ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Παρακολούθηση και Διαχείριση Προσαρμογής Κατανεμημένων Εφαρμογών σε Νεφουπολογιστικά Περιβάλλοντα πολλών Παρόχων(Cloud Resources Management and Monitoring of DistributedApplications hosted in Cloud Environments of more than one Cloud Providers)

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Βασιλείου-Αγγέλου Δαμιανού Στεφανίδης

Αθήνα, Σεπτέμβριος 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Παρακολούθηση και Διαχείριση Προσαρμογής Κατανεμημένων Εφαρμογών σε Νεφουπολογιστικά Περιβάλλοντα πολλών Παρόχων (Cloud Resources Management and Monitoring of DistributedApplications hosted in Cloud Environments of more than one Cloud Providers)

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Συμβουλευτική Επιτροπή : Γρηγόριος Μέντζας, Καθηγητής Ε.Μ.Π. (επιβλέπων)

Παναγιώτης Τσανάκας, Καθηγητής Ε.Μ.Π.

Ηλίας Μαγκλογιάννης , Καθηγητής Πανεπιστημίου Πειραιά

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 11η Σεπτεμβρίου 2024.

.....	.....	.....
Γρηγόριος Μέντζας	Παναγιώτης Τσανάκας	Ηλίας Μαγκλογιάννης
Καθηγητής Ε.Μ.Π. (επιβλέπων)	Καθηγητής Ε.Μ.Π.	Καθηγητής Πανεπιστημίου Πειραιά
	.....	
.....	Δημήτριος Ασκούνης	.....
Ιωάννης Βεργινάδης	Καθηγητής Ε.Μ.Π.	Ιωάννης Ψαράς
Επ. Καθηγητής Οικ. Πανεπ. Αθηνών	.....	Καθηγητής Ε.Μ.Π.
	Θεόδωρος Κωστούλας	
	Αν. Καθ. Παν. Αιγαίου	

Αθήνα, Σεπτέμβριος 2024

.....

Βασίλειος-Άγγελος, Δ. Στεφανίδης

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Η παρούσα εργασία για την Τελική κρίση της Διδακτορικής Διατριβής ασχολείται με καινοτόμες ερευνητικές περιοχές σχετικές με κατανεμημένες εφαρμογές φιλοξενούμενες σε κατανεμημένα συστήματα υπολογιστικού νέφους πολλών παρόχων.

Μετά από αναλυτική και ενδελεχή επισκόπηση της βιβλιογραφίας, προέκυψαν τέσσερα βασικά ερευνητικά ερωτήματα ως βασικό θέμα της παρούσας Διδακτορικής Διατριβής. Μέχρι την ενδιάμεση κρίση, το βασικό ερώτημα της ευέλικτης παρακολούθησης κατανεμημένων εφαρμογών σε υπολογιστικά νέφη πολλών παρόχων αλλά και αυτό της βέλτιστης αναπροσαρμογής των πόρων αναλύθηκαν σε βάθος. Επίσης ένα τρίτο ερώτημα για την μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων τόσο σε επίπεδο κεντρικού νέφους πολλών παρόχων αλλά και ένα τέταρτο τελικό ερώτημα εστιαζόμενο στον υπολογισμό της ακρίβειας προγνώσεων με υπολογισμούς που λαμβάνουν χώρα στο άκρο του νέφους με τον επακόλουθο περιορισμό σε πόρους, είναι αντικείμενα που εξετάζονται στο τελικό στάδιο της παρούσας Τελικής Κρίσης της Διδακτορικής Διατριβής.

Έχοντας ως βάση αυτά τα ερευνητικά ερωτήματα, προτείνουμε λύσεις που βασίζονται σε κατανεμημένα και πολλών επιπέδων σύνθετα συστήματα επεξεργασίας συμβάντων για αποτελεσματική παρακολούθηση της προαναφερθείσας υποδομής. Επίσης προτείνεται και κατάλληλος αλγόριθμος που αποφασίζει την βέλτιστη αναπροσαρμογή των πόρων του υπολογιστικού νέφους με στόχο την βέλτιστη λειτουργία κατανεμημένων εφαρμογών. Τέλος, αναλύονται και υλοποιούνται με πολύ καλά αποτελέσματα Κατανεμημένοι Αλγόριθμοι Federated Learning όσον αφορά την πρόβλεψη ζήτησης πόρων για εφαρμογές νεφουπολογιστικών συστημάτων. Σε αυτού τους αλγορίθμους εξετάζονται και τεχνολογίες που αφορούν το Άκρο του υπολογιστικού Νέφους πολλών παρόχων. Με αυτό τον τρόπο εξασφαλίζεται η ασφαλής και αδιάλειπτη λειτουργία τους στο εγγύς μέλλον τουλάχιστον.

Η προσέγγισή μας αξιολογήθηκε υπό το πρίσμα της κατανάλωσης βασικών πόρων πχ. Μνήμης, Επεξεργαστικής Ισχύος και της στιβαρότητας λειτουργίας του προτεινόμενου συστήματος παρακολούθησης. Όσον αφορά το μοντέλο για την λήψη αποφάσεων σχετικά με την αναπροσαρμογή με βέλτιστο τρόπο των πόρων του υπολογιστικού νέφους, μετρήσεις διεξήχθησαν σχετικά με τους χρόνους εκκίνησης των εικονικών μηχανών των διαφόρων παρόχων δημόσιων και ιδιωτικών που φιλοξενούν εφαρμογές. Αυτές οι βασικές παράμετροι του αλγορίθμου δείχνουν σταθερή συμπεριφορά στους δημόσιους παρόχους. Επίσης αποτελέσματα πειραμάτων σχετικά με τις τελικές τιμές του προτεινόμενου αλγόριθμου με σκοπό την σωστή λήψη απόφασης για αναπροσαρμογή περιβάλλοντος νέφους φανερώνουν την αποτελεσματικότητα της νέας προτεινόμενης μεθόδου. Επίσης, διάφοροι αλγόριθμοι κατανεμημένης αρχιτεκτονικής Federated Learning για πρόβλεψη ζήτησης πόρων νεφουπολογιστικών εφαρμογών έχουν παρουσιαστεί με εξαιρετικά καλά πειραματικά αποτελέσματα προγνώσεων κυρίως λόγω των τεχνικών που εφαρμόστηκαν όπως επιλογής clients αλλά και τεχνολογιών για έλεγχο επαρκούς χρήσης δεδομένων εκπαίδευσης σε multicloud περιβάλλοντα. Τέλος, πολύ καλά αποτελέσματα όσον αφορά την πρόγνωση παρατηρήθηκαν και για Αρχιτεκτονικές Federated Learning προσαρμοσμένες να εκτελούνται και στο Άκρο του Νέφους με λίγους υπολογιστικούς πόρους και χρήση τεχνολογιών κβαντοποίησης αλγορίθμων (Tiny ML).

Λέξεις κλειδιά: υπολογιστικό νέφος, πολλοί πάροχοι, επεξεργασία σύνθετων συμβάντων, federated learning, machine learning, tiny ml, άκρο νέφους, παρακολούθηση νέφους.

SUMMARY

The present work for the final evaluation of the PhD Thesis focusses on innovative research areas that have to do with distributed applications hosted in various distributed infrastructures of many cloud providers.

After extensive research investigation, there are four basic research questions that finally came up as the main theme of the present PhD Thesis. Time wise, and up to intermediate evaluation of PhD thesis that took place, the first two questions that concern the dynamic monitoring of distributed applications in multi-cloud environments and the optimum reconfiguration of the infrastructure of the resources have been thoroughly investigated and analyzed. Apart from those, a third question that concerns the quite bigger accuracy of prediction in resources demand in multi-cloud environment by using federated learning techniques have been analyzed and presented during the final evaluation of PhD Thesis. Moreover, a fourth question regarding the calculation of prediction accuracy regarding the demand at the edge of the cloud with related resources restrictions is a case that has presented during the final evaluation of PhD Thesis.

As per the above, in the current PhD thesis specific solutions are proposed based on the distributed and multi-layer complex event systems for effective monitoring of the infrastructure. Moreover, there is also an innovative algorithm that is proposed for optimum resources management of the multicloud environment by focusing on the optimum operation of the cloud hosted distributed applications. Finally, as part of the specific PhD thesis, two more solutions are proposed. The one concern analysis and implementation of distributed federated learning algorithms in multi cloud application environments with very good prediction demand accuracy results. Moreover, there is the case of algorithms running at the edge of the cloud referring to many providers that are examined and analyzed as well. By that way securely and without any disruption in their operation, the distributed applications are used.

Our solution proposals are evaluated based on consumption of basic cloud resources such as RAM, CPU and Disk Usage and the stability and reliability of the i.e. monitoring system proposed. Regarding the decisioning model for the optimum reconfiguration of the cloud resources, measurements have taken place for the startup times of virtual machines of the public and private cloud providers. These measurements show stable performance for the public cloud providers. Moreover, results from experiments regarding the final values of the proposed algorithm for reconfiguration that takes into account startup times and application response times show the effectiveness of the new proposed method for cloud reconfiguration. What is more, distributed algorithms of federated learning for demand resources prediction show extremely good results in terms of prediction by using innovative client selection methods and technologies inspecting the adequate data volume for the training of the algorithms in multi cloud environments. Last but not least, very good results regarding the prediction accuracy of Federated Learning Architecture at the cloud edge using resources constraints and quantization methods (Tiny ML) have been successfully presented.

keywords: cloud computing, many cloud providers, complex event processing, federated learning, machine learning, tiny ml, cloud edge, cloud monitoring.

ΕΥΧΑΡΙΣΤΙΕΣ

Στα πλαίσια της ολοκλήρωσης του παρόντος πονήματος της Διδακτορικής Διατριβής μου, θα ήθελα να εκφράσω την μέγιστη ευγνωμοσύνη μου στον επιβλέποντα Καθηγητή μου κ. Γρ. Μέντζα. Χωρίς την δική του συμβολή , καθοδήγηση και στήριξη του σε όλες της φάσεις αλλά και τις προσωπικές οικογενειακές δυσκολίες που αντιμετώπισα, κατά την διάρκεια αυτής της Διατριβής, το τελικό επιτυχές αποτέλεσμα δεν θα ήτο δυνατόν. Του εκφράζω ένα απεριόριστο ευχαριστώ. Ένα πολύ μεγάλο ευχαριστώ εκφράζω και στον Καθηγητή κ. Ιωάννη Βεργινάδη για την αμέριστη συμπαράσταση και βοήθεια που μου παρείχε σε όλα τα επίπεδα όπως φυσικά και στον κ. Ιωάννη Πατινιωτάκη Έμπειρο Ερευνητή για τις καίριες βοήθειες και επισημάνσεις του.

Φυσικά ένα μεγάλο ευχαριστώ και στα υπόλοιπα μέλη της Τριμελής μου επιτροπής, Καθηγητές κκ. Παναγιώτη Τσανάκα και Ηλία Μαγκλογιάννη για την όλη στήριξη.

Η παρούσα Διατριβή αφιερώνεται στην μητέρα μου Δέσποινα όπως επίσης και εις μνήμην του πατέρα μου Δαμιανού, διότι μου παρείχαν τα πάντα σε αυτή τη ζωή.

ΠΕΡΙΕΧΟΜΕΝΑ

Γλωσσάριο

ΟΡΟΣ	ΕΠΕΞΗΓΗΣΗ
Federated Learning	Κατανεμημένος συνεργαζόμενος αλγόριθμος μάθησης
Edge Computing	Νεφουπολογιστικό σύστημα στο «άκρο»
Aggregation (of data)	Συγκέντρωση δεοδμένων
Deep Machine Learning	Βαθιά μηχανική μάθηση
Cloud	Νεφουπολογιστικό σύστημα
Middleware	Πλατφόρμα ενδιάμεσου επιπέδου μεταξύ presentation(εφαρμογής) και δεδομένων
Data storage	Αποθήκευση δεδομένων
Data integrity	Ανεξαρτησία δεδομένων
Data confidentiality	Διαφύλαξη δεδομένων
Data availability	Διαθεσιμότητα δεδομένων
Data privacy	Ιδιωτικότητα δεδομένων
“multi-cloud” ή “cross-cloud”	Νεφουπολογιστικό σύστημα πολλών παρόχων
Backup for disaster recovery	Λήψη αντιγράφων για καταστάσεις φυσικής καταστροφής
Servers/ storages /Virtual Machines	Διακομιστές / αποθηκευτικά μέσα/ Εικονικές Μηχανές
Value Added Services	Υπηρεσίες προστιθέμενης αξίας
Services Level Agreement- SLAs	Υπηρεσίες προσφοράς βάση συμφωνιών
horizontal scaling (scaling-in ή scaling-out) or vertical scaling	Οριζόντια ή κάθετη επέκταση στο υπολογιστικό νέφος
provisioning	παροχή
Reinforcement learning engine	Κεντρικοποιημένη μηχανή μάθησης και λήψης απόφασης
Controllers	Ελεγκτές
Global controller	Καθολικοί ελεγκτές
Hosts	Φυσικοί κόμβοι
Response time	Χρόνος Απόκρισης
Bandwidth	Εύρος Ζώνης
Hadoop File System	Σύστημα διαχείρισης αρχείων Hadoop
Tenant	Μισθωτής (λογισμικού)
Elasticity	Ελαστικότητα υποδομής

Complex Event Processing ή CEP	Επεξεργασία Σύνθετων Συμβάντων
Data patterns	Πρότυπα δεδομένων
CEP Engine	Μηχανή επεξεργασίας σύνθετων συμβάντων
Single-Point-of-Failure	Σημείο υψηλού ρίσκου αποτυχίας συστήματος
Throughput	Διεκπεραίωση κίνησης
Message broker	Μονάδα επεξεργασίας μηνυμάτων
Event Processing Network – EPN	Δίκτυο επεξεργασίας Γεγονότων
anomaly/outlier detection	Ανίχνευση ανωμαλιών δεδομένων
Service Oriented Architectures	Υπηρεσιοστραφείς Αρχιτεκτονικές
Event Processing Agents	Μονάδες Επεξεργασίας Γεγονότων
Event Channels	Κανάλια μετάδοσης γεγονότων
Enterprise Service Bus (ESB)	Σύστημα Μεταφοράς και Ολοκλήρωσης Δεδομένων σε Δίαυλο
Raw data	Πρωτογενή δεδομένα
Reconfiguration	Αναπροσαρμογή υποδομής
Software as a Service	Εφαρμογή προσφερόμενη ως υπηρεσία
Reconfiguration time	Χρόνος Αναπροσαρμογής
Key-values pair	Ζεύγος τιμών - κλειδιών
API (Application Programming Interface)	Διεπαφή Εφαρμογής
Spikes	Εκτινάξεις φορτίου

1 Εισαγωγή

1.1 Ερευνητικό Περιβάλλον

Η ερευνητική περιοχή της παρακολούθησης και διαχείρισης των τρόπων προσαρμογής κατανεμημένων εφαρμογών σε υποδομές πέραν του ενός παρόχων υπολογιστικού νέφους παρουσιάζει ιδιαίτερο ενδιαφέρον και προκλήσεις στην εποχή μας. Ειδικότερα η χρήση ειδικών τεχνικών παρακολούθησης μέσω σύνθετης επεξεργασίας συμβάντων, η χρήση συναρτήσεων αποφάσεων και αλγορίθμων προβλέψεων στην ζήτηση πόρων προκειμένου να υπάρχουν αναγκαίες προσαρμογές των εφαρμογών παρουσιάζουν μεγάλα ερευνητικά αποτελέσματα και καινοτομίες. Η χρήση συναρτήσεων μηχανικής μάθησης και συνεργαζόμενων αλγορίθμων βαθιάς μάθησης επίσης δίνει μεγάλη ακρίβεια σε θέματα προβλέψεων ζήτησης πόρων για κατανεμημένες εφαρμογές. Επιπλέον, η περιορισμένη υπολογιστική δυνατότητα αλλά και το μέγεθος της αποθήκευσης των συστημάτων που βρίσκονται στο άκρο του νεφουπολογιστικού συστήματος είναι ένα μεγάλο πρόβλημα προκειμένου να αυξηθεί η απόδοση των συστημάτων μάθησης και η δυνατότητά τους να τρέχουν σε περιορισμένους πόρους των συστημάτων στο άκρο του cloud περιβάλλοντος. Η έρευνα επομένως στην υλοποίηση ενός επαρκούς federated learning συστήματος σε περιβάλλον που βρίσκεται στο άκρο του υπολογιστικού χώρου (edge-cloud environment) είναι εξαιρετικά σημαντική όταν μάλιστα αυξάνει και την ακρίβεια προγνώσεων. Τέλος, ένα βασικό κομμάτι της έρευνας εδράζεται στην χρήση νεφουπολογιστικών περιβαλλόντων που συμπεριλαμβάνουν μεγάλους δημόσιους παρόχους όπως Amazon, Microsoft, Google κτλ. ταυτόχρονα αλλά και ιδιωτικά υπολογιστικά νέφη πχ. πάροχοι εργαστηρίων πανεπιστημίων κτλ.

Πολλές εργασίες σχετικές με επεξεργασία συμβάντων που αφορούν την καταγραφή και ανάλυση ροών δεδομένων για θέματα σχετιζόμενα με εφαρμογές και συνακόλουθα παραγωγή ειδοποιήσεων σχετικά με αυτά έχουν παρουσιάσει αξιόλογα αποτελέσματα. Η επεξεργασία σύνθετων συμβάντων αναφέρεται σε επεξεργασία δεδομένων που συνδυάζει δεδομένα με βάση κάποια πρότυπα και με σκοπό όταν ανιχνεύονται αυτά τα πρότυπα δεδομένων πληροφορίας να ενεργοποιούνται συγκεκριμένες διαδικασίες. Χαρακτηριστικά αναφέρονται εργασίες όπως του Hirzel [38], του Ku [39], του Flouris [41], του Paraiso [40] αλλά και άλλων που αναφέρονται εκτενώς στο ακόλουθο σχετικό κεφάλαιο 4.

Η δυνατότητα αναπροσαρμογής του νέφους και της χρήσης συγκεκριμένων μηχανισμών και συναρτήσεων αποφάσεων αυτής της αναπροσαρμογής είναι ένα πεδίο με επίσης έντονη ερευνητική δραστηριότητα στις μέρες μας. Η δυναμική αναπροσαρμογή των πόρων ανά ζήτηση τόσο από τους χρήστες σε επίπεδο εφαρμογής όσο και σύμφωνα και με το επεξεργαστικό φορτίο είναι ένα από τα μεγαλύτερα πλεονεκτήματα του υπολογιστικού νέφους. Ενδεικτικές εργασίες είναι αυτές των Mao και Humphrey [63], του Salfner [64], του Saniee [66] και πολλές περισσότερες που παρουσιάζονται αργότερα στο κεφάλαιο 5.

Μια άλλη ερευνητική περιοχή που εξετάζεται στην παρούσα Διατριβή αφορά ερευνητικές προσπάθειες που έχουν γίνει σχετικά με τους μηχανισμούς που προσφέρει το Federated Learning στην πρόβλεψη υπολογιστικών πόρων κυρίως σε περιβάλλοντα υπολογιστικού νέφους ακόμα και στο άκρο(edge) πολλών (περισσότερων του ενός) νεφουπολογιστκών παρόχων. Πολλοί παράμετροι είναι αυτοί μου μελετώνται σε πολλές ερευνητικές εργασίες και δημοσιεύσεις όπως ο σύγχρονος ή ασύγχρονος τρόπος επικοινωνίας, η χρήση convex ή non-convex συναρτήσεων απωλειών βαθιάς μηχανικής μάθησης (τοπικές loss functions), η

εξέταση διαφόρων μεθόδων για καθολικό aggregation, οι μέθοδοι επιλογής κόμβων που συμμετέχουν τελικά στο federation καθώς και η ακρίβεια πρόγνωσης που επιτυγχάνεται όπως και ο χρόνος ολοκλήρωσής της [89][90][103][104].

Τέλος, παρά την μεγάλη και ευρεία έρευνα στο χώρο του Federated Learning υπάρχει ένα κενό μεταξύ του Federated Learning και του Edge Learning και ειδικότερα μεταξύ του Tiny Machine Learning case σε συνδυασμό με το Federated Learning. Ειδικότερα το Tiny Machine Learning or Tiny ML υπόσχεται μια νέα ευκαιρία εκμάθησης μοντέλων με αρκετή ευχέρεια ειδικότερα όταν υπάρχει ένα cluster από διακομιστές (Servers) Federated Learning με χαμηλό αριθμό πόρων [149],[101]. Τα σημερινά πιο επιτυχημένα συστήματα Federated Learning που χρησιμοποιούν την περίπτωση των tiny systems εκτελούν την τεχνική του model Inference πάνω στην συσκευή την ίδια που βρίσκεται στο άκρο του υπολογιστικού νέφους[85]. Αρκετή έρευνα παρουσιάζεται στο συγκεκριμένο πεδίο κυρίως όταν η ακρίβεια προγνώσεων παρά τους περιορισμένους υπολογιστικούς πόρους δείχνει να αυξάνεται. Περισσότερες αναφορές και ανάλυση δίδεται στο κεφάλαιο 7.

