ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ & ΠΑΡΑΓΩΓΗΣ

Τεχνητή νοημοσύνη και εφαρμογές στα δίκτυα υπολογιστών νέας γενιάς

από

Ρούσσου Θεοχάρη

Διπλωματική Εργασία

Επιβλέπουσα: Λελίγκου Αικατερίνη-Ελένη

Τμήμα Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής

Πανεπιστήμιο Δυτικής Αττικής

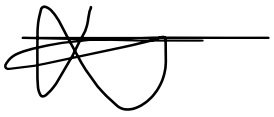
Ιούνιος 2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Ρούσσου Θεοχάρη του Λάσκαρη, με αριθμό μητρώου 71445800, φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι: «Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής 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».

Η Δηλούσα



Μέλη Εξεταστικής Επιτροπής

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	**ΟΝΟΜΑΤΕΠΩΝΥΜΟ**	**ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ**
1	ΛΕΛΙΓΚΟΥ ΑΙΚΑΤΕΡΙΝΗ ΕΛΕΝΗ	Eleni Aikaterini Leligkou Digitally signed by Eleni Aikaterini Leligkou Date: 2022.07.31 20:48:24 +03'00'
2	ΠΑΛΛΗΣ ΕΥΑΓΓΕΛΟΣ	Evangelos Pallis Digitally signed by Evangelos Pallis Date: 2022.08.01 08:10:59 +03'00'
3	ΜΙΧΑΗΛ ΠΑΠΟΥΤΣΙΔΑΚΗΣ	Michail Papoutsidakis Digitally signed by Michail Papoutsidakis Date: 2022.08.01 14:59:59 +03'00'

Πίνακας περιεχομένων

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Περίληψη

Η βέλτιστη χρήση των πόρων υποδομής είναι ένα ιδιαίτερα επιθυμητό , αλλά παράλληλα πολύπλοκο έργο για τους παρόχους δικτυακού service. Αυτό συμβαίνει διότι η βέλτιστη ποσότητα τέτοιων πόρων είναι μία συνάρτηση από διάφορες παραμέτρους, όπως είναι η επιθυμητή / συμφωνημένη ποιότητα υπηρεσιών (QoS) , τα χαρακτηριστικά / προφίλ του service, ο όγκος εργασίας και ο κύκλος ζωής της.

Η θεμελίωση πλαισίων (frameworks) που προβλέπουν τη δυναμική εγκατάσταση και την τοποθέτηση υπηρεσιών και δικτυακών λειτουργιών συμβάλλει περαιτέρω στη μείωση της αποτελεσματικότητας των παραδοσιακών μεθόδων κατανομής πόρων.

Στην έρευνα αυτή, αντιμετωπίζεται αυτό το πρόβλημα με την ανάπτυξη ενός μηχανισμού, ο οποίος αρχικά πραγματοποιεί σκιαγράφηση της υπηρεσίας, και έπειτα εκτελεί μία πρόβλεψη των πόρων που θα χρειαστούν ώστε να οδηγηθεί στο επιθυμητό QoS για κάθε νέα υπηρεσία που αναπτύσσεται.

Τα κύρια στοιχεία της προσέγγισης είναι τα παρακάτω: α) τη συλλογή δεδομένων και από τα τρία επίπεδα των εγκατεστημένων υποδομών (hardware/ υλισμική , virtual/ εικονική, service) αντί για ένα μόνο επίπεδο, για να παρέχεται μία καθαρότερη εικόνα σε πιθανές διακοπές του συστήματος, β) η μελέτη γνωστών εφαρμογών που βασίζονται στην υποδομή των containers ακολουθώντας την αρχιτεκτονική των microservices και γ) τη χρήση μίας διαδικασίας ανάλυσης δεδομένων που χρησιμοποιεί ένα σύνολο αλγορίθμων μηχανικής μάθησης και εκτελεί ακριβείς προβλέψεις για τους πόρους που απαιτούνται για μελλοντικά αιτήματα υπηρεσιών.

Διερευνάται η απόδοση του προτεινόμενου framework χρησιμοποιώντας την εφαρμογή ανοικτού κώδικα ώστε να εξεταστεί η περίπτωση ενός συμπλέγματος Hadoop. Τα αποτελέσματα δείχνουν ότι εκτελώντας έναν μικρό αριθμό δοκιμών είναι εφικτό να αξιολογηθούν τα κύρια σημεία διακοπής του συστήματος και ταυτόχρονα να επιτευχθούν ακριβείς προβλέψεις των πόρων που θα χρειαστούν για μελλοντικά αιτήματα.

Abstract

The optimal use of infrastructure resources is a highly desirable, but at the same time complex task for network service providers. This is because the optimal amount of such resources is a function of several parameters, such as the desired / agreed quality of service (QoS), the characteristics / profile of the service, the workload and its life-cycle.

The establishment of frameworks that predicts the dynamic establishment and placement of service and network functions further contributes to reducing the effectiveness of traditional resource allocation methods.

In this research, this problem is addressed by developing a mechanism, which first performs service profiling, and then a prediction of the resources that will be needed to achieve the desired Qos for each new deployed service.

The main elements of this approach are the following: a) the collection of data of all three layers of the deployed infrastructure (hardware, virtual, service), instead of a single layer, to provide a clearer picture of the possible system break points, b) a study of well-known container based applications following the microservice architecture, and c) the use of a data analysis process that uses a set of machine learning algorithms and performs accurate predictions of the required resources for future service requests.

The performance of the proposed framework is investigated using the open-source implementation to consider the case of a Hadoop cluster. The results show that by running a small number of tests it is possible to assess the main system break points and at the same time achieve accurate predictions of the resources that will be needed for future requests.

Αναγνωρίσεις

Αρχικά, θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου στην κύρια επιβλέπουσα Ελένη - Αικατερίνη Λελίγκου για τη συνεχή υποστήριξη της διπλωματικής μου διατριβής και για την υπομονή καθώς και για τα κίνητρα που μου έδωσε. Με τις διαρκείς συζητήσεις, η κυρία Λελίγκου με βοήθησε να καταλάβω ποια βήματα έπρεπε να ακολουθήσω για να ολοκληρώσω με επιτυχία τη διατριβή μου.

Εκτός από την κύρια επιβλέπουσα μου, θα ήθελα να ευχαριστήσω και τον κύριο Ουζουνίδη , ακαδημαϊκό υπότροφο, και τον κύριο Καρκαζή, Επ. Καθηγητή του τμήματος Μηχανικών πληροφορικής και Υπολογιστών, για την ουσιαστική καθοδήγηση τους η οποία με βοήθησε σε όλο το χρόνο της έρευνας και συγγραφής αυτής της εργασίας. Η ουσιαστική γνώση στο ερευνητικό θέμα της διατριβής μου ήταν πολύ σημαντική. Πάνω από όλα, οφείλω να εκφράσω τις ειλικρινείς μου ευχαριστίες για την εμπιστοσύνη που μου έδειξαν.

Θα ήθελα επίσης να εκφράσω τη βαθύτατη εκτίμηση στους συναδέλφους μου που διάβασαν και σχολίασαν τη διατριβή μου.

Ευχαριστώ τους γονείς μου που με στήριξαν για την ολοκλήρωση της διπλωματικής μου διατριβής. Επίσης για την υπομονή, την ενθάρρυνση και την υποστήριξη.

Ειδικότερα, θέλω να ευχαριστήσω τον άντρα μου γιατί πίστεψε ότι μπορούσα να ολοκληρώσω αυτή τη διατριβή και με στήριξε όταν ένιωθα απογοητευμένη.

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή στο αντικείμενο

Στο πλαίσιο της εικονικοποίησης των λειτουργιών δικτύου (NFV), ένα δικτυακό service(NS),(π.χ. δρομολογητής, τείχος προστασίας, διακομιστής προσωρινής μνήμης κ.λπ.) αποτελείται από μία αλυσίδα διασυνδεδεμένων εικονικών λειτουργιών δικτύου(VNF). Αυτές οι εικονικές λειτουργίες μπορούν να υπάρχουν είτε σε μία, είτε σε πολλαπλές υποδομές VNF (NFVI), παρέχοντας στους χειριστές του δικτύου σημαντική ευελιξία στη διαμόρφωση.

Λαμβάνοντας υπόψιν τη δυναμική φύση των δικτυακών service (NSs), η διαχείριση του κύκλου ζωής τους δεν είναι απλή διαδικασία. Ο λόγος είναι ότι είτε λειτουργεί για σύντομο χρονικό διάστημα μόνο για την παροχή μίας συγκεκριμένης υπηρεσίας, είτε επειδή οι απαιτήσεις του χρήστη μπορεί να αλλάξουν, το δικτυακό service (NS) πρέπει να προσαρμοστεί στις νέες απαιτήσεις εκτελώντας ενέργειες αύξησης ή μείωσης της δυναμικής. Για να αντιμετωπίσει τις προκλήσεις που προκύπτουν σε αυτό το περιβάλλον με υψηλή δυναμικότητα, το ETSI έχει ορίσει αυτοματοποιημένους μηχανισμούς για την αυτόματη κλιμάκωση και θεραπεία του μηχανισμού βάσει κανόνων στο αρχιτεκτονικό πλαίσιο NFV-Management and Orchestrator (MANO).

Ωστόσο, η δυναμική και βέλτιστη κατανομή πόρων παραμένει ένας πολύ ενδιαφέρον ερευνητικός τομέας , καθώς η δυναμική διαχείρισή του δεν μπορεί να βασίζεται αποκλειστικά σε απλοϊκές προσεγγίσεις που βασίζονται σε κανόνες, καθιστώντας απαραίτητη την υιοθέτηση πιο εξελιγμένων εργαλείων , συμπεριλαμβανομένων των αλγορίθμων Τεχνητής Νοημοσύνης (AI)/ Μηχανικής Μάθησης (ML).

Μία από τις πιο βασικές προκλήσεις είναι η μεγιστοποίηση του αριθμού των υποστηριζόμενων υπηρεσιών των χρηστών, εξασφαλίζοντας παράλληλα το προκαθορισμένο QoS για όλη τη διάρκεια ζωής του δικτυακού service. Μία άλλη βασική παράμετρος που δημιουργεί μεγάλο ενδιαφέρον στους παρόχους είναι η ποιότητα

εμπειρίας (QoE), η οποία αποτελεί, στην πράξη, την ποιότητα που βιώνει ο χρήστης και επηρεάζεται κατά κύριο λόγο από την αντιληπτή χρονική καθυστέρηση υπολογισμένη από τον χρήστη στο cloud.

Στην πραγματικότητα, η βέλτιστη κατανομή των πόρων είναι ένα ζήτημα που απασχολεί τους παρόχους δικτυακού service (SP) στην αρχή της δομής του cloud και καθώς οι εικονικές τεχνολογίες εξελίσσονται από εικονικές μηχανές (VM) σε containers και microservices το πρόβλημα βελτιστοποίησης έχει γίνει ακόμα πιο περίπλοκο.

Παραδοσιακά, για την αντιμετώπιση αυτής της πρόκλησης, οι μηχανικοί του δικτύου κατανέμουν σημαντικά μεγαλύτερο αριθμό πόρων από το απαιτούμενο, γεγονός που οδηγεί άμεσα σε σπατάλη πόρων , καθώς δεν αξιοποιούνται στο έπακρο. Προκειμένου να αποφευχθεί αυτή η σπατάλη, είναι επιθυμητό ο μηχανικός δικτύου να διαθέσει τον ακριβή αριθμό των απαιτούμενων πόρων, οδηγώντας, ιδανικά, σε μηδενική υποχρησιμοποίηση.

Για την επίτευξη αυτού του στόχου, είναι απαραίτητη μια ακριβής πρόβλεψη των πόρων που απαιτούνται σε ολόκληρο τον κύκλο ζωής του δικτυακού service. Συγκεκριμένα, αυτή η πρόβλεψη θα καθορίσει το ακριβές σημείο στο οποίο ο αριθμός των πόρων που διατίθενται διασφαλίζει τόσο τη συμμόρφωση με τη συμφωνημένη Service Level Agreement(SLA) όσο και τη μηδενική υποχρησιμοποίηση με την οποία, θα αποφευχθεί η υπερκατανομή και η υποκατανομή των πόρων που μπορούν να οδηγήσουν σε παραβίαση του SLA.

Ο ορισμός αυτού του «κρίσιμου σημείου» είναι ένα περίπλοκο πρόβλημα, καθώς απαιτεί ένα εκτεταμένο προφίλ δικτυακού service για μεγάλο αριθμό διαμορφώσεων , την παρακολούθηση ενός τεράστιου όγκου μετρήσεων συστήματος από διαφορετικές πηγές , όπως bare metal , εικονικές μηχανές, containers, services, δίκτυα κ.λπ, και τέλος, χρειάζεται μία ολοκληρωμένη πλατφόρμα που μπορεί να παρακολουθεί την ανάλυση δεδομένων για όλες τις μετρήσεις και να εξάγει τις πιο σημαντικές που μπορεί, ενδεχομένως, να οδηγήσουν σε υποβάθμιση του QoS. Το ζήτημα του προσδιορισμού του «κρίσιμου σημείου» γίνεται όλο και πιο δύσκολο, καθώς δημιουργούνται καθημερινά νέοι τύποι service και οι αποφάσεις για τους πόρους και την ανάθεσή τους πρέπει να εκτελούνται ταχύτερα και σε πιο λεπτομερές επίπεδο.

Επίσης, άλλη μία σημαντική πρόκληση είναι η ποικιλία της υποδομής που χρησιμοποιείται στα σύγχρονα κέντρα δεδομένων. Για παράδειγμα, ο hardware εξοπλισμός (π.χ. servers, διακόπτες, διακοσμητές κ.λπ) και οι τεχνολογίες εικονικοποίησης που βασίζονται σε virtual machines και containers (π.χ. VMware ESXi, Openstack, AWS, linux containers, docker swarm, Kubernetes κ.λπ) συνδυάζονται ώστε να παρέχουν μία υποδομή υπολογιστικών και δικτυακών πόρων. Ως συνέπεια, οι πάροχοι υπηρεσιών διαδικτύου χρειάζονται μία γενικευμένη λύση, η οποία μπορεί να παρακολουθεί, να αναλύει και να διαχειρίζεται τους κατανεμημένους πόρους σε διάφορα επίπεδα της εγκεκριμένης μεθόδου υλοποίησης.

Σε αυτό το περίπλοκο περιβάλλον, η μηχανική μάθηση μπορεί να βοηθήσει αποτελεσματικά τους μηχανικούς δικτύου να ελαχιστοποιήσουν την απόσταση από το κρίσιμο σημείο λειτουργίας. Επιπλέον μπορεί να εξάγει αυτόματα τα προφίλ των νέων services καθώς δημιουργούνται, αξιοποιώντας μόνο έναν μικρό αριθμό των ανεπτυγμένων ρυθμίσεων παραμέτρων , ελαχιστοποιώντας έτσι το συνολικό χρόνο της δημιουργίας του προφίλ. Αυτά τα προφίλ χρησιμοποιούνται στη συνέχεια για να προβλέψουν τον ακριβή αριθμό των πόρων που θα χρειαστούν για τη διασφάλιση της ποιότητας του service καθ’ όλη τη διάρκεια του κύκλου ζωής του.

Κατά συνέπεια, μία αυτοματοποιημένη διαδικασία δημιουργίας service προφίλ και πρόβλεψης απόδοσης πόρων μπορεί να προσφέρει βελτιωμένη χρήση των πόρων υποδομής, χωρίς να χρειάζεται να θυσιαστεί η ποιότητα ή η διαθεσιμότητα των υπηρεσιών. Το service προφίλ και η ακριβής πρόβλεψη της απόδοσης είναι ένα πρόβλημα το οποίο επεκτείνεται σε διάφορες κατηγορίες υπηρεσιών, όπως είναι τα δίκτυα, η ασφάλεια, τα big data, η αποθήκευση, η εξομοίωση κ.λπ και η λύση που προτείνεται σε αυτή την έρευνα μπορεί εύκολα να προσαρμοστεί σε κάθε κατηγορία υπηρεσιών , καθώς βασίζεται σε τεχνολογία ανοικτού κώδικα και είναι αρθρωτό, ενώ παρουσιάζει μεγάλη ευελιξία ανάπτυξης.

Συγκεκριμένα, σε αυτή την έρευνα , σχεδιάστηκε και αναπτύχθηκε ένα καινοτόμο framework ώστε να δοθεί λύση στο πρόβλημα της δημιουργίας του service προφίλ και να προβλεφθούν τα «κρίσιμα σημεία» του συστήματος, εστιάζοντας σε περίπλοκες υπηρεσίες που εκτελούνται ταυτόχρονα με τα containers.

Η προτεινόμενη λύση είναι μία προσέγγιση με πολλά επίπεδα, καθώς παρακολουθεί και αναλύει δεδομένα και από τα τρία στάδια υλοποίησης , δηλαδή το υλικό , το εικονικό και το service επίπεδο. Επίσης, χρησιμοποιεί τεχνολογία ανοικτού κώδικα , ενώ υποστηρίζει διάφορους τύπους τεχνολογιών εικονικοποίησης, παρέχοντας υψηλό επίπεδο ευελιξίας. Επιπλέον, είναι σε θέση να δημιουργήσει προφίλ διαφόρων τύπων service εκτελώντας δοκιμές “white box” με την συγκέντρωση των μετρήσεων της επίδοσης απευθείας από τις υπό δοκιμή υπηρεσίες ή “black box” δοκιμές ,χρησιμοποιώντας γνωστά εργαλεία συγκριτικής αξιολόγησης (benchmark) (π.χ. jMeter, Apache benchmark, κ.λπ).

Κατά τη διάρκεια της ανάλυσης των δεδομένων η προτεινόμενη εφαρμογή αξιοποιεί το ευρύ φάσμα των αλγορίθμων μηχανικής μάθησης που εκτείνεται από polynomial regression έως και deep neural networks, προσαρμόζοντας τον πιο ακριβή αλγόριθμο για το υπό εξέταση service. Τέλος, η υποψήφια αρχιτεκτονική είναι σε θέση να ορίσει τα πιο εμφανή κρίσιμα σημεία κατά τη διάρκεια δημιουργίας του service προφίλ, ενώ κατά τη φάση ανάλυσης δεδομένων μπορεί να εξαγάγει τα κρυφά κρίσιμα σημεία. Αυτό είναι εφικτό καθώς μπορεί να εκτελεί προβλέψεις για τις ρυθμίσεις παραμέτρων που δεν ήταν δυνατό να εξεταστούν κατά τη διάρκεια δημιουργίας του προφίλ.

Για να εξεταστεί η σκοπιμότητα της προτεινόμενης προσέγγισης δημιουργήθηκε ένα περιβάλλον προστατευμένης εκτέλεσης (sandbox) στο οποίο αναπτύχθηκε ένα Hadoop cluster με σκοπό να δημιουργεί service προφίλ και να εκτελεί τις προβλέψεις, παρακολουθώντας έναν εκτεταμένο αριθμό μετρήσεων του συστήματος (π.χ. CPU usage, memory usage, service throughput, κ.λπ) από τρία επίπεδα, δηλαδή το υλικό, το εικονικό και το service.

Σε αυτή την εργασία, τα καθορισμένα επίπεδα δεν σχετίζονται με τα σχετικά επίπεδα του OSI. Το Hadoop επιλέχθηκε σαν υπηρεσία δοκιμών αφού συνδυάζει κάποια από τα επιθυμητά χαρακτηριστικά, είναι , δηλαδή, μία ευρέως διαδεδομένη λύση για προβλήματα big data, είναι ανοικτού κώδικα, είναι εύκολη η ενσωμάτωσή του με τις υπάρχουσες υποδομές και παρέχει έναν σημαντικό αριθμό δοκιμαστικών σημείων αναφοράς που μπορούν να αξιοποιηθούν ώστε να υπολογιστεί η απόδοση της αναπτυγμένης υποδομής.

1.2 Δομή Εργασίας

Τα υπόλοιπα περιεχόμενα των κεφαλαίων έχουν ώς εξής:

Στο Κεφάλαιο 2 αναλύεται η σχετική έρευνα σε αυτόν τον τομέα με σκοπό να δημιουργηθεί σφαιρική άποψη του προβλήματος που αντιμετωπίζεται.

Στο Κεφάλαιο 3 παρουσιάζεται η αρχιτεκτονική τριών επιπέδων.

Στο Κεφάλαιο 4 αναλύονται, αρχικά, τα προτεινόμενα πεδία δοκιμών που έχουν αναπτυχθεί ώστε να ελεγχθεί η σκοπιμότητα της λύσης και επίσης εξετάζονται τα ερευνητικά ευρήματα από τις δοκιμές που πραγματοποιήθηκαν.

Τέλος, στο Κεφάλαιο 5 αναφέρονται τα συμπεράσματα της εργασίας, καθώς και προτείνονται πιθανές μελλοντικές έρευνες.

Κεφάλαιο 2

Σχετική έρευνα

Σε προηγούμενες μελέτες, έχουν εισαχθεί πολλές διαφορετικές εφαρμογές για το προφίλ και την εκτέλεση προβλέψεων σε κρίσιμες μετρήσεις συστήματος. Στις πηγές [3-6] το προφίλ πραγματοποιήθηκε σε δοκιμαστικά περιβάλλοντα που αναπτύσσονται στο cloud, ενώ στις [7-18] εξετάζονται εφαρμογές στο πλαίσιο του NFV, επειδή εστιάζουν στην ανάπτυξη και επικύρωση των VNF.

Συγκεκριμένα, στο [4] αξιοποιούνται διάφορες μέθοδοι μηχανικής μάθησης για την εκτέλεση προβλέψεων χρησιμοποιώντας μόνο ένα υποσύνολο από τις πιθανές ρυθμίσεις παραμέτρων, δοκιμάζοντας το Tera Sort, το PageRank και τον αλγόριθμο εύρεσης συντομότερων διαδρομών (Single Source Shortest Path), επιτυγχάνοντας υψηλό επίπεδο ακρίβειας παρά τη χρήση περιορισμένου αριθμού παραμέτρων.

Έπειτα, στο [5] χρησιμοποιούνται διάφοροι αλγορίθμοι ,όπως active learning, για να πραγματοποιηθεί ανάλυση της απόδοσης του framework, ενώ στο [6] γίνεται χρήση ενός decision tree ώστε να δοκιμαστεί ο χώρος ανάπτυξης και να βελτιωθεί η συνολική ακρίβεια της πρόβλεψης. Συγκεκριμένα, στο [6] κάθε ρύθμιση ανάπτυξης αντιπροσωπεύει ένα σύνολο παραμέτρων που επηρεάζουν άμεσα τη συνολική απόδοση της εφαρμογής.Για παράδειγμα, στο Hadoop Wordcount, η βασική μέτρηση είναι ο χρόνος της εκτέλεσης ενώ ο χώρος ανάπτυξης περιλαμβάνει τους κόμβους Yarn, τον αριθμό των πυρήνων ανά κόμβο, τη μνήμη ανά κόμβο και το μέγεθος των δεδομένων.

Ένας σημαντικός περιορισμός στην υλοποίηση των [3-5] είναι η αξιοποίηση των δεδομένων μόνο από το επίπεδο της εφαρμογής , ενώ οι μετρήσεις της απόδοσης σε υλισμικό και εικονικό επίπεδο παραμένουν ανεξερεύνητες. Αξίζει να αναφερθεί ότι ο μηχανισμός που παρουσιάζεται στο [6] μπορεί να παρακολουθεί δεδομένα από το υλικό επίπεδο και το επίπεδο της εφαρμογής, αν και το επίπεδο εικονικοποίησης παραμένει ανεξερεύνητο.

Στην συγκεκριμένη εργασία, παρουσιάζεται μία εφαρμογή τριών επιπέδων, η οποία παρακολουθεί και αναλύει δεδομένα από όλα τα επίπεδα με σκοπό να εκτελέσει το service προφίλ και την απόδοση των προβλέψεων χρησιμοποιώντας έναν αριθμό διαφορετικών αλγορίθμων μηχανικής μάθησης, που εκτείνεται από linear regression μέχρι και νευρωνικά δίκτυα. Η προτεινόμενη αρχιτεκτονική έχει την ικανότητα να προσδιορίσει τα εμφανή οριακά σημεία του συστήματος για τα τρία επίπεδα καθ' όλη τη διάρκεια της δημιουργίας των προφίλ και για τα κρυμμένα επίπεδα κατά τη διάρκεια της ανάλυσης. Αυτό είναι πολύ σημαντικό από την οπτική των παρόχων δικτυακού service, καθώς μπορεί να βοηθήσει στην πρόληψη της σπατάλης των πόρων μέσω της σωστής χρήσης τους και μπορεί να αποφευχθεί μία παραβίαση του SLA μέσω της υπερχρησιμοποίησης πόρων. Επιπλέον, η προσέγγιση αυτή μπορεί να συλλέξει και να αναλύσει μετρήσεις όχι μόνο από hypervisor αλλά και από containers (Kubernetes). Μπορεί επίσης να ενσωματωθεί εύκολα με τα ήδη υπάρχοντα MANO frameworks ενώ χρησιμοποιεί εργαλεία ανοικτού κώδικα για να ελαχιστοποιήσει το συνολικό κόστος CapEx. Για να προσδιοριστεί περαιτέρω η εργασία, παρακάτω συνοψίζονται οι πιο σημαντικές λεπτομέρειες της σχετικής βιβλιογραφίας στο πλαίσιο του NFV στον Πίνακα **Πίνακας *1***.

Πίνακας 1: Σχετική έρευνα στο πλαίσιο NVF.

Ref.	Testing Service/App	Examined Metrics	Examined Layers	ML Algorithms
[6]	Spark k-means, Spark Bayes, Hadoop Wordcount, MongoDB	Execution time, throughput	service mainly	Regression Trees
[7]	Clearwater, Snort, Suricata	Successful call rate, CPU, memory and network usage, Packet processing speed vs. traffic	virtual	-
[8]	single Video Encoder (VE),	Number of CPU	virtual	-

	blackbox profiling scenario, entire service chain	cores, CPU time		
[9]	Intrusion Detection System (IDS)	CPU, memory, traffic rate, packet loss	virtual	Rule based
[10]	SIPp prober	Avg. transaction rate, CPU usage	virtual	-
[11]	A chain of three VNFs	Throughput vs. CPU time	virtual	Plug-in arbitrary analysis scripts
[12]	The service Media Gateway (MG) composed of various VNFs	CPU load, network drops, rejected calls, latency	virtual	Stochastic Gradient Descent regressor (SGDR)
[13]	MME, S-GW, HSS, PCRF, PDN-GW	Quality of Decisions (QoD)	virtual	Q-Learning approach
[14]	7 (such as Suricata-IPS, NAT, Tcpdump, Firewall, Netsniffer-ng)	Many from each layer, such as CPU, memory usage, Disk read/write I/O requests and bytes etc.	service virtual physical	Various criteria for behavior classification
[15]	HSS, MME, PGW-U-SGWU, SGW-C-PGW-C, INET-GW and eNB	CPU, CPU cache, memory, bandwidth, Disk	virtual	Decision tree-based multilabel classification technique
[16]	virtual router, switch, firewall and cache server	CPU usage, packet loss, cache response time	virtual	Linear Regression, k-NN, Interpolation, ANN, Curve Fitting
[17]	virtual firewall, (pfSense), virtual streaming server	CPU usage, packet loss, lag ratio	virtual	Support Vector Regression, Random Forest, Gaussian Process, k-NN,

	(Nginx)			Interpolation Method, Curve fitting
[18]	Squid cache, Nginx proxy	CPU, total delay, # of VNF instances	virtual	Linear regression, support vector regression, decision trees, ensemble learning, neural networks
[19]	Scalable monitoring framework	NS, VNFs, NFVI, SDN controllers	virtual	Rule based
This work	Hadoop	CPU usage, disk usage, memory usage, throughput, average I/O rate	service virtual physical	Linear Regression, Polynomial Regression, Decision Tree, Random Forest, Support Vector Regression, k-NN, and Rule based

Κεφάλαιο 3

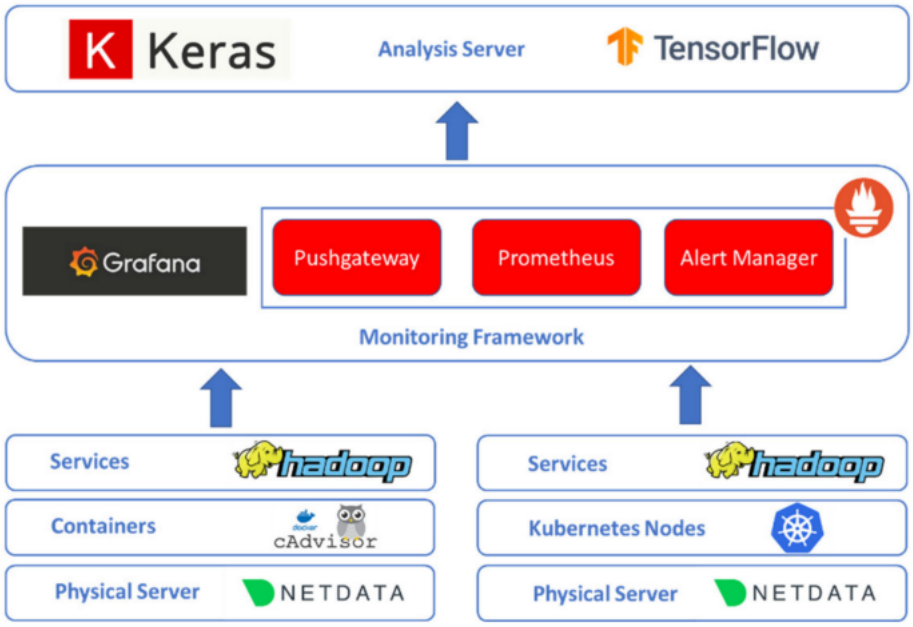
Αρχιτεκτονική

3.1 Εισαγωγή στην αρχιτεκτονική

Παρά το γεγονός πως η βέλτιστη κατανομή των πόρων έχει ήδη διερευνηθεί στη λογοτεχνία, λείπει μία πλήρης, χαμηλού κόστους λύση για τη συλλογή και την ανάλυση πληροφοριών από όλους τους τύπους στόχων παρακολούθησης, κατάλληλη για ενσωμάτωση με τα ήδη υπάρχοντα frameworks.

Το προτεινόμενο framework (**Εικόνα 1**) συνδυάζει εργαλεία παρακολούθησης και ανάλυσης τελευταίας τεχνολογίας με σκοπό να παρέχει ένα αξιόπιστο και οικονομικά αποδοτικό framework για κατανομή πόρων, αποτελούμενο από δύο κύρια στοιχεία , ένα framework παρακολούθησης και έναν server ανάλυσης. Αυτό το framework μπορεί να χρησιμοποιηθεί ως μέρος ενός NFV MANO, ενεργώντας σαν βελτιωμένος διαχειριστής πολιτικής ή μπορεί να χρησιμοποιηθεί σαν εξωτερική συνιστώσα που αναλύει την απόδοση του δικτυακού service και παρέχει προτάσεις στο MANO με τη χρήση των API.

Αυτή η πτυχή ήταν σημαντικό να ληφθεί υπόψιν στον σχεδιασμό και την επιλογή των εργαλείων, παρόλο που η πραγματική ενσωμάτωση με μία συγκεκριμένη εκτέλεση MANO δεν εξετάζεται σε αυτή την εργασία. Οι βασικές απαιτήσεις του προτεινόμενου συστήματος είναι η αυτοματοποιημένη συλλογή δεδομένων παρακολούθησης από όσο το δυνατόν περισσότερους ετερογενείς τύπους πηγών, ένας εξελιγμένος μηχανισμός ανάλυσης βασισμένος σε αλγορίθμους μηχανικής μάθησης και τεχνητής νοημοσύνης για δυναμική κατανομή πόρων, η εύκολη ενσωμάτωσή του με τα ήδη υπάρχοντα orchestrator frameworks, μία επεκτάσιμη και ανθεκτική αρχιτεκτονική έτοιμη να μεταφέρει έναν μεγάλο αριθμό από οντότητες παρακολούθησης που υποστηρίζει την ιεραρχική ανάπτυξη και, τέλος, η ταχύτητα της απόκρισης να είναι σχεδόν σε πραγματικό χρόνο.



Εικόνα 1: Αρχιτεκτονική παρακολούθησης και ανάλυσης

3.2 Framework Παρακολούθησης

Η προτεινόμενη λύση παρακολούθησης συμμορφώνεται με τις παραπάνω προϋποθέσεις με την υιοθέτηση Cloud Native (CN) εργαλείων, τα οποία μπορούν εύκολα να ενσωματώσουν νέους τύπους στόχων παρακολούθησης χωρίς δύσκολες ρυθμίσεις παραμέτρων ή εκτεταμένο χρόνο διακοπής λειτουργίας. Επίσης, όλα τα εργαλεία που χρησιμοποιούνται είναι εφαρμογές ανοικτού κώδικα και υποστηρίζονται από μεγάλες κοινότητες, καθώς χρησιμοποιούνται ευρέως στη βιομηχανία. Αυτά τα χαρακτηριστικά είναι κρίσιμα επειδή εξασφαλίζουν την αξιοπιστία του συστήματος και, την ίδια στιγμή, κάνουν ευκολότερη τη διαδικασία ένταξης με πολλά εξωτερικά συστήματα που χρησιμοποιούν τα ίδια εργαλεία. Έτσι, η επιλογή εφαρμογών ανοικτού κώδικα, οι οποίες υποστηρίζονται από μία μεγάλη και ενεργή κοινότητα, όπως είναι η Linux Foundation, η Apache, και λοιπά, εξασφαλίζει την συμμετοχή πολλών προγραμματιστών για υποστήριξη, διόρθωση σφαλμάτων και αναβαθμίσεις. Επιπλέον, όσο ένα εργαλείο γίνεται δημοφιλέστερο, υιοθετείται από νέα συστήματα, το οποίο κάνει την ενσωμάτωση μεταξύ των συστημάτων ευκολότερη. Το framework παρακολούθησης αποτελείται από εργαλεία παρακολούθησης για συλλογή και αποθήκευση δεδομένων από εκπροσώπους, συγκεντρώνοντας μετρήσεις από διαφορετικές χρονικές περιόδους για τους στόχους.

Η προτεινόμενη υλοποίηση περιλαμβάνει τα παρακάτω εργαλεία παρακολούθησης:

- Ο server Prometheus [21] είναι το κεντρικό σημείο παρακολούθησης, αποθήκευσης και ειδοποίησης. Όλες οι μετρικές της απόδοσης συλλέγονται χρησιμοποιώντας ένα μοντέλο έλξης HTTP και αποθηκεύεται σε μία βάση δεδομένων χρονοσειράς. Κάποια από τα βασικά χαρακτηριστικά που θέτουν αυτόν τον server κατάλληλο για την προτεινόμενη αρχιτεκτονική είναι: (α) η υποστήριξη μίας ευέλικτης γλώσσας προγραμματισμού ερωτημάτων (PromQL), η οποία διευκολύνει τη διασύνδεση με τα εξωτερικά συστήματα, (β) η ύπαρξη μεγάλου αριθμού εφαρμογών ανοικτού κώδικα για την έκθεση των μετρικών παρακολούθησης και η ευκολία δημιουργίας νέων, (γ) η αυτονομία που παρέχεται καθώς δεν βασίζεται σε σύνθετους μηχανισμούς αποθήκευσης και (δ) το γεγονός ότι μπορούν να προστεθούν νέοι στόχοι παρακολούθησης μέσω αναπροσαρμογής των παραμέτρων ή χρησιμοποιώντας τους μηχανισμούς εντοπισμού υπηρεσιών που βασίζονται σε αρχεία.
- Το Prometheus Pushgateway [22] επιτρέπει σε batch εργασίες και μικρής διάρκειας microservices να προβάλλουν τις μετρήσεις τους στον Prometheus. Καθώς αυτή η λειτουργία μπορεί να μην υπάρχει αρκετό καιρό ώστε να αντιγραφούν τα δεδομένα, οι μετρήσεις μπορούν να μετακινηθούν σε ένα Pushgateway, το οποίο με τη σειρά του τις προβάλλει στον Prometheus server.
- Το Alertmanager [23] χειρίζεται τις ειδοποιήσεις που λαμβάνονται από εφαρμογές προγράμματος – πελάτη όπως είναι ο Prometheus server. Είναι υπεύθυνο για αναπαραγωγή, ομαδοποίηση και δρομολόγηση των ειδοποιήσεων στους σωστούς αποδέκτες ενσωμάτωσης, όπως είναι το ηλεκτρονικό ταχυδρομείο, το PagerDuty, ή το OpsGenie. Φροντίζει, επίσης, για τη σίγαση και την αναστολή των ειδοποιήσεων, κάτι που είναι χρήσιμο για τη διαχείριση του αριθμού των ειδοποιήσεων που δημιουργούνται.
- Η Grafana [24] είναι μία λύση ανοικτού κώδικα που λαμβάνει τις μετρήσεις και τις ειδοποιήσεις που είναι καταληπτές από τον Prometheus Server και παρέχει διαδραστικούς πίνακες εργαλείων web απεικόνισης. Αυτοί οι πίνακες απλοποιούν την απεικόνιση των μετρήσεων απόδοσης που έχουν συλλεχτεί, έτσι ώστε μετά από κάθε ειδοποίηση ο χρήστης να μπορεί να ανατρέξει σε συγκεκριμένο πίνακα και να αναγνωρίσει το πρόβλημα παρατηρώντας τα γραφήματα.

Η προτεινόμενη υλοποίηση περιλαμβάνει τα παρακάτω μέσα παρακολούθησης:

- Το Netdata.io [25] είναι ένα ισχυρό μέσο παρακολούθησης σε πραγματικό χρόνο το οποίο συλλέγει χιλιάδες μετρήσεις από συστήματα, μηχανήματα υπολογιστών, εικονικές μηχανές, και εφαρμογές με μηδενική διαμόρφωση. Εκτελείται μόνιμα σε υλικούς/εικονικούς διακομιστές, container, αναπτύξεις cloud, και συσκευές edge/IoT και είναι απόλυτα ασφαλές για εγκατάσταση σε ένα σύστημα στη μέση του περιστατικού χωρίς κάποια προετοιμασία.
- Το cAdvisor [26] παρέχει μετρήσεις για τη χρήση των πόρων και τα χαρακτηριστικά απόδοσης των container που εκτελούνται. Είναι μία διαδικασία daemon που συλλέγει, συγκεντρώνει, επεξεργάζεται και εξάγει πληροφορίες για τα container. Διατηρεί συγκεκριμένες παραμέτρους απομόνωσης πόρων για κάθε container, ιστορικό της χρήσης τους, ιστογράμματα του πλήρους ιστορικού τους και στατιστικά στοιχεία δικτύου.

Αξίζει να αναφερθεί ότι κάποια από τα πιο γνωστά MANO frameworks ανοικτού κώδικα στο πλαίσιο του NFV, όπως είναι το Service Programming and Orchestration for Virtualized Software Networks (SONATA) [27] και το Open Source MANO (OSM) [28], έχουν ήδη υιοθετήσει το Prometheus ως διακομιστή παρακολούθησης. Αυτό καθιστά την επικοινωνία μαζί τους απλή και απαιτεί μόνο την ενημέρωση του αρχείου διαμόρφωσης του διακομιστή παρακολούθησης Prometheus. Επιπλέον, για αναπτύξεις μεγάλης κλίμακας, οι διακομιστές παρακολούθησης Prometheus υποστηρίζουν μία κατανεμημένη (διαδοχική) αρχιτεκτονική: οι τοπικοί διακομιστές του Prometheus συλλέγουν και αποθηκεύουν τα δεδομένα μετρήσεων από τους διαφορετικούς στόχους, ενώ αποστέλλονται στον Prometheus μόνο οι ειδοποιήσεις για περαιτέρω επεξεργασία. Μία άλλη απαραίτητη προϋπόθεση επεκτασιμότητας αφορά τη μεγάλη ροή δεδομένων από τους παράγοντες παρακολούθησης στο διακομιστή παρακολούθησης και την αντίστοιχη βάση δεδομένων του, η οποία μπορεί να επηρεάσει την απόδοση της υπηρεσίας σε ακραίες περιπτώσεις. Για να ξεπεραστούν αυτά τα πιθανά προβλήματα, το σύστημα παρακολούθησης είναι διαμορφωμένο ώστε να αποθηκεύει δεδομένα παρακολούθησης σε συγκεκριμένο χρονικό διάστημα και σε περιπτώσεις μεγάλης ανάπτυξης, η υιοθέτηση της περιγραφόμενης διαδοχικής αρχιτεκτονικής παρέχει μία κατάλληλη λύση.

3.3 Διακομιστής Ανάλυσης

Ο διακομιστής ανάλυσης παρέχεται με δεδομένα που συλλέχθηκαν κατά τη διάρκεια της δημιουργίας του service προφίλ και περιλαμβάνει μία ευρεία γκάμα αλγορίθμων μηχανικής μάθησης που εκτείνονται από linear regression έως και deep neural networks. Ανάλογα με τα χαρακτηριστικά της εξεταζόμενης υπηρεσίας, π.χ.,(α) την πολυπλοκότητα μεταξύ των παραμέτρων της εργασίας, όπως είναι το μέγεθος του αρχείου, ο αριθμός των αρχείων και οι μετρήσεις της απόδοσης, όπως η χρήση της CPU και της μνήμης, και (β) τον αριθμό των εξεταζόμενων χαρακτηριστικών, οι ενεργοποιημένοι αλγόριθμοι μηχανικής μάθησης είναι προσαρμοσμένοι στη συγκεκριμένη υπηρεσία.

Η συνολική ακρίβεια μοντελοποίησης τους εξαρτάται τόσο από το σύνολο δεδομένων (ποσότητα και ποιότητα των εξεταζόμενων δεδομένων), όσο και από τους επιλεγμένους αλγορίθμους μηχανικής μάθησης. Για παράδειγμα, μία πολύπλοκη μαθηματική σχέση μεταξύ των παραμέτρων και της απόδοσης των μετρήσεων, μπορεί να απαιτήσει πιο σύνθετες δομές μηχανικής μάθησης, όπως είναι ένα deep neural network με μεγαλύτερο αριθμό κρυφών επιπέδων.

Επιπλέον, η προτεινόμενη εφαρμογή επιτρέπει την επανεκπαίδευση των αλγορίθμων με πρόσθετα δεδομένα όσο η υπηρεσία συνεχίζει τη δημιουργία προφίλ για μεγαλύτερο χρονικό διάστημα, πιθανών οδηγώντας σε υψηλότερη ακρίβεια μοντελοποίησης. Γενικά, η μεθοδολογία επιλογής των αλγορίθμων μηχανικής μάθησης είναι η εξής: Οι προβλεπόμενες τιμές για κάθε αλγόριθμο ML συγκρίνονται με τις πραγματικές τιμές για κάθε μέτρηση που παρακολουθείται και μόλις η ακρίβεια μοντελοποίησης επαληθεύσει ένα προκαθορισμένο όριο, ο πιο ακριβής αλγόριθμος χρησιμοποιείται από εκείνο το σημείο και έπειτα.

Κεφάλαιο 4

Υλοποίηση

4.1 Υλοποίηση Test Bed – Αξιολόγηση μίας Real-Life υπηρεσίας

4.1.1 Η Ανάπτυξη Sandbox Test-Bed

Σε αυτή την εργασία, δίνεται ιδιαίτερη προσοχή στην αξιολόγηση μίας σύνθετης υπηρεσίας που αποτελείται από πολλά containers που χρησιμοποιούν microservices. Μία τέτοια υπηρεσία είναι το Apache Hadoop, το οποίο είναι ένα software framework ανοικτού κώδικα που προσφέρει υπηρεσίες μεγάλων δεδομένων και ανάλυσης με σχεδόν απεριόριστη επεκτασιμότητα. Χρησιμοποιείται για την αποθήκευση και την επεξεργασία μεγάλου συνόλου δεδομένων με εξαιρετική αποδοτικότητα και μπορεί να αναπτυχθεί αβίαστα σε βασικά προϊόντα hardware. Από το 2011 που κυκλοφόρησε έχει χρησιμοποιηθεί ευρέως στην βιομηχανία είτε ως λύση εσωτερικής εγκατάστασης (on-premise), είτε ως λύση βασισμένη στο cloud (cloud-based). Πολλοί πάροχοι δικτυακού service προσφέρουν το Hadoop ως υπηρεσία, βασισμένοι σε δικές τους αναπτύξεις ή χρησιμοποιούν προμηθευτές software, όπως είναι η Hortonworks ή η MapR. Σήμερα, το Hadoop είναι διαθέσιμο μέσω πολλών παρόχων δικτυακού service, και συγκεκριμένα Cloudera, Hortonworks, MapR, Amazon Elastic Map Reduce, Altiscale, και λοιπά. Για το συγκεκριμένο πείραμα, αναπτύχθηκε ένα σύμπλεγμα Apache Hadoop (Εικόνα 2) που αποτελείται από πολλά microservices, συγκεκριμένα, historyserver, έναν namenode, έναν nodemanager, έναν resource manager, και τρεις data node διακομιστές.

IMAGE	PORTS	NAMES
hadoop-datanode:2.0.0-hadoop3.2.1-java8	9864/tcp	datanode3
hadoop-nodemanager:2.0.0-hadoop3.2.1-java8	8042/tcp	nodemanager
hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8	8088/tcp	resourcemanager
hadoop-datanode:2.0.0-hadoop3.2.1-java8	9864/tcp	datanode2
hadoop-datanode:2.0.0-hadoop3.2.1-java8	9864/tcp	datanode1
hadoop-historyserver:2.0.0-hadoop3.2.1-java8	8188/tcp	historyserver
hadoop-namenode:2.0.0-hadoop3.2.1-java8	0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp	namenode

Εικόνα 2: Υπό δοκιμή σύμπλεγμα Hadoop

Προκειμένου να επικυρωθεί η προτεινόμενη αρχιτεκτονική, χρησιμοποιήθηκε ένα sandbox περιβάλλον αποτελούμενο από δύο διαφορετικούς φυσικούς διακομιστές συνδεδεμένους μέσω ενός τοπικού ethernet δικτύου. Οι τεχνικές προδιαγραφές των φυσικών διακομιστών συνοψίζονται στον Πίνακα 2, και η συσκευή δικτύωσης που χρησιμοποιήθηκε είναι η Cisco SG250 με 18 θύρες Gigabit switch.

Πίνακας 2: Τεχνικές προδιαγραφές φυσικών διακομιστών

PowerEdge R230	
CPU	4 CPUs × Intel(R) Xeon(R) (CPU E3-1220 v6 @ 3.00 GHz)
RAM	16 GB DDR4
HDD	8 TB
NETWORK	2 × 1GbE LOM

Στον πρώτο διακομιστή αναπτύξαμε όλα τα μέρη της προτεινόμενης αρχιτεκτονικής χρησιμοποιώντας σενάρια docker-compose, και στο δεύτερο, ακολουθώντας την ίδια λογική, έγινε ανάπτυξη του συμπλέγματος Hadoop και των μέσων παρακολούθησης (cAdvisor και Netdata.io). Όλα τα εργαλεία λογισμικού (Πίνακας 3) βασίστηκαν στις τελευταίες εκδόσεις των αρχείων που χρησιμοποιούνται ώστε να εκτελεστεί το container από τις εφαρμογές ανοικτού κώδικα (δηλαδή, Prometheus.io, Netdata.io, cAdvisor, Apache Spark, κ.λπ) που κάνουν την προτεινόμενη αρχιτεκτονική αξιόπιστη και επαναχρησιμοποιούμενη. Επιπλέον, η υιοθέτηση από τον διακομιστή παρακολούθησης του Prometheus προετοιμάζει την προτεινόμενη λύση ώστε να ενσωματώσει και να συλλέξει τα δεδομένα παρακολούθησης από τα συμπλέγματα Kubernetes.

Πίνακας 3: Εργαλεία παρακολούθησης και ανάλυσης ανοικτού κώδικα

Software tools Versions	
Prometheus Server	v 2.19.3
Prometheus Pushgateway	v 1.2.0
Prometheus Alertmanager	v 0.23.0
cAdvisor	v 0.32.0
Netdata.io	v 1.23.2
Apache Spark	v 3.2.1

4.1.2 Η Διαδικασία Αξιολόγησης

Το “testDFSIO” είναι ένα πολύ γνωστό τεστ ανάγνωσης και εγγραφής για το κατανεμημένο σύστημα αρχείων Hadoop (HDFS). Συγκεκριμένα, χρησιμοποιείται για να δοκιμάζεται η απόδοση του NameNode και των στοιχείων δικτύου στο HDFS. Αυτό το τεστ εκτελέστηκε χρησιμοποιώντας μια εργασία MapReduce, η οποία πρώτα χωρίζει

τα σύνολα δεδομένων που εισάγονται σε ανεξάρτητα σετ δεδομένων για να επεξεργάζονται παράλληλα (map), και έπειτα οι έξοδοι από την διαδικασία map συνδυάζονται σε ένα μικρότερο σύνολο τιμών (reduce). Επιλέχθηκε το MapReduce, καθώς είναι ένα ευρέως γνωστό μοντέλο προγραμματισμού που παρέχει σημαντικά πλεονεκτήματα όταν πρόκειται για μεγάλο αριθμό δεδομένων που χρειάζεται να επεξεργαστούν. Συγκεκριμένα, είναι εύκολο και έχει δυνατότητα επέκτασης, καθώς μπορεί να επεξεργαστεί μεγάλο αριθμό δεδομένων σε λογικό χρονικό διάστημα και μπορεί να χειριστεί πολλαπλές πηγές και τύπους δεδομένων. Επιπλέον, ο χρήστης μπορεί να επιλέξει και τον αριθμό των αρχείων και το μέγεθός τους για κάθε τεστ. Επιπροσθέτως, ο χρήστης μπορεί να ορίσει τον τύπο του τεστ (ανάγνωση ή εγγραφή) και με την ολοκλήρωσή του συγκεκριμένες μετρήσεις του επιπέδου service μπορούν να απεικονιστούν, όπως είναι η διεκπεραιωτική ικανότητα (throughput) , ο μέσος ρυθμός εισόδου – εξόδου (average I/O rate), την τυπική απόκλιση του I/O rate και τον χρόνο εκτέλεσης του τεστ (execution time). Γενικά, το "testDFSIO" είναι ένας σημαντικός έλεγχος για το σύμπλεγμα Hadoop, καθώς παρέχει μία συνολική αξιολόγηση της απόδοσης της παρεχόμενης υπηρεσίας και μία γρήγορη εκτίμηση της αποτελεσματικότητας του συμπλέγματος όσον αφορά τις εισόδους και τις εξόδους.

Σε αυτή την εργασία, αξιοποιείται αυτή η δοκιμή με σκοπό να δοθεί έμφαση στην υλοποίηση που αναπτύχθηκε ενώ παρακολουθούνται δεδομένα και από τα τρία επίπεδα, ιδιαίτερα από το "testDFSIO", το επίπεδο του service, το υλισμικό και το εικονικό επίπεδο. Έπειτα, τα δεδομένα που συλλέγονται αξιοποιούνται ώστε να εκτελέσουν προβλέψεις μηχανικής μάθησης. Για σκοπούς μοντελοποίησης, συλλέχθηκαν έξι αλγόριθμοι μηχανικής μάθησης για να συσχετιστούν μαθηματικά οι παράμετροι εισόδου, δηλαδή ο αριθμός και το μέγεθος των αρχείων, στις μετρήσεις που παρακολουθούνται, π.χ. CPU και disk usage. Αυτοί οι αλγόριθμοι είναι:

- Πολλαπλή Γραμμική Παλινδρόμηση (Multiple Linear Regression), ο οποίος χρησιμοποιείται για να προβλέψει το αποτέλεσμα μιας μεταβλητής Y γνωρίζοντας την σταθερά Xi και αναφέρεται σε γραμμικά μοντέλα εξισώσεων.

$$y = b_0 + b_1 x_1 + \cdots b_i x_i \quad (1)$$

- Πολυπαραγοντική Πολυωνυμική Παλινδρόμηση (Multivariate Polynomial Regression), ο οποίος είναι εξέλιξη της γραμμικής παλινδρόμησης , αναφέρεται σε πολυωνυμικές εξισώσεις και μπορεί να δώσει σε πιο περίπλοκα μοντέλα λύση.

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \cdots b_i x_1^i \quad (2)$$

- Decision Tree , στο οποίο η επίλυση των συγκεκριμένων προβλημάτων αναπαρίσταται σε δενδροειδή μορφή. Κάθε κλάδος του δέντρου είναι μία πιθανή λύση και κάθε κόμβος μια απόφαση. Ο συγκεκριμένος αλγόριθμος χρησιμοποιείται κυρίως για εύκολα προβλήματα καθώς, σε περίπτωση περιπλοκότητας, δεν βρίσκει πάντα τη βέλτιστη λύση.
- Τυχαίο Δάσος (Random Forest), είναι η εξέλιξη του δέντρου αποφάσεων. Λειτουργεί με κατασκευή πολλαπλών δέντρων και η πλειοψηφία των τελικών αποφάσεων των δέντρων είναι η τελική απόφαση του δάσους.
- Παλινδρόμηση Διανυσμάτων Υποστήριξης (Support Vector Regression), ο οποίος είναι παρόμοιος με τη γραμμική παλινδρόμηση, δεδομένου ότι η εξίσωση της γραμμής είναι :

  $$y = wx_i + b \quad (3)$$

  Στο SVR, αυτή η ευθεία αναφέρεται ως hyperplane και σε αντίθεση με άλλες παλινδρομήσεις που προσπαθούν να ελαχιστοποιήσουν το σφάλμα μεταξύ της πρόβλεψης και του πραγματικού αποτελέσματος, το SVR προσπαθεί να βρει τη βέλτιστη πρόβλεψη μεταξύ ενός ορίου α, θέλοντας να ικανοποιήσει τη σχέση:

  $$-a < y - wx_i + b > a \quad (4)$$
- K-Nearest Neighbors,ο οποίος υπολογίζει την διαφορά των δεδομένων της δοκιμής και της εκπαίδευσης και επιλέγει τις βέλτιστες λύσεις που βρίσκονται πιο κοντά στα δεδομένα της δοκιμής.

Υπάρχουν πολλοί περισσότεροι διαθέσιμοι αλγόριθμοι μηχανικής μάθησης στον διακομιστή ανάλυσης, περιέχοντας ακόμα πιο σύνθετους αλγορίθμους, όπως είναι ο αλγόριθμος deep neural networks (DNN). Ωστόσο, στη συγκεκριμένη ανάλυση δεν συμπεριλήφθηκε ο DNN, καθώς απαιτεί έναν μεγάλο αριθμό από δεδομένα εκπαίδευσης προκειμένου να παρέχει ακριβή αποτελέσματα. Καθώς στη συγκεκριμένη διαδικασία αξιολόγησης εξετάζεται ένας μικρός αριθμός δεδομένων εκπαίδευσης (55 περιπτώσεις), ο DNN θα παρείχε σίγουρα εσφαλμένες εκτιμήσεις. Σε διαφορετικές αναπτύξεις service, όπου το σύνολο των δεδομένων εκπαίδευσης είναι σημαντικά υψηλότερο, πρέπει σίγουρα να λαμβάνονται υπόψιν οι αλγόριθμοι deep learning.

Οι παραπάνω έξι αλγόριθμοι επιλέχθηκαν διότι προσφέρουν υψηλή ακρίβεια μέσα σε ένα σχετικά μικρό υπολογιστικό timeframe, και ταιριάζουν περισσότερο σε προβλήματα που περιέχουν μικρό αριθμό χαρακτηριστικών. Στη συγκεκριμένη υλοποίηση, ως γλώσσα προγραμματισμού χρησιμοποιείται η Python3, η βιβλιοθήκη NumPy για πολλαπλασιασμούς πινάκων, προεπεξεργασία δεδομένων και τμηματοποίηση, η βιβλιοθήκη scikit-learn για την υλοποίηση των αλγορίθμων μηχανικής μάθησης και η βιβλιοθήκη Keras χρησιμοποιώντας την Tensorflow (version 2.0.0) ως backend.

4.2 Αποτελέσματα

Σε αυτό το κεφάλαιο, πρώτα εξετάζονται τα δεδομένα που συλλέχθηκαν από τα τρία επίπεδα κατά τη διάρκεια της διαδικασίας service profiling και έπειτα γίνεται εισαγωγή των δεδομένων στους έξι αλγορίθμους μηχανικής μάθησης με σκοπό να εκτελέσουν προβλέψεις για τις επιλεγμένες μετρήσεις.

4.2.1 Profiling των Κρίσιμων Μετρήσεων Συστήματος από τα Τρία Επίπεδα

Το διαμορφωμένο service δοκιμάζει τα όρια αντοχής του χρησιμοποιώντας το “testDFSIO” προκειμένου να βρεθούν τα κρίσιμα σημεία της υλοποίησης, ή με άλλα λόγια, να εξεταστούν οι παράμετροι εντός των οποίων η υπηρεσία θα λειτουργεί χωρίς προβλήματα. Η κατανόηση αυτών των ορίων σε βάθος είναι πολύ σημαντική από την πλευρά των παρόχων δικτυακού service, καθώς μπορεί να εξασφαλίσει το QoS κατά τη διάρκεια του συνολικού κύκλου λειτουργίας του service.

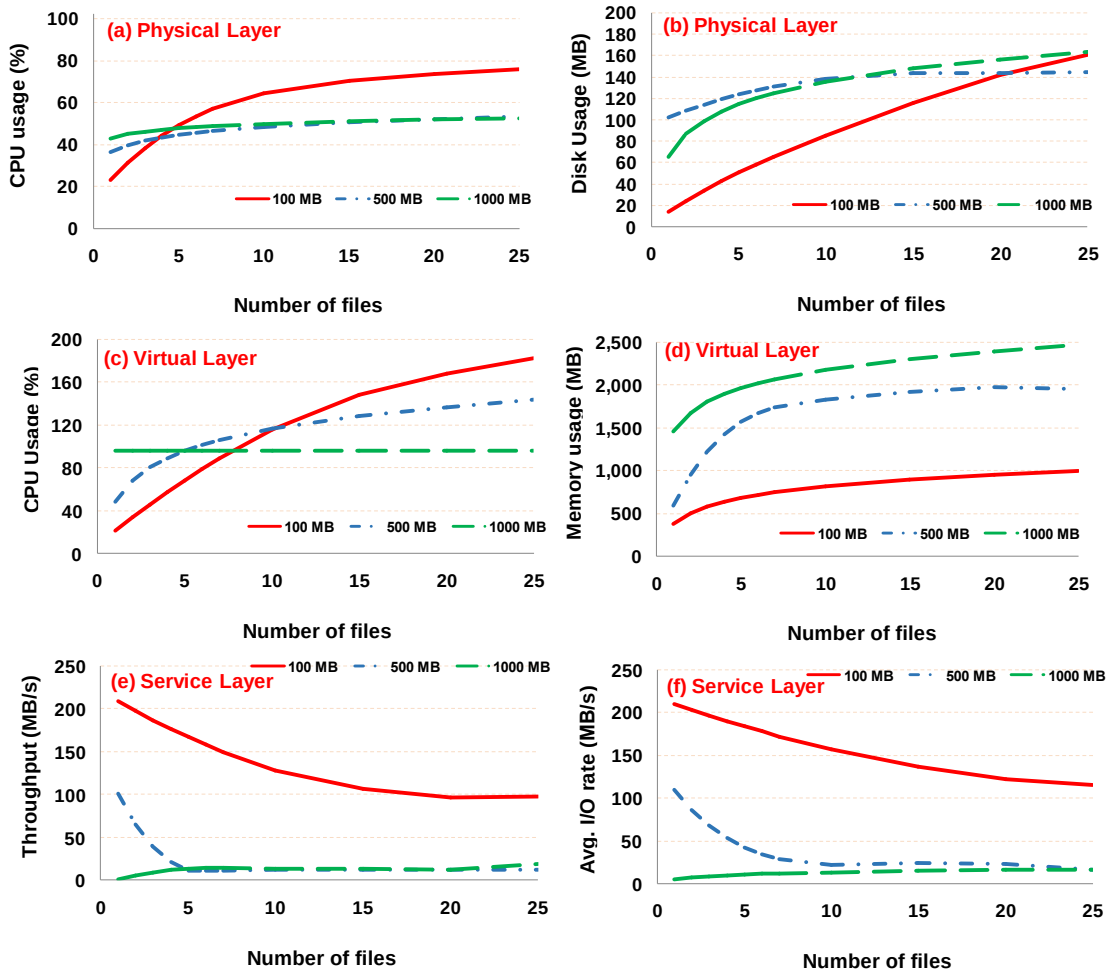
Στις δοκιμές που πραγματοποιήθηκαν, δημιουργήθηκε τεστ εγγραφής για δύο από τα χαρακτηριστικά του “testDFSIO”, συγκεκριμένα , τον αριθμό και το μέγεθος των αρχείων(number of files, file size), με σκοπό να δοκιμαστούν τα όρια του Hadoop Distributed File System που υλοποιείται. Επιπλέον, για να ληφθούν ουσιαστικά αποτελέσματα, εξασφαλίστηκαν οι ίδιες αρχικές ρυθμίσεις για κάθε δοκιμή, διαγράφοντας όλα τα αρχεία που αποθηκεύτηκαν στον δίσκο κατά την προηγούμενη δοκιμή.

Έγινε κατανομή των παραμέτρων για τα τρία επίπεδα, και απεικονίζονται οι πιο αντιπροσωπευτικές μετρήσεις στην Εικόνα 3 (δύο από κάθε επίπεδο). Όπως φαίνεται στην Εικόνα 3, οι μετρήσεις και της CPU usage (Εικόνα 3a) και του disk usage (Εικόνα

3b), αυξάνονται όσο προστίθεται παραπάνω αριθμός αρχείων, καθώς χρειάζεται να κατανεμηθεί ένας μεγάλος αριθμός πόρων του συστήματος ώστε να ολοκληρωθεί η δοκιμή.

Έπειτα, για να επιλεχθούν οι μετρήσεις που θα ληφθούν υπόψη από το εικονικό επίπεδο, έγινε εστίαση στις μετρήσεις απόδοσης που αναφέρονται στο node manager container (μεταξύ των διάφορων κόμβων της υλοποίησης Hadoop), επειδή το node manager είναι ο κόμβος που καταναλώνει τους περισσότερους πόρους, οπότε είναι κρίσιμος ο ρόλος του στο συγκεκριμένο τεστ. Όπως είναι εμφανές, η χρήση της μνήμης του node manager (Εικόνα 3d) αυξάνεται όταν ο αριθμός και το μέγεθος των αρχείων κλιμακώνεται. Επιπλέον, η επίδραση της CPU usage (Εικόνα 3c) εξαρτάται σε μεγάλο βαθμό από το μέγεθος του αρχείου. Για παράδειγμα, σε περίπτωση ενός αρχείου με μέγεθος 100MB και με αριθμό αρχείων από 25 και πάνω, ο αντίκτυπος της CPU usage αναμένεται να φτάσει 200%. Η τιμή της CPU usage υπερβαίνει το 100%, το οποίο αποδίδεται στο γεγονός πως χρησιμοποιήθηκε μία μηχανή πολλαπλών πυρήνων, και πως το τεστ υλοποιείται ως διαδικασία πολλαπλών νημάτων εκτέλεσης(multi-thread).

Στη συνέχεια, από το επίπεδο του service, επιλέχθηκε το throughput (Εικόνα 3e) και το average I/O rate (Εικόνα 3f), τα οποία παρέχονται από το τεστ εγγραφής “testDFSIO”. Τα αποτελέσματα δείχνουν πως όταν η ποσότητα των αρχείων είναι μικρότερη από πέντε, και το throughput και το average I/O μειώνονται όσο το μέγεθος του κάθε αρχείου αυξάνεται. Αντίστοιχα, από τα αποτελέσματα φαίνεται πως όταν η ποσότητα των αρχείων είναι μεγαλύτερη από πέντε, και οι δύο μετρήσεις φτάνουν σε ένα τέλμα. Συγκεκριμένα, όταν το μέγεθος του κάθε αρχείου είναι 500MB και πάνω, οι τιμές τους ελαχιστοποιούνται, αποτελώντας ένα επιπλέον οριακό σημείο στο σύστημα.



Εικόνα 3:Κρίσιμες μετρήσεις συστήματος που παρακολουθούνται από τα τρία επίπεδα. (a,b) μετρήσεις φυσικού επιπέδου, (c,d) μετρήσεις εικονικού επιπέδου, (e,f) μετρήσεις service επιπέδου.

Τα προαναφερθέντα αποτελέσματα επιβεβαιώνουν την ανάγκη για ανάλυση και παρακολούθηση δεδομένων από διάφορα επίπεδα συστήματος, συγκεκριμένα τα φυσικά, εικονικά και service επίπεδα, με σκοπό την αναγνώριση όλων των πιθανών οριακών σημείων του συστήματος και για την εύρεση των ακριβών συνθηκών που βλάπτουν τη συνολική απόδοση της υλοποίησης.

Έπειτα, υπολογίζεται ο συνολικός χρόνος εκτέλεσης για κάθε δοκιμή με διαφορετική ποσότητα ή μέγεθος αρχείου (Εικόνα 4). Αυτή η μέτρηση παρέχεται από την εγγραφή “testDFSIO” του τεστ μετά από την ολοκλήρωση της κάθε δοκιμής. Παρατηρείται ότι ο χρόνος εκτέλεσης της κάθε δοκιμής αυξάνεται σχεδόν γραμμικά με την αύξηση της ποσότητας των αρχείων. Αυτό είναι αναμενόμενο, καθώς η συνολική υλοποίηση χρειάζεται περισσότερο χρόνο να ολοκληρώσει το τεστ εγγραφής λόγω του μεγαλύτερου μεγέθους που απαιτείται για να γίνει εγγραφή στο δίσκο. Ο χρόνος εκτέλεσης μπορεί να

υπερβαίνει, σε κάποιες περιπτώσεις, τα δέκα λεπτά, δείχνοντας πως το αναπτυγμένο σύστημα sandbox έχει φτάσει τα όριά του, καθώς μετά από αυτή την τιμή, π.χ. μετά από τριάντα αρχεία, ο κατανεμημένος χώρος του δίσκου έχει γεμίσει πλήρως με εγγραφές.

Εικόνα 4: Χρόνος εκτέλεσης του testDFSIO από διάφορες περιπτώσεις που εξετάστηκαν

4.2.2 Προβλέψεις με Χρήση Μηχανικής Μάθησης

Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα ανάλυσης της μηχανικής μάθησης στις έξι παραμέτρους που εμφανίζονται στην Εικόνα 3, χρησιμοποιώντας έξι γνωστούς αλγορίθμους μηχανικής μάθησης, συγκεκριμένα τους αλγορίθμους Multiple Linear Regression, Multivariate Polynomial Regression, Decision Tree, Random Forest, Support Vector Regression και K-Nearest Neighbors.

Η ανάλυση των δεδομένων είναι ύψιστης σημασίας για τους παρόχους δικτυακού service, καθώς μπορεί να προβλέψει τη συμπεριφορά του συστήματος που εξετάζεται για κάθε συνδυασμό κρίσιμων χαρακτηριστικών του, όπως είναι η ποσότητα και το μέγεθος των αρχείων, όχι μόνο εκείνα που εξετάστηκαν προηγουμένως, καθώς και να ορίσει επιπλέον κρίσιμα σημεία του συστήματος σε αυτά που προσδιορίστηκαν κατά τη διάρκεια της φάσης profiling, π.χ. τη CPU usage στο node manager. Η διαδικασία μέτρησης πραγματοποιήθηκε σε 55 σημεία για κάθε παράμετρο χρησιμοποιώντας το τεστ εγγραφής του “testDFSIO” ,τα οποία έπειτα χρησιμοποιήθηκαν στην τροφοδοσία των αλγορίθμων μηχανικής μάθησης. Μία αναπαράσταση της σχέσης μεταξύ των παραμέτρων εισόδου (number of files, file size) και των παραμέτρων εξόδου(μετρήσεις απόδοσης, π.χ. CPU usage) για ενδεικτικές περιπτώσεις, αποτυπώνεται στην Εικόνα 3. Ο αριθμός των επιλεγμένων σημείων είναι σχετικά μικρός, καθώς υπάρχει και μικρός αριθμός χαρακτηριστικών στο συγκεκριμένο πρόβλημα, τα οποία είναι ο αριθμός και το μέγεθος του κάθε αρχείου. Αποφασίστηκε να ληφθεί υπόψη ένας μικρός αριθμός

σημείων (55), ώστε να εντοπιστούν οι αλγόριθμοι μηχανικής μάθησης που θα μπορέσουν να οδηγήσουν σε μία αρκετά υψηλή ακρίβεια μοντελοποίησης με περιορισμένο αριθμό δεδομένων για να πετύχουμε γρήγορη δημιουργία service προφίλ. Τα σημεία αυτά επιτεύχθηκαν χρησιμοποιώντας τις παρακάτω παραμέτρους :

- Ποσότητα αρχείων (number of files) : [1,2,3,4,5,6,7,10,15,20,25]
- Μέγεθος αρχείου (file size) σε MB : [100,250,500,750,1000]

Τα βήματα της εκτέλεσης της διαδικασίας μηχανικής μάθησης αναλύονται παρακάτω. Στο πρώτο βήμα, τα 55 ζεύγη εισόδου (number of files και file size) και οι τιμές εξόδου για κάθε παράμετρο (π.χ. CPU usage), αρχικά ανακατεύθηκαν και στη συνέχεια χωρίστηκαν σε τρία υποσύνολα, ένα για την εκπαίδευση, το οποίο περιλάμβανε το 60% των τιμών, ένα για την επικύρωση, το οποίο ενσωματώνει το 20% των τιμών, και ένα για την δοκιμή, το οποίο αποτελεί το τελικό 20% των τιμών. Στη συνέχεια, πραγματοποιήθηκε μία κυκλική χρήση μεταξύ των τριών υποσυνόλων ούτως ώστε να διασφαλιστεί ότι θα δοκιμαστούν όλες οι τιμές που συλλέχθηκαν. Στο δεύτερο βήμα, το υποσύνολο της εκπαίδευσης χρησιμοποιήθηκε ώστε να εκπαιδεύσει τον αλγόριθμο ενώ οι βέλτιστες υπερπαράμετροι προήλθαν από το υποσύνολο της επικύρωσης. Στο τελευταίο βήμα, το υποσύνολο της δοκιμής χρησιμοποιήθηκε για να πραγματοποιήσει τις προβλέψεις και στη συνέχεια οι προβλεπόμενες τιμές συγκρίθηκαν με τις πραγματικές για κάθε μέτρηση. Το σφάλμα πρόβλεψης υπολογίστηκε χρησιμοποιώντας το μέσο απόλυτο σχετικό ποσοστό σφάλματος με σκοπό να επιτευχθούν συγκρίσιμα αποτελέσματα σε όλες τις μετρήσεις και τις μεθόδους μηχανικής μάθησης ως εξής:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i-\hat{y}_i|}{y_i}\times 100\% \quad (5)$$

Όπου το y_i είναι η τιμή του σημείου i που μετρήθηκε χρησιμοποιώντας την αρχιτεκτονική τριών επιπέδων, $\hat{y}_i$ είναι η προβλεπόμενη τιμή από την μέθοδο μηχανικής μάθησης και το n είναι ο συνολικός αριθμός των σημείων.

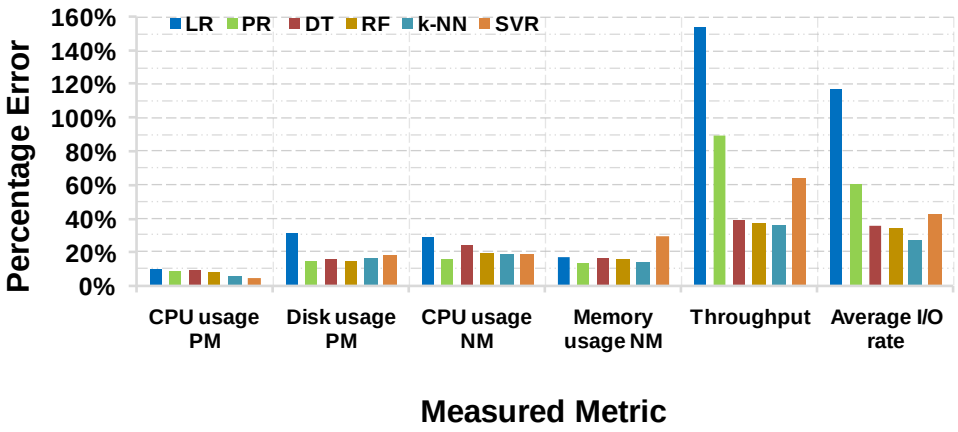
Ο Πίνακας 4 παρέχει τις βέλτιστες τιμές των υπερπαραμέτρων για τις έξι μετρήσεις που εξετάζονται, οι οποίες αποκτήθηκαν χρησιμοποιώντας ένα grid search για τους έξι αλγορίθμους μηχανικής μάθησης. Όπως φαίνεται στον Πίνακα 4, η γραμμική παλινδρόμηση αξιοποιεί ένα πολυώνυμο πρώτης τάξης. Για ένα πολυώνυμο υψηλότερης τάξης, επιλέγουμε μία δεύτερη τάξη, επειδή ένα τέτοιο πολυώνυμο, π.χ. τρίτης τάξης, θα οδηγούσε σε υπερφόρτωση στα δεδομένα εκπαίδευσης. Επιπλέον, όταν χρησιμοποιείται το Decision Tree, ένα δέντρο με βάθος ίσο με πέντε είναι αρκετό. Στη συνέχεια, ο

αριθμός των εκτιμητών του Random Forest εκτείνεται μεταξύ τεσσάρων και πέντε και στην περίπτωση του Support Vector Regression, η παράμετρος γ κυμαίνεται από $5x10^{-5}$ μέχρι 0.01. Τέλος, όταν χρησιμοποιείται ο k-Nearest Neighbors, είναι επαρκές να υπολογιστεί η απόσταση μόνο από έναν ή δύο κοντινότερους γείτονες. Γενικά, η τιμή κάθε υπερπαραμέτρου, μπορεί να επηρεάσει την αλγοριθμική αρχιτεκτονική και την πολυπλοκότητά της, π.χ. μία μεγαλύτερη πολυωνυμική τάξη οδηγεί σε μία πιο σύνθετη δομή. Συνοπτικά, η συγκεκριμένη ανάλυση φανερώνει πως το πρόβλημα αυτό δεν απαιτεί υπερβολικά περίπλοκες δομές μηχανικής μάθησης.

Πίνακας 4: Βέλτιστες υπερπαράμετροι για κάθε αλγόριθμο μηχανικής μάθησης και μέτρηση

Μέθοδος	**Υπερπαράμετρος**	**Βέλτιστη Τιμή/Μέτρηση Υπερπαραμέτρου**					
-	-	CPU Usage PM	Disk Usage PM	CPU Usage NM	Memory Usage NM	Throughput	Average I/O rate
Multiple Linear Regression (LR)	Πολυωνυμική Τάξη	$1^{η}$	$1^{η}$	$1^{η}$	$1^{η}$	$1^{η}$	$1^{η}$
Multivariate Polynomial Regression (PR)	Πολυωνυμική Τάξη	$2^{η}$	$2^{η}$	$2^{η}$	$2^{η}$	$2^{η}$	$2^{η}$
Decision Tree (DT)	Βάθος	5	5	5	5	5	5
Random Forest (RF)	Βάθος, Αριθμός Εκτιμητών	5,4	5,4	5,4	5,5	5,5	5,5
Support Vector Regression (SVR)	C,γ	50, 0.01	50, 0.0075	50, 0.01	50, $5x10^{-5}$	50, $5x10^{-3}$	50, $5x10^{-3}$
k-Nearest Neighbors (k-NN)	k	2	2	2	2	1	1

Τα αποτελέσματα των εφαρμοσμένων αλγορίθμων μηχανικής μάθησης στις έξι μετρήσεις αποτυπώνονται στην Εικόνα 5. Στην γραμμική παλινδρόμηση (LR) εμφανίζεται το μεγαλύτερο σφάλμα σε πέντε από τις έξι μετρήσεις, το οποίο ήταν αναμενόμενο, αφού στην Εικόνα 3 φαίνεται ότι η σχέση μεταξύ των δύο χαρακτηριστικών και των μετρήσεων που αναλύονται δεν είναι, στις περισσότερες περιπτώσεις, γραμμική. Είναι αξιοσημείωτο πως το προσεγγιστικό σφάλμα υπερβαίνει το 100% στις μετρήσεις επιπέδου service, καθώς η προβλεπόμενη τιμή μπορεί να είναι μεγαλύτερη από δύο φορές από την πραγματική τιμή, και ως συνέπεια το κλάσμα $\frac{|y_i - \hat{y}_i|}{y_i}$ στην Εξίσωση (5) μπορεί να είναι μεγαλύτερο του "ένα" για έναν σημαντικό αριθμό των προβλεπόμενων περιπτώσεων. Στη συνέχεια, η πολυωνυμική παλινδρόμηση (PR) είναι μία ικανοποιητική μέθοδος για τέσσερις από τις έξι μετρήσεις, παρέχοντας σφάλμα μικρότερο του 16% στις μετρήσεις φυσικού και εικονικού επιπέδου. Από την άλλη πλευρά, οι μετρήσεις του επιπέδου service είναι οι πιο δύσκολες να προβλεφθούν. Αυτό μπορεί να οφείλεται στο γεγονός ότι η σχέση μεταξύ των δύο χαρακτηριστικών και αυτών των μετρήσεων είναι πιο περίπλοκη, και προκειμένου να βελτιωθεί η ακρίβεια της μέτρησης, πρέπει να διεξαχθεί μεγαλύτερος αριθμός δοκιμών ή να ληφθεί υπόψη μία μέθοδος αύξησης δεδομένων. Το τυχαίο δάσος (RF) ξεπερνά ελαφρώς το δέντρο αποφάσεων (DT) κατά 0.3% έως 4.5%, το οποίο ήταν αναμενόμενο, καθώς περιλαμβάνει πολλά δέντρα. Επιπροσθέτως, η ακρίβεια της παλινδρόμησης διανυσμάτων υποστήριξης (SVR) εξαρτάται σημαντικά από τη μέτρηση που μελετήθηκε, και στις περισσότερες περιπτώσεις, καταφέρνει να πετύχει σφάλμα μικρότερο του 30%. Τέλος, το k-Nearest Neighbors (k-NN) επιτυγχάνει την υψηλότερη ακρίβεια σε μετρήσεις επιπέδου service, οι οποίες είναι το throughput και το average I/O rate, ενώ στους αλγόριθμους Polynomial Regression, Random Forest και k-Nearest Neighbors οι πιο ακριβείς μετρήσεις και του φυσικού και του εικονικού επιπέδου είναι η CPU και η memory usage.



Εικόνα 5: Σύγκριση των έξι ευρέως γνωστών αλγόριθμων μηχανικής μάθησης που εφαρμόζονται σε έξι μετρήσεις συστήματος.

Αυτή η έρευνα επιβεβαιώνει ότι οι αλγόριθμοι μηχανικής μάθησης μπορούν να προβλέψουν με ακρίβεια την απόδοση των κρίσιμων μετρήσεων του συστήματος, χρησιμοποιώντας μόνο ένα μικρό μέρος διαφορετικών συνθηκών συστήματος. Αυτά είναι πολλά υποσχόμενα αποτελέσματα, τα οποία μπορούν να βελτιωθούν περαιτέρω με τεχνικές αύξησης δεδομένων, ή / και εμπλουτισμό των δεδομένων με μεγαλύτερο αριθμό εξεταζόμενων περιπτώσεων, ειδικά για τις μετρήσεις επιπέδου service.

Κεφάλαιο 5

Συμπεράσματα και Μελλοντική Έρευνα

5.1 Συμπεράσματα

Σε αυτή την έρευνα προτείνεται μία υλοποίηση τριών επιπέδων, η οποία μπορεί να εκτελέσει service profiling και να προβλέψει τις πιο σημαντικές μετρήσεις του συστήματος της εφαρμογής που εξετάζεται για κάθε πιθανό σενάριο. Αποδείχθηκε ότι χρησιμοποιώντας μόνο ένα μικρό αριθμό από σημεία δεδομένων μπορεί να επιτευχθεί επαρκής ακρίβεια ακόμα και όταν οι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιούνται φορτώνονται με ένα περιορισμένο σύνολο δεδομένων. Αυτά τα αποτελέσματα μπορούν να αξιοποιηθούν ώστε να ελαχιστοποιηθεί η απόσταση της λειτουργίας ενός συστήματος από το κρίσιμο σημείο του, οδηγώντας σε σημαντική εξοικονόμηση πόρων, διασφαλίζοντας παράλληλα το QoS. Η προτεινόμενη λύση βασίζεται σε εργαλεία ανοικτού κώδικα, και είναι επίσης επεκτάσιμη καθώς επιτρέπει τη συλλογή δεδομένων για διάφορα στοιχεία. Τέλος, είναι ευέλικτη, αφού μπορεί εύκολα να φιλοξενήσει διαφορετικές υπηρεσίες καθώς χρησιμοποιεί διαφορετικούς τύπους εργαλείων benchmark, όπως είναι το testDFSIO, το jMeter και το Apache.

5.2 Μελλοντική Έρευνα

Στο μέλλον, θα μπορούσε να πραγματοποιηθεί αύξηση των δεδομένων (data augmentation), η οποία μπορεί να οδηγήσει σε μία υψηλότερη ακρίβεια μοντελοποίησης, μειώνοντας ταυτόχρονα την ανάγκη για πρόσθετες και χρονοβόρες δοκιμές στην υποδομή που αναπτύσσεται. Επιπλέον, θα ήταν βοηθητικό να δοκιμαστεί η υλοποίηση με υπηρεσίες σε άλλους αντιπροσωπευτικούς τομείς, όπως είναι σε εφαρμογές πολυμέσων, ασφάλεια, κ.λπ. Ακόμα, είναι επιθυμητό να πραγματοποιηθεί online εκπαίδευση μηχανικής μάθησης για κάθε νέο δικτυακό service, καθώς και να ενσωματωθεί η προτεινόμενη αρχιτεκτονική με ορισμένα από τα MANO frameworks ανοικτού κώδικα, προκειμένου να διατεθεί ένα εντελώς αυτόνομο σύστημα διαχείρισης έτοιμο να χρησιμοποιηθεί στο πλαίσιο NFV.

Αναφορές

1.https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/041/04.01.01_60/gr_NFV-IFA041v040101p.pdf (accessed on 23 August 2021).

2. Palumbo, F.; Aceto, G.; Botta, A.; Ciuonzo, D.; Persico, V.; Pescapé, A. Characterization and analysis of cloud-to-user latency: The case of Azure and AWS. Comput. Netw. 2020, 184, 107693. [CrossRef]

3. Wood, T.; Cherkasova, L.; Ozonat, K.; Shenoy, P. Profiling and Modeling Resource Usage of Virtualized Applications. In Proceedings of the ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Leuven, Belgium, 1–5 December 2008.

4. Giannakopoulos, I.; Tsoumakos, D.; Papailiou, N.; Koziris, N. PANIC: Modeling Application Performance over Virtualized Resources. In Proceedings of the 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, USA, 9–13 March 2015; pp. 213–218.

5. Duplyakin, D.; Brown, J.; Ricci, R. Active Learning in Performance Analysis. In Proceedings of the 2016 IEEE International Conference on Cluster Computing (CLUSTER), Taipei, Taiwan, 12–16 September 2016.

6. Giannakopoulos, I.; Tsoumakos, D.; Koziris, N. A decision tree based approach towards adaptive modeling of big data applications. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017.

7. Cao, L.; Sharma, P.; Fahmy, S.; Saxena, V. NFV-VITAL: A framework for characterizing the performance of virtual network functions. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 93–99.

8. Peuster, M.; Karl, H. Understand Your Chains: Towards Performance Profile-Based Network Service Management. In Proceedings of the 2016 Fifth European Workshop on Software-Defined Networks (EWSDN), Den Haag, The Netherlands, 10–11 October 2016.

9. Rossem, S.V.; Tavernier, W.; Peuster, M.; Colle, D.; Pickavet, M.; Demeester, P. Monitoring and debugging using an SDK for NFV-powered telecom applications. In Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 7–10 November 2016.

10. Rosa, R.V.; Bertoldo, C.; Rothenberg, C.E. Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking. IEEE Commun. Mag. 2017, 55, 110–117. [CrossRef]
11. Peuster, M.; Karl, H. Profile your chains, not functions: Automated network service profiling in DevOps environments. In Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017.
12. Iglesias, J.O.; Aroca, J.A.; Hilt, V.; Lugones, D. Orca: An orchestration automata for configuring VNFS. In Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Las Vegas, NV, USA, 11–15 December 2017; pp. 81–94.
13. Sciancalepore, V.; Yousaf, F.Z.; Costa-Perez, X. z-TORCH: An Automated NFV Orchestration and Monitoring Solution. IEEE Trans. Netw. Serv. Manag. 2018, 15, 1292–1306. [CrossRef]
14. Nam, J.; Seo, J.; Shin, S. Probius: Automated Approach for VNF and Service Chain Analysis in Software-Defined NFV. In Proceedings of the Symposium on SDN Research (SOSR'18), Los Angeles, CA, USA, 28–29 March 2018.
15. Khan, M.G.; Bastani, S.; Taheri, J.; Kassler, A.; Deng, S. NFV-Inspector: A Systematic Approach to Profile and Analyze Virtual Network Functions. In Proceedings of the 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, Japan, 22–24 October 2018.
16. Van Rossem, S.; Tavernier, W.; Colle, D.; Pickavet, M.; Demeester, P. Profile-Based Resource Allocation for Virtualized Network Functions. IEEE Trans. Netw. Serv. Manag. 2019, 16, 1374–1388. [CrossRef]
17. Van Rossem, S.; Tavernier, W.; Colle, D.; Pickavet, M.; Demeester, P. Optimized Sampling Strategies to Model the Performance of Virtualized Network Functions. J. Netw. Syst. Manag. 2020, 28, 1482–1521. [CrossRef]
18. Schneider, S.; Satheeschandran, N.P.; Peuster, M.; Karl, H. Machine Learning for Dynamic Resource Allocation in Network Function Virtualization. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020
19. Trakadas, P.; Karkazis, P.; Leligou, H.C.; Zahariadis, T.; Papadakis, A. Scalable monitoring for multiple virtualized infrastructures for 5G services. In Proceedings of the International Symposium on Advances in Software Defined Networking and Network Functions Virtualization, Athens, Greece, 22–26 April 2018.

20. Al-Hazmi, Y.; Gonzalez, J.; Rodriguez-Archilla, P.; Alvarez, F.; Orphanoudakis, T.; Karkazis, P.; Magedanz, T. Unified representation of monitoring information across federated cloud infrastructures. In Proceedings of the IEEE 2014 26th International Teletraffic Congress (ITC), Karlskrona, Sweden, 9–11 September 2014.
21. GitHub—Prometheus/Prometheus: The Prometheus Monitoring System and Time Series Database. Available online: https://github.com/prometheus/prometheus (accessed on 23 August 2021).
22. GitHub—Prometheus/Pushgateway: Push Acceptor for Ephemeral and Batch Jobs. Available online: https://github.com/prometheus/pushgateway (accessed on 23 August 2021).
23. GitHub—Prometheus/Alertmanager: Prometheus Alertmanager. Available online: https://github.com/prometheus/alertmanager (accessed on 23 August 2021).
24. GitHub—Grafana/Grafana. Available online: https://github.com/grafana/grafana (accessed on 23 August 2021).
25. GitHub—Netdata/Netdata: Real-Time Performance Monitoring. Available online: https://github.com/netdata/netdata (accessed on 23 August 2021).
26. GitHub—Google/Cadvisor. Available online: https://github.com/google/cadvisor (accessed on 23 August 2021).
27. GitHub—Sonata-nfv. Available online: https://github.com/sonata-nfv (accessed on 23 August 2021).
28. “OSM ETSI” Git. Available online: https://osm.etsi.org/gitweb/ (accessed on 23 August 2021).