ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

Ειδίκευση Δικτύων Επικοινωνιών και Κατανεμημένων Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μετρήσεις απόδοσης και διαμόρφωση δικτύου νεφοϋπολογιστικής υποδομής με περιέκτες

Μηνάς Θεοδωρακάκος

Α.Μ. 18039

Εισηγητής: Δημήτριος Καλλέργης, Λέκτορας Εφ.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

Ειδίκευση Δικτύων Επικοινωνιών και Κατανεμημένων Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μετρήσεις απόδοσης και διαμόρφωση δικτύου νεφοϋπολογιστικής υποδομής με περιέκτες

Μηνάς Θεοδωρακάκος

Α.Μ. 18039

Τριμελής εξεταστική επιτροπή

1. **Επιβλέπων καθηγητής: Δημήτριος Καλλέργης,** Λέκτορας Εφ. Πανεπιστημίου Δυτικής Αττικής

   Digitally signed by Dimitrios Kallergis
   Date: 2023.07.11 00:54:36 +03'00'

2. **Μέλος: Βασίλειος Μάμαλης,** Καθηγητής Πανεπιστημίου Δυτικής Αττικής

   Vasileios Mamalis
   Digitally signed by Vasileios Mamalis
   Date: 2023.07.12 01:48:13 +03'00'

3. **Μέλος: Παναγιώτης Καρκαζής,** Αναπλ. Καθηγητής Πανεπιστημίου Δυτικής Αττικής

   Panagiotis Karkazis
   Digitally signed by Panagiotis Karkazis
   Date: 2023.07.11 10:53:47 +03'00'

Αθήνα, Μάρτιος 2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος, Θεοδωρακάκος Μηνάς του Θεοδώρου, με αριθμό μητρώου 18039, φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμης και Τεχνολογίας της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Digitally signed by Dimitrios Kallergis
Date: 2023.07.11 00:55:00 +03'00'

Περίληψη

Η ανάπτυξη και επέκταση της υπολογιστικής νέφους έχουν επιταχυνθεί τα τελευταία χρόνια λόγω της μεγάλης αποδοχής του από όλο και περισσότερους χρήστες. Η εξέλιξη αυτή απαιτεί συνεχείς αλλαγές στις υποδομές του για την όσο το δυνατό μεγαλύτερη ικανοποίηση αυτών των χρηστών όσον αφορά στην εμπειρία τους σε αυτές. Επιπλέον, οι μελέτες αυτών των υποδομών που αποσκοπούν στη σύγκριση τους ως προς την αξιοποίηση των διαθέσιμων υπολογιστικών πόρων έχουν αποκτήσει μεγάλο ενδιαφέρον. Ο σκοπός της συγκεκριμένης εργασίας είναι η αξιολόγηση της απόδοσης δυο δικτυακών τεχνολογιών μιας υποδομής του νέφους η οποία βασίζεται σε περιέκτες που εκτελούν μικροϋπηρεσίες. Για αυτό το λόγο πραγματοποιήθηκαν κάποιες μετρήσεις που είχαν ως βασικό στόχο τη βελτιστοποίηση της δικτυακής απόδοσης της υποδομής σε συνθήκες υψηλών απαιτήσεων μέσα από την σύγκριση αυτών των δυο τεχνολογιών. Οι έλεγχοι που έγιναν ανέδειξαν τις διαφορές τους καθώς και την καλύτερη επιλογή ανάμεσα τους για την αποδοτικότερη δικτυακή λειτουργία της υποδομής.

Λέξεις-Κλειδιά: Περιέκτες, Δικτυακές τεχνολογίες, Υπολογιστική Νέφους, Βελτίωση Απόδοσης.

Abstract

In recent years, the development and expansion of cloud computing has accelerated owing to its wide acceptance by an increasing number of users. Infrastructures have thus had to constantly develop to improve user experience and satisfaction. Comparison studies of these infrastructures in terms of their utilization of available computing resources are therefore of great importance. The purpose of this work is to evaluate the performance of two cloud infrastructure network technologies based on containers running microservices. For this reason, a series of measurements were carried out with the main objective of optimizing the network performance of the infrastructure in conditions of high demand to allow the comparison of these two technologies. The testing made highlighted their differences as well as allowing the assessment of the best network performance.

Keywords: Containers, Network Technologies, Cloud, Network Performance Assessment.

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου Καλλέργη Δημήτριο, για την αμέριστη βοήθεια, τη συνεχή καθοδήγησή του και την εμπιστοσύνη που μου έδειξε.

Ακόμα, θα ήθελα να ευχαριστήσω την οικογένειά μου, για την υπομονή τους, καθώς και την υποστήριξη την οποία έλαβα όλα αυτά τα χρόνια παρά τις δυσκολίες ειδικά του διαστήματος της πανδημίας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

1. ΕΙΣΑΓΩΓΗ

1.1 ΠΡΟΛΟΓΟΣ

Η συγκεκριμένη εργασία μπορεί να ενταχθεί στο πεδίο των αρχιτεκτονικών δικτυακών υποδομών για υπηρεσίες που βασίζονται στην Υπολογιστική Νέφους (Cloud Computing). Η ανάπτυξη αυτού του κλάδου της Πληροφορικής οφείλεται σε μεγάλο βαθμό στην μείωση του κόστους για τον χρήστη καθώς και στη δημιουργία υπηρεσιών που χρησιμοποιούνται από πολλά εκατομμύρια χρήστες σε όλο τον κόσμο οι οποίες δε θα μπορούσαν να είναι λειτουργικές σε τόσο μεγάλο εύρος με τις υπάρχουσες τεχνολογίες. Ένα από τα βασικά της χαρακτηριστικά είναι η διαχείριση πόρων ανάλογα με τη ζήτηση έτσι ώστε κάθε στιγμή να αξιοποιούνται αυτοί που είναι διαθέσιμοι με τον καλύτερο δυνατό τρόπο.

Η εικονικοποίηση η οποία αποτελεί την βασικότερη τεχνική στην Υπολογιστική Νέφους μπορεί να αποκρύπτει όλη τη δομή και το υλικό ενός υπολογιστικού συστήματος, ενώ εμφανίζει στο χρήστη μόνο εικονικούς πόρους που μπορεί να αξιοποιήσει σύμφωνα με τις ανάγκες του. Έτσι, ανάλογα με τον σκοπό που προορίζεται το κάθε σύστημα ή μια εφαρμογή, δεσμεύονται οι κατάλληλοι πόροι οι οποίοι στη συνέχεια γίνονται διαθέσιμοι προς χρήση. Η αποδοτικότερη διαχείριση αυτών των εικονικών πόρων είναι μια από τις πιο σημαντικές προκλήσεις στην Υπολογιστική Νέφους.

Η εικονικοποίηση που βασίζεται σε περιέκτες σε συνδυασμό με τις μικροϋπηρεσίες που φιλοξενούνται σε αυτούς φαίνεται να έχουν αποκτήσει μεγάλη απήχηση τα τελευταία χρόνια στην υπολογιστική νέφους. Η δικτυακή υποδομή των περιεκτών αποτελεί σημαντική παράμετρο στην υλοποίηση τους και έχουν αναπτυχθεί διάφορες δικτυακές τεχνολογίες που εξυπηρετούν τις ποικίλες ανάγκες των υπηρεσιών που προσφέρονται από το νέφος. Η αξιολόγηση αυτών των τεχνολογιών κρίνεται αναγκαία όσον αφορά στη σχέση ανάμεσα στην απόδοση των υπηρεσιών και των διαθέσιμων δικτυακών υποδομών έτσι ώστε να αξιοποιούνται πλήρως, ενώ παράλληλα να ικανοποιούνται οι απαιτήσεις των χρηστών για απρόσκοπτη λειτουργία.

Η παρούσα εργασία ασχολείται με την αξιολόγηση δυο δικτυακών τεχνολογιών μιας δικτυακής υποδομής του νέφους η οποία βασίζεται σε περιέκτες που εκτελούν μικροϋπηρεσίες. Οι έλεγχοι που πραγματοποιήθηκαν είχαν ως βασικό στόχο τη σύγκριση αυτών των δυο τεχνολογιών, ώστε να εξαχθούν ασφαλή συμπεράσματα ως προς την δικτυακή τους απόδοση σε συνθήκες υψηλών απαιτήσεων. Οι μετρήσεις που έγιναν έδειξαν τα πλεονεκτήματα και τα

μειονεκτήματα αυτών των τεχνολογιών καθώς και το προβάδισμα ανάμεσα τους ως προς τη βελτιστοποίηση της απόδοσης.

1.2 ΣΚΟΠΟΣ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΟ ΜΕΛΕΤΗΣ

Ο σκοπός της μεταπτυχιακής εργασίας είναι η μέτρηση της απόδοσης και η σχετική διαμόρφωση της δικτυακής υποδομής μιας υπηρεσίας του νέφους που βασίζεται σε περιέκτες υπό συνθήκες υψηλής ζήτησης για βελτιστοποίηση της απόδοσης της. Η δικτύωση αυτής της υπηρεσίας υλοποιήθηκε με τη βοήθεια του εργαλείου Docker Compose της πλατφόρμας Docker πάνω σε ελεύθερη υποδομή του νέφους, ενώ για την προσομοίωση της επιθυμητής δικτυακής κίνησης χρησιμοποιήθηκε μια γεννήτρια φορτίου. Το αντικείμενο της μελέτης για την παρούσα εργασία είναι η αποδοτικότερη διαχείριση των δικτυακών πόρων της εν λόγω υποδομής σε πελάτες που συμμετέχουν στην παρεχόμενη υπηρεσία.

1.3 ΔΟΜΗ

Η δομή της παρούσας εργασίας αναλύεται παρακάτω:

- **Κεφάλαιο 1:** Περιλαμβάνει την εισαγωγή με σύντομες αναφορές στο θεωρητικό υπόβαθρο, το σκοπό και το αντικείμενο της παρούσας διπλωματικής εργασίας καθώς και τη δομή της.
- **Κεφάλαιο 2:** Αναλύει την υπάρχουσα κατάσταση για το πρόβλημα της χωρητικότητας των νεφοϋπολογιστικών υποδομών σε σχέση με την απόδοση τους σε συνθήκες υψηλής ζήτησης. Ακόμα, παρουσιάζει τις νέες τάσεις που εμφανίζονται στην εικονικοποίηση που μπορεί να υλοποιηθεί και να αξιοποιηθεί στο υπολογιστικό νέφος. Τέλος, εξηγεί την αναγκαιότητα των ελέγχων που πρέπει να γίνονται στις υποδομές του νέφους και αναφέρει τις τρέχουσες εξελίξεις τους.
- **Κεφάλαιο 3:** Περιγράφει το πειραματικό μέρος της εργασίας, τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν και τους λόγους που αυτά επιλέχθηκαν. Παρουσιάζει αναλυτικά την πειραματική διάταξη δίνοντας έμφαση στη δικτύωση της με τις δυο προκαθορισμένες μεθόδους που υλοποιήθηκαν και μετρήθηκαν.
- **Κεφάλαιο 4:** Παρουσιάζει τα αποτελέσματα των μετρήσεων που έγιναν για τους δυο τρόπους δικτύωσης των περιεκτών μιας δοκιμαστικής εφαρμογής. Αναφέρει τους περιορισμούς που υφίστανται με την επιλογή του καθενός από αυτούς αλλά και γενικότερα με την συγκεκριμένη υποδομή του υπολογιστικού νέφους.
- **Κεφάλαιο 5:** Κάνει μια συνολική αποτίμηση των εργασιών που έγιναν και των συμπερασμάτων που εξάγονται από την παρούσα εργασία. Παραθέτει πιθανές

μελλοντικές ερευνητικές κατευθύνσεις όπου κάποιες από αυτές θα μπορούσαν να βασιστούν πάνω στη συγκεκριμένη εργασία.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 ΣΧΕΔΙΑΣΜΟΣ ΧΩΡΗΤΙΚΟΤΗΤΑΣ ΚΑΙ ΑΠΟΔΟΣΗ ΝΕΦΟΫΠΟΛΟΓΙΣΤΙΚΩΝ ΥΠΟΔΟΜΩΝ

2.1.1 Το υπολογιστικό νέφος και οι υπηρεσίες του

Το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (National Institute of Standards and Technology - NIST) ορίζει το υπολογιστικό νέφος ως ένα μοντέλο για χορήγηση πρόσβασης χρηστών σε μια διαμοιραζόμενη ομάδα διαμορφώσιμων υπολογιστικών πόρων οι οποίοι περιλαμβάνουν δίκτυα, εξυπηρετητές (servers), αποθηκευτικούς χώρους (storage), εφαρμογές (applications) και υπηρεσίες (services) που προσφέρονται ταχύτατα και με μικρή αλληλεπίδραση τόσο στην διαχείριση όσο και στην παροχή τους. Το υπολογιστικό νέφος δεν είναι μια καινούργια τεχνολογία από μόνο του αλλά ένας νέος τρόπος να ενσωματωθούν σε αυτό υπάρχουσες τεχνολογίες. Με άλλα λόγια, αυτό είναι απλά ένας τρόπος για να έχει μια επιχείρηση πρόσβαση σε υπολογιστικούς πόρους και να τους διαχειρίζεται χωρίς να ενδιαφέρεται για την συντήρηση ή την υφιστάμενη υποδομή (infrastructure) τους [14].

Το υπολογιστικό νέφος έχει διαμορφώσει τον τρόπο με τον οποίο χρησιμοποιούνται από τους πελάτες οι διάφορες υπηρεσίες και οι αντίστοιχες υποδομές τους ενώ παράλληλα έχει δώσει το έναυσμα για την ανάδειξη του ως η πέμπτη υπηρεσία κοινής ωφέλειας (utility service). Ύστερα από την εμφάνιση του έχει γίνει αποδεκτό από βιομηχανικούς οργανισμούς, κυβερνητικά ιδρύματα και την πανεπιστημιακή κοινότητα ενώ παράλληλα η υιοθέτηση του γνωρίζει ραγδαία εξέλιξη. Το πρότυπο αυτό έχει αναπτυχθεί στη ραχοκοκαλιά της σύγχρονης οικονομίας προσφέροντας αν και όταν ζητηθεί (on demand) πρόσβαση σε συνδρομητικούς πληροφοριακούς πόρους (Information Technology - IT), αναδεικνύοντας όχι μόνο τον τρόπο με τον οποίο αποκτιέται πρόσβαση σε υπηρεσίες κοινής ωφέλειας αλλά και την εξάρτηση της σύγχρονης κοινωνίας από αυτούς [1].

Οι υπηρεσίες του νέφους ταξινομούνται σε τρείς κύριες κατηγορίες οι οποίες αφορούν την υποδομή, την πλατφόρμα (platform) και το λογισμικό (software). Ωστόσο, ο έλεγχος ως υπηρεσία (Testing-as-a-Service) έχει γίνει πρόσφατα γνωστός ως καινούργιος και απαιτητικός τύπος μοντέλου υπηρεσίας του νέφους [15].

- Στην **υποδομή ως υπηρεσία (Infrastructure-as-a-Service - IaaS)** ο πελάτης μπορεί να αποκτήσει διάφορους υπολογιστικούς πόρους οι οποίοι

συμπεριλαμβάνουν αποθήκευση δεδομένων (data storage), δικτυακές συσκευές (network devices) και επεξεργαστική ισχύ (processing power).

- Στην **πλατφόρμα ως υπηρεσία (Platform-as-a-Service - PaaS)** ο πελάτης μπορεί να αποκτήσει πρόσβαση σε ένα σύνολο προγραμματιστικών περιβαλλόντων, όπως είναι τα λειτουργικά συστήματα, οι βάσεις δεδομένων (databases) και οι διαδικτυακοί εξυπηρετητές (web servers).
- Στο **λογισμικό ως υπηρεσία (Software-as-a-Service - SaaS)** ο πελάτης έχει πρόσβαση σε εφαρμογές μέσω μιας διασύνδεσης πελάτη (client interface), όπως για παράδειγμα ο φυλλομετρητής (web browser), χωρίς την πολυπλοκότητα των αντίστοιχων εγκαταστάσεων.
- Στον **έλεγχο ως υπηρεσία (Testing-as-a-Service - TaaS)** ο πελάτης μπορεί να αποκτήσει πρόσβαση σε εξομοιωμένα περιβάλλοντα ελέγχου και να παρακολουθήσει σύνθετες συμπεριφορές προγραμμάτων αδιάκοπα.

2.1.2 Χωρητικότητα και απόδοση νεφοϋπολογιστικών υποδομών

Με την αλματώδη πρόοδο των υπολογιστικών νεφών και των δικτυακών τεχνολογιών έχει αναπτυχθεί από τους παρόχους υπηρεσιών ένα μεγάλο φάσμα υπηρεσιών που βασίζονται στο νέφος (cloud-based services) οι οποίες έχουν χρησιμοποιηθεί από πολυάριθμους οργανισμούς ως αναπόσπαστα συστατικά των πληροφοριακών τους συστημάτων. Η απόδοση (performance) αυτών των υπηρεσιών έχει σημαντική επίδραση στις επιδόσεις των μελλοντικών πληροφοριακών υποδομών, οπότε η διεξοδική αξιολόγηση τους είναι κρίσιμη και ωφέλιμη τόσο για τους παρόχους υπηρεσιών όσο και για τους πελάτες τους [9]. Συνεπώς, οι έλεγχοι και οι μετρήσεις της απόδοσης τέτοιων υπηρεσιών είναι αναπόφευκτα για τις μελλοντικές βελτιώσεις και εξελίξεις των υπολογιστικών νεφών [2].

Οι πάροχοι υπηρεσιών του νέφους οφείλουν να έχουν βαθιά γνώση για τη σχέση ανάμεσα στην απόδοση αυτών των υπηρεσιών και των διαθέσιμων πόρων έτσι ώστε να αξιοποιούν πλήρως τις πληροφοριακές υποδομές τους ενώ παράλληλα να ικανοποιούν τις απαιτήσεις των πελατών τους για απρόσκοπτη λειτουργία. Οι υποδομές του νέφους μπορούν να γίνουν εφικτές εναλλακτικές των αντίστοιχων συμβατικών εφόσον αυτές παρέχουν ένα κατάλληλο επίπεδο

απόδοσης. Επιπλέον, οι ερευνητές θα πρέπει να ξεκινούν την αξιολόγηση της απόδοσης των προσφερόμενων υπηρεσιών μόλις αυτές γίνονται διαθέσιμες ώστε να παρέχουν έναν οδηγό σε ενδεχόμενους χρήστες οι οποίοι μετά θα αποφασίσουν αν θα τις υιοθετήσουν [9].

Η πρόσφατη ανάπτυξη των δικτυακών υποδομών έχει βελτιώσει τις εφαρμογές του διαδικτύου (Web-based applications) διαφόρων κλάδων και επηρεάζει σημαντικά τις ζωές των σημερινών ανθρώπων. Η μεγάλη αλλαγή στις υποδομές δικτύωσης έχει δώσει τη δυνατότητα στη χωρητικότητα (capacity) τους να παίζει σημαντικό ρόλο στην παροχή μιας προσδοκώμενης υπηρεσίας. Η πρόκληση γίνεται μεγαλύτερη με την αύξηση του μεγέθους των δεδομένων που διακινούνται στα διάφορα δίκτυα. Για παράδειγμα, μια κυκλοφοριακή συμφόρηση (bottleneck) σε ένα δίκτυο αποτελεί ένα σοβαρό εμπόδιο στην αύξηση της απόδοσης του, η οποία στη συνέχεια μπορεί να καταλήξει σε μια μεγάλη καθυστέρηση μεταφοράς δεδομένων (latency). Συνεπώς, η όσο το δυνατόν πιο ποιοτική υλοποίηση των σημερινών δικτύων αποτελεί πλέον μια επιτακτική ανάγκη [18].

Για να εξασφαλιστεί μια υψηλής ποιότητας απόδοση, για παράδειγμα σε κρίσιμες εφαρμογές του νέφους οι οποίες βασίζονται στην δικτυακή επικοινωνία, θα πρέπει να υφίσταται η απαιτούμενη ταχύτητα μεταφοράς δεδομένων (bandwidth) χωρίς να διευρύνεται ή υπερτροφοδοτείται με πόρους το δίκτυο. Με τις υφιστάμενες συνθήκες παράμετροι των δικτύων, όπως είναι η ταχύτητα μεταφοράς δεδομένων ή η διεκπεραιωτικότητα (throughput) δεν μπορούν να διασφαλιστούν [8]. Για να ικανοποιηθεί αποτελεσματικά ένα υψηλό επίπεδο ζήτησης υπηρεσιών απαιτούνται συνεχείς αλλαγές στις υποδομές υποστήριξης τους [13].

Οι απαιτήσεις των υπηρεσιών του νέφους έχουν μεγαλώσει τα τελευταία χρόνια από τη μία λόγω της ετερογένειας τους και από την άλλη λόγω της αναπόφευκτης δυναμικής τους διαχείρισης. Μέσα σε αυτό το πλαίσιο η αύξηση των δικτυακών απαιτήσεων αυτών των υπηρεσιών σε συνδυασμό με την περιορισμένη ταχύτητα μεταφοράς δεδομένων δικτύου έχουν καταστήσει αναγκαία τη χρήση μηχανισμών βελτίωσης έτσι ώστε να μπορέσουν τα δίκτυα που τις εξυπηρετούν να λειτουργούν με καλύτερη αξιοποίηση της χωρητικότητας τους [13]. Ωστόσο, η πρόκληση στο συγκεκριμένο εγχείρημα είναι η δυσκολία που προκύπτει για να υπάρχει η γνώση των διαφορετικών δικτύων λόγω των πολλών νέων σχετικών τεχνολογιών που παρέχονται από την αύξηση της χρήσης τους [18].

2.2 ΕΙΚΟΝΙΚΟΠΟΙΗΣΗ (VIRTUALIZATION) ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

2.2.1 Εικονικοποίηση και Υπολογιστικό Νέφος

Μια από τις βασικές τεχνολογίες υπολογιστικής νέφους είναι η εικονικοποίηση, η οποία περιλαμβάνει την εκχώρηση από τους παρόχους υπηρεσιών νέφους (Cloud service providers) εικονικών υπολογιστικών, αποθηκευτικών και δικτυακών πόρων κατά παραγγελία (on demand) σε πελάτες [4]. Η εικονικοποίηση είναι καίριας σημασίας στην περιγραφή του υπολογιστικού νέφους, ενώ οι πόροι που εικονικοποιούνται σε αυτό μπορούν να γίνουν πιο αποδοτικά διαχειρίσιμοι. Η εικονοκοποιημένη υποδομή του νέφους παρέχει την απαιτούμενη «αφαίρεση» (abstraction) για να εξασφαλιστεί ότι μια εφαρμογή ή υπηρεσία μιας επιχείρησης είναι ανεξάρτητη από την υποκείμενη φυσική (hardware) υποδομή την οποία μπορεί να αποτελούν εξυπηρετητές, αποθηκευτικοί χώροι ή ακόμα και δικτυακές συσκευές [19].

Στο μοντέλο υποδομή ως υπηρεσία του νέφους μπορούν να δημιουργηθούν πολλοί εικονικοί εξυπηρετητές (instances) σε μια φυσική μηχανή ενώ υπάρχουν πολλοί πάροχοι υπηρεσιών νέφους που προσφέρουν τέτοιες δυνατότητες. Ωστόσο, η hypervisor-based εικονικοποίηση, όπως αποκαλείται, περιλαμβάνει επιβαρύνσεις (overhead) πόρων και μπορεί να περιορίσει την επεκτασιμότητα και προσαρμοστικότητα, ιδιαίτερα όταν διαφορετικοί χρήστες με ποικίλες ανάγκες επιθυμούν να αξιοποιήσουν αυτούς τους υπολογιστικούς πόρους. Για να ικανοποιηθούν όλοι αυτοί οι χρήστες, μια εναλλακτική λύση η οποία έχει κερδίσει ευρεία αποδοχή είναι οι υπηρεσίες micro-hosting ή μικροϋπηρεσίες (micro-services) και η εικονικοποίηση που βασίζεται σε περιέκτες (containers) [4].

2.2.2 Η χρήση των μικροϋπηρεσιών στη δημιουργία λογισμικού

Οι μικροϋπηρεσίες δεν είναι καινούργια ιδέα στην αρχιτεκτονική του λογισμικού και στις μέρες μας είναι ευρέως αναγνωρισμένες ως μια αποδοτική λύση στην ανάπτυξη εφαρμογών. Η βασική ιδέα της αρχιτεκτονικής αυτού του είδους υπηρεσιών είναι το «διαίρει και βασίλευε», δηλαδή η δημιουργία της υποδομής μιας εφαρμογής η οποία αποτελείται από ξεχωριστές μικροϋπηρεσίες σε περιέκτες που λειτουργούν ανεξάρτητα μεταξύ τους. Ουσιαστικά, αυτές αντικαθιστούν ένα μεγάλο κώδικα εντολών προγραμματισμού μιας εφαρμογής με έναν πολύ μικρότερης κλίμακας ο οποίος περιορίζεται σε μικρές και ευέλικτες ομάδες τέτοιων εντολών. Το πλεονέκτημα αυτής της ιδέας είναι ότι καθεμία ομάδα εντολών μπορεί να λειτουργεί ως ξεχωριστή μικροϋπηρεσία απομονωμένα και προστατευμένα από τις άλλες [4].

Πριν την εμφάνιση των μικροϋπηρεσιών η γενική προσέγγιση για την ανάπτυξη μιας υπηρεσίας ήταν η δημιουργία κατά κύριο λόγο μονολιθικών εφαρμογών, δηλαδή η ύπαρξη ενός και μόνο υπολογιστικού περιβάλλοντος το οποίο διαχειριζόταν τα πάντα. Σε περιβάλλοντα του νέφους πολλά από αυτά τα θέματα μπορούν να ξεπεραστούν μέσω γλωσσών σεναρίου (scripting languages) οι οποίες υποστηρίζουν και τα τρία μοντέλα υπηρεσιών του νέφους, δηλαδή υποδομής, πλατφόρμας και λογισμικού. Παρόλα αυτά, αυτές οι λύσεις αποτελούν μια πρόκληση αφού υπάρχουν πολλές εργασίες (projects) συγκεκριμένου σκοπού (ad hoc) και κοινότητες που εμπλέκονται με δικό τους λογισμικό και απαιτήσεις δεδομένων που ποικίλουν. Έτσι, σε περιβάλλοντα του νέφους οι λύσεις που βασίζονται σε αυτό είναι ωφέλιμες και μια τέτοια είναι η χρήση του μοντέλου των μικροϋπηρεσιών [4].

Από το 2014 υπάρχει ένας αυξανόμενος αριθμός εταιριών που ενδιαφέρονται να υιοθετήσουν την αρχιτεκτονική που χρειάζεται για να λειτουργήσουν οι μικροϋπηρεσίες. Λόγω της αποκεντρωμένης φύσης της και της σύνδεσης μη περιορισμένων μικροϋπηρεσιών που προσφέρει έχει γίνει μια δημοφιλής αρχιτεκτονική για μεγάλες εφαρμογές. Ωστόσο, αυτή χρειάζεται μια υποδομή νέφους για να υπάρξει επικοινωνία ανάμεσα σε μικροϋπηρεσίες, βάσεις δεδομένων ή ακόμα και υπηρεσίες τρίτων (third-party). Επιπλέον, οι δικτυακές συνδέσεις που χρησιμοποιούνται σε αυτή την υποδομή προσθέτουν πολυπλοκότητα ακόμα και στο να γίνουν έλεγχοι στην εφαρμογή που εξυπηρετούν [20].

2.2.3 Περιέκτες και εικονικοποίηση

Οι περιέκτες ως μια μορφή εικονικοποίησης μπορούν να χωρίσουν το λειτουργικό σύστημα μιας φυσικής μηχανής σε λογικά απομονωμένα περιβάλλοντα. Εφόσον οι περιέκτες μοιράζονται το λειτουργικό σύστημα της φυσικής μηχανής αντί να τρέχουν ξεχωριστά λειτουργικά, όπως συμβαίνει με την hypervisor-based εικονικοποίηση, προκύπτει μια μείωση στην επιβάρυνση της επεξεργασίας τόσο στην μνήμη όσο και στη χρήση του σκληρού δίσκου. Οι περιέκτες χρησιμοποιούνται από πολύ μεγάλες εταιρίες, όπως η Google και η Amazon, ενώ μια από τις πιο δημοφιλής υλοποίησή τους είναι με τη χρήση της πλατφόρμας Docker [17]. Εξ ορισμού σε περιέκτες, όπως για παράδειγμα οι Docker, η δικτύωση τους διαμορφώνεται για να παρέχει το καλύτερο δυνατό (best effort) σε όλη την κίνηση δεδομένων. Έτσι, μια εφαρμογή που απαιτεί εντατική επικοινωνία (communication-intensive) και είναι εγκατεστημένη σε περιέκτες μπορεί να υποστεί μη αποδεκτή απόδοση όταν φιλοξενείται σε εξυπηρετητές μαζί με άλλες παρόμοιες εφαρμογές [8].

Έχει γίνει αρκετή έρευνα σχετικά με την απόδοση της εικονικοποίησης και έχει αναγνωριστεί ότι τεχνολογίες οι οποίες χρησιμοποιούν hypervisor-based εικονικοποίηση έχουν μεγάλες επιβαρύνσεις απόδοσης. Τελευταία, τόσο η εικονικοποίηση που βασίζεται σε περιέκτες όσο και οι μικροϋπηρεσίες έχουν αποκτήσει μεγάλη απήχηση. Αυτό οφείλεται στο γεγονός ότι προσφέρουν μια «ελαφριά» (lightweight) λύση η οποία επιτρέπει να προσφέρονται εφαρμογές μαζί με δεδομένα με ένα πιο απλό και προσανατολισμένο στην απόδοση τρόπο καθώς και ότι μπορούν να εκτελούνται σε διαφορετικές υποδομές του νέφους [4].

Οι περιέκτες παρέχουν ένα εναλλακτικό τρόπο για να διαχωρίζουν τους υλικούς (hardware) πόρους ανάμεσα στους χρήστες και να επιταχύνουν την ανάπτυξη μιας εφαρμογής. Σε σύγκριση με την hypervisor-based εικονικοποίηση οι περιέκτες υφίστανται λιγότερη επιβάρυνση και επιτρέπουν πολύ καλύτερη αναλογία σταθεροποίησης (consolidation), δηλαδή του αριθμού των εικονικών περιβαλλόντων που μπορούν να εκτελούνται σε μια φυσική μηχανή. Παρόλα αυτά, η δικτύωση η οποία αποτελεί ένα από τα κρίσιμης σημασίας συστατικά τους δεν έχει γίνει ακόμα καλά κατανοητή σε ένα περιβάλλον νέφους αν και έχουν αναπτυχθεί πολλές τεχνικές για την παροχή σύνδεσης τους σε αυτό [5].

Επιπροσθέτως, λαμβάνοντας υπόψη ότι οι περιέκτες περιλαμβάνουν μόνο σχετιζόμενες εξαρτήσεις (λειτουργικά συστήματα, βάσεις δεδομένων κ.α.) για να εκτελέσουν μια εφαρμογή, υπάρχουν ελάχιστες αντιγραφές (duplication) ίδιων διεργασιών (processes) μέσα σε ένα σύστημα. Παρόλα αυτά, η δικτυακή τους απόδοση είναι ίσως η πιο σημαντική τους πλευρά με δεδομένο ότι οι περιέκτες σχετίζονται όλο και περισσότερο με τις μικροϋπηρεσίες λόγω των εργασιών ενορχήστρωσης (orchestration), ομαδοποίησης (clustering) και αντιγραφής [4].

2.2.4 Η σημασία της δικτυακής απόδοσης υποδομών που βασίζονται σε περιέκτες

Τα δίκτυα της επόμενης γενιάς οφείλουν να υποστηρίξουν την εικονικοποίηση. Οι διαχειριστές αυτών των δικτύων είναι υπεύθυνοι να διαμορφώσουν πολιτικές και να χρησιμοποιήσουν εργαλεία παρακολούθησης και μηχανισμούς μέτρησης για να ανιχνεύσουν και να ανταποκριθούν σε ένα μεγάλο φάσμα δικτυακών γεγονότων και υπηρεσιών. Ωστόσο, η μεγάλη κλίμακα και ποικιλία της παραγόμενης δικτυακής κίνησης από αυτές τις υπηρεσίες καθιστά δύσκολο το έργο των διαχειριστών του δικτύου να μετρήσουν σε μικρά χρονικά διαστήματα την κατάσταση και τη δυναμική του δικτύου με αποτελεσματικούς και αποδοτικούς τρόπους. Επιπλέον, αυτοί είναι υποχρεωμένοι να εξασφαλίσουν την παροχή εφαρμογών/υπηρεσιών χωρίς υποβάθμιση της ποιότητας υπηρεσίας (Quality-of- Service) το

οποίο σημαίνει την ακριβή παρακολούθηση της δικτυακής κίνησης και την ανίχνευση της δικτυακής συμφόρησης με την παρουσία μηχανισμών μέτρησης όπου αυτοί διατίθενται [6].

Τα τελευταία χρόνια, οι τεχνολογίες που χρησιμοποιούνται για να γίνεται η διαχείριση και ανάπτυξη εφαρμογών/υπηρεσιών και οι οποίες βασίζονται σε περιέκτες έχουν αποκτήσει μεγάλη δυναμική στη βιομηχανία έναντι των hyper-visored εικονικών μηχανών. Παρά τα πολλά πλεονεκτήματα τους παραμένουν ακόμα κάποιες δυσκολίες να αντιμετωπιστούν όσον αφορά στην επικοινωνία ανάμεσα στους περιέκτες, κυρίως λόγω του ότι η υπάρχουσα δικτύωση σε αυτές τις τεχνολογίες δεν καταφέρνει να ικανοποιήσει τις απαιτήσεις των εφαρμογών που βασίζονται σε αυτούς σε σχέση με την απόδοση και την απομόνωση (isolation). Συνεπώς, θα πρέπει να ληφθούν υπόψιν οι συμβιβασμοί ανάμεσα σε αυτές τις παραμέτρους οι οποίοι καταλήγουν κατά κανόνα σε βάρος της απόδοσης [25].

Με βάση το γεγονός ότι η δικτυακή απόδοση είναι σημαντική όσον αφορά την εμπειρία του χρήστη σε εφαρμογές που βασίζονται σε περιέκτες (container-based applications) είναι αποφασιστικής σημασίας να επιλεγεί το σωστό δίκτυο για αυτούς. Ωστόσο, αυτό δεν είναι εύκολη υπόθεση επειδή πολλές τεχνικές έχουν αναπτυχθεί για την εικονικοποίηση δικτύων με περιέκτες στο λειτουργικό σύστημα του εξυπηρετητή τους έτσι ώστε να δημιουργηθούν απομονωμένα δίκτυα για καθένα από αυτούς ξεχωριστά [5]. Παρόλα αυτά, οι υπάρχουσες λύσεις δεν εξετάζουν την ποικιλομορφία των απαιτήσεων τέτοιων δικτύων για αντίστοιχους πόρους και κατά κάποιο τρόπο αδιαφορούν για αυτούς τους πόρους με αποτέλεσμα τον ανταγωνισμό τους [7].

2.3 ΕΛΕΓΧΟΙ ΣΤΟ ΝΕΦΟΣ (CLOUD TESTING)

2.3.1 Έλεγχοι υποδομών στο νέφος

Η αλματώδης ανάπτυξη στην τεχνολογία της πληροφορίας προκάλεσε σημαντική αύξηση των ερευνών για το υπολογιστικό νέφος. Ανάμεσα στις ερευνητικές δραστηριότητες είναι οι έλεγχοι (testing) των υποδομών του νέφους οι οποίοι αποτελούν πλέον έναν πολύ σημαντικό ερευνητικό τομέα. Οι έλεγχοι αυτοί μπορούν να ερμηνευτούν τόσο ως δοκιμές των εφαρμογών του νέφους για να διασφαλιστεί ότι η απόδοση του είναι τέτοια ώστε να μπορεί να καλύψει στο ακέραιο τις δεσμεύσεις που έχει δώσει ο πάροχος στον πελάτη όσο και ως δοκιμές για μεγάλης κλίμακας προσομοιώσεις των αλληλεπιδράσεων των χρηστών με τις εν λόγω εφαρμογές σε πραγματικό χρόνο [15].

Οι έλεγχοι στο νέφος είναι μια μορφή αξιολόγησης κατά την οποία οι εφαρμογές που ελέγχονται χρησιμοποιούν τις υποδομές του νέφους ως υπολογιστικό περιβάλλον για να

εξομοιώσουν κίνηση δεδομένων της πραγματικής ζωής αξιοποιώντας τις υπάρχουσες τεχνολογίες του. Οι εν λόγω έλεγχοι δίνουν τη δυνατότητα να εξετάζεται μια εφαρμογή με χρήση των υποδομών του νέφους, όπως για παράδειγμα το υλικό (hardware), η ταχύτητα του δικτύου (network bandwidth) και ο όγκος της κίνησης δεδομένων (workload), με αποτέλεσμα να εξομοιώνονται καλύτερα οι συνθήκες και οι παράμετροι του πραγματικού κόσμου [16].

2.3.2 Τρόποι ελέγχου στο νέφος

Οι διαδικτυακές εφαρμογές γίνονται όλο και πιο σύνθετες και αυτό έχει ως αποτέλεσμα οι δοκιμές τους με υπολογιστικούς πόρους της ίδιας της επιχείρησης να είναι αναποτελεσματικές. Αυτό οφείλεται στο ότι οι λειτουργίες ελέγχου που προσφέρουν αυτοί οι πόροι, από τη μια πρέπει να ρυθμίζονται και να συντηρούνται προσεκτικά, από την άλλη μένουν ανεκμετάλλευτες όταν δεν γίνονται αυτές οι δοκιμές. Επιπλέον, οι λειτουργίες που χρησιμοποιούνται για τον έλεγχο της εκάστοτε επιχείρησης δεν είναι τόσο ισχυρές ώστε να εξομοιώσουν την κίνηση δεδομένων των χρηστών του πραγματικού κόσμου. Έτσι, το νέφος, το οποίο παρέχει καλύτερα τέτοιες δυνατότητες μπορεί να γίνει ο μεσάζων ή το ενδιάμεσο λογισμικό (middleware) για να πραγματοποιηθούν αξιολογήσεις των υποδομών διαφόρων εφαρμογών του διαδικτύου σε πολλαπλά σενάρια προσφέροντας μια ακόμα υπηρεσία την ονομαζόμενη έλεγχος ως υπηρεσία [15].

Ο έλεγχος και οι μετρήσεις σε συστήματα υπολογιστικού νέφους πραγματοποιούνται είτε με τη χρήση εργαλείων που προσφέρουν οι πάροχοι των υπηρεσιών του όταν χρειάζονται απλά στατιστικά είτε εγκαθιστώντας κατάλληλες υποδομές για να ελέγχουν λεπτομερώς την απόδοση ενός ή περισσότερων υποσυστημάτων [23]. Τα εργαλεία της πρώτης περίπτωσης είναι ικανοποιητικά για μια γενική εικόνα σε αριθμούς του συστήματος του νέφους αλλά δε μπορούν να δώσουν αναλυτικά στατιστικά. Αρκετές φορές χρησιμοποιούνται προσαρμοσμένα πειραματικά περιβάλλοντα που απευθύνονται σε κάποιο συγκεκριμένο κομμάτι του νέφους. Σε τέτοιες περιπτώσεις οι έλεγχοι μπορούν να γίνονται χειροκίνητα ή αυτοματοποιημένα, ενώ διατίθενται ακόμα και υπηρεσίες για ελέγχους που επιδέχονται αρκετή παραμετροποίηση τόσο σε επίπεδο συστήματος, το οποίο κάνει τον έλεγχο, όσο και στην υποδομή η οποία ελέγχεται [23].

Πολλά εργαλεία ελέγχου είναι διαθέσιμα για τον έλεγχο της απόδοσης συστημάτων ή δικτύων ενώ η εστίαση καθενός από αυτά είναι διαφορετική. Κάποια από αυτά εστιάζουν στην απόδοση του ρυθμού γραφής/ανάγνωσης, άλλα στον έλεγχο της ταχύτητας μεταφοράς δεδομένων ή των υπολογιστικών διεργασιών κ.α. Το Apache Jmeter που αποτελεί ένα από τα

πιο ισχυρά εργαλεία ελέγχου, είναι μια εφαρμογή που σχεδιάστηκε να μετράει την απόδοση της λειτουργικής συμπεριφοράς. Αρχικά, αυτό το εργαλείο σχεδιάστηκε για να ελέγχει εφαρμογές του διαδικτύου αλλά στη συνέχεια επεκτάθηκε και σε άλλες λειτουργίες ελέγχου. Το Apache Jmeter μπορεί να χρησιμοποιηθεί για να εξομοιωθεί το υπερβολικό φορτίο ενός εξυπηρετητή, μιας ομάδας εξυπηρετητών ή ενός δικτύου για να αναλυθεί η συνολική του απόδοση με διαφορετικούς τύπους φορτίου (workload) [24].

Οι δοκιμές (testing) των υποδομών που δημιουργούνται στο νέφος θα πρέπει να είναι εξαντλητικές και να καλύπτουν στο μέτρο του δυνατού κάθε πιθανό σενάριο υλοποίησης. Εξαιτίας της πολυπλοκότητας των υποδομών οι έλεγχοι είναι απαραίτητοι για να διασφαλιστεί αφενός η ομαλή λειτουργία τους και αφετέρου ότι η απόδοση του νέφους είναι τέτοια, ώστε να μπορεί να υποστηρίξει όλα τα σενάρια χρήσης, ενώ ταυτόχρονα καλύπτονται οι δεσμεύσεις του παρόχου απέναντι στον πελάτη. Επιπλέον, οι δοκιμές βοηθούν να ανακαλυφθεί τι γίνεται σε περιπτώσεις που ξαφνικά το νέφος δέχεται πολύ περισσότερα αιτήματα (requests) από τα αναμενόμενα, κάτι το οποίο εμφανίζεται συχνά και για διάφορους λόγους. Με αυτόν τον τρόπο μπορεί να μελετηθεί η συμπεριφορά της υποδομής σε τέτοιες περιπτώσεις και να βρεθεί αν οι πόροι που χρησιμοποιούνται μπορούν να διαχειριστούν τέτοια σενάρια [21].

2.3.3. Είδη ελέγχου των υποδομών του νέφους

Ορισμένα από τα είδη ελέγχου που πραγματοποιούνται στο νέφος είναι τα παρακάτω (Εικόνα 2.1) [16]:

Έλεγχος ευελιξίας και επεκτασιμότητας (Elasticity and Scalability Testing)

Ένα από τα κύρια χαρακτηριστικά αυτού του είδους ελέγχου στο νέφος είναι η αυτόματη προσαρμοστικότητα, όπως για παράδειγμα, η ικανοποίηση ζήτησης πόρων εν λειτουργία με ευέλικτο τρόπο. Επομένως, είναι απαραίτητο να ελέγχεται τόσο η κάθετη (vertical) όσο και η οριζόντια (horizontal) επεκτασιμότητα των υποδομών του. Η μεν κάθετη επεκτασιμότητα σημαίνει την αντικατάσταση των υφιστάμενων πόρων με πιο ισχυρούς για να ικανοποιηθεί η αυξημένη ζήτηση. Η δε οριζόντια επεκτασιμότητα σημαίνει την προσθήκη περισσότερων πόρων ίδιου τύπου για αύξηση της απόδοσης, ανάλογα με τις απαιτήσεις. Επίσης, οι πόροι θα πρέπει να παρέχονται απρόσκοπτα όταν δημιουργούνται οι ανάγκες.

Έλεγχος ασφάλειας (Security Testing)

Οι πλατφόρμες και οι εφαρμογές του νέφους εκτίθενται σε διάφορες αδυναμίες ασφάλειας με πιο σημαντική αυτή της ασφάλειας διέλευσης (traversal). Αυτή υφίσταται όταν ένας πελάτης έχει τη δυνατότητα να μεταπηδήσει από τον δικό του χώρο σε μια εικονική μηχανή στο χώρο

άλλου πελάτη της ίδιας μηχανής με ότι συνέπειες μπορεί να έχει αυτό. Έτσι, γίνεται ένας τέτοιου είδους έλεγχος για να διασφαλιστεί ότι δεν θα μπορούν να προκύψουν τέτοιες καταστάσεις.

Έλεγχος απόδοσης (Performance Testing)

Παρά τα μοναδικά χαρακτηριστικά των περιβαλλόντων του νέφους θα πρέπει να εξετάζεται η ορθότητα των δεδομένων που βρίσκονται στο νέφος μετρώντας την καθυστέρηση μεταφοράς και τη ρυθμαπόδοση/διεκπεραιωτικότητα. Με αυτό το είδος ελέγχων διαπιστώνεται αν η απόδοση ανάλογα με το φόρτο των μηχανημάτων είναι η αναμενόμενη σε πολλά σενάρια και συνθήκες λειτουργίας.

Απευθείας έλεγχος αναβάθμισης (Live Upgrade Testing)

Αυτός σχετίζεται με την κατανόηση της απόδοσης του συστήματος όταν γίνεται μια αναβάθμιση λογισμικού και της δυνατότητας να συνεχίζεται η παροχή των υπηρεσιών μιας επιχείρησης ακόμα και όταν η αναβάθμιση είναι σε εξέλιξη. Μέσω αυτού του ελέγχου είναι απαραίτητο να διαπιστώνεται ότι η επιχείρηση δεν σταματά τη λειτουργία της σε περιπτώσεις που γίνεται αναβάθμιση ή συντήρηση στον υλικό εξοπλισμό της.

Έλεγχος προσομοίωσης ακραίων καταστάσεων (Stress Testing)

Αυτός ο έλεγχος χρησιμοποιείται για να διαπιστώσει τη δυνατότητα μιας εφαρμογής ή υποδομής να κρατήσει ένα συγκεκριμένο επίπεδο αποδοτικότητας σε ακραίες συνθήκες λειτουργίας. Είναι ουσιώδες για μια εφαρμογή να συνεχίσει να λειτουργεί ακόμα και σε τέτοιες καταστάσεις και επειδή το κόστος δημιουργίας τέτοιων σεναρίων είναι πολύ μεγάλο, το νέφος προσφέρει μια αρκετά οικονομική λύση για την υλοποίηση τους.

Έλεγχος λειτουργικότητας (Functional Testing)

Ο έλεγχος αυτός έχει να κάνει με τη λειτουργία μιας εφαρμογής ή υποδομής σύμφωνα με τις προκαθορισμένες απαιτήσεις.

Έλεγχος συμβατότητας (Compatibility Testing)

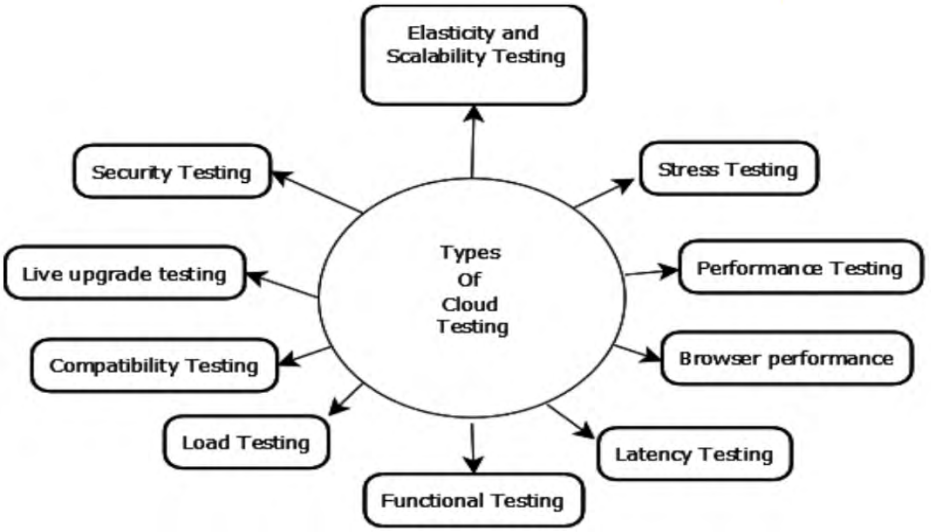
Με τη χρήση των περιβαλλόντων του νέφους οι έλεγχοι αυτοί γίνονται χωρίς κόπο, όπως για παράδειγμα στα διαφορετικά λειτουργικά συστήματα που δημιουργούνται ύστερα από κατά παραγγελία.

Έλεγχος απόδοσης προγραμμάτων περιήγησης (Browser Performance)

Ο έλεγχος αυτός επιβεβαιώνει την υποστήριξη μια εφαρμογής από διάφορους τύπους προγραμμάτων περιήγησης και η μέτρηση της απόδοσης της γίνεται πολύ εύκολα.

Έλεγχος καθυστέρησης μεταφοράς (Latency Testing)

Ελέγχει την καθυστέρηση ανάμεσα στο αίτημα (request) που γίνεται σε μια εφαρμογή που δημιουργήθηκε στο νέφος και την αντίστοιχη απάντηση (response) που λαμβάνεται από αυτή.



Εικόνα 2.1 - Είδη ελέγχου που πραγματοποιούνται στο νέφος

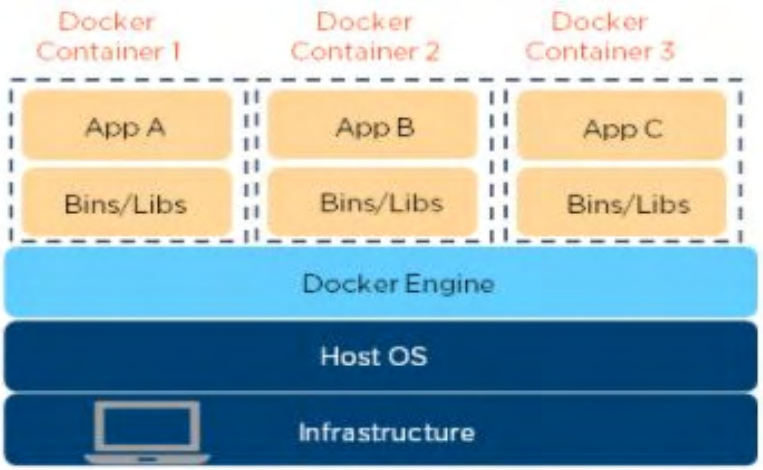
3. ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΤΑΞΗ - ΜΕΘΟΔΟΙ

3.1 ΥΠΟΔΟΜΗ (INFRASTRUCTURE)

Για την πραγματοποίηση των δοκιμών χρησιμοποιήθηκε ένα περιβάλλον υπολογιστικού νέφους στο οποίο έγινε η υλοποίηση της πειραματικής διάταξης και πραγματοποιήθηκαν όλες οι μετρήσεις. Αυτό δημιουργήθηκε με τη χρήση των υπολογιστικών υπηρεσιών σε μορφή δημόσιας υποδομής-ως-υπηρεσία που προσφέρει ο ΩΚΕΑΝΟΣ (Okeanos) λόγω του ότι υπάρχει η δυνατότητα πρόσβασης σε λογαριασμό που μας επιτρέπει να έχουμε έναν ικανοποιητικό αριθμό υπολογιστικών πόρων.

Πιο συγκεκριμένα, η πειραματική υποδομή αποτελείται από μια εικονική μηχανή με λειτουργικό σύστημα Ubuntu `16.04` LTS (Xenial Xerus) το οποίο διαθέτει οκτώ εικονικούς επεξεργαστές, 8 GB μνήμης RAM και σκληρό δίσκο χωρητικότητας 40 GB. Σε αυτή είναι εγκατεστημένο το εργαλείο μετρήσεων Apache JMeter απευθείας πάνω στο λειτουργικό της σύστημα. Επίσης, η εικονική μηχανή έχει εγκατεστημένο το Docker Engine `18.09.7` και το Docker Compose `1.29.2`. Η εφαρμογή που χρησιμοποιείται για τις πειραματικές δοκιμές είναι εγκατεστημένη σε περιέκτες οι οποίοι δημιουργούνται με τη χρήση του εργαλείου Docker Compose.

Στο Docker Engine, το οποίο σχηματικά απεικονίζεται στην παρακάτω εικόνα, υπάρχει η δυνατότητα να εγκατασταθεί μια εφαρμογή σε γλώσσα Python που ονομάζεται Docker Compose η οποία μπορεί να χρησιμοποιηθεί για την ανάπτυξη και διαχείριση μιας παραγωγικής εφαρμογής, η οποία είναι εγκατεστημένη σε πολλούς περιέκτες αλλά σε έναν μόνο εξυπηρετητή (single host). Μέσω αυτού του εργαλείου, αντί να υπάρχει ένας μόνο περιέκτης που να έχει εγκατεστημένα όλα τα συστατικά της εφαρμογής αυτά, μπορούν να χωριστούν σε περισσότερους περιέκτες όπου καθένας από αυτούς έχει μια συγκεκριμένη αποστολή. Για παράδειγμα, ένας περιέκτης μπορεί να περιλαμβάνει τον κώδικα εντολών της εφαρμογής, ένας δεύτερος να έχει την βάση δεδομένων για αποθήκευση κ.α. Με αυτόν τον τρόπο αυτοί μπορούν να εκτελούνται, να αναβαθμίζονται και να ελέγχονται ανεξάρτητα ο ένας από τον άλλον [27].



Εικόνα 3.1 - Docker engine και περιέκτες

Το Docker Compose λειτουργεί σε όλα τα περιβάλλοντα μιας εφαρμογής, δηλαδή παραγωγής (production), οργάνωσης (staging), ανάπτυξης (development) και ελέγχου. Για την χρήση του απαιτείται μια διαδικασία τριών βημάτων η οποία αναλύεται στη συνέχεια [35]:

1. Καθορίζεται το περιβάλλον της εφαρμογής/υπηρεσίας με το αρχείο *Dockerfile* για να μπορεί να αναπαραχθεί οπουδήποτε.
2. Καθορίζονται οι μικροϋπηρεσίες που αποτελούν την εφαρμογή με το αρχείο *docker-compose.yml* για να μπορούν να εκτελούνται μαζί σε ένα απομονωμένο περιβάλλον.
3. Εκτελείται η εντολή του Docker *docker compose up* για να ξεκινήσει η εφαρμογή.

Το αρχείο *Dockerfile* είναι ένα αρχείο κειμένου που περιέχει όλες τις απαραίτητες οδηγίες που χρειάζονται για να συγκεντρωθούν όλα μαζί τα μέρη της εφαρμογής ή υπηρεσίας σε ένα αρχείο εικόνας (image) για τη δημιουργία του Docker περιέκτη (Docker container image). Ένα αρχείο εικόνας του Docker είναι ένα είδος «πακεταρίσματος» (packaging) που περιέχει οτιδήποτε απαιτείται για να εκτελεστεί μια εφαρμογή. Επίσης, μέσα από αυτό δίνεται η δυνατότητα κλήσης άλλων έτοιμων αρχείων εικόνας τα οποία μπορεί να περιλαμβάνουν ακόμα και ένα περιορισμένο λειτουργικό σύστημα. Έτσι, ένα αρχείο *Dockerfile* μπορεί να περιέχει τον κώδικα της εφαρμογής, τις εξαρτήσεις (dependencies) της και δομές/κατασκευές ενός λειτουργικού συστήματος. Ένα δείγμα από ένα τέτοιο αρχείο φαίνεται στη συνέχεια:

```
# εγκατάσταση του ubuntu 22.04 OS από αρχείο εικόνας
FROM ubuntu:22.04

# εγκατάσταση των εξαρτήσεων της εφαρμογής
RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip install flask==2.1.*

# εγκατάσταση της εφαρμογής
COPY hello.py /

# τελικές παραμετροποιήσεις
ENV FLASK_APP=hello
EXPOSE 8000
CMD flask run --host 0.0.0.0 --port 8000
```

Τα αρχεία εικόνας μπορούν να αναζητηθούν αυτόματα κατά την εκτέλεση του Docker Compose από ένα «μητρώο» (registry) με το πιο κοινό να είναι το Docker Hub, αλλά υπάρχουν και άλλα. Τα αρχεία αυτά μπορούν να «κατέβουν» (download) στον τοπικό εξυπηρετητή του Docker (Docker host) όπου μπορούν να χρησιμοποιηθούν στη συνέχεια για να ξεκινήσουν έναν ή περισσότερους περιέκτες [28]. Σε αυτή την εφαρμογή που θα χρησιμοποιηθεί στην παρούσα εργασία τα αρχεία εικόνας προέρχονται από το Docker Hub.

Το αρχείο *docker-compose.yml* που χρειάζεται για την πειραματική εφαρμογή της εργασίας είναι αυτό που φαίνεται παρακάτω:

```
version: "3.9"
services:
 db:
   image: mysql:5.7
   volumes:
    - db_data:/var/lib/mysql
   restart: always
```

```
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data: {}
  wordpress_data: {}
```

Το πρώτο πράγμα που μπορεί να παρατηρηθεί σε αυτό το αρχείο είναι τα τρία «κλειδιά» (keys) *versions:*, *services:* και *volumes:* τα οποία εμφανίζονται σε ανώτερο επίπεδο λόγω των εσοχών που υπάρχουν στις γραμμές του αρχείου που βρίσκονται κάτω από αυτά. Το κλειδί *versions:* είναι υποχρεωτικό, βρίσκεται πάντοτε στην πρώτη γραμμή του αρχείου και ορίζει την έκδοση του. Το κλειδί *services:* είναι το μέρος που ορίζονται οι διάφορες μικροϋπηρεσίες της εφαρμογής σε δικούς τους περιέκτες. Στην δοκιμαστική εφαρμογή αυτής της εργασίας ορίζονται δυο μικροϋπηρεσίες με ονόματα db και wordpress. Το κλειδί *volumes:* ορίζει ειδικούς καταλόγους μέσα στους περιέκτες οι οποίοι αντιστοιχούν σε καταλόγους του εξυπηρετητή για την αποθήκευση των δεδομένων της εφαρμογής [28].

Μετά την εκτέλεση της εντολής *docker compose up* από την γραμμή εντολών του λειτουργικού συστήματος της εικονικής μηχανής θα δημιουργηθεί καθεμία από αυτές τις μικροϋπηρεσίες σε περιέκτες οι οποίοι θα έχουν τα ίδια ονόματα με αυτά που αναφέρονται στο παραπάνω αρχείο. Επίσης, είναι σημαντικό να τονιστεί η διάκριση ανάμεσα στους δυο αριθμούς πορτών οι οποίοι εμφανίζονται στο προηγούμενο *docker-compose.yml* αρχείο στις γραμμές:

```
ports:
    - "8000:80"
```

όπου ο αριθμός πόρτας 8000 ονομάζεται HOST_PORT ενώ ο αριθμός πόρτας 80 έχει το όνομα CONTAINER_PORT και ο πρώτος χρησιμοποιείται για την πρόσβαση της μικροϋπηρεσίας wordpress έξω από το Docker Compose ενώ ο δεύτερος για την επικοινωνία μεταξύ των περιεκτών της εφαρμογής.

Για τον ορισμό και την εκτέλεση της εφαρμογής που χρησιμοποιείται στην εργασία με χρήση του Docker Compose ακολουθούνται τα παρακάτω βήματα:

- Δημιουργείται ένας άδειος φάκελος στην εικονική μηχανή ο οποίος θα περιέχει τους πόρους που θα χρειαστούν για να δημιουργηθεί το γενικό πλαίσιο ή το περιβάλλον για το αρχείο εικόνας της εν λόγω εφαρμογής.
- Χρησιμοποιείται το αρχείο *docker-compose.yml* που αναλύθηκε παραπάνω το οποίο θα τοποθετηθεί στον νέο φάκελο.
- Εκτελείται η εντολή *docker compose up* μέσα από τον ίδιο φάκελο.
- Εκτελείται η εφαρμογή WordPress στην πόρτα `8000` της εικονικής μηχανής (Docker host) πληκτρολογώντας σε έναν φυλλομετρητή ενός άλλου συστήματος τη διεύθυνση πλοήγησης `http://83.212.116.223:8000`.

Είναι σημαντικό να τονιστεί ότι το Docker Compose κατά τις επανεκκινήσεις του διατηρεί τους ειδικούς καταλόγους (volumes) που χρησιμοποιούνται από τις μικροϋπηρεσίες του. Ακόμα, όταν γίνονται αλλαγές στα αρχεία αυτών των καταλόγων έξω από τον αντίστοιχο περιέκτη που έχει τον κώδικα μιας διαδικτυακής εφαρμογής, αυτές οι αλλαγές θα εφαρμοστούν άμεσα με μια απλή ανανέωση της ιστοσελίδας της. Επίσης, κάθε αρχείο εικόνας (image) που

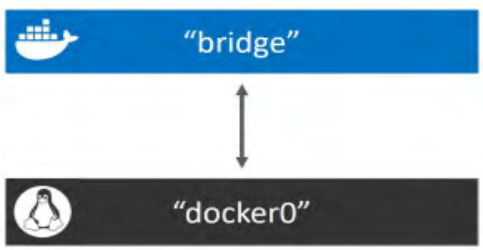
δημιουργείται με την εντολή *docker compose up* θα παραμείνει στο σύστημα οπότε κάθε μελλοντική ανάπτυξη της εφαρμογής θα γίνεται γρηγορότερα [28].

3.2 DOCKER COMPOSE NETWORKING (BRIDGE/HOST MODE)

3.2.1 Η δικτύωση στο Docker

Η δημοφιλής πλατφόρμα λογισμικού Docker έχει αλλάξει τον τρόπο ανάπτυξης λογισμικού τα τελευταία χρόνια και έχει βοηθήσει την κοινότητα των προγραμματιστών καθώς και τις εταιρίες να δημιουργούν εύκολα και γρήγορα τις εφαρμογές τους. Χρησιμοποιεί την αρχιτεκτονική πελάτη-εξυπηρετητή (client-server) όπου ο Docker client επικοινωνεί με τον Docker server ή Docker Host ή Docker Daemon ο οποίος αναλαμβάνει το βαρύ φορτίο να δημιουργήσει, να τρέξει και να διαμοιράσει περιέκτες. Ο πελάτης και ο εξυπηρετητής του Docker μπορούν να εκτελούνται στο ίδιο σύστημα ή σε απομακρυσμένα συστήματα. Ένα είδος Docker client είναι και το Docker Compose το οποίο επιτρέπει να εκτελούνται εφαρμογές που αποτελούνται από μια ομάδα περιεκτών [41].

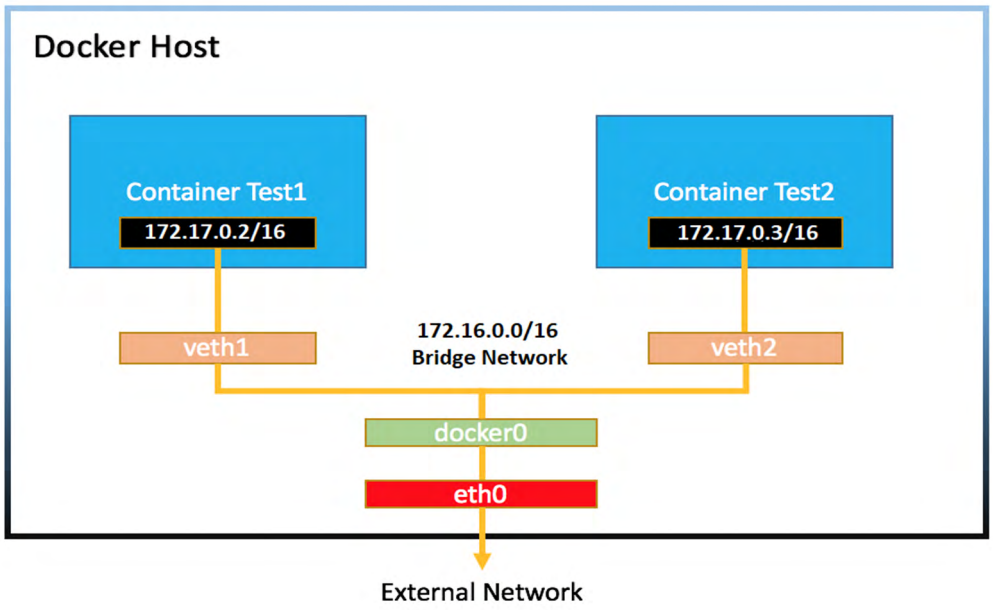
Η πλατφόρμα Docker που χρησιμοποιείται για την υλοποίηση περιεκτών σε έναν Linux εξυπηρετητή δημιουργεί ένα προκαθορισμένο (bridge) δίκτυο που αντιστοιχίζεται σε ένα άλλο υποκείμενο δίκτυο, που προϋπάρχει στον πυρήνα (kernel) του Linux εδώ και πολλά χρόνια, μέσω μιας εικονικής γέφυρας δικτύωσης (virtual ethernet bridge) η οποία ονομάζεται docker0 [26].



Εικόνα 3.2 - Το προκαθορισμένο bridge δίκτυο στο Docker

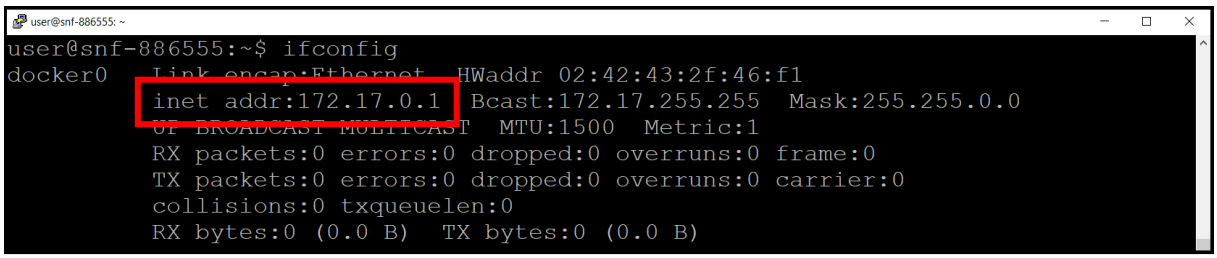
Όλοι οι περιέκτες που δημιουργούνται στον Docker εξυπηρετητή ρυθμίζονται αυτόματα να ανήκουν στο ίδιο υποδίκτυο (subnet) για να συνδεθούν σε αυτήν την εικονική γέφυρα και να μπορούν να επικοινωνήσουν μεταξύ τους. Ωστόσο, υπάρχει η επιλογή να ανήκουν σε άλλα δίκτυα του ίδιου μηχανήματος τα οποία μπορούν να ορίζονται από τον διαχειριστή του Docker (user-defined networks). Για κάθε περιέκτη δημιουργείται μια εικονική διασύνδεση δικτύου

(virtual ethernet interface - veth or endpoint) και του ανατίθεται μια ιδιωτική διεύθυνση δικτύου (private IP address) από το Docker [26].



Εικόνα 3.3 - Τοπολογία του προκαθορισμένου δικτύου στο Docker

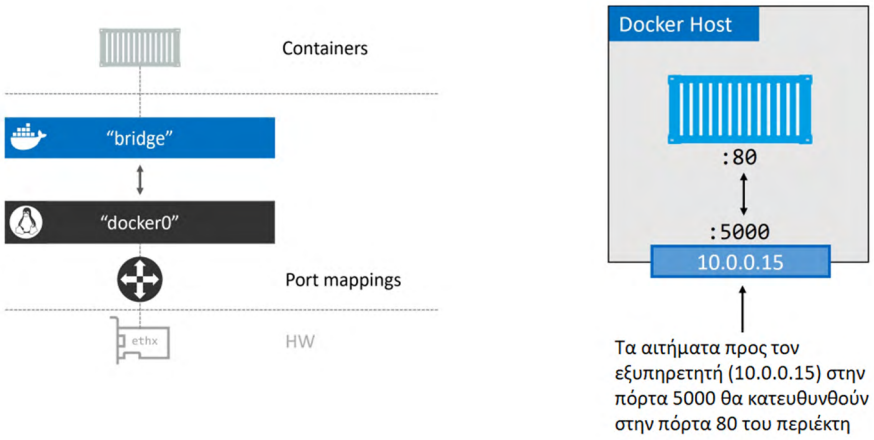
Στην προηγούμενη εικόνα οι περιέκτες Test1 και Test2 είναι συνδεδεμένοι στο υποδίκτυο `172.17.0.0/16` ενώ έχουν αποκτήσει από το Docker τις διευθύνσεις δικτύου `172.17.0.2` και `172.17.0.3` στις δυο εικονικές τους διασυνδέσεις veth1 και veth2 αντίστοιχα. Στην επόμενη εικόνα με την εκτέλεση της εντολής *ifconfig* στη γραμμή εντολών του λειτουργικού συστήματος της εικονικής μηχανής εμφανίζεται η διεύθυνση δικτύου (`172.17.0.1`) της εικονικής γέφυρας η οποία λειτουργεί ως πύλη (gateway) μεταξύ του εξυπηρετητή Docker και του υποδικτύου.



Εικόνα 3.4 - Η διεύθυνση δικτύου της εικονικής γέφυρας

Για να μπορεί να υπάρχει πρόσβαση σε έναν περιέκτη από ένα διαφορετικό σύστημα χρησιμοποιείται η αντιστοίχιση πόρτας (port mapping) η οποία επιτρέπει να αντιστοιχηθεί μια πόρτα (port) ενός περιέκτη σε μια ελεύθερη πόρτα του εξυπηρετητή που τον φιλοξενεί. Έτσι, όταν γίνεται ένα αίτημα που έρχεται από άλλο σύστημα προς τον εξυπηρετητή σε αυτήν την

πόρτα, αυτό μεταβιβάζεται μέσω αυτού του μηχανισμού στον περιέκτη που έχει γίνει η εν λόγω αντιστοίχιση [28].



Εικόνα 3.5 - Η αντιστοίχιση πόρτας στους περιέκτες του Docker

Στο προηγούμενο σχεδιάγραμμα η εφαρμογή που εκτελείται στον περιέκτη λειτουργεί στην πόρτα 80 η οποία έχει αντιστοιχηθεί με την πόρτα 5000 στη διασύνδεση δικτύου του εξυπηρετητή. Έτσι, όλη η δικτυακή κίνηση που φτάνει σε αυτήν την πόρτα του εξυπηρετητή `10.0.0.15:5000` ανακατευθύνεται στην πόρτα 80 του περιέκτη.

Μέχρι αυτό το σημείο δεν υπάρχουν δυνατότητες για τον διαμοιρασμό του δικτύου, για παράδειγμα του εύρους ζώνης (bandwidth) του, όπως μπορεί να γίνει με τον επεξεργαστή και την μνήμη του Docker εξυπηρετητή. Επομένως, οι εν λόγω περιέκτες μοιράζονται το ίδιο εύρος ζώνης στο δίκτυο που ανήκουν [26]. Επιπλέον, οι περιέκτες που δημιουργούνται με Docker σε ένα σύστημα Linux λειτουργούν πιο κοντά στο υλικό και μοιάζουν με τις διεργασίες που εκτελούνται σε αυτό το λειτουργικό σύστημα. Ωστόσο, το γεγονός αυτό περιορίζει τη συνδεσιμότητα (connectivity) τους εφόσον δεν έχουν πρόσβαση στη διευθυνσιοδότηση του δικτύου (network addressing) που ανήκουν. Για να ξεπεραστεί αυτό το εμπόδιο και να προσεγγίσει η επικοινωνία των περιεκτών εκείνη των συμβατικών δικτύων χρησιμοποιούνται οι οδηγοί δικτύου (network drivers) οι οποίοι παρέχουν διάφορες μορφές συνδεσιμότητας [17].

Οι εφαρμογές που εκτελούνται μέσα σε περιέκτες της πλατφόρμας Docker χρειάζεται να επικοινωνήσουν με πολλά και διαφορετικά δίκτυα που σημαίνει ότι αυτή πρέπει να είναι εξοπλισμένη με κάποιες δυνατότητες δικτύωσης. Το Docker προσφέρει λύσεις για δικτύωση τόσο μεταξύ περιεκτών στον ίδιο εξυπηρετητή όσο και μεταξύ αυτών με υφιστάμενα δίκτυα εξωτερικών συστημάτων, τα οποία, για παράδειγμα, μπορεί να είναι εξυπηρετητές σε φυσικές

ή εικονικές μηχανές. Έτσι, για να αντιμετωπισθούν οι πιο συνηθισμένες δικτυακές απαιτήσεις στις εν λόγω εφαρμογές το Docker περιλαμβάνει τους οδηγούς δικτύου που αναφέρθηκαν προηγουμένως για τη δημιουργία Docker δικτύων ενός μόνο εξυπηρετητή, με πολλούς εξυπηρετητές (multi-host networks) ή για τη σύνδεση με άλλα δίκτυα. Πέρα από τα προηγούμενα δίνεται ακόμη η δυνατότητα για την κατασκευή προσαρμοσμένων οδηγών δικτύου για ειδικές περιπτώσεις δικτύωσης [28].

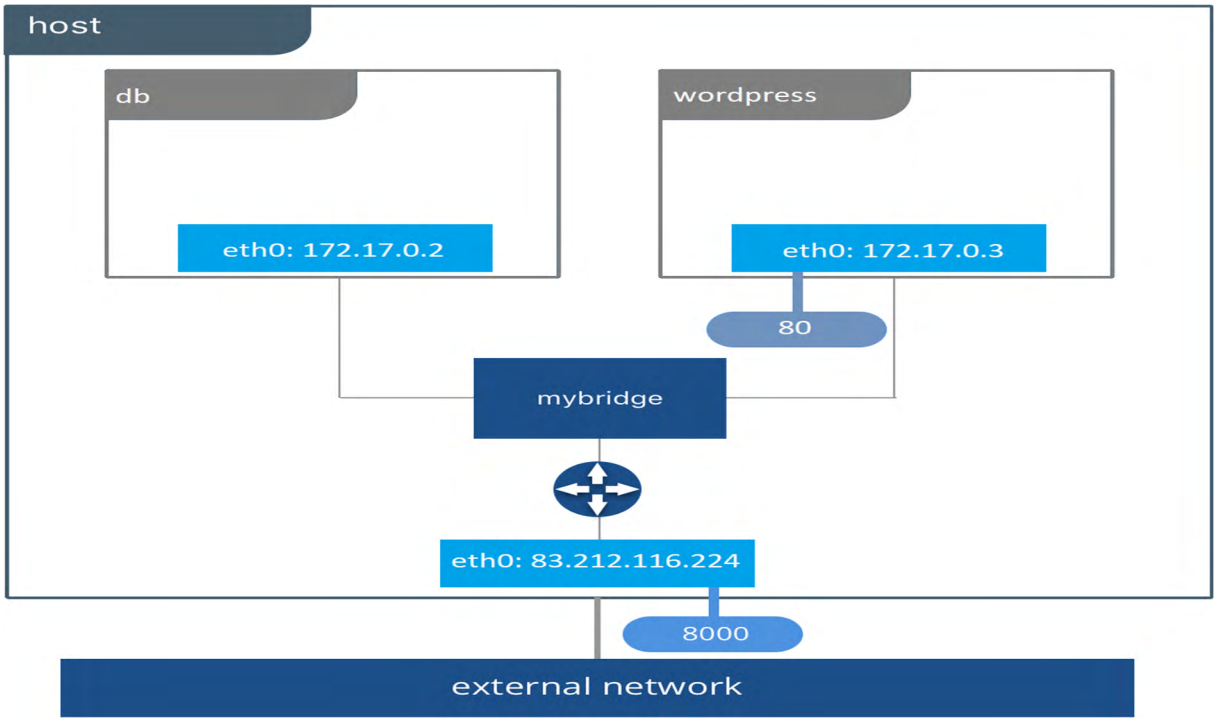
Στη συνέχεια παρατίθενται πιο αναλυτικά οι οδηγοί δικτύου που χρησιμοποιεί το Docker για να παρέχει βασική λειτουργικότητα δικτύωσης στους περιέκτες [33]:

- **bridge:** Ο προκαθορισμένος οδηγός δικτύου αν δεν οριστεί κάποιος άλλος και χρησιμοποιείται συνήθως σε περιπτώσεις που η εφαρμογή φιλοξενείται σε αυτοδύναμους περιέκτες οι οποίοι χρειάζεται να επικοινωνήσουν.
- **host:** Προορίζεται για αυτόνομους περιέκτες, αφαιρεί την απομόνωση ανάμεσα στον περιέκτη και τον εξυπηρετητή χρησιμοποιώντας απευθείας τη δικτύωση του.
- **overlay:** Συνδέει πολλές μηχανές (engines) Docker (daemons) και επιτρέπει στις υπηρεσίες που εξυπηρετούν οι περιέκτες αυτών των μηχανών να επικοινωνούν μεταξύ τους.
- **macvlan:** Επιτρέπει να αντιστοιχηθεί μια φυσική διεύθυνση (MAC address) σε έναν περιέκτη καθιστώντας τον ως μια πραγματική συσκευή σε ένα δίκτυο. Μερικές φορές ο οδηγός αυτός είναι η καλύτερη λύση όταν αφορά παλιές εφαρμογές που συνδέονται στο φυσικό δίκτυο.
- **none:** Με αυτόν τον οδηγό απενεργοποιείται όλη η δικτύωση σε έναν περιέκτη και χρησιμοποιείται συνήθως με έναν προσαρμοσμένο οδηγό δικτύου.

3.2.2. Η δικτύωση στο Docker Compose

Η δικτύωση στο Docker Compose υλοποιείται με τη δημιουργία ενός προκαθορισμένου bridge δικτύου το οποίο συνδέει κάθε περιέκτη που συμμετέχει στην εφαρμογή που εξυπηρετεί το εργαλείο. Έτσι, κάθε ένας από αυτούς είναι προσβάσιμος από τους άλλους περιέκτες της εφαρμογής ενώ μπορεί να εντοπισθεί σε αυτό το δίκτυο με το όνομα που του δίνεται από το

Docker Compose. Σε αυτό το δίκτυο δίνεται το όνομα του φακέλου μέσα στον οποίο είναι αποθηκευμένα όλα τα αρχεία που χρειάζονται για να εκτελεστεί η εν λόγω εφαρμογή και χρησιμοποιεί τον προκαθορισμένο οδηγό δικτύου bridge αν δεν γίνει κάποια άλλη επιλογή μέσω εντολών του Docker [34].



Εικόνα 3.6 - Τοπολογία του προκαθορισμένου δικτύου στο Docker Compose

Το προκαθορισμένο δίκτυο που δημιουργείται από το Docker Compose παρέχει τη δυνατότητα του εντοπισμού μιας μικροϋπηρεσίας σε περιέκτη από αυτές που συμμετέχουν σε μια εφαρμογή μέσω του DNS (Domain Name System). Το DNS είναι ένα σύστημα αντιστοίχισης της ονομασίας συστημάτων στις διευθύνσεις δικτύου τους και αυτός ο μηχανισμός εντοπισμού ονομάζεται DNS service discovery. Το Docker Compose δημιουργεί εγγραφές (DNS entries) για τις μικροϋπηρεσίες που εξυπηρετεί χρησιμοποιώντας τα ονόματα τους και μέσω αυτών μπορούν να επικοινωνούν μεταξύ τους. Επίσης, όταν μια μικροϋπηρεσία κάνει επανεκκίνηση για κάποιο λόγο θα αποκτήσει μεν μια καινούργια διεύθυνση δικτύου αλλά το όνομα της δεν θα αλλάξει [27].

Αν για κάποιο λόγο δεν είναι θεμιτό να χρησιμοποιηθεί το προκαθορισμένο δίκτυο η άλλη επιλογή που υπάρχει στο Docker Compose είναι να ρυθμιστεί με τον οδηγό δικτύου host μέσω του αρχείου *docker-compose.yml*. Για να γίνει αυτό χρησιμοποιείται η εντολή *network_mode: host* η οποία εισάγεται σε καθεμία από τις μικροϋπηρεσίες που δηλώνονται στο εν λόγω αρχείο, όπως δείχνεται παρακάτω. Επίσης, να επισημανθεί ότι με αυτόν τον οδηγό δικτύου δεν

λειτουργεί ο μηχανισμός της αντιστοίχισης πόρτας ο οποίος όμως μπορεί να υλοποιηθεί με τον προκαθορισμένο τρόπο δικτύωσης όπως αναφέρθηκε προηγουμένως.

```
version: "3.9"
services:
 db:
  image: mysql:5.7
  volumes:
   - db_data:/var/lib/mysql
  restart: always
  environment:
   MYSQL_ROOT_PASSWORD: somewordpress
   MYSQL_DATABASE: wordpress
   MYSQL_USER: wordpress
   MYSQL_PASSWORD: wordpress
  network_mode: host
 wordpress:
  depends_on:
   - db
  image: wordpress:latest
  volumes:
   - wordpress_data:/var/www/html
  restart: always
  environment:
   WORDPRESS_DB_HOST: db
   WORDPRESS_DB_USER: wordpress
   WORDPRESS_DB_PASSWORD: wordpress
   WORDPRESS_DB_NAME: wordpress
  network_mode: host
volumes:
 db_data: {}
 wordpress_data: {}
```

Με αυτόν τον τρόπο δικτύωσης οι περιέκτες που έχουν τις μικροϋπηρεσίες ενώνονται με το δίκτυο του εξυπηρετητή τους και χρησιμοποιούν την δική του διεύθυνση δικτύου για την επικοινωνία τους. Αυτό έχει ως αποτέλεσμα να μην υφίσταται η απομόνωση τους από τον

εξυπηρετητή αφού ανήκουν στο ίδιο δίκτυο. Για παράδειγμα, αν εκτελείται μια μικροϋπηρεσία σε έναν περιέκτη στην πόρτα 80 αυτή θα είναι διαθέσιμη για πρόσβαση από ένα άλλο σύστημα στην πόρτα 80 της διεύθυνσης δικτύου του εξυπηρετητή [32].

Με την χρήση του οδηγού δικτύου host δημιουργείται ένα ακόμα ζήτημα το οποίο έχει να κάνει με την αδυναμία του εντοπισμού μιας μικροϋπηρεσίας σε περιέκτη μέσω του DNS από αυτές που συμμετέχουν στην εφαρμογή του Docker Compose. Έτσι, δεν μπορεί να λειτουργήσει η εφαρμογή, μιας και δεν υπάρχει η επικοινωνία μεταξύ των μικροϋπηρεσιών της με χρήση των ονομάτων τους, που έχουν οριστεί στο αρχείο *docker-compose.yml*. Για να ξεπεραστεί αυτό το πρόβλημα, μπορεί να χρησιμοποιηθεί η εντολή *extra_hosts:* και με βάση το προηγούμενο *docker-compose.yml* αρχείο να γίνει χειροκίνητη (manual) αντιστοίχιση των ονομάτων των δυο μικροϋπηρεσιών στην διεύθυνση δικτύου (`172.17.0.1`) που αντιστοιχεί στο docker0 του εξυπηρετητή τους. Οπότε, το εν λόγω αρχείο της δοκιμαστικής εφαρμογής ύστερα από αυτές τις αλλαγές διαμορφώνεται πλέον ως εξής:

```
version: "3.9"
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    network_mode: host
    extra_hosts:
      - "wordpress:172.17.0.1"
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
```

```
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    network_mode: host
    extra_hosts:
      - "db:172.17.0.1"
volumes:
  db_data: {}
  wordpress_data: {}
```

Κάποιες παλιές εφαρμογές όπως και κάποια εργαλεία παρακολούθησης της δικτυακής κίνησης αναμένουν την απευθείας σύνδεση τους στο φυσικό δίκτυο. Σε αυτή την περίπτωση μπορεί να χρησιμοποιηθεί ο macvlan οδηγός δικτύου ο οποίος μπορεί να εκχωρήσει μια φυσική διεύθυνση (MAC address) σε κάθε εικονική κάρτα δικτύου περιέκτη για να εμφανίζεται σαν φυσική κάρτα δικτύου που συνδέεται στο φυσικό δίκτυο. Εδώ χρειάζεται να οριστεί μια κάρτα δικτύου του εξυπηρετητή για χρήση από το macvlan ή ακόμα και να απομονωθούν τα macvlan δίκτυα σε αυτόν με την χρήση διαφορετικών καρτών δικτύου [38]. Με βάση τα παραπάνω δεν επιλέχθηκε για τις δοκιμές ο εν λόγω οδηγός όπου πέρα από το διαχειριστικό κόστος δημιουργεί και μια αυξημένη πολυπλοκότητα.

Όταν πρόκειται να γίνει σύνδεση πολλών μηχανών Docker (daemons) για την εξυπηρέτηση μιας υπηρεσίας που εκτελείται σε περιέκτες τους μπορεί να χρησιμοποιηθεί ο οδηγός δικτύου overlay έτσι ώστε να πραγματοποιηθεί αυτή η επικοινωνία (multi-host communication). Για να γίνει αυτό θα πρέπει οι μηχανές που συμμετέχουν σε αυτήν την επικοινωνία να έχουν ενεργοποιημένο το swarm mode για να σχηματίσουν μια συστοιχία που ονομάζεται Swarm cluster [34]. Έτσι, ο οδηγός δικτύου overlay είναι κατάλληλος για την επικοινωνία πολλών εξυπηρετητών για μια υπηρεσία πράγμα το οποίο είναι πέρα από το πεδίο μελέτης αυτής της εργασίας.

3.3 ΓΕΝΝΗΤΡΙΕΣ ΦΟΡΤΙΟΥ (LOAD GENERATORS)

Η ανάπτυξη και εξέλιξη των συστημάτων που βασίζονται στο διαδίκτυο είναι ραγδαία τα τελευταία χρόνια και αυτό οφείλεται κυρίως σε παράγοντες όπως η κοινωνική δικτύωση (social

network) και η κινητή τηλεφωνία. Η ανάπτυξη αυτή, η οποία προϋποθέτει την ικανοποίηση εκατομμυρίων χρηστών που έχουν πρόσβαση σε εφαρμογές του διαδικτύου με μια αρκετά καλή ποιότητα υπηρεσίας, απαιτεί συνεχείς αλλαγές στις υποδομές έτσι ώστε να βελτιωθεί η εμπειρία του χρήστη και να γίνει η διαχείριση των νέων αναγκών που προκύπτουν. Επομένως, οι μελέτες που αποσκοπούν στη σύγκριση των διαφορετικών υποδομών με χρήση γεννητριών φορτίου και ανιχνεύουν συμφόρηση συστημάτων ή ελέγχουν εφαρμογές παρουσιάζουν μεγάλο ενδιαφέρον [13].

Οι γεννήτριες φορτίου περιλαμβάνουν κάποιες έννοιες που θα πρέπει να αποσαφηνιστούν πριν από τον ορισμό τους και αυτές έχουν ως εξής [31]:

- **Φορτίο εργασίας (workload)** το οποίο μπορεί να οριστεί ως ένα σύνολο καταχωρήσεων που λαμβάνει ένα σύστημα ή μια υποδομή από το περιβάλλον που ανήκουν σε μια δεδομένη χρονική περίοδο.
- **Συνθετικό φορτίο εργασίας (synthetic workload)** το οποίο είναι ένα σύνολο παραμετροποιήσιμων συνθετικών ή τεχνητών προγραμμάτων που χρησιμεύουν ως φορτίο εργασίας του εξεταζόμενου συστήματος. Τα προγράμματα αυτά επιτρέπουν στο χρήστη να δημιουργήσει διαφορετικές απαιτήσεις πόρων ενός συστήματος οι οποίες θα πρέπει να είναι παρόμοιες με τις πραγματικές ενώ η συμπεριφορά αυτού θα πρέπει να είναι ίδια και στις δυο περιπτώσεις.
- **Σημείο αναφοράς ή συγκριτικό πρότυπο (benchmark)** που είναι ένα τυποποιημένο σύστημα λογισμικού που χρησιμοποιείται για να μετρήσει την απόδοση μιας πλατφόρμας ή ενός άλλου συστήματος με τα οποία αλληλοεπιδρά και συνήθως αποτελείται από γεννήτριες φορτίου και εφαρμογές υπό έλεγχο.

Η γεννήτρια φορτίου αποτελεί τον πυρήνα πολλών εργαλείων ελέγχου απόδοσης (performance testing tools) και είναι υπεύθυνη τόσο για να ερμηνεύσει τα σενάρια (scripts) που αναφέρονται στα τεστ σε συστήματα όσο και για τη δημιουργία φορτίου σε αυτά [30]. Συνοπτικά, η γεννήτρια φορτίου είναι ένα εργαλείο λογισμικού το οποίο παράγει συνθετικό φορτίο εργασίας για σύγκριση με κάποιο πρότυπο ή για μια πραγματική εφαρμογή για να προσομοιώσει τη συμπεριφορά των τελικών χρηστών [13].

Η γεννήτρια φορτίου μπορεί να δημιουργήσει και να διαχειριστεί πολλούς εικονικούς χρήστες οι οποίοι εκτελούνται τοπικά ή με κατανεμημένο τρόπο στα διάφορα σενάρια. Στην πρώτη περίπτωση επηρεάζεται η ποιότητα των αποτελεσμάτων που λαμβάνονται μέσα από τους ελέγχους της απόδοσης εφόσον αυτά βασίζονται στην ποσότητα φορτίου που μπορεί να δημιουργηθεί και συντηρηθεί από το ίδιο το μηχάνημα το οποίο ελέγχεται ταυτόχρονα. Αυτό οφείλεται στο γεγονός ότι περιορίζεται η δημιουργία φορτίου σε ένα μόνο μηχάνημα που μπορεί να προκαλέσει συμφόρηση ακόμα και σε αυτό το ίδιο. Αντίθετα, στη δεύτερη περίπτωση το φορτίο που παράγεται κατανέμεται σε περισσότερα μηχανήματα προσθέτοντας όμως ένα ακόμα επίπεδο πολυπλοκότητας στις γεννήτριες φορτίου καθώς οι διαχειριστές τους θα πρέπει να εγκαταστήσουν πολλαπλά μηχανήματα και/ή υπηρεσίες του νέφους [30].

Οι μέθοδοι μέτρησης (metrics) που χρησιμοποιούν οι γεννήτριες φορτίου περιλαμβάνουν μετρήσιμα χαρακτηριστικά που μπορεί να αφορούν τόσο την εφαρμογή όσο και το σύστημα που βρίσκεται υπό έλεγχο. Μερικά από αυτά μπορούν να συνοψιστούν στα παρακάτω:

- **Throughput**: Δείχνει την παραγωγικότητα/διεκπεραιωτικότητα του εξυπηρετητή ανά δευτερόλεπτο μετρώντας τον πραγματικό ρυθμό με τον οποίο ολοκληρώνονται τα αιτήματα προς αυτόν.
- **Hits/Requests Per Second:** είναι ο αριθμός των αιτημάτων ανά δευτερόλεπτο.
- **Latency:** δείχνει τη χρονική καθυστέρηση ή τον χρόνο που χρειάζεται για να ταξιδέψουν τα δεδομένα από τον αποστολέα στον παραλήπτη, δηλαδή ανάμεσα στο αίτημα του χρήστη και στην λήψη της απάντησης.
- **Transaction**: Κατά τη διάρκεια του τεστ παράγονται δεδομένα από τις διάφορες συναλλαγές των εικονικών χρηστών τα οποία μπορούν να αξιολογηθούν με βάση κάποιους από τους παρακάτω σχετικούς δείκτες:
  - **Transaction Response Time (TRT):** Δείχνει τις τιμές του χρόνου απόκρισης κάτω από διαφορετικό φορτίο, όπως για παράδειγμα ο μέσος χρόνος απόκρισης.
  - **Transaction Per Second (TPS):** Δείχνει τον αριθμό των συναλλαγών που γίνονται ανα δευτερόλεπτο.
  - **Transaction Success Rate (TSR):** Δείχνει τον αριθμό των επιτυχημένων, αποτυχημένων ή σταματημένων συναλλαγών.

- **Bytes Total/Sec**: Δείχνει τη συνολική ποσότητα των δεδομένων που στέλνονται και λαμβάνονται.
- **Server Bytes Total/Sec**: Δείχνει τη συνολική ποσότητα των δεδομένων που στέλνονται και λαμβάνονται σε έναν εξυπηρετητή.
- **Error %**: Επί τις εκατό ποσοστό λαθών στη λήψη απαντήσεων από έναν εξυπηρετητή.
- **Connections Established**: Δείχνει τον αριθμό των δικτυακών συνδέσεων σε έναν εξυπηρετητή.
- **Standard Deviation:** Μετράει την αστάθεια ή μεταβλητότητα ενός συνόλου δεδομένων.

3.4 APACHE JMETER ΩΣ ΓΕΝΝΗΤΡΙΑ ΦΟΡΤΙΟΥ

3.4.1 Επιλογή εργαλείου ελέγχου απόδοσης

Υπάρχουν πολλά εργαλεία δοκιμών για τον έλεγχο της απόδοσης ενός υπολογιστή ή ενός δικτύου με καθένα από αυτά να έχει διαφορετικό επίκεντρο, όπως για παράδειγμα στον έλεγχο της ταχύτητας του δικτύου ή του υπολογιστή. Το Apache Jmeter αποτελεί ένα ισχυρό εργαλείο που σχεδιάστηκε για να ελέγχει τη συμπεριφορά και να μετράει την απόδοση αυτών που ελέγχει. Μπορεί να χρησιμοποιηθεί για να προσομοιώσει πολύ μεγάλο φορτίο σε έναν ή περισσότερους εξυπηρετητές ή ακόμα και σε ένα δίκτυο για να αναλυθεί η συνολική απόδοση κάτω από διαφορετικές συνθήκες λειτουργίας [24].

Με βάση τη μελέτη [30] για την αξιολόγηση των εργαλείων που χρησιμοποιούνται για τη δημιουργία φορτίου και την μέτρηση των αποτελεσμάτων προέκυψε ότι αυτά που αναφέρονται τις περισσότερες φορές στη βιβλιογραφία είναι τα LoadRunner και Apache Jmeter, όπου το μεν πρώτο είναι εμπορικό προϊόν ενώ το δεύτερο ανοιχτού κώδικα. Τα αποτελέσματα της συγκεκριμένης μελέτης δείχνουν σαφή προτίμηση και μεγαλύτερη υιοθέτηση αυτών των εργαλείων ελέγχου και για αυτούς τους λόγους στην παρούσα έρευνα επιλέχθηκε το Apache Jmeter.

3.4.2. Το Apache Jmeter

Το Apache Jmeter είναι ένα εργαλείο ελέγχου απόδοσης γραμμένο σε γλώσσα προγραμματισμού java που δημιουργεί αιτήματα τα οποία στέλνει σε μια συγκεκριμένη διεύθυνση δικτύου (ip address) και μετράει τους χρόνους απάντησης. Στα πλεονεκτήματα του

συγκαταλέγονται το γεγονός ότι δεν απαιτεί κάποια χρέωση στη χρήση του, η οποία μπορεί να γίνεται χωρίς περιορισμούς σε εύρος ζώνης ή άλλους περιορισμούς που υπάρχουν στα περισσότερα online εργαλεία καθώς Επίσης, προσφέρει πολλές δυνατότητες παραμετροποίησής μέσω ρυθμίσεων ή πρόσθετων (plugins) τα οποία μπορούν να εγκατασταθούν πολύ εύκολα.

Το Apache JMeter είναι ένα εργαλείο το οποίο μπορεί να χρησιμοποιηθεί για έλεγχο απόδοσης εφαρμογών που χρησιμοποιούν Hyper Text Transfer Protocol (HTTP) ή File Transfer Protocol (FTP) εξυπηρετητές και είναι σε μεγάλο βαθμό επεκτάσιμο. Ο έλεγχος της απόδοσης μιας εφαρμογής μπορεί να πραγματοποιηθεί όταν δέχεται μεγάλο φορτίο σε συνδυασμό με την πολλαπλή και ταυτόχρονη παρουσία χρηστών πάνω σε αυτή. Επίσης, έχει τη δυνατότητα να αναλύσει και να μετρήσει την απόδοση μιας διαδικτυακής (web) εφαρμογής καθώς και μια ποικιλία άλλων υπηρεσιών.

Ένα τυπικό τεστ με το Apache JMeter προϋποθέτει ότι θα δημιουργηθεί μια επανάληψη του τεστ (loop) και ένας αριθμός χρηστών. Η επανάληψη προσομοιώνει διαδοχικά αιτήματα προς έναν εξυπηρετητή με μια προεπιλεγμένη καθυστέρηση ενώ η ομάδα χρηστών σχεδιάζεται για την προσομοίωση παράλληλου φορτίου δεδομένων. Η καθυστέρηση επιτυγχάνεται με χρήση των timers οι οποίοι προσομοιώνουν τον χρόνο που χρειάζονται οι πραγματικοί χρήστες για να κάνουν την επόμενη κίνηση σε μια διαδικτυακή εφαρμογή. Το εργαλείο παρέχει μια διασύνδεση χρήστη με την οποία μπορεί να δημιουργηθεί ένα σχέδιο για το τεστ (test plan) που περιλαμβάνει μια σειρά λειτουργιών για να εκτελεστούν. Το πιο απλό σχέδιο για ένα τεστ συνήθως περιέχει τα ακόλουθα στοιχεία [29]:

- **Αριθμός χρηστών (Thread group):** χρησιμοποιείται για να προσδιοριστεί ο αριθμός των ενεργών χρηστών και η περίοδος έντονης δραστηριότητας (ramp up period). Κάθε thread προσομοιώνει έναν χρήστη και η περίοδο έντονης δραστηριότητας προσδιορίζει το χρόνο που χρειάζεται για να δημιουργηθούν όλοι οι χρήστες. Για παράδειγμα, τα 5 thread με ramp-up period 10 seconds θα χρειαστούν 2 δευτερόλεπτα ανάμεσα στη δημιουργία του κάθε thread. Ο μετρητής επανάληψης (loop count) ορίζει το χρόνο εκτέλεσης για ένα thread ενώ ο χρονοπρογραμματιστής (scheduler) καθορίζει την αρχή και το τέλος του χρόνου εκτέλεσης του τεστ.

- **Δείγματα αιτημάτων (Samplers):** είναι παραμετροποιήσιμα HTTP ή FTP αιτήματα προς έναν εξυπηρετητή. Σε αυτά μπορούν να δηλωθούν ο αριθμός της πόρτας, η μέθοδος του πρωτοκόλλου κ.α.
- **«Ακροατές» (Listeners):** χρησιμοποιούνται για την μετέπειτα επεξεργασία των δεδομένων του αιτήματος τα οποία έχουν την μορφή ενός αρχείου ή γραφήματος. Ένας από τους ακροατές με όνομα Summary Report που υπάρχει στο Apache Jmeter χρησιμοποιείται στις μετρήσεις της εργασίας.

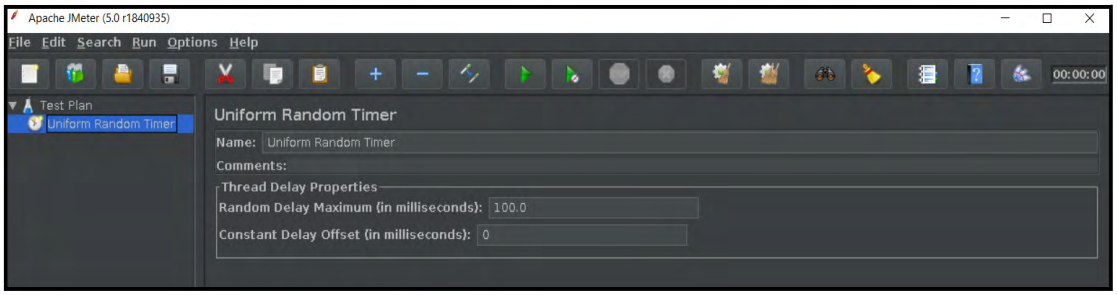
Το Apache Jmeter εκτελεί όλα τα δείγματα αιτημάτων το ένα μετά το άλλο χωρίς να υπάρχει καθυστέρηση ανάμεσα στις εκτελέσεις του κάθε εικονικού χρήστη. Αυτό δημιουργεί μη ρεαλιστικά τεστ επειδή κανένας χρήστης δεν ανοίγει μια ιστοσελίδα, κάνει κλικ σε έναν σύνδεσμο (link) και αμέσως μετά κάνει ένα νέο αίτημα. Συνήθως, ο χρήστης σκέφτεται, διαβάζει και γράφει σε μια διαδικτυακή εφαρμογή ενώ όλοι αυτοί οι χρόνοι που χρειάζονται είναι τα βασικά χαρακτηριστικά ενός σεναρίου για ένα τεστ. Στους ελέγχους απόδοσης αυτό ονομάζεται «think time» και στο Apache JMeter μπορεί να υλοποιηθεί με τους timers. Έτσι, με την χρήση τους μπορούν να δημιουργηθούν τυπικά τεστ τα οποία μπορούν να προσομοιώσουν περιπτώσεις που εμφανίζονται σε πραγματικές συνθήκες.

3.4.3 Οι timers του Apache JMeter

Οι timers οι οποίοι μπορούν να ρυθμίσουν την καθυστέρηση που δημιουργεί το Apache Jmeter στην αποστολή διαδοχικών αιτημάτων του ίδιου χρήστη βασίζονται σε διάφορες κατανομές πιθανοτήτων και μεταξύ αυτών είναι οι Uniform και Poisson. Στην στατιστική η uniform κατανομή είναι ένας όρος που χρησιμοποιείται για να περιγράψει μια μορφή κατανομής πιθανοτήτων όπου κάθε ενδεχόμενο αποτέλεσμα έχει την ίδια πιθανότητα να συμβεί. Η πιθανότητα αποτελεί μια σταθερά εφόσον κάθε μεταβλητή που χρησιμοποιείται έχει τις ίδιες πιθανότητες να είναι το αποτέλεσμα. Η Poisson κατανομή χρησιμοποιείται για να δείξει πόσες φορές ένα γεγονός ενδέχεται να συμβεί σε μια συγκεκριμένη χρονική περίοδο. Αυτή είναι μια κατανομή μέτρησης και συχνά χρησιμοποιείται για να γίνουν αντιληπτά ανεξάρτητα γεγονότα τα οποία συμβαίνουν με σταθερό ρυθμό μέσα σε μια ορισμένη χρονική περίοδο.

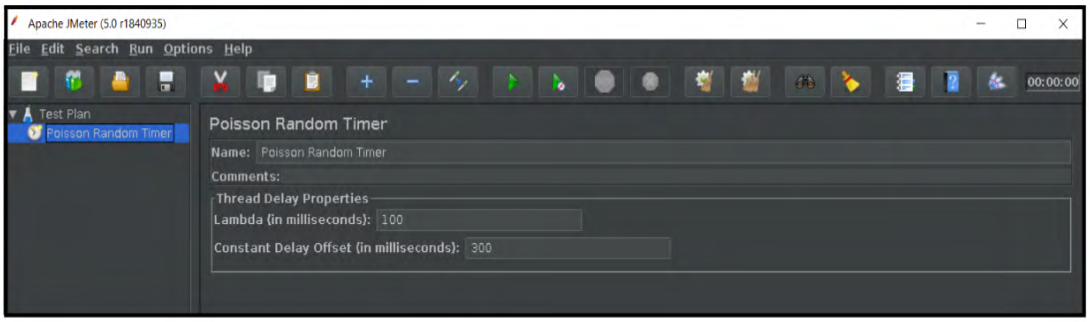
Ο Uniform Random Timer διακόπτει τον εικονικό χρήστη (thread) κατά έναν συντελεστή που προκύπτει από την επόμενη ψευδοτυχαία (pseudorandom) τιμή στο εύρος τιμών από `0.0`

μέχρι 1.0 (χωρίς αυτή την τιμή) που πολλαπλασιάζεται με το Random Delay Maximum και στο αποτέλεσμα προστίθεται το Constant Delay Offset. Το Random Delay Maximum είναι η μέγιστη καθυστέρηση που μπορεί να υπάρχει ανάμεσα στα αιτήματα του εικονικού χρήστη. Έτσι, στην προκαθορισμένη παραμετροποίηση που φαίνεται στην επόμενη εικόνα, ο timer θα διακόψει την αποστολή αιτημάτων από τον ίδιο χρήστη κατά έναν αριθμό milliseconds στο εύρος τιμών από 0 έως 99 milliseconds καθώς ο τύπος από τον οποίο προκύπτει αυτή η καθυστέρηση διαμορφώνεται ως εξής: 0.X * 100 + 0, όπου το X μπορεί να είναι ένας αριθμός από 1 έως και 9 [37].



Εικόνα 3.7 – Παραμετροποίηση Uniform Random Timer

Ο Poisson Random Timer χρησιμοποιεί τον επόμενο τυχαίο αριθμό μέσω της κατανομής Poisson που προκύπτει από την πρόσθεση της παραμέτρου λ ή «Lambda» με το Constant Delay Offset. Όταν χρησιμοποιούνται οι προκαθορισμένοι παράμετροι, δηλαδή 100 millisecond ως Lambda και 300 millisecond ως Constant Delay Offset θα δημιουργήσει μια καθυστέρηση στο εύρος από 375 ως 425 milliseconds περίπου όπως δείχνεται παρακάτω [37].



Εικόνα 3.8 - Παραμετροποίηση Poisson Random Timer

4. ΜΕΤΡΗΣΕΙΣ – ΣΥΖΗΤΗΣΗ

4.1 ΠΕΡΙΓΡΑΦΗ ΜΕΤΡΗΣΕΩΝ

Με δεδομένες τις ανάγκες των σύγχρονων δικτύων που φιλοξενούν τις εφαρμογές του διαδικτύου για την αποδοτικότερη υλοποίηση τους και για την καλύτερη αξιοποίηση της χωρητικότητας τους είναι σημαντικό να εξεταστεί η απόδοση τους σε διάφορες συνθήκες. Τέτοιες συνθήκες μπορεί να είναι μια υπερβολικά μεγάλη αύξηση της κίνησης σε ένα δίκτυο για κάποιο χρονικό διάστημα με αποτέλεσμα τόσο την μεγάλη καθυστέρηση μεταφοράς των δεδομένων μέσα σε αυτό όσο και την πολύ χαμηλή διεκπεραιωτικότητα/ρυθμαπόδοση του για την εξυπηρέτηση των χρηστών του.

Με βάση τα προηγούμενα, αυτό που θα ελεγχθεί και θα μετρηθεί ως προς την απόδοση στην παρούσα εργασία είναι η ταχύτητα με την οποία θα παίρνει ο χρήστης την απάντηση της επιλογής του από μια απλή web εφαρμογή που ονομάζεται WordPress εξετάζοντας δυο παραμέτρους/μετρικές του δικτύου των περιεκτών που περιέχουν τις μικροϋπηρεσίες της. Αυτές οι δυο μετρικές του συγκεκριμένου δικτύου είναι η καθυστέρηση μεταφοράς και η διεκπεραιωτικότητα/ρυθμαπόδοση του, οι οποίες θα εξεταστούν με τους δυο τρόπους δικτύωσης (bridge/host) περιεκτών στο Docker Compose όπου είναι εγκατεστημένη η εν λόγω εφαρμογή.

Η καθυστέρηση μεταφοράς που μετράει το Apache Jmeter είναι ο χρόνος από τη στιγμή ακριβώς πριν σταλεί το αίτημα μέχρι ακριβώς τη στιγμή που λαμβάνεται η πρώτη απόκριση. Έτσι, αυτός ο χρόνος περιλαμβάνει όλη τη διάρκεια της επεξεργασίας που απαιτείται για την συναρμολόγηση όλης της απάντησης αλλά και τη λήψη της πρώτης απόκρισης από το σύστημα που υπέβαλε το αίτημα. Όσον αφορά στην διεκπεραιωτικότητα/ρυθμαπόδοση αυτή μετριέται σε αιτήματα/μονάδα χρόνου συνήθως σε δευτερόλεπτα. Ο χρόνος εδώ υπολογίζεται από την αρχή του πρώτου αιτήματος μέχρι το τέλος του τελευταίου και περιλαμβάνει τις όποιες διακοπές/παύσεις ανάμεσα τους καθώς υποτίθεται ότι αναπαριστάνεται το φορτίο του συστήματος που βρίσκεται υπό έλεγχο. Ο τύπος υπολογισμού της είναι: Throughput = (number of requests) / (total time) [40].

Μέσα από τις δοκιμές θα πραγματοποιηθεί ένας έλεγχος απόδοσης ή ειδικότερα ένας έλεγχος φορτίου (load testing) με τη χρήση των προαναφερόμενων παραμέτρων/μετρικών οι οποίες θα βοηθήσουν να εξεταστεί η απόδοση της υποδομής που βρίσκεται η εφαρμογή. Για το σκοπό αυτό θα χρησιμοποιηθεί το Apache Jmeter το οποίο ύστερα από κατάλληλη παραμετροποίηση θα δημιουργήσει τα ανάλογα φορτία σύμφωνα με τα σενάρια που θα

υλοποιηθούν για τις δοκιμές. Τα αιτήματα που θα δημιουργήσουν τα προαναφερθέντα φορτία στην εφαρμογή θα γίνουν με υλοποίηση δυο διαφορετικών τρόπων (modes) δικτύωσης ανάμεσα στους δυο περιέκτες οι οποίοι συνιστούν την εν λόγω εφαρμογή.

Πιο συγκεκριμένα, θα μελετηθεί η διαφορά στην απόδοση της υποδομής της εφαρμογής που είναι υλοποιημένη με Docker Compose ανάμεσα σε αυτούς τους δυο διαφορετικούς τρόπους δικτύωσης των δυο περιεκτών που την αποτελούν. Οι δυο τρόποι που έχουν επιλεγεί είναι οι bridge και host επειδή αυτοί είναι οι προκαθορισμένοι τρόποι που μπορούν να χρησιμοποιηθούν στο Docker Compose για να συνδέσουν τους περιέκτες του με άλλα δίκτυα που δεν ανήκουν στον εξυπηρετητή τους. Τέλος, θα υπάρχει μια προκαθορισμένη χρονική καθυστέρηση μεταξύ των αιτημάτων από τον ίδιο αποστολέα για καλύτερη προσομοίωση με τις πραγματικές συνθήκες λειτουργίας.

Όπως έχει ήδη αναφερθεί ο προκαθορισμένος τρόπος δικτύωσης στο Docker Compose είναι ο bridge και οι μετρήσεις που έγιναν με αυτή την υλοποίηση αποτέλεσαν τη βάση για να γίνει η σύγκριση με εκείνες που έγιναν με τον host τρόπο δικτύωσης. Για τον host τρόπο δικτύωσης έγινε κάποια επιπλέον δικτυακή παραμετροποίηση έτσι ώστε να πραγματοποιηθεί τόσο η επικοινωνία μεταξύ των περιεκτών του Docker Compose όσο και της εφαρμογής που συνιστούν με τον έξω κόσμο. Στη συνέχεια, συγκρίνοντας τις δυο μεθόδους δικτύωσης μπόρεσαν να εξαχθούν συμπεράσματα για την επίδραση/επιβάρυνση (overhead) που έχει η καθεμία από αυτές στην δικτυακή απόδοση της υποδομής της δοκιμαστικής εφαρμογής στις δυο περιπτώσεις όσον αφορά στην καθυστέρηση μεταφοράς και τη διεκπεραιωτικότητα/ρυθμαπόδοση μεταξύ των δυο περιεκτών της.

Τα τέσσερα σενάρια που υλοποιήθηκαν περιλάμβαναν έναν αρχικό αριθμό 250 ταυτόχρονων εικονικών χρηστών (threads) που έκαναν αιτήματα στην εφαρμογή ο οποίος διπλασιαζόταν μετά από κάθε τεστ φτάνοντας τους 4000 χρήστες στο τελευταίο σενάριο μετρήσεων. Ο αριθμός των ανθρώπων που ασχολούνται με μια ιστοσελίδα δίνει μια καλή ένδειξη των δυνατοτήτων της όσον αφορά στον αριθμό των πετυχημένων αιτημάτων που υποβάλλουν σε αυτή. Εξάλλου είναι πιο εύκολο να συζητάμε, για παράδειγμα, για το επερχόμενο κύμα καταναλωτών τις ημέρες των Χριστουγέννων που επισκέπτονται μια ιστοσελίδα παρά για τον αριθμό των αιτημάτων που μπορεί να έρχονται κάθε δευτερόλεπτο σε αυτή.

Για να προσομοιωθούν οι πραγματικές συνθήκες λειτουργίας δημιουργήθηκε μια τεχνητή καθυστέρηση ανάμεσα στα αιτήματα του ίδιου χρήστη μέσω δυο κατανομών (distributions)

στατιστικών πιθανοτήτων της Uniform και της Poisson. Η επιλογή αυτών των δυο κατανομών στατιστικών πιθανοτήτων έγινε για να εξασφαλιστεί η όσο το δυνατόν μεγαλύτερη αξιοπιστία των μετρήσεων λόγω της διαφορετικής μαθηματικής λογικής που ακολουθούν στην παραγωγή της τυχαίας χρονικής καθυστέρησης μεταξύ των διαδοχικών αιτημάτων που γίνονται από τους χρήστες της εφαρμογής. Πιο συγκεκριμένα, με την κατανομή Uniform πραγματοποιήθηκαν περισσότερα αιτήματα ανά δευτερόλεπτο εφόσον ο χρόνος καθυστέρησης ανάμεσα στα αιτήματα του ίδιου χρήστη είναι από 0 έως 99 milliseconds ενώ της κατανομής Poisson ο αντίστοιχος χρόνος είναι από 375 έως 425 milliseconds. Έτσι, για το ίδιο χρονικό διάστημα των δεκαπέντε λεπτών το φορτίο στην πρώτη περίπτωση ήταν μεγαλύτερο. Επομένως, με αυτήν την επιλογή έγινε προσπάθεια να πραγματοποιηθεί μια ακόμη καλύτερη προσομοίωση των πραγματικών συνθηκών που υπάρχουν στη χρήση της υποδομής μιας διαδικτυακής εφαρμογής.

Η διάρκεια του κάθε τεστ ήταν 15 λεπτά και αυτά επαναλήφθηκαν αρκετές φορές για το κάθε σενάριο χωρίς να διαπιστωθούν σημαντικές διαφορές ανάμεσα τους ενώ σε περιπτώσεις που συνέβη αυτό κρατήθηκε ο μέσος όρος των μετρήσεων. Επίσης, τα τεστ στις περισσότερες περιπτώσεις δεν έγιναν το ένα μετά το άλλο ενώ όταν συνέβαινε αυτό μεσολαβούσε επανεκκίνηση της εικονικής μηχανής για να μην υπάρχει επίδραση ανάμεσα τους όσον αφορά κυρίως στη χρήση του επεξεργαστή και της μνήμης του εξυπηρετητή. Ακόμα, έγιναν διάφοροι έλεγχοι με εργαλεία του Linux που κάνουν έλεγχο στην χρησιμοποίηση των πόρων (επεξεργαστή, μνήμης) της εικονικής μηχανής έτσι ώστε να διαπιστωθεί η κατάσταση της πριν την εκκίνηση του κάθε τεστ. Με όλα τα παραπάνω έγινε προσπάθεια να εξασφαλιστεί στο μεγαλύτερο δυνατό βαθμό η ορθότητα και η ακεραιότητα των μετρήσεων.

Οι μετρήσεις με το Apache Jmeter πραγματοποιήθηκαν με βάση κάποιες πρακτικές για τον περιορισμό της χρήσης υπολογιστικών πόρων. Σύμφωνα με αυτές [36] για την εκκίνηση του κάθε τεστ αντί για το Graphical User Interface (GUI) χρησιμοποιήθηκε μέσα από τη γραμμή εντολών (CLI mode) του λειτουργικού συστήματος η εντολή: *jmeter -n -t test_name.jmx -l test_result.csv* [39]. Η εντολή εκτελέστηκε με τη βοήθεια του προγράμματος απομακρυσμένης πρόσβασης PUTTY, όπου με την παράμετρο -n ορίζεται η εκτέλεση του τεστ μέσω της γραμμής εντολών (CLI mode), με την -t καθορίζεται το όνομα του αρχείου που έχει τις ρυθμίσεις του τεστ (Test Plan) και με την -l προσδιορίζεται το όνομα του αρχείου των αποτελεσμάτων του τεστ (*test_result.csv*) το οποίο μπορεί να είναι σε μορφή JTL ή CSV. Ακόμα, κατά τη διάρκεια των εκτελέσεων των τεστ δεν έγινε καμία χρήση Listeners για όσο

το δυνατό λιγότερη επιβάρυνση σε υπολογιστικούς πόρους, όπως για παράδειγμα View Results Tree ή View Results in Table.

Η συνοπτική αναφορά (Summary Report) είναι ο Listener που χρησιμοποιήθηκε για τη διεκπεραιωτικότητα/ρυθμαπόδοση μετά από την εκτέλεση των τεστ και την εξαγωγή των αρχείων csv με τα αποτελέσματα των μετρήσεων. Τα αρχεία αυτά μελετήθηκαν και παραμετροποιήθηκαν για την όσο το δυνατό πιο ορθή μέτρηση της καθυστέρησης μεταφοράς στα διάφορα σενάρια. Επίσης, χρησιμοποιήθηκε μια επέκταση του Apache Jmeter με το όνομα Report Dashboard η οποία είναι για την παραγωγή γραφικών απεικονίσεων και στατιστικών από ένα τεστ. Η συγκεκριμένη επέκταση παρέχει διάφορες μετρικές που μπορούν να μελετηθούν ενώ αυτές που χρησιμοποιήθηκαν στις παρούσες μετρήσεις για την καθυστέρηση μεταφοράς ήταν η Latencies Over Time και η Connect Time Over Time. Μέσα από επεξεργασία των γραφημάτων των συγκεκριμένων μετρικών κατέστη εφικτό να εξαχθούν καινούργια γραφήματα με τους μέσους χρόνους των καθυστερήσεων απόκρισης της υποδομής.

4.2 ΠΕΡΙΟΡΙΣΜΟΙ

Υπάρχουν κάποιοι περιορισμοί όσον αφορά στη δικτύωση των περιεκτών στο Docker και ειδικότερα στον οδηγό δικτύου host ο οποίος λειτουργεί μόνο σε εξυπηρετητές Linux1 Επιπλέον, όταν χρησιμοποιείται αυτός ο οδηγός για ένα δίκτυο περιεκτών δεν υπάρχει απομόνωση αυτού του δικτύου από εκείνο του εξυπηρετητή τους λόγω του ότι δεν αποκτά κάθε περιέκτης την δικιά του διεύθυνση δικτύου αλλά χρησιμοποιεί εκείνη που έχει ο εξυπηρετητής που τους φιλοξενεί. Για παράδειγμα, όταν ένας περιέκτης τρέχει μια εφαρμογή που εκτελείται στην πόρτα 80 η εφαρμογή αυτή είναι διαθέσιμη εξωτερικά από τον περιέκτη στην πόρτα 80 του εξυπηρετητή [32]. Έτσι, από τη μία δεσμεύεται η εκάστοτε πόρτα από τον περιέκτη που τη χρησιμοποιεί με κίνδυνο της διένεξης της με μια ίδια πόρτα στον ίδιο τον εξυπηρετητή ή σε έναν άλλο περιέκτη από την άλλη περιορίζεται ο αριθμός των περιεκτών που μπορούν να εκτελούνται σε αυτόν τον εξυπηρετητή.

Το Docker compose δεν είναι παραδοσιακά προσανατολισμένο σε παραγωγικά υπολογιστικά περιβάλλοντα αλλά σε εκείνα της ανάπτυξης και του ελέγχου εφαρμογών. Ωστόσο, σε κάθε καινούργια έκδοση του γίνεται προσπάθεια να δημιουργηθούν νέες λειτουργίες που θα είναι πιο κοντά σε ένα παραγωγικό περιβάλλον. Επίσης, το εργαλείο αυτό δεν μπορεί να βασιστεί σε εξισορροπητές φορτίου (load balancers) ενώ κάθε ανάπτυξη μιας νέας εφαρμογής περιλαμβάνει χρόνο εκτός λειτουργίας της (downtime) ο οποίος μπορεί να είναι σύντομος αλλά πολύ κρίσιμος σε περίπτωση που αυτή έχει πολλούς χρήστες [35].

Το Docker Compose έχει τη δυνατότητα με μια εντολή να ξεκινήσει ή να σταματήσει πολλούς περιέκτες μαζί πράγμα το οποίο δεν είναι εφικτό με τυπικές εντολές του Docker. Ωστόσο, τα δίκτυα ενός και μόνο εξυπηρετητή Docker που υποστηρίζει είναι χρήσιμα για μικρές εφαρμογές γενικότερα. Παρόλα αυτά, να διευκρινιστεί ότι αν το παραγωγικό περιβάλλον μιας εφαρμογής σημαίνει τη χρήση ενός και μόνο εξυπηρετητή ο οποίος θα περιλαμβάνει σε περιέκτες όλα όσα χρειάζονται για να λειτουργήσει η εν λόγω εφαρμογή τότε μπορεί πολύ εύκολα να χρησιμοποιηθεί το Docker Compose για την υλοποίηση του.

Το Apache JMeter που χρησιμοποιήθηκε για τις μετρήσεις τρέχει μέσα στο ίδιο εικονικό μηχάνημα με την δοκιμαστική εφαρμογή και αυτό έγινε για δύο λόγους. Αρχικά, λόγω περιορισμένων πόρων θα ήταν προβληματική η επιλογή να αφιερωθεί μια ολόκληρη εικονική μηχανή στο Jmeter επειδή θα έπρεπε να βρεθεί άλλη μια για την εγκατάσταση της δοκιμαστικής εφαρμογής που χρησιμοποιείται για τις μετρήσεις. Έπειτα, η συγκεκριμένη τοποθέτηση του Apache JMeter αφαιρεί την πιθανή καθυστέρηση που θα υπήρχε στο δίκτυο ανάμεσα στην εικονική μηχανή του Jmeter και αυτή της εφαρμογής. Έτσι, η μη μετρήσιμη καθυστέρηση λόγω δικτύου θα επηρέαζε πιθανότατα σε απρόβλεπτο βαθμό τα αποτελέσματα των δοκιμών.

4.3 ΑΠΟΤΕΛΕΣΜΑΤΑ

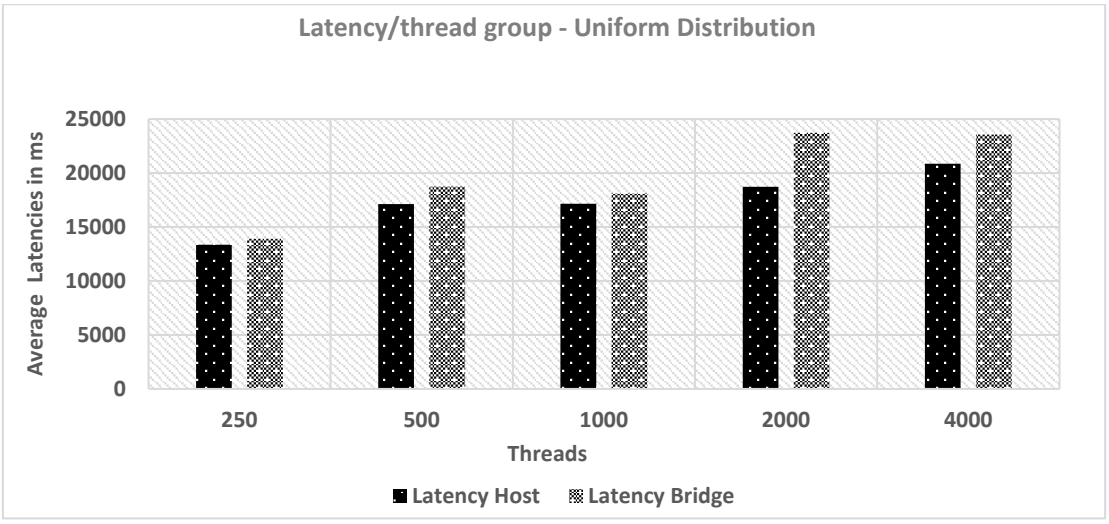
Σε αυτό το κεφάλαιο παρουσιάζονται και αναλύονται τα τελικά αποτελέσματα των μετρήσεων που πραγματοποιήθηκαν. **Εδώ είναι σημαντικό να αναφερθεί ότι οι χρόνοι των καθυστερήσεων απόκρισης προέκυψαν από την αφαίρεση του χρόνου σύνδεσης (connect time) από τον χρόνο της καθυστέρησης μεταφοράς που μετράει το Apache Jmeter**. Ο χρόνος σύνδεσης είναι ο χρόνος που απαιτείται για να εγκατασταθεί μια σύνδεση ανάμεσα σε δυο συστήματα και η παραπάνω αφαίρεση έγινε επειδή το Apache Jmeter ενσωματώνει αυτό τον χρόνο μέσα στην καθυστέρηση μεταφοράς που μετράει.

Ένα άλλο που πρέπει να αναφερθεί είναι ότι στην καθυστέρηση μεταφοράς συμπεριλαμβάνεται και ο χρόνος επεξεργασίας (processing time) του αιτήματος ο οποίος είναι το χρονικό διάστημα που χρειάζεται ένα σύστημα για να επεξεργαστεί ένα αίτημα που του γίνεται. Αν θεωρηθεί ότι αυτός ο χρόνος μπορεί να επηρεαστεί από αλλαγές στον κώδικα της δοκιμαστικής εφαρμογής ή στην βάση δεδομένων που χρησιμοποιεί ή ακόμα και από βελτιώσεις/αναβαθμίσεις στο υλικούς πόρους του συστήματος τότε 7δεν είναι σημαντική η επίδραση του στα αποτελέσματα των μετρήσεων. Ακόμα, είναι σημαντικό να επισημανθεί ότι αυτά τα αποτελέσματα προέκυψαν από την ανάλυση των επιτυχημένων αιτημάτων, δηλαδή αυτών που δεν εμφάνισαν λάθη κατά τις δοκιμές.

Οι παρακάτω πίνακες με τα αντίστοιχα διαγράμματα παρουσιάζουν τους μέσους όρους των καθυστερήσεων απόκρισης της υποδομής της εφαρμογής με τους δυο διαφορετικούς τρόπους δικτύωσης του Docker Compose και για τις δυο κατανομές στατιστικών πιθανοτήτων. Ο μέσος όρος είναι μια μέτρηση που επιτρέπει να έχουμε μια σαφή εικόνα για την απόδοση ενός συστήματος. Όσο μεγαλύτεροι είναι ο συγκεκριμένοι μέσοι όροι τόσο περισσότερος χρόνος χρειάζεται για την ολοκλήρωση της αποστολής των απαντήσεων στα αιτήματα.

Πίνακας 4-1 - Καθυστέρηση μεταφοράς στην Uniform κατανομή στατιστικών πιθανοτήτων

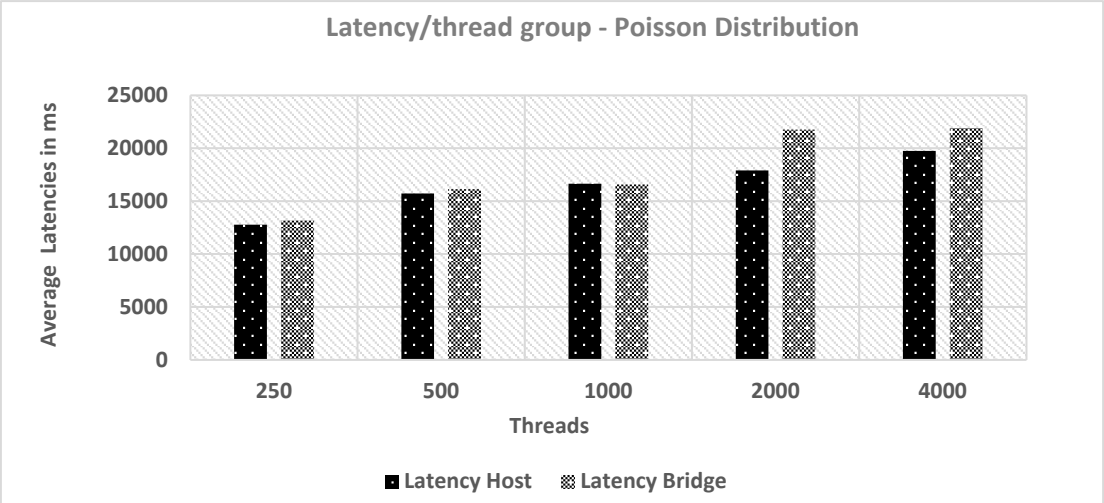
Uniform Distribution					
Average Latency (ms)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	13901,863	18730,450	18081,920	23724,090	23553,127
Network Host mode	13357,708	17124,406	17142,882	18724,322	20859,661



Εικόνα 4.1 - Γράφημα καθυστέρηση μεταφοράς στην Uniform κατανομή στατιστικών πιθανοτήτων

Πίνακας 4-2 - Καθυστέρηση μεταφοράς στην Poisson κατανομή στατιστικών πιθανοτήτων

Poisson Distribution					
Average Latency (ms)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	13163,659	16128,920	16563,125	21755,505	21900,382
Network Host mode	12778,799	15725,380	16644,900	17899,342	19736,435



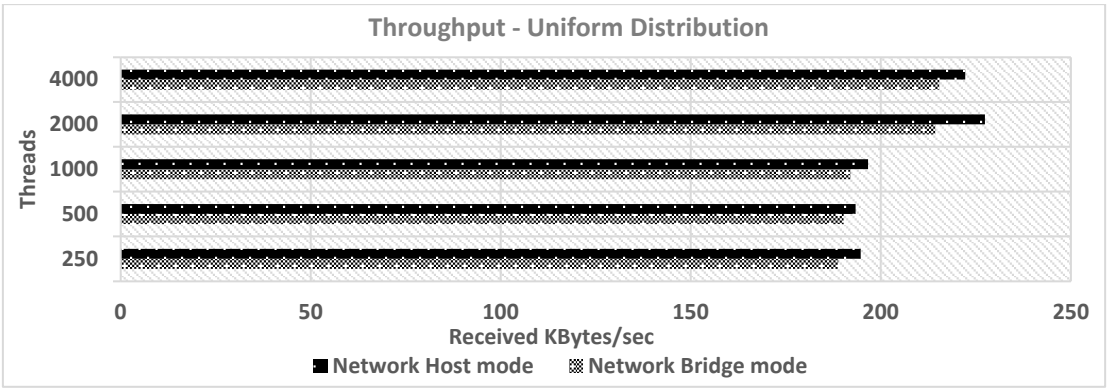
Εικόνα 4.2 - Γράφημα καθυστέρηση μεταφοράς στην Poisson κατανομή στατιστικών πιθανοτήτων

Με βάση τα προηγούμενα εξάγονται χρήσιμα συμπεράσματα για την δικτυακή απόδοση των διαφορετικών σεναρίων υλοποίησης των δοκιμών. Αυτό που παρατηρείται στις δυο κατανομές στατιστικών πιθανοτήτων είναι ότι σε όλες τις περιπτώσεις η μέση καθυστέρηση απόκρισης των επιτυχημένων αιτημάτων είναι μικρότερη στον host τρόπο δικτύωσης με τις πιο μεγάλες διαφορές να εντοπίζονται στα σενάρια με τους 2000 και 4000 εικονικούς χρήστες (threads). Αυτό δείχνει ότι ο host τρόπος δικτύωσης ανταποκρίνεται ακόμα καλύτερα στο μεγάλο αριθμό ταυτόχρονων χρηστών στο σύστημα είτε τα αιτήματα γίνονται σε μικρά χρονικά διαστήματα όπως στην Uniform κατανομή είτε σε μεγαλύτερα χρονικά διαστήματα όπως στην Poisson κατανομή.

Στα παρακάτω διαγράμματα με τους αντίστοιχους πίνακες εμφανίζεται η ρυθμαπόδοση σε KB/sec των επιτυχημένων αποκρίσεων της δοκιμαστικής υποδομής της εφαρμογής με τους δυο διαφορετικούς τρόπους δικτύωσης στο Docker Compose για τις δυο κατανομές στατιστικών πιθανοτήτων.

Πίνακας 4-3 - Ρυθμαπόδοση στην Uniform κατανομή στατιστικών πιθανοτήτων

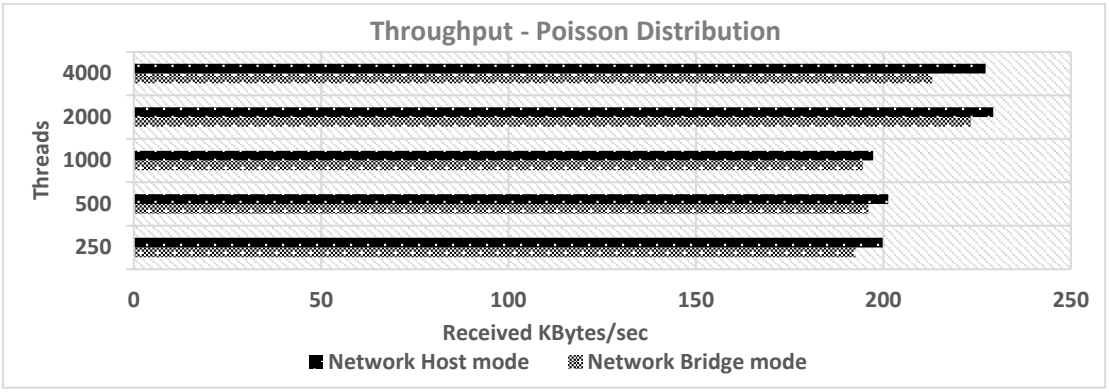
Uniform Distribution					
Throughput in KBytes/sec (successful responses)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	188,84	190,21	192,03	214,35	215,38
Network Host mode	194,72	193,37	196,67	227,38	222,22



Εικόνα 4.3 - Γράφημα ρυθμαπόδοσης στην Uniform κατανομή στατιστικών πιθανοτήτων

Πίνακας 4-4 - Ρυθμαπόδοση στην Poisson κατανομή στατιστικών πιθανοτήτων

Poisson Distribution					
Throughput in KBytes/sec (successful responses)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	192,58	195,98	194,53	223,34	213,02
Network Host mode	199,80	201,28	197,29	229,32	227,24



Εικόνα 4.4 - Γράφημα ρυθμαπόδοσης στην Poisson κατανομή στατιστικών πιθανοτήτων

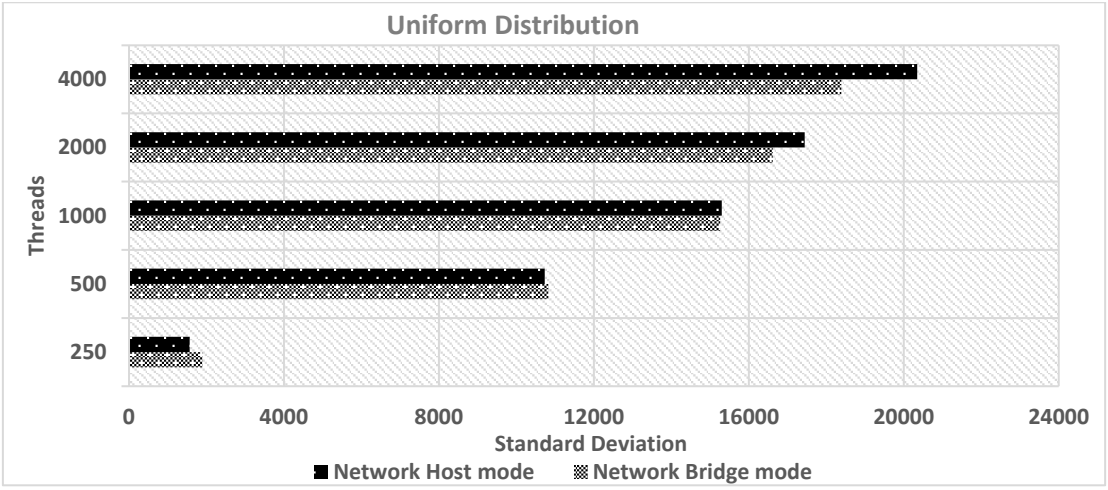
Αυτό που παρατηρείται τόσο στην κατανομή Uniform όσο και στην κατανομή Poisson είναι ότι σε όλες τις περιπτώσεις η ρυθμαπόδοση σε KB/sec των επιτυχημένων αποκρίσεων είναι μεγαλύτερη στον host τρόπο δικτύωσης. Αυτό που αξίζει να επισημανθεί είναι ότι στην ρυθμαπόδοση ο host τρόπος δικτύωσης ανταποκρίνεται ακόμα καλύτερα με 2000 και 4000 ταυτόχρονους χρήστες στο σύστημα είτε τα αιτήματα γίνονται πολύ γρήγορα όπως στην Uniform κατανομή είτε πιο αργά όπως στην Poisson κατανομή. Το ίδιο παρατηρήθηκε και στα διαγράμματα της καθυστέρησης μεταφοράς που παρουσιάστηκαν πιο πάνω.

Αυτή η διαφορά στην ρυθμαπόδοση αλλά και στην καθυστέρηση μεταφοράς είναι πιθανό να οφείλεται στην μεγαλύτερη πολυπλοκότητα σύνδεσης που υπάρχει στον bridge τρόπο δικτύωσης επειδή ο κάθε περιέκτης αποκτά την δική του διεύθυνση δικτύου σε σχέση με τον host τρόπο όπου όλοι οι περιέκτες έχουν τη διεύθυνση δικτύου του εξυπηρετητή τους. Έτσι, στον host τρόπο δικτύωσης οι περιέκτες για να επικοινωνήσουν χρησιμοποιούν τη διευθυνσιοδότηση του εξυπηρετητή τους σε αντίθεση με τον bridge τρόπο που χρησιμοποιεί την δικιά του διευθυνσιοδότηση. Αυτό έχει ως αποτέλεσμα να μην χρησιμοποιείται ο μηχανισμός DNS service discovery αντιστοίχισης της ονομασίας των μικροϋπηρεσιών που εκτελούνται στις διευθύνσεις δικτύου τους σε αντίθεση με τον bridge τρόπο δικτύωσης.

Η τυπική απόκλιση (Standard Deviation) που είναι μια πολύ σημαντική μετρική θεωρήθηκε χρήσιμη για τους σκοπούς των δοκιμών. Με αυτή γίνεται αντιληπτό το πως οι χρόνοι απόκρισης των αιτημάτων διασκορπίζονται γύρω από το μέσο χρόνο απόκρισης και όσο πιο μικρή τιμή έχει τόσο πιο μεγάλη είναι η αξιοπιστία αυτού που ελέγχεται. **Ο χρόνος απόκρισης (response/elapsed time) είναι ο χρόνος από τη στιγμή ακριβώς πριν σταλεί το αίτημα σε ένα σύστημα μέχρι τη στιγμή ακριβώς της λήψης όλης της απάντησης από αυτό.** Μια υποδομή νέφους είναι ουσιώδες να μπορεί να προσφέρει στους πελάτες σταθερότητα στην απόδοση ώστε η συμπεριφορά των εμπλεκόμενων συστημάτων να είναι όσο το δυνατό πιο αναμενόμενη. Όταν υπάρχει μικρή τυπική απόκλιση τότε όλες οι μετρήσεις δεν κυμαίνονται μακριά από το μέσο όρο και έτσι η απόδοση είναι πιο σταθερή.

Πίνακας 4-5 - Τυπική απόκλιση στην Uniform κατανομή στατιστικών πιθανοτήτων

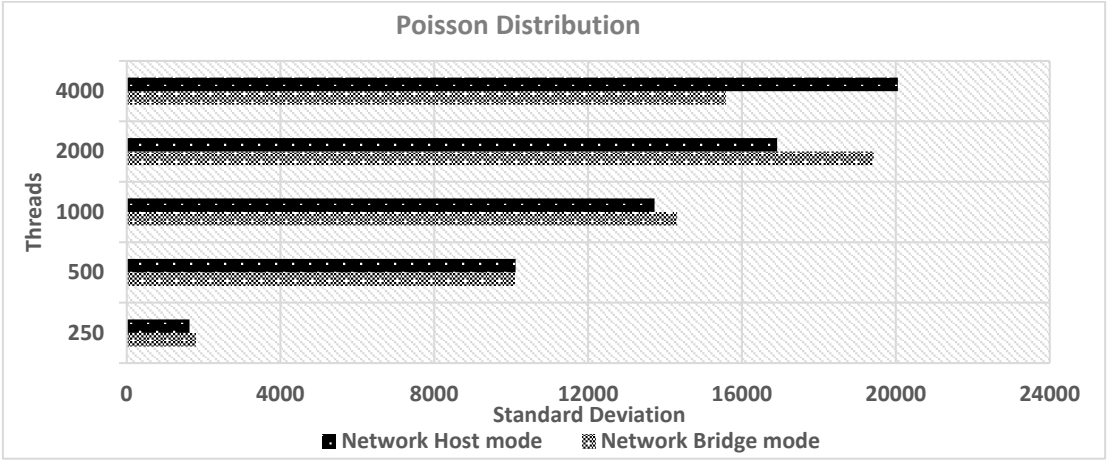
Uniform Distribution					
Standard Deviation (ms)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	1891,35	10825,73	15260,64	16615,40	18389,51
Network Host mode	1566,36	10732,23	15302,28	17442,67	20346,51



Εικόνα 4.5 - Γράφημα τυπικής απόκλισης στην Uniform κατανομή στατιστικών πιθανοτήτων

Πίνακας 4-6 - Τυπική απόκλισης στην Poisson κατανομή στατιστικών πιθανοτήτων

Poisson Distribution					
Standard Deviation (ms)	**250 (threads)**	**500 (threads)**	**1000 (threads)**	**2000 (threads)**	**4000 (threads)**
Network Bridge mode	1803,15	10095,59	14311,41	15576,54	15576,54
Network Host mode	1632,73	10111,03	13724,26	16911,89	20051,37



Εικόνα 4.6 - Γράφημα τυπικής απόκλισης στην Poisson κατανομή στατιστικών πιθανοτήτων

Αυτό που διαπιστώνεται από την εξέταση των τυπικών αποκλίσεων στις δυο στατιστικές κατανομές είναι ότι παρουσιάζουν μεγάλες διαφορές ανάμεσα στους δυο τρόπους δικτύωσης στα σενάρια δοκιμών με τους περισσότερους χρήστες, δηλαδή με τους 2000 και 4000. Αυτό που παρατηρείται ανάμεσα σε αυτά τα δυο σενάρια είναι ότι μόνο με τους 2000 εικονικούς χρήστες στην κατανομή Poisson υπερτερεί ο host τρόπος δικτύωσης ενώ στα υπόλοιπα ο bridge τρόπος. Επομένως, μπορεί να συναχθεί το συμπέρασμα ότι και στις δυο στατιστικές κατανομές με τους δυο τρόπους δικτύωσης υπάρχει μια διακύμανση της σταθερότητας στην απόδοση της υποδομής με ένα μικρό προβάδισμα στον bridge τρόπο δικτύωσης γενικότερα.

5. ΕΠΙΛΟΓΟΣ

5.1 ΣΥΝΟΨΗ

Η υπολογιστική νέφους κερδίζει συνεχώς έδαφος τα τελευταία χρόνια και γίνεται απαραίτητο εργαλείο για μικρές και μεγάλες εταιρείες, όχι μόνο στον κλάδο της πληροφορικής αλλά και σε πολλούς άλλους. Το γεγονός αυτό δημιουργεί την ανάγκη για την ταχεία βελτίωση της και αυτό μπορεί να συμβεί με την καλύτερη αξιοποίηση ενός μεγάλου συνόλου στοιχείων είτε σε υποδομές είτε σε πρωτόκολλα που χρησιμοποιούνται σε υπηρεσίες της υπολογιστικής νέφους.

Ένας τομέας πάνω στον οποίο γίνονται συνεχώς έρευνες με αποτέλεσμα τη σταδιακή βελτίωση είναι η δικτύωση των συστημάτων του νέφους. Αυτή αποτελεί μία από τις πιο σημαντικές λειτουργίες για μια υποδομή υπολογιστικού νέφους και με δεδομένη την αύξηση των δικτυακών απαιτήσεων των υπηρεσιών που προσφέρονται από αυτό θεωρείται ένας από τους βασικούς πυλώνες της υπολογιστικής νέφους και το κλειδί για την επιτυχία της.

Στα πλαίσια αυτής της εργασίας μελετήθηκε το πεδίο της δικτύωσης των Docker περιεκτών μιας εφαρμογής σε μια υποδομή του νέφους ως προς την αποδοτικότερη αξιοποίηση των δυνατοτήτων αυτής της υποδομής. Η υλοποίηση της έγινε με τη βοήθεια μιας εικονικής μηχανής της πλατφόρμας Okeanos όπου πάνω σε αυτή εγκαταστάθηκε το Docker Compose για την δοκιμαστική εφαρμογή μαζί με το Apache Jmeter για την διεξαγωγή των μετρήσεων.

Πραγματοποιήθηκαν αρκετά πειράματα πάνω σε αυτήν την υποδομή ώστε να εξαχθούν χρήσιμα συμπεράσματα και να αξιολογηθεί στον καλύτερο δυνατό βαθμό η πειραματική διάταξη. Τέλος, οι μετρήσεις που πραγματοποιήθηκαν συμπεριέλαβαν όσο το δυνατό περισσότερες παραμέτρους που σχετίζονται με αυτό που πραγματικά μπορεί να συμβαίνει σε πραγματικές καταστάσεις λειτουργίας.

Στη συνέχεια, αφού εντοπίστηκαν κάποια πιθανά σημεία βελτίωσης της απόδοσης της δικτυακής κίνησης ανάμεσα στους περιέκτες της εφαρμογής στο Docker Compose έγιναν οι απαραίτητες παραμετροποιήσεις για την λειτουργία της υποδομής και στους δυο τρόπους δικτύωσης ή αλλιώς οδηγούς δικτύου (bridge/host) που προσφέρονται από το Docker Compose. Πιο συγκεκριμένα, στο bridge τρόπο δικτύωσης δεν χρειάστηκε κάτι ιδιαίτερο αλλά στον host τρόπο έπρεπε να γίνουν κάποιες αλλαγές στο αρχείο ανάπτυξης της εφαρμογής για να μπορέσει να λειτουργήσει. Τελικά, αυτές οι δυο μέθοδοι για τη δικτυακή σύνδεση των περιεκτών μετρήθηκαν και αξιολογήθηκαν ακριβώς στις ίδιες συνθήκες όσον αφορά στον

αριθμό των χρηστών οι οποίοι έκαναν ίδιου τύπου αιτήματα στην ίδια χρονική διάρκεια στον εξυπηρετητή.

Μια συνολική αποτίμηση είναι ότι προέκυψαν κάποια ενδιαφέροντα συμπεράσματα μέσα από τη μελέτη της απόδοσης των προκαθορισμένων τρόπων δικτύωσης του Docker Compose σε διάφορα σενάρια φορτίου. Αρκετά προγραμματιστικά εργαλεία και μέθοδοι δοκιμάστηκαν και χρησιμοποιήθηκαν για να προκύψουν τελικά εκείνα που υιοθετήθηκαν με βάση τα πλεονεκτήματα και μειονεκτήματα τους. Η υλοποίηση των προσφερόμενων δικτυακών τεχνικών στο Docker Compose έδωσε ένα μικρό προβάδισμα στην μέθοδο που χρησιμοποιείται ο host οδηγός δικτύου και μπορεί να δώσει την κατεύθυνση ή να αποτελέσει τη βάση για περαιτέρω μελέτη.

5.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Με δεδομένο ότι η τοπολογία πάνω στην οποία πραγματοποιήθηκαν οι δοκιμές καθώς και τα εργαλεία υποστήριξης τους είναι έτοιμα θα είχε ενδιαφέρον να επεκταθούν οι μετρήσεις σε διαφορετικούς τύπους φορτίου, δηλαδή πέρα από τα απλά αιτήματα που χρησιμοποιήθηκαν σε αυτήν την εργασία, για να διαπιστωθεί αν η δικτυακή απόδοση της υποδομής είναι περίπου ίδια ή σχετίζεται μόνο με το συγκεκριμένο είδος αιτημάτων.

Αυτό που θα είχε ερευνητικό ενδιαφέρον το οποίο αναφέρθηκε σε προηγούμενο κεφάλαιο αλλά δεν εξετάσθηκε στη συγκεκριμένη εργασία είναι η μελέτη των συμβιβασμών ανάμεσα στην απόδοση και στην απομόνωση των δικτύων των περιεκτών με σκοπό την καλύτερη αξιοποίηση των διατιθέμενων πόρων από τους παρόχους του υπολογιστικών υπηρεσιών σε συνδυασμό με την ικανοποίηση των χρηστών τους.

Επιπλέον, τα αποτελέσματα των μετρήσεων που έγιναν ίσως να φανούν σημαντικά επειδή θα μπορούσαν να ληφθούν υπόψη στη δημιουργία εφαρμογών οι οποίες βασίζονται σε μεγάλο βαθμό στην δικτυακή απόδοση. Ακόμα, θα είχε ενδιαφέρον να διερευνηθεί η δικτυακή απόδοση των περιεκτών σε μια συστοιχία από εξυπηρετητές που φιλοξενούν μια υπηρεσία η οποία μπορεί να υλοποιηθεί με το αποκαλούμενο Docker Swarm σε διάφορους υποστηριζόμενους τρόπους δικτύωσης του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L. M., Netto, M. A., Toosi, A. N., Rodriguez, M. A., Llorente, I. M., Vimercati, S. D., Samarati, P., Milojicic, D., Varela, C., Bahsoon, R., Assuncao, M. D., … Shen, H. (2019). A manifesto for future generation cloud computing. ACM Computing Surveys, 51(5), 1-38. https://doi.org/10.1145/3241737

[2] Al-Said Ahmad, A., & Andras, P. (2019). Scalability analysis comparisons of cloud-based software services. Journal of Cloud Computing, 8(1). https://doi.org/10.1186/s13677-019-0134-y

[3] Nair, S. K., & Novak, D. C. (2007). A traffic shaping model for optimizing network operations. European Journal of Operational Research, 180(3), 1358-1380. https://doi.org/10.1016/j.ejor.2006.04.036

[4] Kozhirbayev, Z., & Sinnott, R. O. (2017). A performance comparison of container-based technologies for the cloud. Future Generation Computer Systems, 68, 175-182. https://doi.org/10.1016/j.future.2016.08.025

[5] Suo, K., Zhao, Y., Chen, W., & Rao, J. (2018). An analysis and empirical study of container networks. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. https://doi.org/10.1109/infocom.2018.8485865

[6] Yassine, A., Rahimi, H., & Shirmohammadi, S. (2015). Software defined network traffic measurement: Current trends and challenges. IEEE Instrumentation & Measurement Magazine, 18(2), 42-50. https://doi.org/10.1109/mim.2015.7066685

[7] Gu, L., Guan, J., Wu, S., Jin, H., Rao, J., Suo, K., & Zeng, D. (2019). CNTC: A container aware network traffic control framework. Green, Pervasive, and Cloud Computing, 208-222. https://doi.org/10.1007/978-3-030-19223-5_15

[8] Herbein, S., Dusia, A., Landwehr, A., McDaniel, S., Monsalve, J., Yang, Y., Seelam, S. R., & Taufer, M. (2016). Resource management for running HPC applications in container clouds. Lecture Notes in Computer Science, 261-278. https://doi.org/10.1007/978-3-319-41321-1_14

[9] Duan, Q. (2017). Cloud service performance evaluation: Status, challenges, and opportunities – a survey from the system modeling perspective. Digital Communications and Networks, 3(2), 101-111. https://doi.org/10.1016/j.dcan.2016.12.002

[10] Bertolino, A., Angelis, G. D., Gallego, M., García, B., Gortázar, F., Lonetti, F., & Marchetti, E. (2019). A systematic review on cloud testing. ACM Computing Surveys (CSUR), 52(5), pp. 1-42. https://doi.org/10.1145/3331447

[11] Ghalwash, H., & Huang, C. (2019). QoS for SDN-based fat-tree networks. Lecture Notes in Networks and Systems, 691-705. https://doi.org/10.1007/978-3-030-12385-7_49

[12] Mesquita, C., Cavalca, U., Pereira, A. C., & Carrano, E. G. (2013). A traffic shaping optimization methodology for web systems. Proceedings of the 19th Brazilian symposium on Multimedia and the web - WebMedia '13. https://doi.org/10.1145/2526188.2526190

[13] Curiel, M., & Pont, A. (2018). Workload generators for web-based systems: Characteristics, current status, and challenges. IEEE Communications Surveys & Tutorials, 20(2), 1526-1546. https://doi.org/10.1109/comst.2018.2798641

[14] Hong, J., Dreibholz, T., Schenkel, J.A., & Hu, J.A. (2019). An Overview of Multi-cloud Computing. In: Barolli, L., Takizawa, M., Xhafa, F., Enokido, T. (eds) Web, Artificial Intelligence and Network Applications. WAINA 2019. Advances in Intelligent Systems and Computing, vol 927. Springer, Cham. https://doi.org/10.1007/978-3-030-15035-8_103

[15] Yao, J., Maleki Shoja, B., & Tabrizi, N. (2019). An overview of cloud computing testing research. Cloud Computing – CLOUD 2019, 303-313. https://doi.org/10.1007/978-3-030-23502-4_21

[16] Nachiyappan, S., & Justus, S. (2015). Cloud testing tools and its challenges: A comparative study. Procedia Computer Science, 50, 482-489. https://doi.org/10.1016/j.procs.2015.04.018

[17] Mentz, L. L., Loch, W. J., & Koslovski, G. P. (2020). Comparative experimental analysis of docker container networking drivers. 2020 IEEE 9th International Conference on Cloud Networking (CloudNet). https://doi.org/10.1109/cloudnet51028.2020.9335811

[18] Jiang, H., & Sun, X. (2017). Understanding networking capacity management in cloud computing. Lecture Notes in Computer Science, 516-526. https://doi.org/10.1007/978-3-319-52015-5_53

[19] Barik, R. K., Lenka, R. K., Rao, K. R., & Ghose, D. (2016). Performance analysis of virtual machines and containers in cloud computing. 2016 International Conference on Computing, Communication and Automation (ICCCA). https://doi.org/10.1109/ccaa.2016.7813925

[20] Sotomayor, J. P., Allala, S. C., Alt, P., Phillips, J., King, T. M., & Clarke, P. J. (2019). Comparison of runtime testing tools for Microservices. 2019 IEEE 43rd Annual Computer

Software and Applications Conference (COMPSAC). https://doi.org/10.1109/compsac.2019.10232

[21] Murthy, M.S.N., Suma, V. (2014). A Study on Cloud Computing Testing Tools. In: Satapathy, S., Avadhani, P., Udgata, S., Lakshminarayana, S. (eds) ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol I. Advances in Intelligent Systems and Computing, vol 248. Springer, Cham. https://doi.org/10.1007/978-3-319-03107-1_66

[22] Sahu, Y., & Pateriya, R. K. (2013). Cloud computing overview with load balancing techniques. International Journal of Computer Applications, 65(24), pp. 40-44.

[23] Yan, M., Sun, H., Liu, X., Deng, T., & Wang, X. (2014). Delivering web service load testing as a service with a global cloud. Concurrency and Computation: Practice and Experience, 27(3), 526-545. https://doi.org/10.1002/cpe.3246

[24] Qusef, A., Issa, L., Ayoubi, E., & Murad, S. (2019). Challenges and opportunities in cloud testing. Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems - DATA '19. https://doi.org/10.1145/3368691.3368706

[25] Abbasi, U., Bourhim, E. H., Dieye, M., & Elbiaze, H. (2019). A performance comparison of container networking alternatives. IEEE Network, 33(4), 178-185. https://doi.org/10.1109/mnet.2019.1800141

[26] Dusia, A., Yang, Y., & Taufer, M. (2015). Network quality of service in docker containers. 2015 IEEE International Conference on Cluster Computing. https://doi.org/10.1109/cluster.2015.96

[27] Smith, R. (2017). Docker Orchestration. Packt Publishing.

[28] Poulton, N. (2017). Docker Deep Dive. Independently published.

[29] Nevedrov, D. (2006, February 8). Oracle Technical Articles. Retrieved July 19, 2022, from

https://www.oracle.com/technical-resources/articles/enterprise-architecture/jmeter-performance-testing-part1.html

[30] Costa, V., Girardon, G., Bernardino, M., Machado, R., Legramante, G., Neto, A., Basso, F. P., & De Macedo Rodrigues, E. (2020). Taxonomy of performance testing tools. Proceedings of the 35th Annual ACM Symposium on Applied Computing. https://doi.org/10.1145/3341105.3374006.

[31] Kiskis, D. L., & Shin, K. G. (1996). A synthetic workload for a distributed real-time system. Real-Time Systems, 11(1), 5-18. https://doi.org/10.1007/bf00365518...

[32] Use host networking. (2022, August 1). Docker Documentation. Retrieved August 3, 2022, from https://docs.docker.com/network/host/

[33] Networking overview. (2022, July 19). Docker Documentation. Retrieved July 21, 2022, from https://docs.docker.com/network/

[34] Networking in Compose. (2022, July 23). Docker Documentation. Retrieved July 25, 2022, from https://docs.docker.com/compose/networking/

[35] Overview. (2022, July 26). Docker Documentation. Retrieved July 28, 2022, from https://docs.docker.com/compose/

[36] Apache JMeter - User's Manual: Best Practices. (n.d.). Retrieved July 15, 2022, from https://jmeter.apache.org/usermanual/best-practices.html

[37] Tikhanski, D. (2020, July 8). A Comprehensive Guide to Using JMeter Timers. Blazemeter. Retrieved October 15, 2022, from

https://www.blazemeter.com/blog/jmeter-timer

[38] Use macvlan networks. (2022, August 1). Docker Documentation. Retrieved August 1, 2022, from https://docs.docker.com/network/macvlan/

[39] Apache JMeter - User's Manual: Getting Started. (n.d.). Retrieved July 31, 2022, from https://jmeter.apache.org/usermanual/get-started.html

[40] Apache JMeter - User's Manual: Glossary. (n.d.-b). Retrieved July 20, 2022, from https://jmeter.apache.org/usermanual/glossary.html

[41] Docker overview. (2022, August 1). Docker Documentation. Retrieved August 1, 2022, from https://docs.docker.com/get-started/overview/