1.2 Προκλήσεις και Συνεισφορά

Στην παρούσα αναφορά για την Τελική κρίση της Διδακτορικής Διατριβής αναλύονται διάφορα ερωτήματα που προκύπτουν από την σχετική έρευνα που έχει γίνει σε θέματα διαχείρισης και προσαρμογής κατανεμημένων εφαρμογών σε νεφουπολογιστικά περιβάλλοντα πολλών παρόχων. Παρουσιάζονται σχετικές προτάσεις επίλυσης τους με καινοτόμες τεχνικές. Πιο συγκεκριμένα παρουσιάζονται, αναλύονται και προτείνονται λύσεις για τα ακόλουθα ερωτήματα:

- Ποιος ο καταλληλότερος τρόπος για ευέλικτη παρακολούθηση κατανεμημένων εφαρμογών σε περιβάλλοντα υπολογιστικού νέφους πολλών παρόχων?
- Πώς μπορούμε να πετύχουμε βέλτιστη προσαρμογή των πόρων σε ένα περιβάλλον υπολογιστικού νέφους πολλών παρόχων?
- Πώς θα επιτευχθεί μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων που φιλοξενεί κατανεμημένη εφαρμογή με βέλτιστη αξιοποίηση της υπάρχουσας υποδομής?
- Πως επηρεάζεται η ακρίβεια στην πρόβλεψη ζήτησης πόρων σε περιπτώσεις υπολογισμών σε μικρές συσκευές στο Άκρο (edge) του νεφουπολογιστικού συστήματος πολλών παρόχων?

Πιο συγκεκριμένα, σχετικά με τα παραπάνω ερωτήματα και προκλήσεις, η Διατριβή προτείνει μια νέα τεχνική επεξεργασίας σύνθετων συμβάντων ακολουθώντας ιεραρχική Αρχιτεκτονική και διαμοιρασμό της επεξεργασίας των δεδομένων και του υπολογιστικού φόρτου σε διάφορα επίπεδα. Επίσης η εφαρμογή της συγκεκριμένης πρότασης σε περιβάλλοντα πολλών παρόχων υπολογιστικού νέφους και όχι μόνο σε ένα, αποτελεί ισχυρό πλεονέκτημά της. Αναφερόμενοι σε σχετική περιοχή, παρουσιάζεται μια άλλη πρόταση που αφορά την εφαρμογή καινοτόμου μεθοδολογίας για βέλτιστη δυναμική αναπροσαρμογή των πόρων σε κατανεμημένες εφαρμογές με βασική παράμετρο τον χρόνο. Και εδώ η μεθοδολογία θεωρεί την συνύπαρξη πολλών παρόχων υπολογιστικού νέφους την ίδια χρονική στιγμή. Επίσης μια άλλη ανάγκη που παρουσιάζεται στην παρούσα Διδακτορική εργασία είναι η δυνατότητα για πρόβλεψη της ζήτησης πόρων για την Αρχιτεκτονική του συστήματος που προαναφέρθηκε.

Μελετάται και υλοποιείται μια συγκεκριμένη δομή κατανεμημένου και όχι συγκεντρωτικού αλγορίθμου αρχιτεκτονικής Federated Learning με χρήση παράλληλα τοπικών συναρτήσεων βαθιάς μηχανικής μάθησης (Deep machine learning). Στόχος αυτής της ανάπτυξης είναι η επίτευξη μέγιστης ακρίβειας στην σωστή πρόβλεψη αναγκαίων πόρων νεφουπολογιστικού συστήματος κατανεμημένων εφαρμογών πολλών παρόχων. Και σε αυτή την πρόταση η μεθοδολογία θεωρεί την συνύπαρξη πολλών παρόχων υπολογιστικού νέφους την ίδια χρονική στιγμή. Τέλος, και προκειμένου να προταθεί μια καινοτόμος λύση όσον αφοράτην περιορισμένη υπολογιστική δυνατότητα αλλά και το μέγεθος της αποθήκευσης των συστημάτων που βρίσκονται στο άκρο του νεφουπολογιστικού συστήματος, εξετάζεται η απόδοση και η ακρίβεια συγκεκριμένα προγνώσεων των συστημάτων μάθησης και η δυνατότητά τους να τρέχουν σε περιορισμένους πόρους στο άκρο του cloud περιβάλλοντος. Πιο συγκεκριμένα, η αρχιτεκτονική εκτέλεσης του Inference βήματος ενός κατανεμημένου συστήματος στο άκρο (edge) του υπολογιστικού νέφους με χρήση τεχνολογίας Tiny ML σε περιβάλλον περιορισμένων επεξεργαστικών δυνατοτήτων εξετάζεται επιτυχώς με πολύ καλά αποτελέσματα. Η συνύπαρξη πολλών παρόχων υπολογιστικού νέφους την ίδια χρονική στιγμή θεωρείται μέρος της προτεινόμενης αρχιτεκτονικής λύσης.

1.3 Σχέση με τις δημοσιεύσεις

Μεγάλο μέρος της έρευνας που διεξήχθη στο πλαίσιο της διατριβής έχει παρουσιαστεί σε επιστημονικά συνέδρια και έχει δημοσιευτεί σε διεθνή περιοδικά. Ο πλήρης κατάλογος των δημοσιεύσεων και ανακοινώσεων δίνεται στο τέλος της διατριβής, στο παράρτημα «Δημοσιεύσεις Διατριβής» και αφορά τις δημοσιεύσεις σχετιζόμενες με την παρούσα Τελική Κρίση της Διατριβής.

1.4 Σχέση με Ερευνητικά Έργα

Η παρούσα Διδακτορική Διατριβή έχει υποστηριχθεί σε ένα μεγάλο κομμάτι από την Ευρωπαϊκή Επιτροπή μέσω του Ερευνητικού Έργου Horizon 2020 με τον αριθμό σύμβασης 731664 και με την ονομασία “Melodic”. Πιο συγκεκριμένα στο συγκεκριμένο έργο, αναπτύχθηκε μια πλατφόρμα επιπέδου middleware η οποία χρησιμοποιεί τα πλεονεκτήματα του υπολογιστικού νέφους για να εξυπηρετήσει εφαρμογές Μεγάλων Δεδομένων. Επίσης χρησιμοποιεί περιβάλλοντα δημόσιου αλλά και ιδιωτικού υπολογιστικού νέφους με στόχο δυναμική βέλτιστη χρήση πόρων, προσαρμοζόμενο στις ιδιωτικές ανάγκες του χρήστη και τις ζητούμενες υπηρεσίες αλλά και αποφεύγοντας ταυτόχρονα την εξάρτηση από έναν και μοναδικό πάροχο. Ένα κομμάτι της παρούσας Διδακτορικής Διατριβής έχει υποστηριχθεί και μέσω του Ερευνητικού Έργου Horizon 2020 με τον αριθμό σύμβασης 101070516 και με την ονομασία “ NebulOuS”. Το συγκεκριμένο έργο ανέπτυξε μια πλατφόρμα που αντιμετωπίζει προβλήματα fog συστημάτων σε συνδυασμό με πόρους σε περιβάλλοντα πολλών νεφουπολογιστικών παρόχων με σκοπό να αντιμετωπίσει προβλήματα καθυστερήσεων στην απόκριση των εφαρμογών. Προφανώς και εδώ η εξάρτηση από ένα μόνο πάροχο αποφεύγεται.

1.5 Δομή του κειμένου για την τελική κρίση της Διδακτορικής Διατριβής

Το παρόν πόνημα για την τελική κρίση της Διδακτορικής Διατριβής έχει δομηθεί ως εξής:

Στο κεφάλαιο 2 παρουσιάζονται οι βασικότερες ερευνητικές εργασίες και αποτελέσματα όσον αφορά τον γενικότερο τομέα της διαχείρισης πόρων υπολογιστικού νέφους πολλών παρόχων για κατανεμημένες εφαρμογές.

Στο κεφάλαιο 3 παρουσιάζεται η διαμόρφωση του προβλήματος και η διατύπωση των σχετικών ερευνητικών ερωτημάτων, όπου απαντώνται στην παρούσα τελική κρίση της διατριβής.

Στο κεφάλαιο 4 παρουσιάζεται η ερευνητική μελέτη για εύρεση κατάλληλου τρόπου ευέλικτης παρακολούθησης κατανεμημένων εφαρμογών σε υπολογιστικό νέφος. Η υλοποίηση αφορά ένα νέο σύστημα επεξεργασίας σύνθετων συμβάντων κατανεμημένης αρχιτεκτονικής το οποίο στο τέλος αξιολογείται με βάση πειραματικά δεδομένα μετρήσεων που γίνονται σε διάφορα νεφο-υπολογιστικά περιβάλλοντα πολλών παρόχων.

Στο κεφάλαιο 5 παρουσιάζεται η ερευνητική μελέτη για βέλτιστη προσαρμογή πόρων σε περιβάλλον υπολογιστικού νέφους πολλών παρόχων. Η υλοποίηση αφορά ένα σύστημα λήψης απόφασης για βέλτιστη προσαρμογή των πόρων υπολογιστικού νέφους με βασική παράμετρο το μέγεθος του χρόνου. Η αξιολόγηση που ακολουθεί βασίζεται σε πειραματικές μετρήσεις.

Στο κεφάλαιο 6 παρουσιάζεται η ερευνητική μελέτη για επίτευξη υψηλής ακρίβειας στην πρόγνωση ζήτησης πόρων υπολογιστικού νέφους για κατανεμημένες εφαρμογές σε περιβάλλοντα πέραν του ενός παρόχου. Οι τεχνικές που εξετάζονται είναι αυτές του federated learning. Η υλοποίηση αφορά ένα σύστημα αρχιτεκτονικής πολλών παρόχων server-client με τεχνική Federated Learning και τεχνικές με επιλογή των καταλλήλων clients και αποφυγή του θορύβου καθώς και μετατροπή του Inference. Η αξιολόγηση που ακολουθεί βασίζεται σε πειραματικές μετρήσεις που φανερώνουν τα θετικά αποτελέσματα της υλοποιημένης αρχιτεκτονικής.

Στο κεφάλαιο 7 παρουσιάζεται η ερευνητική μελέτη για επίτευξη υψηλής ακρίβειας στην πρόγνωση ζήτησης πόρων υπολογιστικού νέφους για κατανεμημένες εφαρμογές σε περιβάλλοντα πέραν του ενός παρόχων ευρισκόμενες όμως στο άκρο (Edge) του υπολογιστικού νέφους. Οι τεχνικές που εξετάζονται είναι αυτές του federated learning σε συνδυασμό με το Tiny ML που αφορά εκτέλεση αλγορίθμου σε συσκευές με μικρή μνήμη ή/και επεξεργαστική ισχύ. Η υλοποίηση αφορά ένα σύστημα αρχιτεκτονικής πολλών παρόχων server-client κατανεμημένου αλγορίθμου με τεχνική Federated Learning και Inference Tiny ML εκτέλεση στο Άκρο(Edge) του υπολογιστικού νέφους. Τα πειραματικά αποτελέσματα φανερώνουν τα θετικά αποτελέσματα της υλοποιημένης αρχιτεκτονικής κυρίως στο κομμάτι των προγνώσεων και της εξοικονόμησης πόρων.

Στο τελευταίο κεφάλαιο 8 του κειμένου της τελικής κρίσης για την Διδακτορική Διατριβή παρουσιάζονται τα συμπεράσματα της έρευνας που έχει διεξαχθεί και υλοποιημένων προτάσεων μαζί με σχετικές αξιολογήσεις.

2 Επισκόπηση Βιβλιογραφίας

2.1 Εισαγωγή

Στην σημερινή εποχή το υπολογιστικό νέφος έχει αναγνωριστεί ως η κύρια μέθοδος και το βασικό στάνταρ για φιλοξενία και προσφορά υπηρεσιών μέσω Διαδικτύου. Ειδικότερα, το υπολογιστικό νέφος δεν είναι μια νέα τεχνολογία από μόνη της αλλά θα έλεγε κάποιος ότι αποτελεί ένα νέο τρόπο να συμπεριληφθούν και να χρησιμοποιηθούν υπάρχουσες τεχνολογίες [1]. Οι πλατφόρμες μέσω υπολογιστικού νέφους υιοθετούνται ολοένα και περισσότερο λόγω του ότι παρέχουν πολλά οφέλη όπως χρέωση ανά χρήση (βελτιστοποίηση κόστους), επεκτασιμότητα μεγαλύτερη από την φιλοξενία τους σε παραδοσιακά κέντρα δεδομένων, μεγαλύτερη διαθεσιμότητα σε πόρους και αποθήκευση πχ δεδομένων με την δυνατότητα πρόσβασης από οποιαδήποτε σημείο, καλύτερη επαναφορά από βλάβη και διασφάλιση υψηλής διαθεσιμότητας υπηρεσίας. Με άλλα λόγια οι πλατφόρμες μέσω υπολογιστικού νέφους προσφέρουν ευελιξία όσον αφορά τη δυνατότητα να καλύπτουν αποτελεσματικά τόσο τη σταδιακή ανάπτυξη όσο και την κυκλική ζήτηση. Το νέφος παρέχει ευέλικτες δυνατότητες υπολογιστικής με διακομιστές και λογισμικό, αποθηκευτικές δυνατότητες και δυνατότητες επικοινωνίας μέσω διαδικτυακών συνδέσεων. Η υπολογιστική νέφους επιτρέπει σε κάθε πελάτη να αυξάνει ή να μειώνει τη χρήση υπηρεσιών νέφους ανά πάσα στιγμή. Ο πελάτης πληρώνει μόνο για τις υπηρεσίες που χρησιμοποιεί. Ωστόσο η χρησιμοποίηση ενός και μόνο παρόχου υπολογιστικού νέφους για παροχή υπηρεσιών μέσω Ιντερνέτ παρουσιάζουν κάποιες δυσκολίες ως ακολούθως:

- Ασφάλεια δεδομένων και ιδιωτικότητα: Σε πολλές αναφορές [1,2,3,4] θέματα ασφάλειας πληροφοριών που αφορούν το υπολογιστικό νέφος έχουν αναφερθεί και αναλυθεί. Συγκεκριμένα αναφέρονται θέματα ελλιπούς ασφάλειας σε τομείς του data storage, data integrity, data confidentiality, data availability, data privacy.
- Νομικές αναφορές σχετικά με κανόνες για αποθήκευση δεδομένων ανάλογα με την τοποθεσία.
- Περιπτώσεις απώλειας-καταστροφής δεδομένων π.χ. εάν ένας πάροχος.

  υπολογιστικού νέφους χρεωκοπήσει τι θα απογίνουν τα δεδομένα που φιλοξενεί?
- Δυσκολίες στην δια-συνδεσιμότητα μεταξύ διαφόρων συστημάτων.
- Αργή μεταφορά δεδομένων.
- Η προσήλωση σε έναν μόνο πάροχο υπολογιστικού νέφους ειδικά όταν αναφερόμαστε σε πελάτες που υποχρεώνονται να πληρώνουν υψηλά κόστη αποτελεί πρόβλημα.

Προκειμένου να μπορέσουν να περιοριστούν κάποια από τα θέματα που αναφέρθηκαν, η λογική μετάβαση και εξέλιξη είναι σε ένα περιβάλλον υπολογιστικού νέφους όπου θα υπάρχουν περισσότεροι του ενός πάροχοι υπηρεσιών υπολογιστικού νέφους. Αυτό το περιβάλλον ονομάζεται “multi-cloud” ή “cross-cloud”. Με αυτό τον τρόπο διάφοροι συνδυασμοί σε υποδομές και πόρους προσφέρουν καλύτερες λύσεις. Συγκεκριμένα τα ακόλουθα χαρακτηριστικά πλεονεκτήματα παρουσιάζονται σε περιβάλλοντα όπου λειτουργούν περισσότεροι του ενός πάροχοι υπολογιστικού νέφους:

- Επιτυχής διαχείριση αυξημένης ζήτησης σε υπηρεσίες και αιτήματα (requests) χρησιμοποιώντας εναλλακτικές τρίτες πηγές.
- Βελτιστοποίηση κόστους.
- Βελτίωση της ποιότητας των προσφερόμενων υπηρεσιών υπολογιστικού νέφους.
- Προσαρμογές σε αλλαγές από τους παρόχους του υπολογιστικού νέφους.
- Συμμόρφωση με νέους νόμους και περιορισμούς όπως πχ. νέες τοποθεσίες.
- Αποφυγή της εξάρτησης από ένα μόνο εξωτερικό πάροχο υπολογιστικού νέφους.
- Εξασφάλιση υψηλής διαθεσιμότητας προσφερόμενων υπηρεσιών και πόρων.
- Καλύτερη διασφάλιση backup (λήψη αντιγράφων) σε καταστάσεις disaster recovery (ανάκαμψη από φυσική καταστροφή).
- Δυνατότητα επαυξημένων υπηρεσιών αναλόγως των συμφωνιών με άλλους παρόχους.

Σε μεγάλα (cloud) νεφουπολογιστικά περιβάλλοντα συνύπαρξης πολλών παρόχων υπολογιστικού νέφους έχει ιδιαίτερη σημασία η αποτελεσματική διαχείριση των πόρων καθώς αυτή επηρεάζει τόσο την απόδοση των κατανεμημένων εφαρμογών όσο και τα κόστη της συντήρησης αξιόπιστων υπηρεσιών για τους τελικούς χρήστες. Το σημαντικότερο σημείο είναι να βρίσκεται μια χρυσή τομή ανάμεσα στους διαθέσιμους πόρους και την βιώσιμη παροχή υπηρεσιών και εφαρμογών με προβλεπόμενη απόδοση. Ταυτόχρονα στόχος είναι η επίτευξη αποδοτικότερης υποδομής από άποψη κόστους και ενεργειακής κατανάλωσης των κέντρων δεδομένων. Πολλοί είναι οι τρόποι διαχείρισης πόρων που περιλαμβάνονται σε αυτή την διαδικασία βελτιστοποίησης χρήσης τους:

- Διαχείριση εικονικών μηχανών.
- Διαχείριση συν-φιλοξενίας πολλών εφαρμογών σε κοινή υποδομή υπολογιστικού νέφους ή εικονικής μηχανής.
- Αυτόματη διαχείριση παροχής υπηρεσιών και ευελιξίας αύξησης ή ελάττωσης της υποδομής.
- Διαχείριση της κατεύθυνσης υπολογιστικής κίνησης και της ετερογένειας του δικτύου της υποδομής υπολογιστικού νέφους.
- Διαχείριση της ενέργειας που καταναλώνεται στα κέντρα δεδομένων.
- Διαχείριση της αποθήκευσης δεδομένων.
- Διαχείριση της απόδοσης των εφαρμογών.

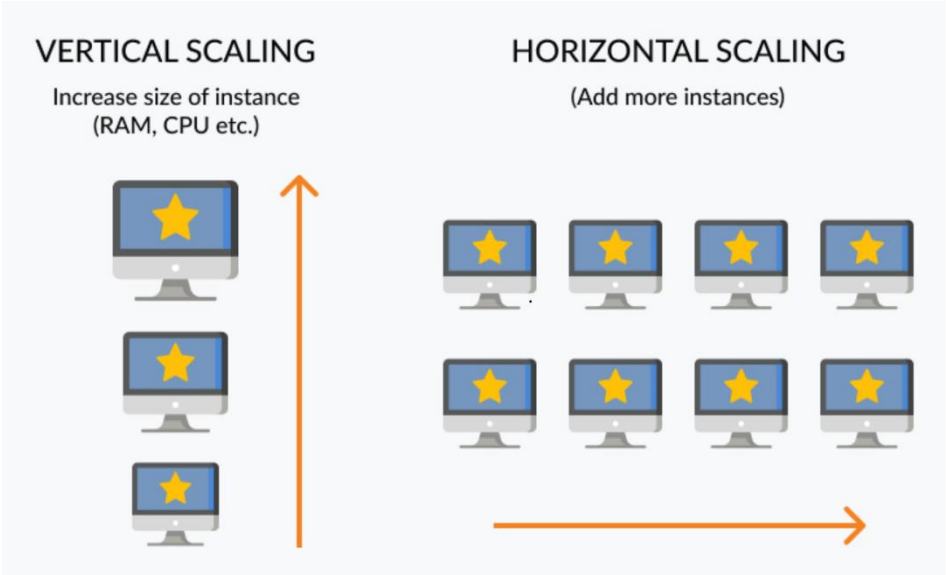
2.2 Θεωρίες διαχείρισης πόρων υπολογιστικού νέφους πολλών παρόχων για κατανεμημένες εφαρμογές

Με την επαυξημένη ανάγκη για παροχή υπολογιστικών πόρων σε χαμηλό σχετικά κόστος για τους τελικούς χρήστες οι πάροχοι υποδομών υπολογιστικού νέφους αναζητούν τρόπους να διατηρήσουν και να προστατέψουν τα έσοδα τους. Ένας βασικός τρόπος να επιτευχθεί αυτό

είναι μέσω της διατήρησης σε χαμηλό επίπεδο του λειτουργικού κόστους των υποδομών τους. Μια πολλά υποσχόμενη μεθοδολογία για να επιτευχθεί αυτό είναι να αναπροσαρμόζεται συνεχώς η χρήση των πόρων του Νεφουπολογιστικού Συστήματος ανάλογα με τις ανάγκες του φορτίου των υπηρεσιών που τρέχουν οι τελικοί χρήστες.

Για παράδειγμα ύστερα από συγκεκριμένη ανάλυση μπορεί να προκύψει ότι χρειάζεται να κλείσουν συγκεκριμένοι διακομιστές (servers) αν βρεθεί ότι υπολειτουργούν ελαττώνοντας έτσι και την σχετιζόμενη με αυτούς κατανάλωση ισχύος. Οι πάροχοι υποδομών και υπηρεσιών υπολογιστικού Νέφους διαχειρίζονται την υποδομή τους που αφορά διακομιστές (servers), αποθηκευτικά μέσα (storages) και διασύνδεση δικτυακή, μέσω της χρήσης τεχνολογίας εικονικών μηχανών (διακομιστών – Virtual Machines).

Επιπλέον πολλοί πάροχοι υπηρεσιών νοικιάζουν τις παραπάνω προσφερόμενες υπηρεσίες υποδομών και τις μετά-πωλούν παρέχοντας υπηρεσίες παροχής ηλεκτρονικών πλατφορμών ή παροχής υπηρεσιών λογισμικού μέσω του υπολογιστικού νέφους. Αυτό τους εξασφαλίζει την δυνατότητα να παρέχουν υπηρεσίες προστιθέμενης αξίας (Value Added Services). Εκείνο που πρέπει να τονιστεί είναι ότι πάροχοι υποδομών υπολογιστικού νέφους παράγουν έσοδα μέσω της ικανοποιήσεως συγκεκριμένων συμφωνιών για συμφωνημένη ποιότητα επιπέδου υπηρεσιών που παρέχουν στους πελάτες (Services Level Agreement- SLAs). Ο τρόπος να το επιτύχουν αυτό είναι με την περιοδική προσαρμογή της υποδομής τους και κυρίως αφορά την απόφαση για horizontal scaling (scaling-in ή scaling-out) ή vertical scaling της υποδομής τους αναλόγως του φορτίου που παράγεται από τις υπηρεσίες που χρησιμοποιούνται, χωρίς φυσικά να διαταράσσονται σε καμία περίπτωση τα επίπεδα των Services Level Agreement - SLAs. Με αυτό τον τρόπο η εξοικονόμηση πόρων που επιτυγχάνεται μπορεί να χρησιμοποιηθεί για να εξυπηρετήσει νέα φορτία υπηρεσιών ή να ελαττώσει την κατανάλωση ισχύος οδηγώντας σε ένα αυξημένο τελικό κέρδος για τον εκάστοτε πάροχο. Οι δυνατότητες προσαρμογής για οριζόντια (horizontal) ή κάθετη (vertical scaling) ανάπτυξη/επέκταση φαίνονται γραφικά ως ακολούθως στο σχήμα 1:



Σχήμα 1: Οριζόντια και κάθετη ανάπτυξη/επέκταση υποδομής υπολογιστικού νέφους

2.2.1 Διαχείριση Επεξεργαστικής Ισχύος και Μνήμης

Ως βασικοί πόροι του Υπολογιστικού Νέφους η Επεξεργαστική ισχύς και η Μνήμη έχουν χρησιμοποιηθεί και ερευνηθεί πάρα πολύ τα τελευταία χρόνια. Παρότι πολλή βιβλιογραφία αναλώνεται στην απλά «οριζόντια» επέκταση (horizontal scale-in/out) του υπολογιστικού νέφους απλά προσθέτοντας προκαθορισμένου τύπου εικονικές μηχανές όπως στην περίπτωση του Lama & Zhou [5] η οποία είναι μια σχετικά εύκολα διαχειριστική διαδικασία, εντούτοις συγκρινόμενη με την βελτιστοποίηση της χρήσης της επεξεργαστικής ισχύος και της μνήμης, θεωρείται ότι οδηγεί σε χάσιμο πόρων του υπολογιστικού νέφους και όχι βέλτιστη χρήση τους όπως επίσης και σε μεγαλύτερη δαπάνη ενέργειας. Κατά συνέπεια το συμπέρασμα της συγκεκριμένης δημοσίευσης είναι ότι μια «κάθετη» (vertical) προσαρμογή των πόρων πχ. Επεξεργαστικής ισχύος, μνήμης κτλ. μιας εικονικής μηχανής είναι συν-εκτιμωμένων και των συνθηκών πιο συμφέρουσα από την απλά μόνο «οριζόντια» επέκταση.

Στην εργασία που αφορά την δημοσίευση των Dawoud, Takouna and Meinel [6] στο περιοδικό “Global Trends Computing Communication Systems”, οι συγγραφείς προτείνουν ξεκάθαρα το λεπτομερή καθορισμό του εύρους της επεξεργαστικής δυνατότητας και της μνήμης προκειμένου να αποφευχθεί ο υπερπλεονασμός για provisioning (παροχή) νέων εικονικών μηχανών όπως επίσης και να αποφευχθούν παραβιάσεις σε όρους παροχής υπηρεσιών συμβολαίων (Services Level Agreement).

Στην εργασία των Addis, Ardagna, Panicucci, Squillante and Zhang [7] και ακολουθώντας μια πιο πρακτική αντιμετώπιση προβλημάτων οι συγγραφείς αναφέρονται σε μια καινοτόμα μεθοδολογία προκειμένου να παρατάσσουν μια κατανεμημένη εφαρμογή σε διάφορα nodes όπως αναφέρει (εικονικές μηχανές), προκειμένου όμως παράλληλα να διατηρούν την χρήση της μονάδας επεξεργαστικής ισχύος (CPU) σε κάθε εικονική μηχανή χαμηλότερα από το όριο του 60%. Ένας αλγόριθμος τοπικής αναζήτησης βελτιστοποιεί την αρχική τοποθέτηση/παράταξη της κατανεμημένης εφαρμογής στις διάφορες εικονικές μηχανές με βάση κάποιους περιορισμούς που έχουν τεθεί αρχικά όπως πχ αυτό της επεξεργαστικής ισχύος.

Στην εργασία [10] του Han R, προτείνεται ένα απλό και πρακτικό μοντέλο διαχείρισης πόρων βασισμένο στην χρονική απόκριση της εφαρμογής, που λαμβάνεται με βάση την εμπειρία του πελάτη, αλλά και με γνώμονα το υπολογιστικό κόστος ή φορτίο. Η συγκεκριμένη προσέγγιση και αλγόριθμος προσπαθεί να ικανοποιήσει τις αυξανόμενες ανάγκες λόγω μεγαλύτερου υπολογιστικού φορτίου δίνοντας ένα καλύτερο χρόνο απόκρισης της υπηρεσίας που προσφέρεται στον πελάτη αυξάνοντας σταδιακά την επεξεργαστική ισχύ και την χρησιμοποιούμενη μνήμη στο νεφουπολογιστικό σύστημα. Σε αυτή την αναπροσαρμογή των πόρων του νεφουπολογιστικού συστήματος λαμβάνονται υπόψιν και συγκεκριμένοι περιορισμοί που αφορούν τον πελάτη όπως κόστη και προϋπολογισμοί για να είναι δυνατή η εκάστοτε επέκταση.

Με βάση την έως τώρα βιβλιογραφία και έρευνα, οι πιο πολλές αρχιτεκτονικές νεφουπολογιστικών συστημάτων αφορούν κεντρικοποιημένες αρχιτεκτονικές διαχείρισης, όπου η Μνήμη και η Επεξεργαστική ισχύς αλλάζουν με μια εντολή που δίνεται τελικά κεντρικά και διαχέεται σε όλη την υποδομή σε όλους τους σχετικούς πόρους. Σε αυτή την περίπτωση αρκετές φορές παρουσιάζονται καθυστερήσεις στο τελικό επανακαθορισμό όλης της νεφουπολογιστικής υποδομής όταν μάλιστα μιλάμε για χιλιάδες πόρους. Χαρακτηριστική είναι η περίπτωση που έχει ήδη αναφερθεί σε εργασία των Bu, Rao, Xu [8], όπου οι συγγραφείς χρησιμοποιήσαν ένα κεντρικοποιημένο σύστημα λήψης απόφασης

(reinforcement learning engine) όπου έτρεχε τον σχετικό αλγόριθμο και παρατηρήθηκε ότι ο χρόνος που χρειάζεται προκειμένου να σταθεροποιηθεί η αναπροσαρμοσμένη υποδομή αυξάνει με το μέγεθος και τον αριθμό των πόρων ή αλλιώς το cluster size δηλαδή μέγεθος της υποδομής.

Για να βελτιωθεί η επεκτασιμότητα και η απόκριση των μηχανών αποφάσεων ερευνητές ασχολήθηκαν και με αποκεντρωμένες αρχιτεκτονικές μηχανών λήψης αποφάσεων και σχετικών αλγορίθμων όπως ιεραρχικές και κατανεμημένες. Στην περίπτωση της εργασίας του Jung G. [9] οι συγγραφείς προτείνουν μια ιεραρχική δομή από ελεγκτές (controllers) που είναι μοιρασμένοι σε διάφορα ομαδοποιημένα σύνολα μηχανών (clusters) όπου το κάθε cluster το διαχειρίζεται ένας τοπικός controller. Οι ιεραρχικής δομής controllers τρέχουν σε διάφορα χρονικά διαστήματα με έναν τοπικό σε cluster controller να τρέχει πιο συχνά για την εξαγωγή συμπερασμάτων σε σχέση με έναν global controller. Με αυτό τον τρόπο οι χρόνοι προκειμένου να ληφθεί η απόφαση αναπροσαρμογής των πόρων του νεφουπολογιστικού συστήματος και σταθεροποίησης της νέας υποδομής μετά μειώνονται αισθητά εν σχέση με την κεντρικοποιημένες αρχιτεκτονικές που μελετήθηκαν σε προηγούμενες εργασίες.

Συγκριτικά οι ανωτέρω εργασίες και μελέτες παρουσιάζονται στον παρακάτω πίνακα 1:

ΕΡΓΑΣΙΑ	*Κεντρικοποιημένη Αρχιτεκτονική*	*Κατανεμημένη Αρχιτεκτονική*	*Κάθετη Επέκταση (Vertical Scaling)*	*Οριζόντια Επέκταση (Horizontal Scaling)*	*Χρήση περιορισμών και αλγορίθμων αποφάσεων*
Lama P, Zhou X (2012) Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud.	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***
Dawoud W, Takouna I, Meinel C (2011) Elastic virtual machine for fine-grained cloud resource provisioning.	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***
Addis B, Ardagna D, Panicucci B, Squillante MS, Zhang L (2013) A hierarchical approach for the resource management of very large cloud platforms	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***
Han R, Guo L, Ghanem MM, Guo Y (2012) Lightweight resource scaling for cloud applications	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΝΑΙ***
Bu X, Rao J, Xu C-Z (2011) Model-free learning approach for coordinated configuration of virtual machines and appliances.	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***
Jung G, Hiltunen MA, Joshi KR, Schlichting RD, Pu C (2010) Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures.	***ΌΧΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***

Πίνακας 1: Σύγκριση εργασιών για διαχείριση εικονικών μηχανών και πόρων

2.2.3 Μεταφορά των Εικονικών Μηχανών και προσαρμογή του μεγέθους των κόμβων

Οι φυσικοί κόμβοι (host) των νεφουπολογιστικών συστημάτων έχουν την δυνατότητα γενικά να φιλοξενούν εικονικές μηχανές σε μεγάλο αριθμό. Παρόλα αυτά κάποια στιγμή κάποιες από αυτές τις εικονικές μηχανές αυξάνουν τόσο πολύ την ανάγκη κατανάλωσης σε πόρους (μνήμη, επεξεργαστική ισχύς) λόγω επαυξημένης πχ. ζήτησης από τις εφαρμογές που φιλοξενούν, ώστε αυτές οι ανάγκες να μην μπορούν να καλυφθούν από τους συγκριμένους κόμβους. Έτσι εμφανίζεται η περίπτωση για μεταφορά (migration) συγκεκριμένων εικονικών μηχανών από έναν κόμβο σε άλλο κόμβο (host) ο οποίος διαθέτει την απαραίτητη επάρκεια σε πόρους. Στην διεθνή βιβλιογραφία εξετάζεται πως αυτό το migration μπορεί να συμβαίνει σε ομογενές (ενός παρόχου) ή όχι (πολλών παρόχων) νεφουπολογιστικό περιβάλλον και να επηρεάζει διάφορες μετρικές όπως πχ. την ταχύτητα απόκρισης (response time) φιλοξενούμενων εφαρμογών σε αυτές τις μεταφερόμενες εικονικές μηχανές. Μια άλλη παράμετρος που εξετάζεται είναι η μεταβολή-μείωση στην κατανάλωση ισχύος αυτών των μεταφερόμενων εικονικών μηχανών και φυσικά η όσο το δυνατόν λιγότερη παραβίαση συμβολαίων φιλοξενούμενων υπηρεσιών στο νεφουπολογιστικό σύστημα των πελατών πχ. Services Level Agreement.

Η πρώτη σημαντική αναφορά που εξετάζεται σχετικά με την τεχνική και τις μεθόδους επαναδιαμόρφωσης πόρων του Νεφουπολογιστικού Συστήματος δια μέσου της μεταφοράς εικονικών μηχανών είναι η εργασία [11]. Σε αυτή την εργασία χρησιμοποιείται μια πολύ πρακτική μέθοδος προσαρμογής και εξετάζονται 3 πολιτικές: ελαχιστοποίησης των μεταφορών των εικονικών μηχανών, της μεγιστοποίησης της ανάπτυξης τους (των μεταφορών) και της τυχαίας επιλογής μεταφοράς τους.

Η βασική ιδέα είναι να μετακινούνται πρώτα όλα τα Virtual Machines (εικονικές μηχανές) από κόμβους που φιλοξενούνται (φυσικές μηχανές) και υπό-χρησιμοποιούνται και τελικά μπορούν να αποσυρθούν ή και να μετακινούνται και κάποια Virtual Machines από κόμβους που είναι πολύ φορτωμένοι από φορτίο (over-loaded). Τέλος, τα Virtual Machines που έχουν μετακινηθεί και διατηρούνται ενεργά (δεν αποσύρονται) θα πρέπει με την βοήθεια κάποιου αλγορίθμου να τοποθετηθούν σε έναν νέο επιλεγμένο κόμβο (host). Για την τελευταία αυτή ανάγκη ο αλγόριθμος που λέγεται “Modified Best Fit Decreasing” (MBFD) κατηγοριοποιεί τα Virtual Machines σε φθίνουσα σειρά όσον αφορά το φόρτο και τοποθετεί καθένα από αυτά στην φυσική μηχανή με την καλύτερη ενεργειακή απόδοση και επαρκή χώρο σε πόρους για να φιλοξενηθεί. Σχετικά με την περίπτωση μεταφοράς Virtual Machines εκτός κόμβου που έχει υπερφορτωθεί, διάφοροι «heuristics αλγόριθμοι» έχουν χρησιμοποιηθεί στην βιβλιογραφία και συγκεκριμένα:

- Ο αλγόριθμος ελάχιστης μεταφοράς Virtual Machines (εικονικών μηχανών) ώστε παράλληλα ο φυσικός κόμβος που τα φιλοξενεί να αποσυμφορηθεί. Ενδεικτικά παρουσιάζεται ακολούθως στο σχήμα 2:

```
1  Input: hostList  Output: migrationList
2  foreach h in hostList do
3      vmList ← h.getVmList()
4      vmList.sortDecreasingUtilization()
5      hUtil ← h.getUtil()
6      bestFitUtil ← MAX
7      while hUtil > THRESH_UP do
8          foreach vm in vmList do
9              if vm.getUtil() > hUtil − THRESH_UP then
10                 t ← vm.getUtil() − hUtil + THRESH_UP
11                 if t < bestFitUtil then
12                     bestFitUtil ← t
13                     bestFitVm ← vm
14             else
15                 if bestFitUtil = MAX then
16                     bestFitVm ← vm
17                 break
18         hUtil ← hUtil − bestFitVm.getUtil()
19         migrationList.add(bestFitVm)
20         vmList.remove(bestFitVm)
21     if hUtil < THRESH_LOW then
22         migrationList.add(h.getVmList())
23         vmList.remove(h.getVmList())
24 return migrationList
```

Σχήμα 2: Αλγόριθμος ελάχιστης μεταφοράς εικονικών μηχανών

- Ο αλγόριθμος Highest Potential Growth (HPG) επιλέγει τα Virtual Machines (εικονικές μηχανές) που έχουν το χαμηλότερο λόγο τρέχοντος φορτίου. Ενδεικτικά παρουσιάζεται ακολούθως στο σχήμα 3:

$$R = \begin{cases} \left\{ S | S \in \mathcal{P}(V_j), u_j - \sum_{v \in S} u_a(v) < T_u, \right. \\ \left. \sum_{v \in S} \frac{u_a(v)}{u_r(v)} \rightarrow \min \right\}, & \text{if } u_j > T_u; \\ V_j, & \text{if } u_j < T_l; \\ \emptyset, & \text{otherwise} \end{cases}$$

Σχήμα 3: Αλγόριθμος Highest Potential Growth (HPG)

- Ο αλγόριθμος Random Choice (RC) που επιλέγει με τυχαίο τρόπο τα Virtual Machines που θα μετακινηθούν. Ενδεικτικά παρουσιάζεται ακολούθως στο σχήμα 4:

$$R = \begin{cases} \left\{ S|S \in \mathcal{P}(V_j), u_j - \sum_{v \in S} u_a(v) < T_u, \right. \\ \qquad \left. X \stackrel{d}{=} U(0, |\mathcal{P}(V_j)| - 1) \right\}, & \text{if } u_j > T_u; \\ V_j, & \text{if } u_j < T_l; \\ \emptyset, & \text{otherwise} \end{cases}$$

Σχήμα 4: Αλγόριθμος Random Choice (RC)

Το συμπέρασμα που προκύπτει είναι ότι η πολιτική για ελαχιστοποίηση μεταφορών των εικονικών μηχανών μπορεί να εξασφαλίσει οικονομίες στην ενεργειακή κατανάλωση. Βέβαια υπάρχουν και περιπτώσεις όπου παρατηρούνται μικρές παραβιάσεις σε όρους των Services-Level-Agreement προκειμένου να επιτευχθεί η εξοικονόμηση ενέργειας. Η πολιτική ελαχιστοποίησης μεταφοράς εικονικών μηχανών επιλέγει εικονικές μηχανές με την υψηλότερη κατανάλωση επεξεργαστικής ισχύος προκειμένου να μεταφέρει τις συγκεκριμένες μηχανές σε άλλους φυσικούς κόμβους (hosts). Το μειονέκτημα αυτής της μεθόδου είναι ότι μεταφέρονται εικονικές μηχανές οι οποίες είναι ήδη σε ρίσκο παραβίασης των Services-Level-Agreement των πελατών εξαιτίας της υψηλής κατανάλωσης επεξεργαστικής ισχύος και επιπλέον αυξάνει το συγκεκριμένο ρίσκο λόγω της επικινδυνότητας που προστίθεται εξαιτίας του κόστους της εν λειτουργία μετάβασης (“live migration”) της συγκεκριμένης μηχανής που μεταφέρεται.

Τέλος, η εργασία [12] παρουσιάζει μια λιγότερο γνωστή και λιγότερο συνηθισμένη τεχνική για μεταφορά εικονικών μηχανών χρησιμοποιώντας ανάλυση χρονοσειρών προκειμένου να προβλεφθεί η ανάγκη για μεταφορά πόρων δια μέσου ενός Fast Fourier Transform[78] αλγορίθμου. Οι συγγραφείς στην συγκεκριμένη εργασία προβλέπουν και εκτελούν μέσω του αλγορίθμου την μετακίνηση της εικονικής μηχανής πριν αυτή χρειαστεί και συγκεκριμένα πριν υπερφορτωθεί ώστε να ελαχιστοποιούν το κόστος της μετακίνησης. Σε αντίθετη περίπτωση μη έγκαιρης μετακίνησης, η καθυστερημένη μεταφορά μιας υπερφορτωμένης εικονικής μηχανής συνήθως προκαλεί προβλήματα διαθεσιμότητας της υπηρεσίας που παρέχει προς τους τελικούς χρήστες με αποτέλεσμα μικρές παραβιάσεις σε όρους των Services-Level-Agreement.

2.2.3 Προσαρμογή του χώρου και της υποδομής Αποθήκευσης Δεδομένων

Η πιο συνηθισμένη μέθοδος που χρησιμοποιείται ερευνητικά για την προσαρμογή σε όλα τα επίπεδα των πόρων αποθηκευτικών μέσων δεδομένων είναι η θεωρία Ελέγχου συστημάτων. Συγκεκριμένα, η εργασία [13] χρησιμοποιεί ελεγκτές εφαρμογών (application controllers) με χρήση αλγορίθμων ανάδρασης για να υπολογίσει το απαιτούμενο εύρος ζώνης (Bandwidth) για Input/Output που χρειάζεται ο εκάστοτε κόμβος που φιλοξενεί αποθηκευτικό χώρο δεδομένων. Επιπλέον, στην εργασία [14] χρησιμοποιείται η θεωρία ελέγχου προκειμένου να προσαρμόζεται κάθε φορά το κεντρικό αποθηκευτικό σύστημα δεδομένων με την βοήθεια του Hadoop File System. Σε αυτή την αρχιτεκτονική χρησιμοποιείται αλγόριθμος που χρησιμοποιεί μη πραγματικού χρόνου δεδομένα (offline profiling dataset) προκειμένου να «εκπαιδευτεί» και να διαμορφώσει τα σωστά βάρη και άρα την σωστή συνάρτηση μεταφοράς του μοντέλου που χρησιμοποιεί. Αυτή η συνάρτηση μεταφοράς αργότερα

χρησιμοποιεί και πραγματικού χρόνου (OnLine) δεδομένα από μετρήσεις κατανάλωσης επεξεργασίας (CPU) των φυσικών κόμβων(host) που φιλοξενούν τους αποθηκευτικούς χώρους δεδομένων και παράλληλα ανανεώνει (update) τα σχετικά αρχεία δεδομένων (dataset). Τελικά, η επαναδιαμόρφωση των αποθηκευτικών χώρων γίνεται σε 2 φάσεις:

- Ο πρώτος controller του αλγορίθμου που χρησιμοποιείται προσθέτει είτε αφαιρεί κόμβους οι οποίοι φιλοξενούν αποθηκευτικά μέσα.
- Ο δεύτερος controller του αλγορίθμου που χρησιμοποιείται φροντίζει για την σωστή κατανομή των δεδομένων στην νέα δομή των κόμβων που έχει διαμορφωθεί.

Με βάση την ανωτέρω λογική επιτυγχάνεται η βέλτιστη επαναδιαμόρφωση του νεφουπολογιστικού συστήματος όσον αφορά την σωστή διαμόρφωση των αποθηκευτικών χώρων δεδομένων.

2.2.4 Multi-tenancy

Multi-tenancy είναι μια βασική λειτουργία του υπολογιστικού νέφους και ορίζεται ως η δυνατότητα όπου οι κατανεμημένες και μη κατανεμημένες εφαρμογές που ανήκουν σε διαφορετικούς χρήστες «συν-φιλοξενούνται» σε μια κοινή υποδομή ενός η περισσότερων κέντρων δεδομένων. Η συγκεκριμένη ιδιότητα έχει περιγραφεί αναλυτικά στην εργασία [15]. Η διαδικασία multi-tenancy υπόσχεται υψηλή διαθεσιμότητα των πόρων των συστημάτων των διαφόρων παρόχων υπολογιστικού νέφους και παράλληλα εξασφαλίζει όσο το δυνατόν αποδοτικότερη λειτουργία της υποδομής των παρόχων από πλευράς κόστους.

Από την άλλη οι υποδομές των νεφουπολογιστικών συστημάτων πολλών παρόχων που φιλοξενούν κατανεμημένες εφαρμογές εισάγουν διάφορες προκλήσεις όσον αφορά την ασφάλεια και την απόδοση, όπως αναφέρουν οι εργασίες [16],[17]. Η πιο σημαντική είναι με το να παρέχεται η δυνατότητα ο κάθε tenant της εφαρμογής να «δρα» αυτόνομα και να μην επηρεάζεται η απόδοση των υπολοίπων. Αυτό περιγράφεται εκτενώς και στις εργασίες [18],[19]. Προηγούμενη έρευνα που έχει γίνει [20],[21],[22] έδειξε ότι ο διαμοιρασμός των πόρων σε περιβάλλον υπολογιστικού νέφους πολλών παρόχων μεταξύ διαφόρων tenants μπορεί να προκαλέσει απρόβλεπτη μεταβολή στην απόδοση κατανεμημένων εφαρμογών που φιλοξενούνται εκεί.

Επίσης ένα σημαντικό θέμα που αφορά το Multi-tenancy νεφουπολογιστικού συστήματος είναι ο διαχωρισμός μεταξύ δικτύου και απόδοσης. Η εργασία [23] παρέχει μια δίκαια πολιτική μεταξύ διαφόρων εικονικών μηχανών που ζητούν πόρους δικτυακούς. Παρόλα αυτά, καθώς η πολιτική αυτή εφαρμόζεται σε εικονικές μηχανές αντί στους διάφορους tenants, ένας tenant μπορεί πρακτικά να αυξάνει το μέγεθος του εύρους ζώνης που θέλει να καταναλώσει με το να εκκινεί περισσότερες εικονικές μηχανές που θα χρησιμοποιεί. Μια πιο πρόσφατη μελέτη, αυτή της EyeQ [24] χρησιμοποιεί τεχνικές ελέγχου κίνησης προκειμένου να προβλέψει το αναγκαίο εύρος ζώνης που θέλει ένας tenant μιας εικονικής μηχανής. Αυτό επιτυγχάνεται κυρίως με το να απομονώνει το σύστημα του tenant προωθώντας την κίνηση προς το άκρο του υπολογιστικού νέφους με τεχνικές edge computing.

Το πιο σημαντικό θέμα σχετικά με το Multi-tenancy νεφουπολογιστικού συστήματος πολλών παρόχων είναι το να βρίσκεται η χρυσή τομή μεταξύ χρήσης συστημάτων και απόδοσης εφαρμογών μεταξύ των διαφόρων tenants. Σε αυτό σημαντική παράμετρος είναι η παρακολούθηση τόσο της σχετικής υποδομής όσο και των κατανεμημένων εφαρμογών που φιλοξενούνται πράγμα που παρουσιάζεται εκτενώς στην εργασία [25]. Με αυτό τον τρόπο

όποιες «συγκρούσεις» μεταξύ των διαφόρων tenants αποφεύγονται λόγω επαρκούς απομόνωσης/διαχωρισμού που εξασφαλίζεται μεταξύ των εφαρμογών τους.

2.2.5 Αυτόματη παροχή υπηρεσιών μέσω νεφουπολογιστικού περιβάλλοντος πολλών παρόχων

Οι υπηρεσίες νέφους ορίζονται ως υπηρεσίες λογισμικού που χρησιμοποιούν πόρους υποδομής του υπολογιστικού νέφους. Μια υπηρεσία υπολογιστικού νέφους αποτελείται από κατάλληλα διαμορφωμένα τμήματα λογισμικού τα οποία φιλοξενούνται και λειτουργούν από πόρους της υποδομής νέφους οι οποίοι διαμορφώνονται δυναμικά και όχι στατικά. Μια βασική λειτουργία για τις πλατφόρμες που «σερβίρονται» μέσω υπολογιστικού νέφους είναι η δυνατότητα να δεσμεύουν και να αποδεσμεύουν πόρους υπολογιστικού νέφους κατά βούληση με βάση τις ανάγκες τους. Με άλλα λόγια με βάση την παρακολούθηση του φορτίου στο νεφο υπολογιστικό σύστημα σε επίπεδο διακομιστών εγκαταστάσεων αποθήκευσης και δικτύων λαμβάνουν χώρα αυτόματα οι αναγκαίες βελτιστοποιήσεις. Πιο συγκεκριμένα λαμβάνουν χώρα βελτιστοποιήσεις τόσο για αρχικές διαθέσεις όσο και για μετέπειτα αλλαγές. Παράδειγμα η αρχική τοποθέτηση εικονικών μηχανών και περιεκτών για την εξισορρόπηση του φορτίου μεταξύ φυσικών διακομιστών, η ελαχιστοποίηση της καθυστέρησης μεταξύ εφαρμογών και αποθηκευτικών συσκευών και η ελαχιστοποίηση της κίνησης δικτύου. Άλλα παραδείγματα βελτιστοποίησης είναι ή μετεγκατάσταση εικονικών μηχανών με αποτέλεσμα την αύξηση της απόδοσης ή την ελαχιστοποίηση της κατανάλωσης ενέργειας. Η διαδικασία αυτή παροχής τέτοιων υπηρεσιών γίνεται πολλές φορές αυτόματα με βάση μηχανισμούς δέσμευσης και αποδέσμευσης πόρων στο υπολογιστικό νέφος με διπλό σκοπό, αφενός να ικανοποιούνται τα service level objectives και αφετέρου να ελαχιστοποιούνται τα λειτουργικά κόστη.

Η εργασία [26], και με σκοπό τα όσα αναφέρθηκαν στην προηγούμενη παράγραφο, προτείνει μια αρχιτεκτονική που να επιτρέπει την δήλωση υπηρεσιών υπολογιστικού νέφους χρησιμοποιώντας μια περιγραφική γλώσσα για τον σχεδιασμό της τοπολογίας της υποδομής που χρησιμοποιείται κάθε φορά για την προσφερόμενη υπηρεσία. Επιπρόσθετα, μια γλώσσα περιγραφής στοιχείων χρησιμοποιείται για να προσδιορίσει την συμπεριφορά των στοιχείων του λογισμικού. Μια τέτοια γλώσσα λέγεται TOSCA: Topology and Orchestration Specification for Cloud Applications και προσφέρει την δυνατότητα διαχείρισης όπως αυτόματη προσφορά υπηρεσίας, λήξη υπηρεσίας, αυτόματη επέκταση της υποδομής που φιλοξενεί μια συγκεκριμένη υπηρεσία, λήψη αντιγράφων ασφαλείας της υπηρεσίας κτλ.

2.3 Προκλήσεις και προβλήματα

Στην σύγχρονη εποχή και σε αναπτυσσόμενα περιβάλλοντα νεφουπολογιστικών συστημάτων πολλών παρόχων η διαχείριση κατανεμημένων εφαρμογών είναι μια περιοχή που αφορά τεχνικές για λεπτομερειακή παρακολούθηση (monitoring), βελτιστοποίηση απόδοσης τους και υψηλής διαθεσιμότητας των εφαρμογών ακόμα και σε περιβάλλοντα περιορισμένων υπολογιστικών πόρων. Με άλλα λόγια ο βασικός στόχος στους μηχανισμούς διαχείρισης και απόδοσης κατανεμημένων εφαρμογών περιλαμβάνει την παρακολούθηση των εφαρμογών, την ανίχνευση πιθανών προβλημάτων απόδοσης της εφαρμογής καθώς και την διατήρηση προβλεπόμενου επιπέδου υπηρεσίας [27]. Οι κύριες προκλήσεις εστιάζονται στα παρακάτω τέσσερα βασικά σημεία:

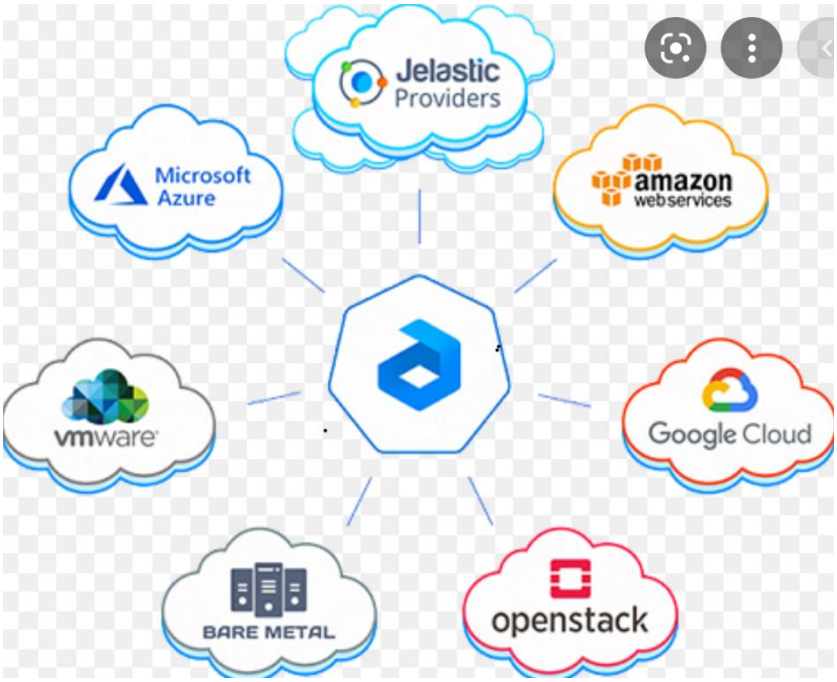
2.3.1 Ευέλικτη παρακολούθηση νεφουπολογιστικού συστήματος πολλών παρόχων

Την σύγχρονη εποχή παρατηρούμε μια ολοένα αυξανόμενη χρήση διασυνδεδεμένων συσκευών και υπηρεσιών που παράγουν δεδομένα και μεταδίδουν διαφόρων τύπων μηνύματα. Η γέννηση πολλών ετερογενών δεδομένων και η μετάδοσή τους με την βοήθεια πληθώρας αρχιτεκτονικών διασυνδέσεων δημιούργησε την ανάγκη για επαρκείς και οικονομικά συμφέρουσες υποδομές. Μια τέτοια ανάγκη ήρθε να εκπληρώσει η χρησιμοποίηση του υπολογιστικού νέφους και ειδικότερα για τις περιπτώσεις κατανεμημένων εφαρμογών που διαχειρίζονται μεγάλο όγκο δεδομένων. Τέτοια υποδομή θεωρητικά προσφέρει μεγάλες επιλογές και διαθεσιμότητες πόρων για φιλοξενία πολλών τύπων κατανεμημένων εφαρμογών.

Ωστόσο μια υποδομή υπολογιστικού νέφους που παρέχεται από πολλούς παρόχους θέτει μια σειρά προκλήσεων η κυριότερη των οποίων είναι η διασφάλιση της ποιότητας της παρεχόμενης υπηρεσίας και με βάση τις συμφωνίες με τους εκάστοτε πελάτες. Επίσης όταν διάφορες εφαρμογές χρησιμοποιούνται από διάφορους χρήστες εμφανίζονται συχνά η πιθανότητα μείωσης της απόδοσης λειτουργίας είτε της υποδομής είτε της εφαρμογής, οι κίνδυνοι για αποτυχία τμημάτων του συστήματος όπως και το ρίσκο για επιθέσεις που αφορούν την ασφάλεια των φιλοξενούμενων στις υποδομές νέφους δεδομένων και ευαίσθητων πληροφοριών.

Για τους παραπάνω λόγους είναι πολύ σημαντικό να παρακολουθούνται με ευέλικτο τρόπο και άρα να αναλύονται τα σχετικά δεδομένα από τις υποδομές υπολογιστικού νέφους πολλών παρόχων προκειμένου να ανιχνεύονται περιπτώσεις που θα μπορούσαν να οδηγήσουν σε αναπροσαρμογή των πόρων της υπάρχουσας χρησιμοποιούμενης υποδομής. Έτσι δίνεται η δυνατότητα δυναμικής αναπροσαρμογής της, είτε επεκτείνοντας την είτε συρρικνώνοντας την αναλόγως κάθε φορά του φορτίου που πρέπει να διαχειριστεί. Με άλλα λόγια η παρακολούθηση της νεφουπολογιστικής υποδομής παρέχει απαραίτητη γνώση για κατάλληλες αποφάσεις σχετικά με φιλοξενία κατανεμημένων εφαρμογών σε τέτοια περιβάλλοντα. Με αυτό τον τρόπο απαντώνται οι διάφορες προκλήσεις που προαναφέρθηκαν.

Τέλος, μια σημαντική ακόμα πρόκληση που παρουσιάζεται είναι το γεγονός ότι ακόμα και αν υπάρχουν κάποια εργαλεία παρακολούθησης υπολογιστικού νέφους αυτά περιορίζονται σε έναν μόνο πάροχο και είναι αυστηρώς κλειστά και ιδιωτικά. Δημιουργείται έτσι η ανάγκη για τρόπους και εφαρμογές ευέλικτης παρακολούθησης που θα εφαρμόζονται ανεξαρτήτως παρόχου[28]. Διάφοροι μεγάλοι πάροχοι έχουν ήδη αναπτύξει έξυπνα συστήματα παρακολούθησης με εφαρμογή στο καθένα όμως ξεχωριστά, ενδεικτικά φαίνονται οι ακόλουθοι στο σχήμα 5:



Σχήμα 5: Δημόσιοι και Ιδιωτικοί πάροχοι που έχουν αναπτύξει Proprietary συστήματα παρακολούθησης

2.3.2 Επανακαθορισμός των πόρων σε περιβάλλον υπολογιστικού νέφους πολλών παρόχων

Μια από τις πιο ευεργετικές υπηρεσίες του υπολογιστικού νέφους είναι η δυνατότητα να παρέχει κατά βούληση και με γνώμονα τις ανάγκες των τελικών χρηστών πόρους που θα φιλοξενούν κατανεμημένες εφαρμογές. Δίνεται έτσι η δυνατότητα του “elasticity” (ελαστικότητας) αναλόγως του φορτίου που χρειάζεται η κάθε εφαρμογή, δηλαδή της δυνατότητας να δεσμεύονται ή να αποδεσμεύονται πόροι σύμφωνα με τις ανάγκες των εφαρμογών των χρηστών που φιλοξενούνται στις υποδομές. Ακόμα όμως και επαναλαμβανόμενες αναπτύξεις της ίδιας κατανεμημένης εφαρμογής στους ίδιους παρόχους νεφουπολογιστικού νέφους δεν εγγυόνται κάθε φορά την ίδια απόδοση της και μπορεί να οδηγεί σε διαφορετικές τιμές πχ. χρόνου απόκρισης της. Αυτό οφείλεται στο γεγονός ότι το πιο πιθανό είναι να γίνεται διαφορετική επιλογή των πόρων του υπολογιστικού νέφους που φιλοξενεί την εφαρμογή σε κάθε διαφορετική ανάπτυξη. Γι’ αυτό τον λόγο της δυναμικής επέκτασης και φιλοξενίας της εφαρμογής, οι εφαρμογές χρειάζεται όπως αναφέρθηκε στην προηγούμενη παράγραφο να παρακολουθούνται και να αναπροσαρμόζονται με ένα κατάλληλο σύστημα αποφάσεων. Αυτό διασφαλίζει τον βέλτιστο τρόπο φιλοξενίας της εφαρμογής, την ικανοποίηση των χρηστών και των συμβολαίων και των περιορισμών που μπορεί να τίθενται όπως πχ. χρόνος εκτέλεσης κάποιων εντολών.

Με βάση την παραπάνω πρόκληση αναπτύχθηκε το έργο του PaaSage [29], όπου οι αναπτύξεις των διαφόρων εφαρμογών ακολουθούν μια λογική βασισμένη σε χρήση μοντέλων χωρίς όμως να λαμβάνει υπόψιν ωστόσο την σημερινή ανάγκη της επεξεργασίας μεγάλου όγκου δεδομένων καθώς και της συνύπαρξης πολλών παρόχων.

2.3.3 Μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων

Στη σύγχρονη εποχή της εκτεταμένης ανάπτυξης νεφουπολογιστικών υποδομών ο βασικός σκοπός της πρόγνωσης στη διάθεση πόρων είναι να ενεργοποιούνται και να χρησιμοποιούνται εκ προοιμίου πόροι που θα φιλοξενούν εφαρμογές προκειμένου να βελτιώσουν την απόδοσή τους όσον αφορά την ποιότητα της προσφερόμενης υπηρεσίας και την χρονική τους απόκριση. Πολλές τέτοιες πρώιμες τεχνικές έχουν ήδη παρουσιαστεί στο παρελθόν [30],[31].

Το βασικό πρόβλημα στο υπολογιστικό νέφος είναι ότι για την διαδικασία της πρόβλεψης ζήτησης πόρων, η βασική παράμετρος που είναι δυσεύρετη είναι τα δεδομένα των χρηστών που προσδιορίζουν τις ανάγκες τους. Ειδικότερα στις μέρες μας με την μεγάλη ανάπτυξη νεφουπολογιστικών περιβαλλόντων πέραν τους ενός παρόχου, η έλλειψη γνώσης για αυτές τις ανάγκες σε ένα δαιδαλώδες και εν πολλοίς άγνωστο υπολογιστικό νέφος γίνεται ολοένα και μεγαλύτερη με αποτέλεσμα μεγαλύτερη δυσχέρεια στην πρόγνωση της απόδοσης των κατανεμημένων εφαρμογών. Επίσης σε νεφουπολογιστικά περιβάλλοντα πολλών παρόχων είναι δυσεύρετη και η συλλογή διαφόρων δεδομένων σχετικά με την ιστορική κατανάλωση πόρων πχ. κατανάλωση επεξεργαστικής ισχύος σε πολλούς παρόχους υπολογιστικού νέφους ταυτόχρονα. Λίγες ερευνητικές προσπάθειες υπάρχουν με επαρκή αποτελέσματα στην ακρίβεια των προγνώσεων ειδικότερα όταν μιλάμε και για υπολογιστικά περιβάλλοντα πολλών παρόχων [32]. Συνεπώς οι τεχνικές αποτελεσματικών προγνώσεων για την χρήση των πόρων νεφουπολογιστικών συστημάτων πολλών παρόχων με υποδομές εκτεταμένες σε διάφορες γεωγραφικές περιοχές ανά τον κόσμο είναι λίγες αποτελώντας έτσι ένα χώρο που παρουσιάζει μεγάλο ερευνητικό ενδιαφέρον και ισχυρές προκλήσεις.

2.3.4 Μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων με χρήση συστημάτων στο Άκρο (Edge Cloud Systems)

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, ο βασικός σκοπός της πρόγνωσης στη διάθεση πόρων είναι να ενεργοποιούνται και να χρησιμοποιούνται εκ προοιμίου πόροι που θα φιλοξενήσουν εφαρμογές προκειμένου να βελτιώσουν την απόδοσή τους. Χρησιμοποιώντας δεδομένα συμπεριφοράς χρηστών στις διάφορες εφαρμογές όπως π.χ. κατανάλωση επεξεργαστικής ισχύος εφαρμογών σε πολλούς παρόχους υπολογιστικού νέφους, αναπτύσσουμε τεχνικές προγνώσεων σε υποδομές εκτεταμένες σε διάφορες γεωγραφικές περιοχές ανά τον κόσμο ανά διαφόρους παρόχους. Στις μέρες μας, υπάρχει μια μεγάλη τάση για μαζική παραγωγή δεδομένων από μεγάλο αριθμό συνεχώς αυξανόμενων συσκευών ευρισκόμενες στο «άκρο» ή “edge” του υπολογιστικού νέφους όπως λέγεται. Επομένως γεννάται το ερώτημα και για λόγους από-κεντρικοποίησης, εάν μέρος της διαδικασίας πρόγνωσης αυτής θα μπορούσε να λαμβάνει χώρα σε τέτοιες συσκευές.

Παρά την μεγάλη και ευρεία έρευνα στο χώρο του Federated Learning για προγνώσεις διάθεσης πόρων, υπάρχει ένα κενό μεταξύ του Federated Learning και του Edge Learning όπως προαναφέρθηκε, δηλαδή της χρήσης συσκευών περιορισμένων πόρων ευρισκόμενες στο άκρο του υπολογιστικού νέφους για τους αναγκαίους υπολογισμούς αλγορίθμων ή

αλλιώς αλγορίθμων Tiny Machine Learning[149],[101]. