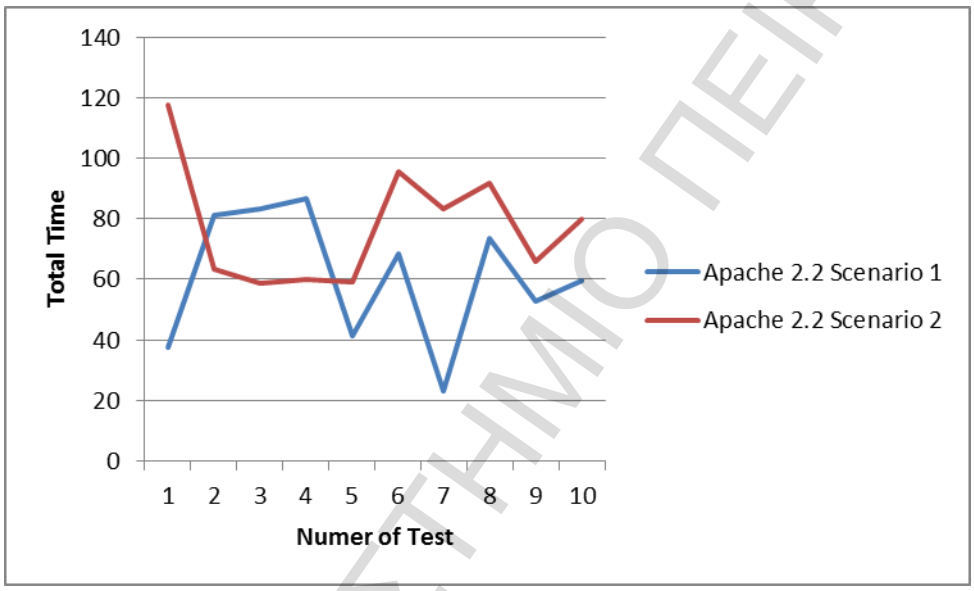
Ταυτόχρονες αιτήσεις (Concurrent requests)= 20

Συνολικές αιτήσεις(Request) = 100000

Η δοκιμή επαναλήφτηκε 10 φόρες για την κάθε τον κάθε web server και στα δυο σενάρια ώστε να αναλύσουμε καλύτερα τα αποτελέσματα.

5.2.1 Apache2.2

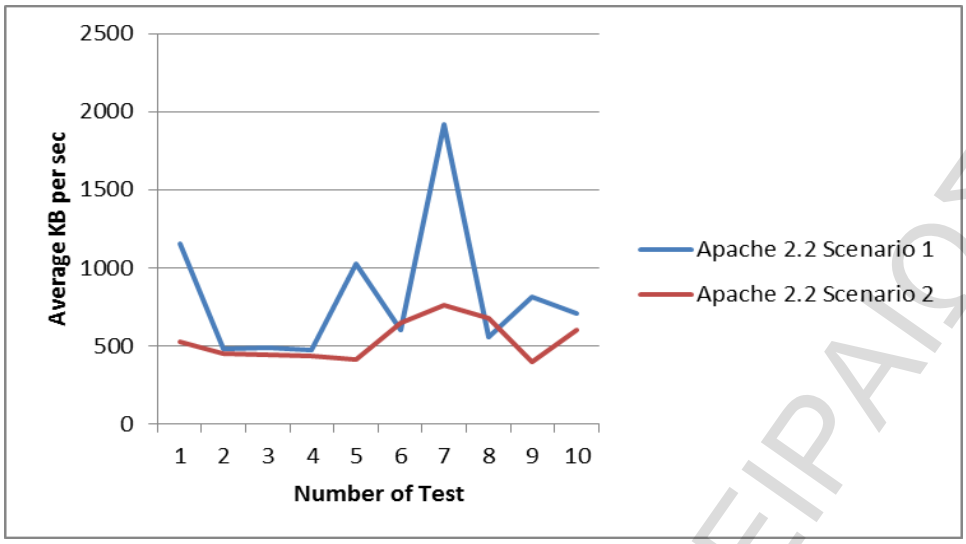
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 17: Συνολικός χρόνος εκτέλεσης δοκιμής για Apache 2.2

Στο γράφημα αυτό παρατηρούμε ότι ο συνολικός χρόνος εκτέλεσης των δοκιμών είναι μεγαλύτερος στο δεύτερο σενάριο αλλά όχι σε όλες της επαναλήψεις. Στις επαναλήψεις 2,3,4,5 οι χρόνοι είναι ακόμα μικρότεροι και από τις επανάληψης του πρώτου σεναρίου και αυτό πολύ πιθανών να οφείλεται στο γεγονός ότι η υποδομή cloud ανακάμπτει από κάποια αποτυχία και επειδή η εικονικοί μηχανή με τις δόκιμες ανακτάτε πρώτη έχει στην διάθεση της όλους τους πόρους της υποδομής.

Ο Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 18: Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο για Apache 2.2

Στο γράφημα αυτό βλέπουμε ότι στις περισότερες επαναλήψεις της δοκιμής ο μέσος ρυθμός μεταφοράς KB είναι σχεδόν ίδιος και στα δυο σενάρια εκτός από της επαναλήψεις 1,4,7 οπού βλέπουμε ότι η υπερφόρτωση της υποδομής cloud (δεύτερο σενάριο) επηρεάζει και μειώνει τον ρυθμό αυτό.

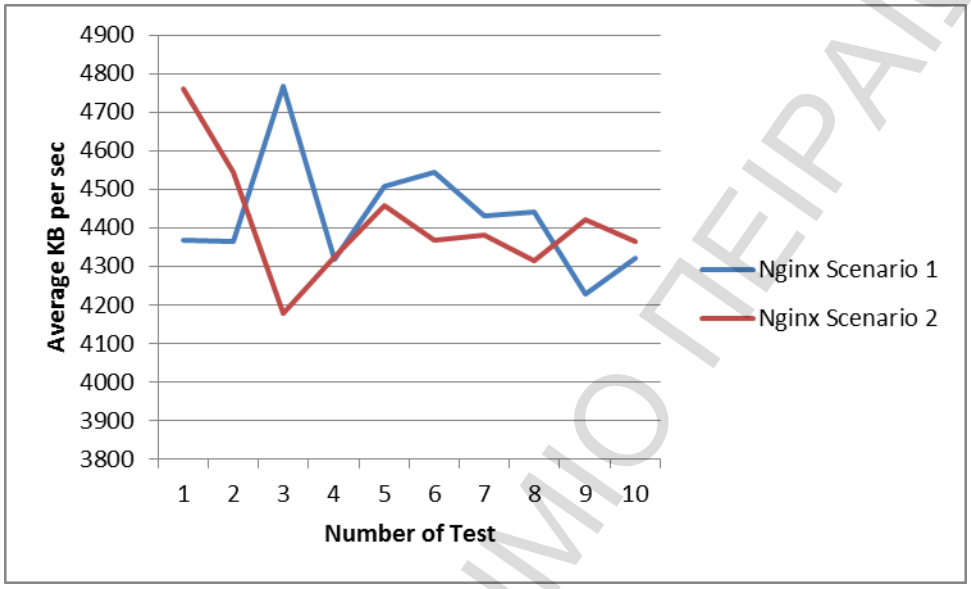
5.2.2 Nginx

Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:

Γράφημα 19: Συνολικός χρόνος εκτέλεσης δοκιμής για Nginx

Αυτό που παρατηρούμε γρήγορα στους συνολικούς χρόνους εκτέλεσης των δοκιμών στον web server Nginx είναι η μεγάλη διαφορά τους με τους χρόνους εκτέλεσης των δοκιμών για των web server Apache 2.2. Η διαφορά αυτή οφείλεται γενικά ανώτερη απόδοση του Nginx ανεξαρτήτως υποδομής και όπως αναμέναμε οι χρόνοι του δεύτερου σεναρίου είναι υψηλότεροι.

Ο Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο των δοκιμών παρουσιάζεται παρακάτω:

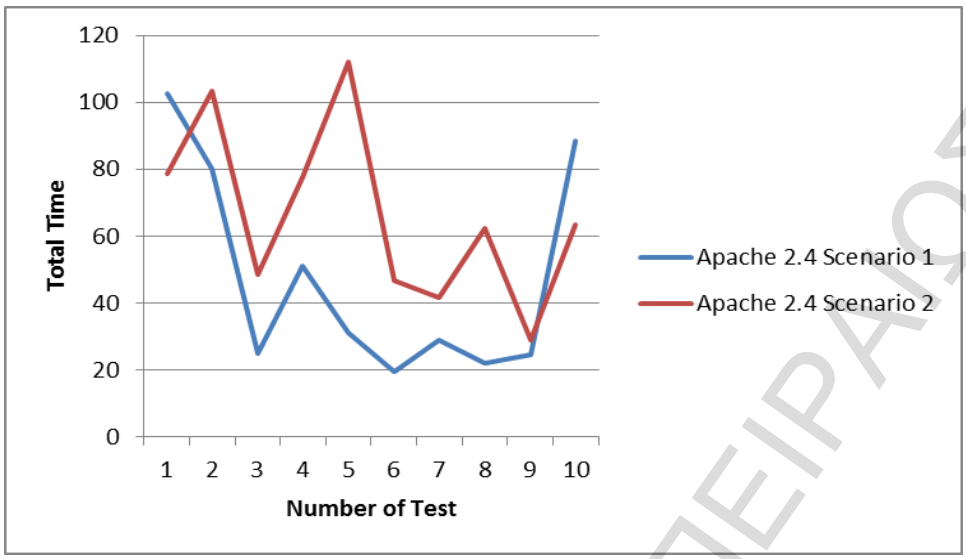


Γράφημα 20: Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο για Nginx

Και σε αυτό το γράφημα παρατηρούμε την ανωτερότητα του Nginx με τον ρυθμό μεταφορά KB να είναι αρκετά μεγάλος και στα δυο σενάρια. Γενικά βλέπουμε ότι το Nginx δεν επηρεάζετε σε μεγάλο βαθμό από την υπερφόρτωση της υποδομής cloud.

5.2.3 Apache2.4

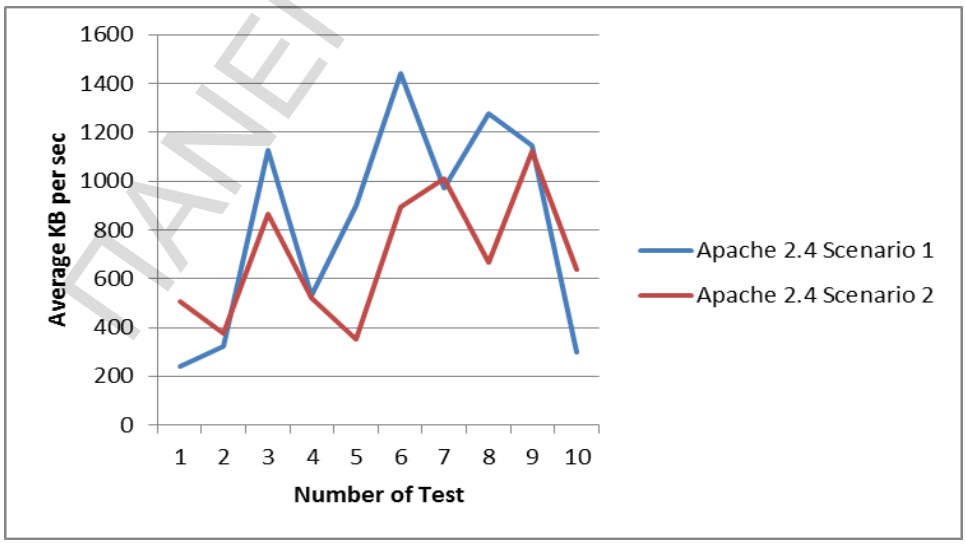
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 21: Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο για Apache 2.4

Στο γράφημα αυτό παρατηρούμε ότι το web server Apache 2.4, το οποίο είναι μια βελτιωμένη έκδοση του Apache 2.2, οι χρόνοι εκτέλεσης της δοκιμής επηρεάζονται σε αρκετά μεγάλο βαθμό από την υπερφόρτωση της υποδομής cloud όποτε ένας χρήστης που χρησιμοποίει Apache 2.4 πρέπει να είναι ιδιαίτερα προσεκτικός στο ποιος πάροχος έχει την καλύτερη απόδοση υπό συνθήκες μεγάλου φόρτου εργασιών.

Ο Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο των δοκιμών παρουσιάζεται παρακάτω:

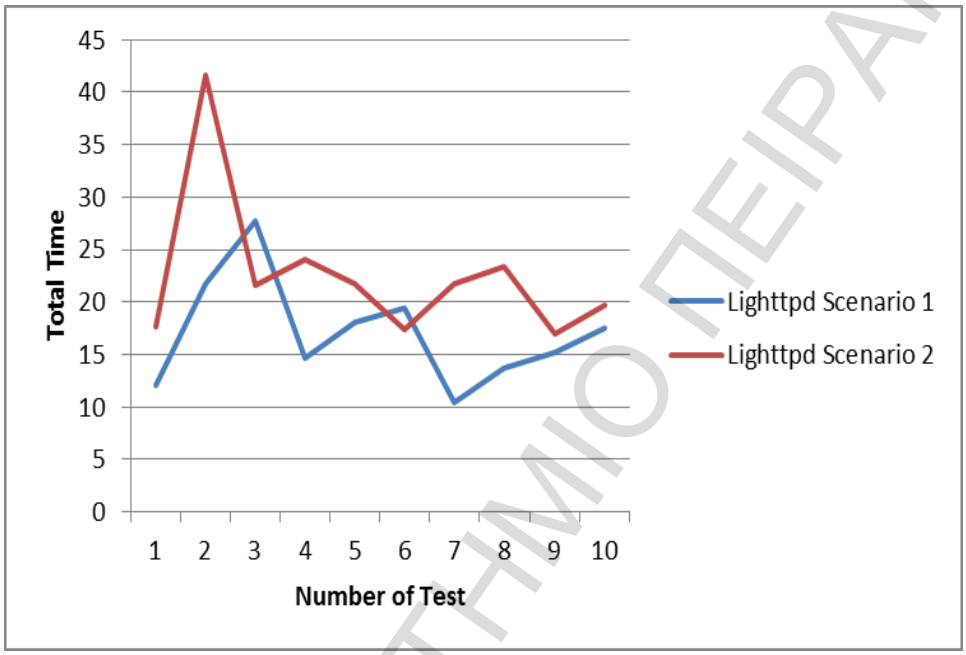


Γράφημα 22: Συνολικός χρόνος εκτέλεσης δοκιμής για Apache 2.4

Στο γράφημα του μέσου ρυθμού μεταφοράς παρατηρούμε το ίδιο με το γράφημα συνολικού χρόνου εκτέλεσης δοκιμών με τις διαφορές μεταξύ των δυο σεναρίων να μην είναι τόσο μεγάλες ωστόσο.

5.2.4 Lighttpd

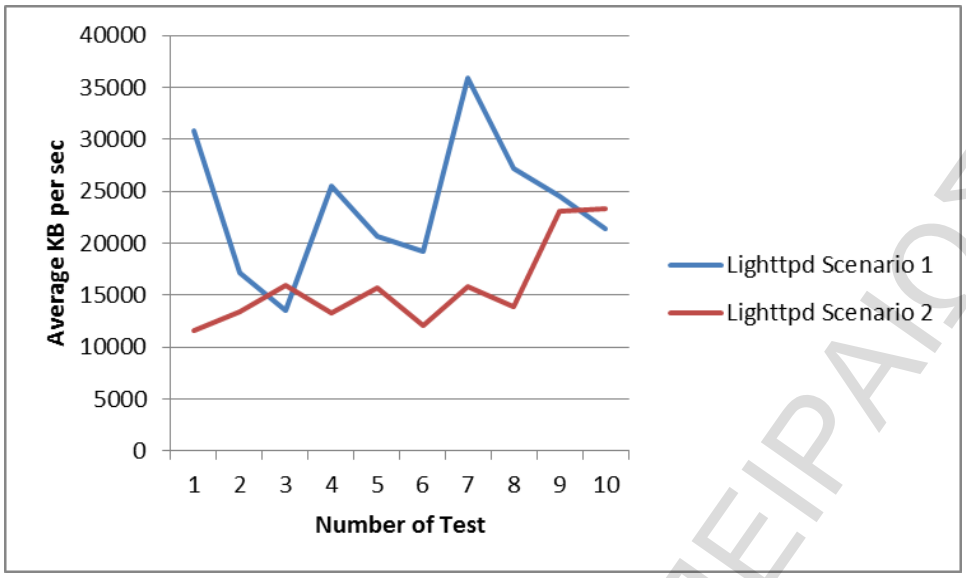
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 23: Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο για Lighthttpd

Στον web server Lighthttpd οι συνολικοί χρόνοι εκτέλεσης του τεστ ακλουθούνε το ίδιο μοτίβο όπως στα αποτελέσματα του Nginx αλλά με λίγο μεγαλύτερος χρόνους. Τα αποτελέσματα του δεύτερου σεναρίου είναι τα αναμενόμενα ,οντάς ποιο μεγάλοι χρόνοι σε σχέση με το πρώτο σενάριο.

Ο Μέσος ρυθμός μεταφοράς Kb ανά δευτερόλεπτο των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 24: Συνολικός χρόνος εκτέλεσης δοκιμής για Lighthttpd

Ο ρυθμός μεταφοράς KB όπως και στον Apache 2.4 βλέπουμε ότι επηρεάζετε αρκετά από την υπερφόρτωση της υποδομής cloud και αυτό είναι πολύ σημαντικό στοιχείο για έναν χρηστή αυτόν τον web server καθώς αν ο web server μεταφέρει πολλά δεδομένα θα δει την απόδοση του web server να μειώνετε αν η υποδομή του παρόχου που έχει επιλέξει έχει μεγάλο φόρτο εργασίας .

5.2.5 Συμπεράσματα για Βάσεις Δεδομένων

Όπως και στης βάσεις δεδομένων έτσι και στους web servers παρατηρούμαι τα ίδια αποτελέσματα με τις αποκλίσεις όμως στα δυο σενάρια να μην είναι τόσο μεγάλα όσο στις βάσεις δεδομένων. Αυτό οφείλετε κυρίως στον μέγεθος των υπολογιστικών πόρων που καταναλώνει μια web server εφαρμογή . Ένας web server που ουσιαστικά εξυπηρετεί σελίδες ισότοπου (web pages) den δεν χρειάζεται τόσους υπολογιστικούς πόρους όσο μια βάση δεδομένων που λειτούργει σε full capacity. Τα αποτελέσματα πολύ πιθανών να ήταν πολύ διαφορετικά εάν είχαμε χρησιμοποιήσει application servers ,οι όποιοι χρειάζονται

αρκετούς υπολογιστικούς πόρους. Ο πόρος που παίζει σημαντικό ρολό σε ένα web server είναι το δίκτυο και η απόδοση του[22]. Όταν η υποδομή cloud βρίσκεται στα όρια του το εσωτερικό δίκτυο της υποδομής χάνει φυσικά την απόδοση του αλλά όχι τόσο όσο η μνήμη και οι επεξεργαστές και αυτός είναι ο λόγος που δεν βλέπουμε τόσο μεγάλες αποκλίσεις στα δυο σενάρια.

5.3 Αποτελέσματα για Hadoop-MapReduce

Οι δοκιμές που εκτελεστήκαν για το περιβάλλον Hadoop-MapReduce είναι TestDFSIO , NNBench , MRBench και TeraGen-TeraSort. Τα κάθε είδος δοκιμής στο υπό εξέταση το περιβάλλον Hadoop-MapReduce εκτελέστηκε και στα δυο σενάρια με τις ίδιες ακριβώς παραμέτρους έτσι ώστε να δούμε πώς επηρεάζετε η απόδοση τους από το φόρτο εργασίας της υποδομής Cloud .

Τα αποτελέσματα που επιλέξαμε προς σύγκριση για δοκιμή TestDFSIO είναι ο συνολικός χρόνος εκτέλεσης της δοκιμής και το Throughput MB/sec από τις φασεις write και read. Η φάση delete δεν παράγει αποτελέσματα.

Η φάση write,read και delete και στα δυο σενάρια εκτελέστηκε με τις παρακάτω παραμέτρους :

Αριθμός αρχείων προ δημιουργία :3

Μέγεθος αρχείων : 1000 MB

Τα αποτελέσματα που επιλέξαμε προς σύγκριση για την δοκιμή TeraGen-TeraSort είναι ο συνολικός χρόνος εκτέλεσης της δοκιμής από τις teragen ,terasort και teravalidate.

Η δοκιμή εκτελέστηκε και στα δυο σενάρια εκτελέστηκε με την παρακάτω παράμετρο :

Μέγεθος αρχείων προς δημιουργία : 100000 bytes

Τα αποτελέσματα που επιλέξαμε προς σύγκριση για την δοκιμή NNBench ο μέσος χρόνος εκτέλεσης των διεργασιών MapReduce και ο ρυθμός συναλλαγών ανά δευτερόλεπτο(TPS) από τις φάσεις write, read και delete.

Η φάση write,read και delete και στα δυο σενάρια εκτελέστηκε με τις παρακάτω παραμέτρους :

maps : 1

reduces : 1

blockSize : 1

bytesToWrite : 10

numberOfFiles : 1000

bytesPerChecksum : 1

replicationFactorPerFile : 1

Τα αποτελέσματα που επιλέξαμε προς σύγκριση για την δοκιμή MRBench είναι ο μέσος χρόνος εκτέλεσης των διεργασιών MapReduce.

Η δοκιμή εκτελέστηκε και στα δυο σενάρια εκτελέστηκε με τις παρακάτω παραμέτρους :

maps: 10

reduces: 10

inputLines: 300000 lines

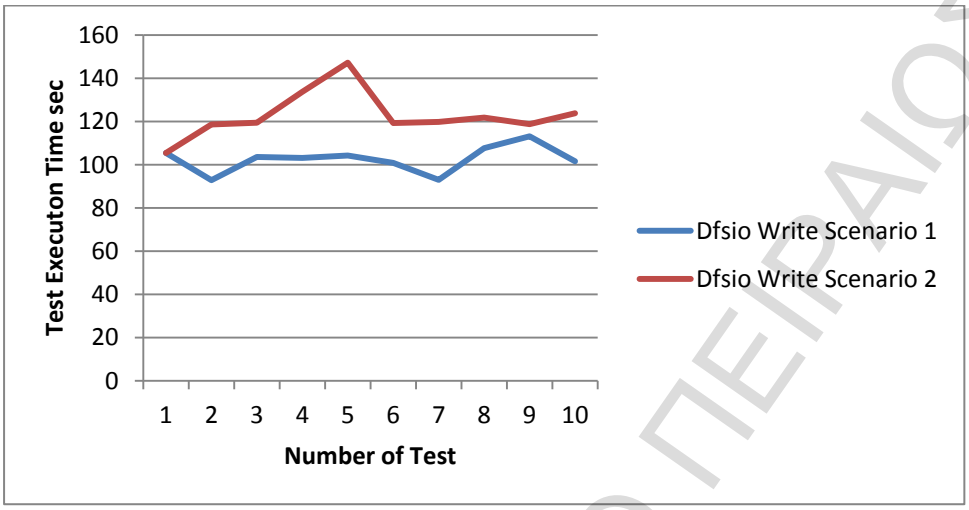
inputType: random

Η δοκιμή επαναλήφτηκε 10 φόρες για την κάθε βάση δεδομένων και στα δυο σενάρια ώστε να αναλύσουμε καλύτερα τα αποτελέσματα.

5.3.1 TestDFSIO

Φάση Write

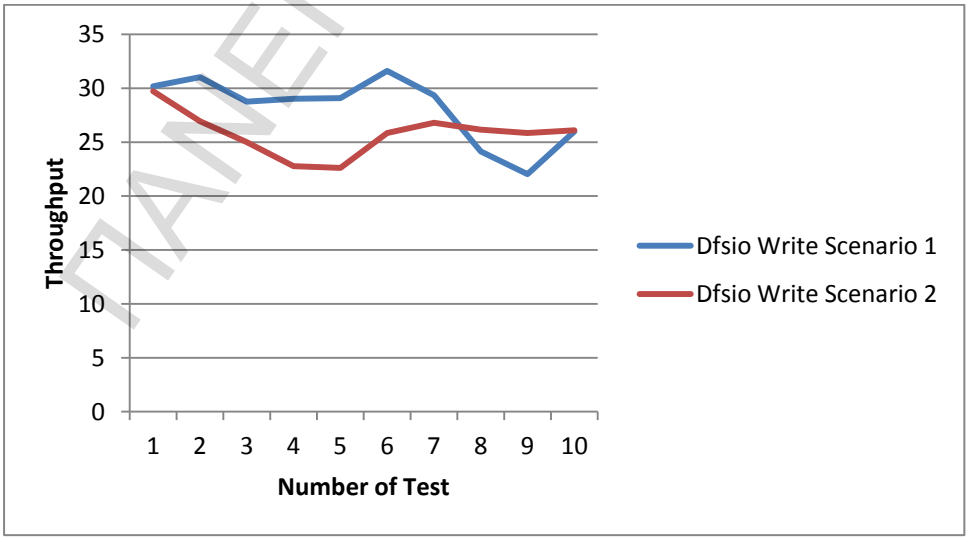
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 25: Συνολικός χρόνος εκτέλεσης δοκιμής για TestDFSIO στην φάση write

Όπως ήταν αναμενόμενο ο συνολικός χρόνος εκτέλεσης του δεύτερου σεναρίου είναι μεγαλύτερος από του πρώτου σεναρίου. Αυτό μας δείχνει ότι κατά την φάση write, οπού γράφονται και τα αρχεία στο HDFS, υπερφόρτωση της υποδομής επηρεάζει αρκετά την απόδοση μιας εφαρμογής που γραφεί πολλά δεδομένα στο HDFS.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:

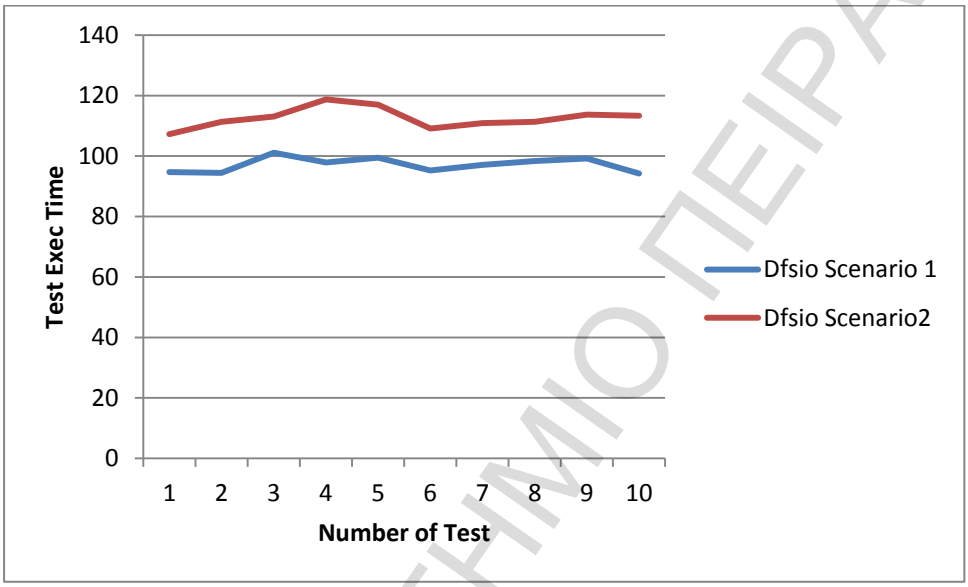


Γράφημα 26: Throughput για TestDFSIO στην φάση write

Ο ρυθμός εγράφης MB, σε αυτήν την φάση της δοκιμής, όπως είναι λογικό επηρεάζετε αρνητικά από την υπερφόρτωση της υποδομής οπού πόροι όπως μέσο αποθήκευσης και επεξεργαστική ισχύ χάνουν την απόδοση τους. Το ίδιο όμως παρατηρούμε και για το πρώτο σενάριο στις 3 τελευταίες επαναλήψεις που όμως πιθανόν να οφείλετε σε αστοχία της υποδομής καθώς στην υποδομή cloud στο σενάριο αυτό δεν υπάρχουν άλλες εικονικές μηχανές που να καταναλώνουν πόρους.

Φάση Read

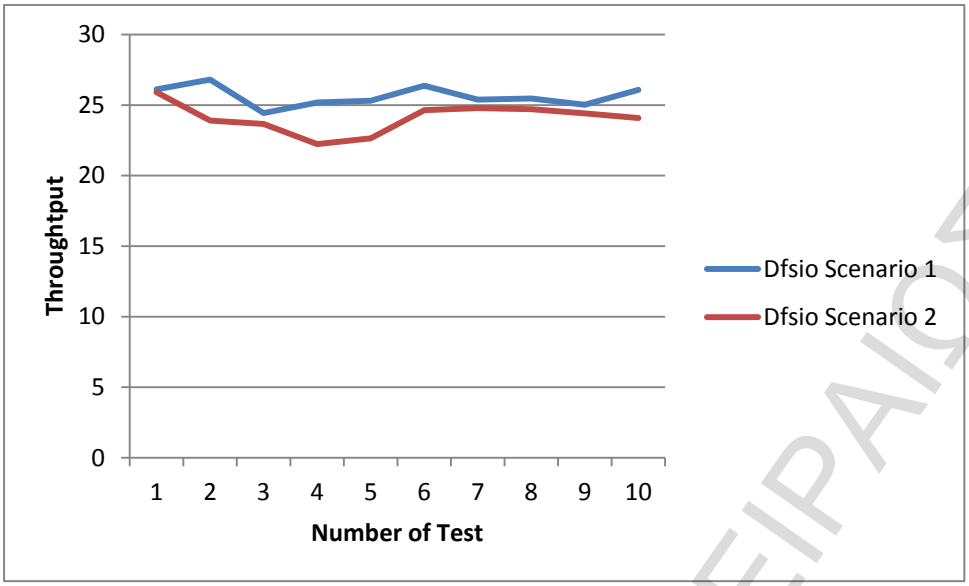
Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 27: Συνολικός χρόνος εκτέλεσης δοκιμής για TestDFSIO στην φάση read

Ο συνολικής χρόνος εκτέλεσης δοκιμής στην φάση read βλέπουμε ότι δεν επηρεάζετε τόσο πολύ από την υπερφόρτωση της υποδομής cloud καθώς στην φάση αυτή δεν γίνονται διεργασίες που απαιτούν πολλούς πόρους και για αυτό δεν βλέπουμε μεγάλες αποκλίσεις ανάμενα στα δυο σενάρια.

Το Throughput των δοκιμών παρουσιάζεται παρακάτω:



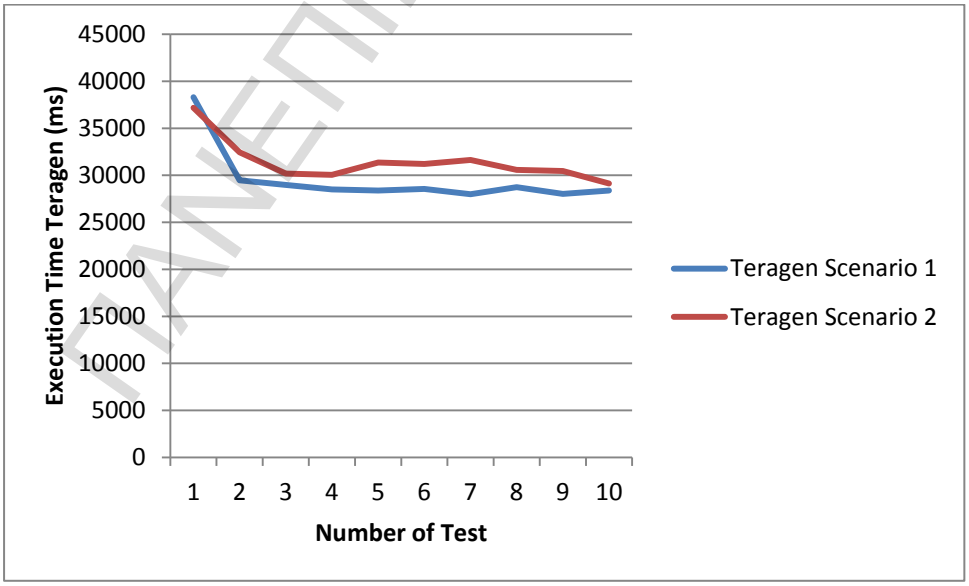
Γράφημα 28: Throughput για TestDFSIO στην φάση read

Όπως και το προηγούμενο γράφημα και σε αυτήν την δόκιμη βλέπουμε ότι το throughput στην φάση αυτή δεν επηρεάζετε πολύ από την υπερφόρτωση της υποδομής cloud.

5.3.2 TeraGen-TeraSort

Φάση teragen

Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:

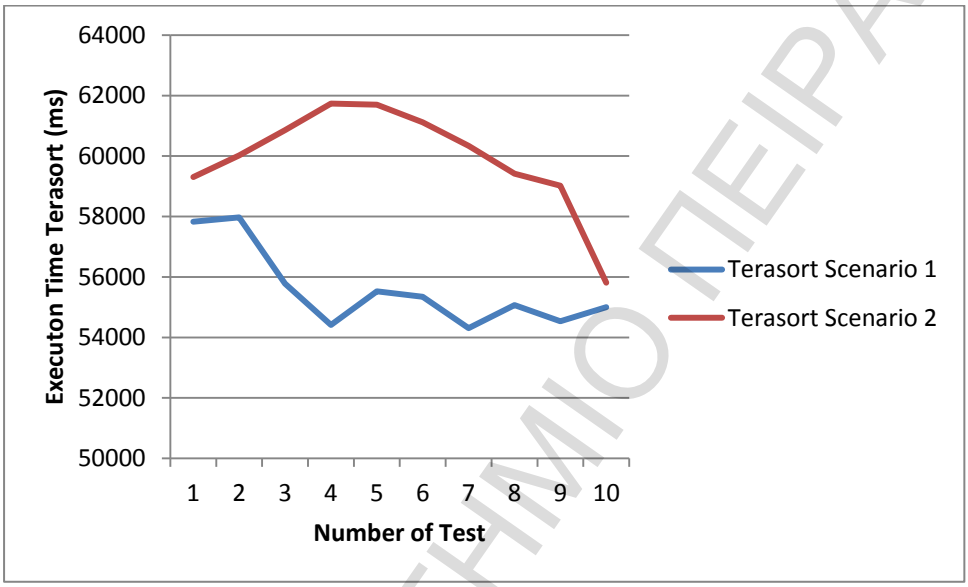


Γράφημα 29: Συνολικός χρόνος εκτέλεσης δοκιμής για Teragen

Από το παραπάνω γράφημα βλέπουμε ότι στην φάση teragen, που δημιουργούμε τα αρχεία στο HDFS, ο συνολικός χρόνος εκτέλεσης των δοκιμών στο δεύτερο σενάριο έχει μεγάλη διαφορά από τους αντίστοιχους χρόνους του προτού σεναρίου. Οι χρόνοι αναφέρονται x10000. Αυτό είναι λογικό από την στιγμή που η εγράφη δεδομένων στο HDFS απαιτεί αρκετούς υπολογιστικούς πόρους και στην περίπτωση του δεύτερου σεναρίου οι πόροι δεν αποδίδουν τόσο καλά.

Φάση terasort

Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:

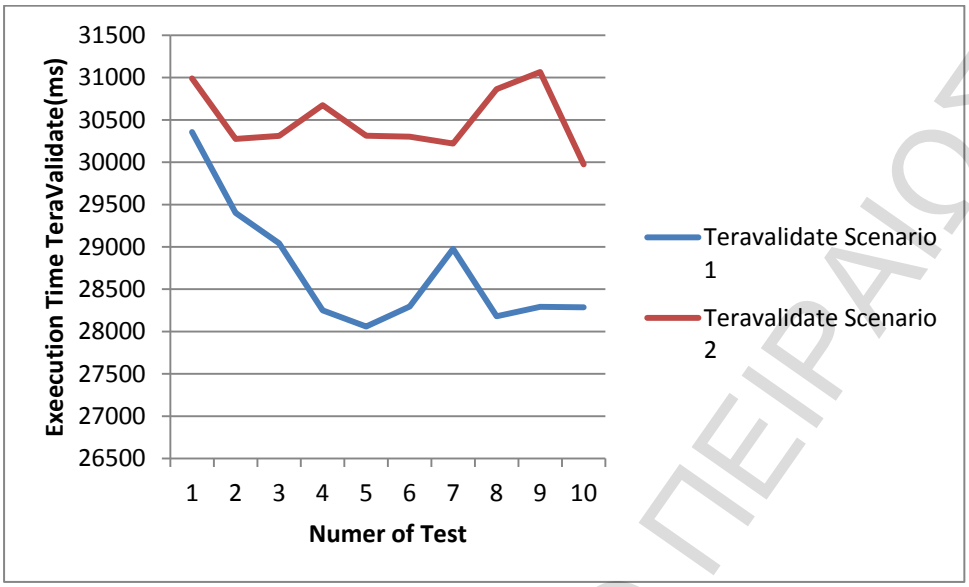


Γράφημα 30: Συνολικός χρόνος εκτέλεσης δοκιμής για Terasort

Όπως και στο προηγούμενο γράφημα έτσι και στην φάση terasort η υπερφόρτωση της υποδομής cloud αρνητικά το συνολικό χρόνο εκτέλεσης των δοκιμών. Στην φάση αυτή η διαφορά είναι ακόμα πιο έντονη γιατί έκτος από την ανάγνωση των αρχείων, γίνετε ταξινόμηση και αποθήκευση των αποτελεσμάτων της ταξινόμησης . Οι διεργασίες αυτές απαιτούν πολλούς υπολογιστικούς πόρους και όπως είναι λογικό στο δεύτερο σενάριο με τους πόρους την υποδομής όχι τόσο αποδοτικούς ο χρόνος εκτέλεσης των δοκιμών αυξάνεται.

Φάση teravalidate

Ο συνολικός χρόνος εκτέλεσης των δοκιμών παρουσιάζεται παρακάτω:



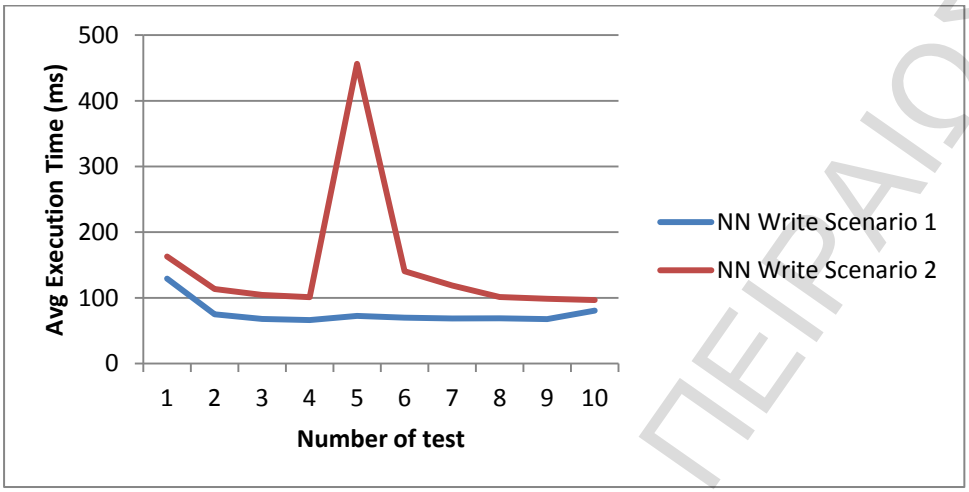
Γράφημα 31: Συνολικός χρόνος εκτέλεσης δοκιμής για Teravalidate

Επειδή στην φάση teravalidate εκτελούνται οι ίδιες διεργασίες εκτός από την ταξινόμηση (την αντικαθιστά διεργασία για επαλήθευση αποτελεσμάτων) οι χρόνοι εκτέλεσης της δοκιμής του δεύτερου σεναρίου είναι πιο μεγάλοι από του πρώτου σεναρίου. Γενικά εφαρμογές αυτού του τύπου είναι απαιτητικές σε υπολογιστικούς πόρους και είναι σημαντικό για έναν χρήστη να γνωρίζει πόσο αποδοτικός είναι ένας πάροχος σε συνθήκες μεγάλου φορτίου εργασιών.

5.3.3 NNBench

Φάση Write

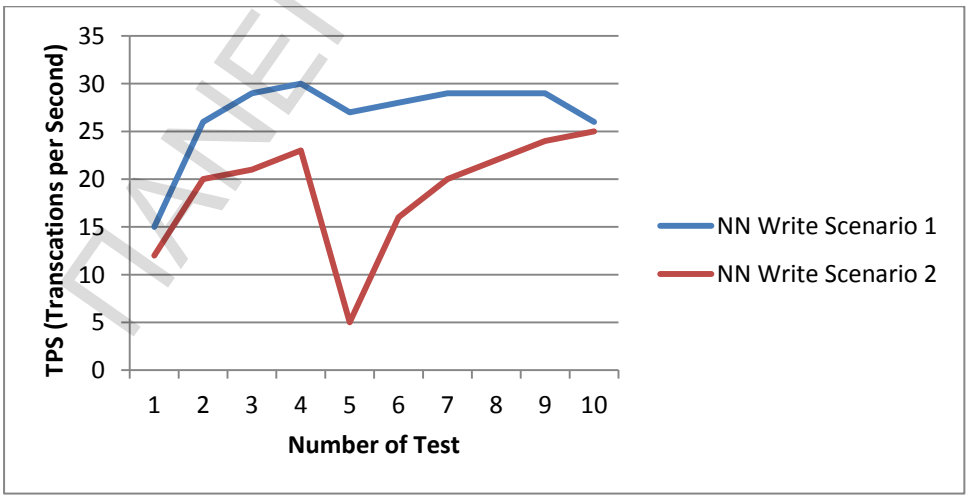
Ο μέσος χρόνος εκτέλεσης των διεργασιών των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 32: Μέσος χρόνος εκτέλεσης για NNBench στην φάση write

Από το γράφημα βλέπουμε ότι γενικά ο χρόνος εκτέλεσης δοκιμών στην φάση write επηρεάζετε αρκετά από την υπερφόρτωση της υποδομής (σενάριο 2). Να σημειώσουμε ότι στο NNBench εκτελούνται πολύ μικρές εργασίες όποτε μια διαφορά του χρόνου της τάξης των 90 ms είναι αρκετά μεγάλη διαφορά. Επίσης παρατηρούμε ότι η 5η επανάληψη έχει πολύ μεγάλη τιμή για τον χρόνο κα μάλλον οφείλεται σε αστοχία των πόρων της υποδομής cloud λόγο της υπερφόρτωσης των πόρων.

Το transaction per second των δοκιμών παρουσιάζεται παρακάτω:

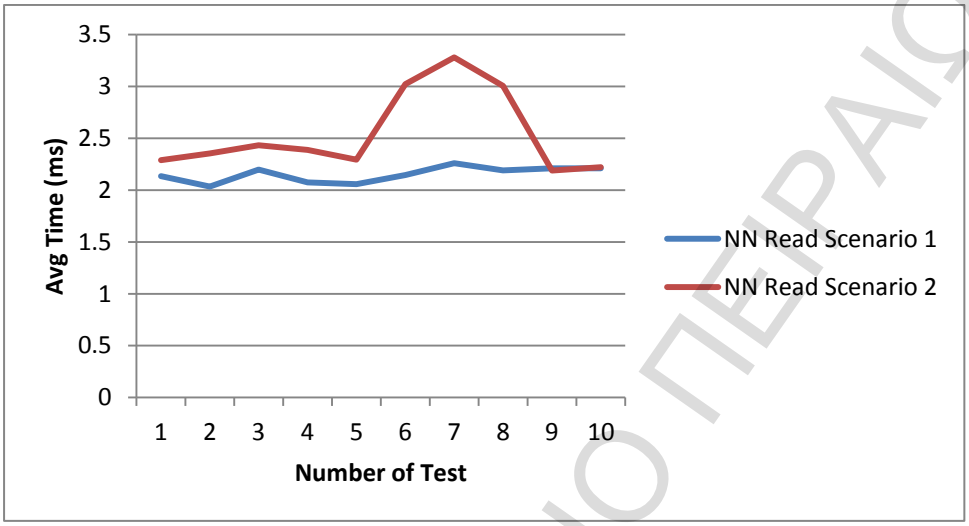


Γράφημα 33: TPS για NNBench στην φάση write

Το TPS(transaction per second) ακολουθεί το ίδιο μοτίβο με το προηγούμενο γράφημα και οι χρόνοι του δεύτερου σεναρίου επηρεάζονται σημαντικά από το μεγάλο φορτίο εργασιών που έχει η υποδομή cloud.

Φάση Read

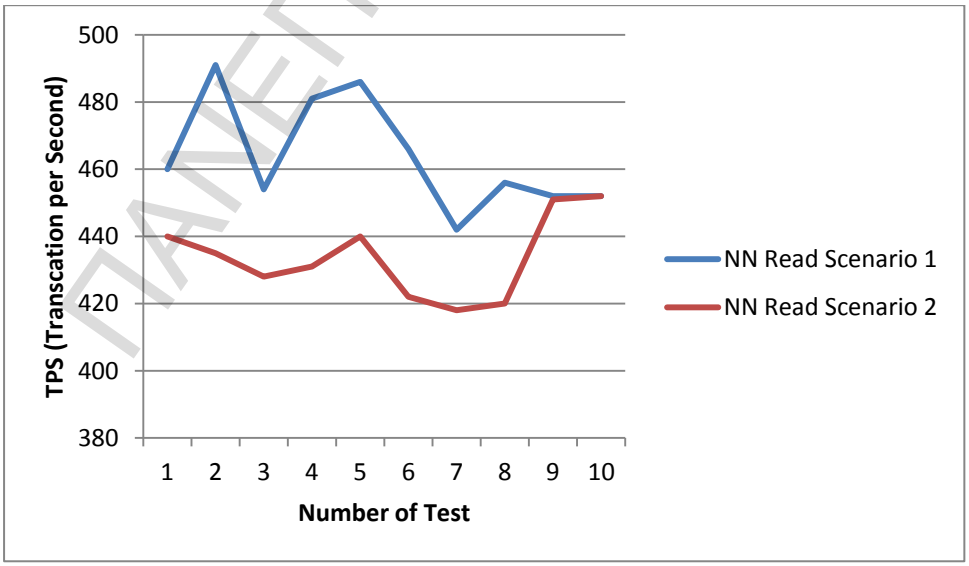
Ο μέσος χρόνος εκτέλεσης των διεργασιών των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 34: Μέσος χρόνος εκτέλεσης για NNBench στην φάση read

Η φάση του read δεν απαιτεί πολλούς υπολογιστικούς πόρους και είναι αυτός ο λόγος που οι χρόνοι και στα δυο σενάρια είναι αρκετά μικροί αλλά όπως αναμέναμε οι χρόνοι του δεύτερου σεναρίου είναι μεγαλύτεροι στο δεύτερο σενάριο

Το transaction per second των δοκιμών παρουσιάζεται παρακάτω:

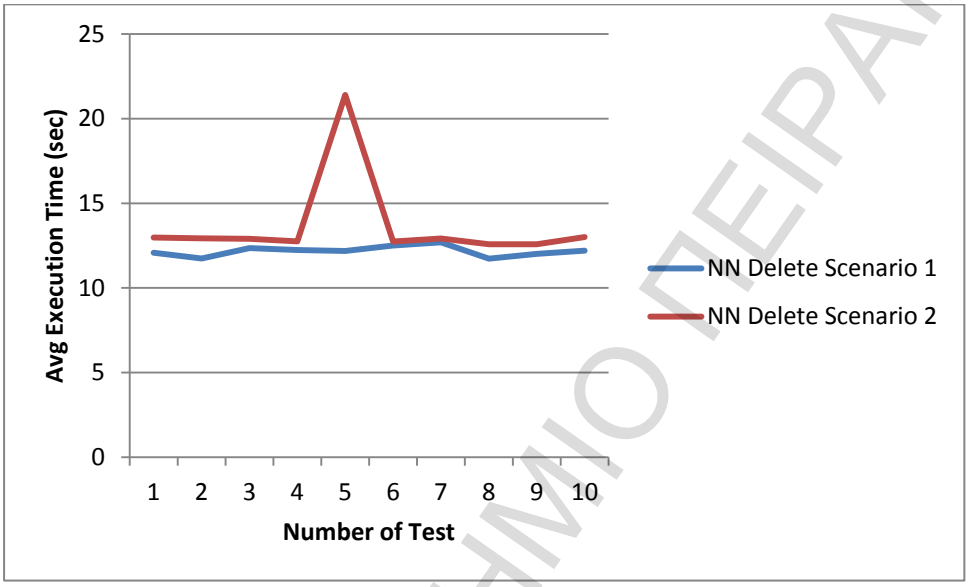


Γράφημα 35: TPS για NNBench στην φάση read

Επίσης και σε αυτό το γράφημα παρατηρούμε στην φάση του read το δεύτερο σενάριο έχει χειρότεροι απόδοση από το σε σχεδόν όλες της δόκιμες. Η άνοδος του TPS στη 8η,9η,10η επανάληψη πιθανόν να οφείλεται στο γεγονός ότι οι υπόλοιπες 5 εικονικές μηχανές δεν απασχολούν όλους τους πόρους που προορίζονται για αυτές και αυτό δίνει την δυνατότητα στην δική μας εικονική μηχανή να αυξήσει την δική της απόδοση.

Φάση Delete

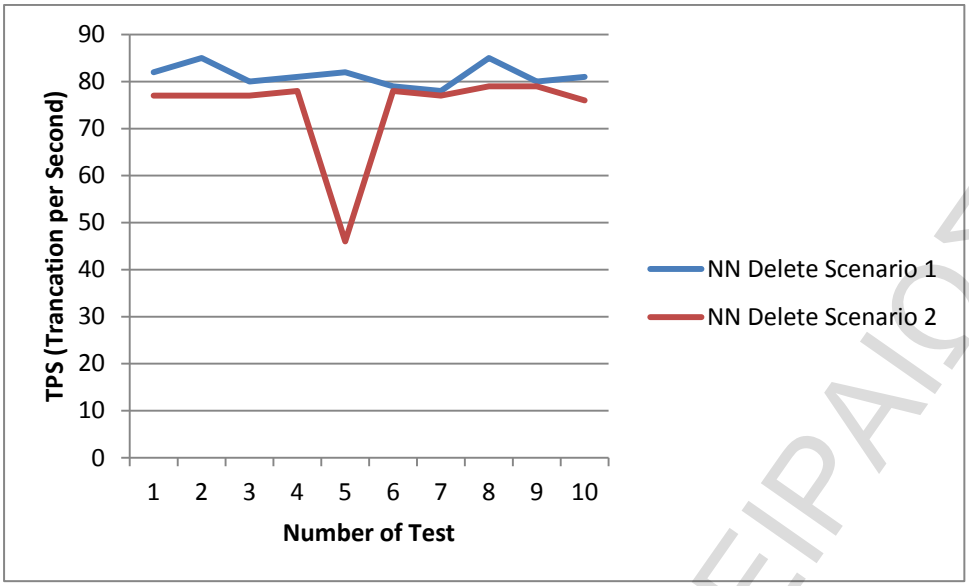
Ο μέσος χρόνος εκτέλεσης των διεργασιών των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 36: Μέσος χρόνος εκτέλεσης για NNBench στην φάση delete

Παρατηρούμε στο παραπάνω γράφημα ότι στην φάση delete ο μέσος χρόνος εκτέλεσης των διεργασιών στην δόκιμη δεν επηρεάζετε τόσο από την υπερφόρτωση της υποδομής εκτός από την 5^{η} επανάληψη. Αυτό μας οδηγεί στο συμπέρασμα ότι στην πέμπτη δόκιμη πιθανόν να υπήρχε κάποια αστοχία στην υποδομή και όχι ότι η χρόνος εκτέλεσης επηρεάστηκε από την υπερφόρτωση της υποδομής .

Το transaction per second των δοκιμών παρουσιάζεται παρακάτω:

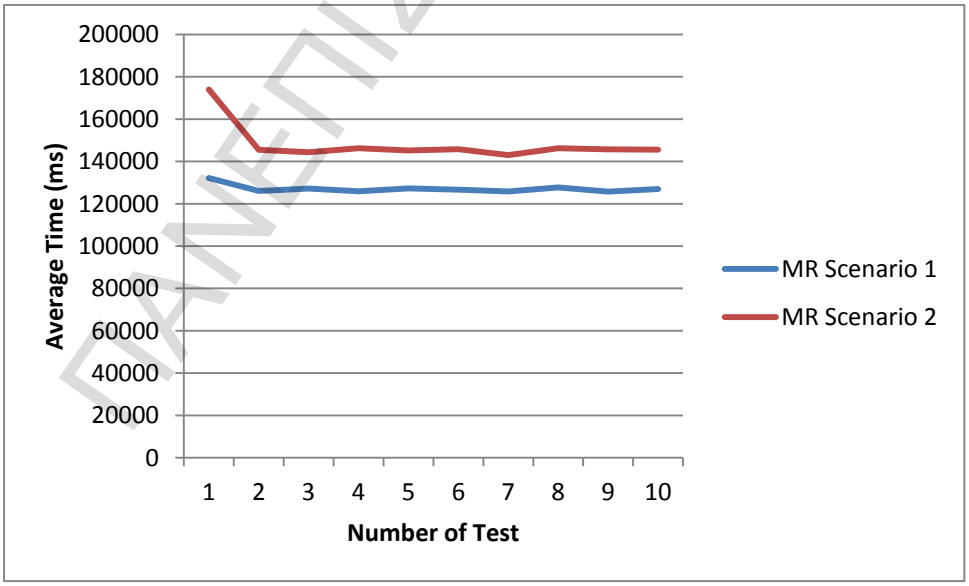


Γράφημα 37: TPS για NNBench στην φάση delete

Όπως και στο προηγούμενο γράφημα το ίδιο παρατηρούμε και για το TPS. Γενικότερα η φάση delete δεν απαιτεί πολλούς πόρους και για αυτό βλέπουμε ότι το δεύτερο σενάριο δεν επηρεάζετε τόσο πολύ από το μεγάλο φόρτο εργασίας της υποδομής.

5.3.4 MRBench

Ο μέσος χρόνος εκτέλεσης των διεργασιών των δοκιμών παρουσιάζεται παρακάτω:



Γράφημα 38: Μέσος χρόνος εκτέλεσης για MRBench

Βλέπουμε από το παραπάνω γράφημα ότι ο μέσος χρόνος εκτέλεσης διεργασιών σε ένα MRBench επηρεάζετε σε μεγάλο βαθμό από το πόσο εργασίας έχει η υποδομή cloud καθώς όλοι οι χρόνοι του δεύτερου σεναρίου είναι πολύ μεγαλύτερη σε σχέση με το πρώτο σενάριο. Όπως και στις προηγούμενες εφαρμογές του Hadoop έτσι και εδώ είναι σημαντικό ένας χρήστη αν έχει εφαρμογή τέτοιου τύπου να γνωρίζει πώς αποδίδει η υποδομή ενός παρόχου σε συνθήκες μεγάλου φορτίου εργασιών αλλιώς η εφαρμογή του δεν θα είναι αποδοτική.

5.3.5 Συμπεράσματα για Hadoop-MapReduce

Τα αποτελέσματα όπως ήταν αναμενόμενο στις δοκιμές για εφαρμογές τύπου MapReduce επηρεάζονται σε μεγάλο βαθμό από το φόρτο εργασίας που έχει η υποδομή cloud. Όλοι οι χρόνοι εκτέλεσης των δοκιμών , μέσοι και συνολικοί , στο δεύτερο σενάριο έχουν αρκετά μεγάλη απόκλιση σε σχέση με το πρώτο σενάριο. Το ίδιο συμβαίνει με τον ρυθμό συναλλαγών και το throughput. Γενικά ένα περιβάλλον Hadoop και οι εφαρμογές που τρέχουν σε αυτό είναι πολύ απαιτητικές σε υπολογιστικούς πόρους (επεξεργαστική ισχύ, μνήμη, δίκτυο και χώρος αποθήκευσης) [41]σε σχέση με τα αλλά δυο είδη εφαρμογών που παρουσιάσαμε (Βάσεις Δεδομένων και Web Servers) όποτε είναι πολύ σημαντικό για έναν χρήστη να γνωρίζει πότε και ποιος παρόχος αποδίδει καλύτερα για αυτού του είδους των εφαρμογών. Όπως είναι φυσικό η απόδοση αυτόν των εφαρμογών πέφτει όταν το δικό μας private cloud βρίσκεται στα όρια του.

6 Μελλοντική Μελέτη

Για το web tool front-end θα μπορούσε να χρησιμοποιηθεί τεχνολογία JQuery και Ajax για πιο γρήγορη φόρτωση των αποτελεσμάτων. Θα μπορούσε επίσης να δημιουργηθεί ένα add-on etool το οποίο θα ήταν υπεύθυνο για την εξαγωγή των αποτελεσμάτων σε ένα αρχείο με τη μορφή pdf ή xls. Στη σελίδα των αποτελεσμάτων είναι πιθανή , η ύπαρξη πιο εξειδικευμένων συναρτήσεων φόρτωσης αποτελεσμάτων για να μπορεί ο χρήστης να συγκρίνει ταυτόχρονα πολλαπλά αποτελέσματα. Στα αποτελέσματα που επιστρέφονται αμέσως μετά την εκτέλεση της δοκιμής, θα ήταν δυνατόν να χρησιμοποιηθούν πιο εξειδικευμένα γραφήματα που να παρουσιάζουν περισσότερα αποτελέσματα και πιο συγκεκριμένα.

Σαν βάση δεδομένων για την αποθήκευση των αποτελεσμάτων θα μπορούσε να χρησιμοποιηθεί , μη σχεσιακή βάση δεδομένων , έτσι ώστε τα αρχεία αποτελεσμάτων ή αποτελέσματα να αποθηκεύονται πιο γρήγορα και αποδοτικά.

Θα μπορούσε το εργαλείο μας , από την μεριά του back-end να υποστηρίζει περισσότερα είδη εφαρμογών , όπως Scientific, Graphic, Data Streaming, Media Streaming, Software testing και Web Search καθώς και να επιστρέφει αποτελέσματα των δοκιμών σε XML μορφή ή JSON μορφή..

Επίσης, το εργαλείο θα μπορούσε να υποστηρίζει εκτέλεση των δοκιμών σε περισσότερους παρόχους δημόσιου cloud.

Τέλος, το εργαλείο μας θα μπορούσε να παρέχει καλύτερη ανατροφοδότηση σχετικά με τα λάθη που συμβαίνουν σε επίπεδο περιβάλλοντος υπολογιστικού νέφους και ενδιαμέσου εργαλείου Jclouds ώστε ο χρήστης να είναι πιο ενήμερος σχετικά με την κατάσταση και τα σφάλματα της υποδομής όταν δημιουργεί πόρους στο υπολογιστικό νέφος.

Βιβλιογραφικές Αναφορές

[1] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, Special Publication 800-145, National Institute of Standards and Technology, September 2011

[2] http://conta.uom.gr/conta/ekpaideysh/metaptyxiaka/technologies_diktywn/ergasies/2013/A%20review%20in%20Grid%20and%20Cloud%20Computing.pdf

[3] (Mell & Grance, 2011; Velte κ.ά., 2010; Θανάσου Παρασκευή, 2011)

[4] I. Foster, C. Kesselman, S. Tuecke, “*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*” International Journal Supercomputer Applications, Vol. 15, No. 3, 2001.

[5] http://apprenda.com/library/glossary/definition-cloud-instance-single-multi/

[6] Scientific Cloud Computing: Early Definition and Experience Lizhe Wang and Gregor von Laszewski October 26, 2008

[7] Fair Benchmarking for CloudComputing Systems, Lee Gillam March 2012

[8] https://cloudsleuth.net/

[9] http://en.wikipedia.org/wiki/First-come,_first-served

[10] The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis (Shengsheng Huang)

[11] Hauck, M., Huber, M., Klems, M., Kounev, S., Muller-Quade, J., Pretschner, A., Reussner, R., and Tai, S. Challenges and opportunities of Cloud computing. Karlsruhe Reports in Informatics 19, Karlsruhe Institute of Technology - Faculty of Informatics, 2010.

[12] George Kousiouris, Dimosthenis Kyriazis, Andreas Menychtas and Theodora Varvarigou, "Legacy Applications on the Cloud: Challenges and enablers focusing on application performance analysis and providers characteristics", in Proceedings of the 2012 2nd IEEE International Conference on Cloud Computing and Intelligence Systems (IEEE CCIS 2012), Oct. 30th ~ Nov. 1st,Hangzhou, China, 2012.

[13] Zhonghong Ou, Hao Zhuang, Jukka K. Nurminen, Antti Ylä-Jääski, and Pan Hui. 2012. Exploiting hardware heterogeneity within the same instance type of Amazon EC2. In

Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing (HotCloud'12). USENIX Association, Berkeley, CA, USA, 4-4,2012.

[14] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An analysis of performance interference effects in virtual environments. In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 200–209, April 2007.

[15] Lee Gillam, Bin Li, John O'Loughlin,Anuz Pranap,Singh Tomar,Fair Benchmarking for Cloud Computing Systems,2012

[16] http://openbenchmarking.org/index.php?c=result&i=1202066-AR-IBOJILOV651

[17] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. CloudCmp: comparing public Cloud providers. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10). ACM, New York, NY, USA, 1-14. DOI=10.1145/1879141.1879143 http://doi.acm.org/10.1145/1879141.1879143

[18] http://labs.yahoo.com/news/yahoo-cloud-serving-benchmark/

[19] http://redmine.lighttpd.net/projects/weighttp/wiki

[20] http://blogs.msdn.com/b/seliot/archive/2010/03/04/what-the-heck-is-cloud-computing-another-re-look-with-pretty-pictures.aspx

[21] http://blogs.msdn.com/b/seliot/archive/2010/03/04/what-the-heck-is-cloud-computing-another-re-look-with-pretty-pictures.aspx

[22] http://httpd.apache.org/

[23] https://hadoop.apache.org/docs/current2/hadoop-project-dist/hadoop-common/SingleNodeSetup.html

[24] https://hadoop.apache.org/docs/current2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

[25] http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/

[26] TeraByte Sort on Apache Hadoop,Owen O'Malley,May 2008

[27] http://wiki.apache.org/hadoop/NameNode

[28] http://jclouds.apache.org/start/what-is-jclouds/

[29] http://www.ubuntu.com/cloud/private-cloud/openstack

[30] http://www.rackspace.com/cloud/openstack/

[31] http://www.openstack.org/community/

[32] https://access.redhat.com/site/documentation/en-US/Red_Hat_OpenStack/3/html-single/Installation_and_Configuration_Guide/images/2440.png

[33] http://docs.openstack.org/developer/swift/

[34] https://wiki.openstack.org/wiki/Cinder

[35] http://docs.openstack.org/developer/glance/

[36] http://www.openstack.org/software/openstack-shared-services/

[37] http://www.openstack.org/software/openstack-networking/

[38] http://docs.openstack.org/developer/ceilometer/

[39] http://docs.openstack.org/developer/horizon/

[40] http://highscalability.com/blog/2013/11/25/how-to-make-an-infinitely-scalable-relational-database-manag.html

[41] http://hadoop.apache.org/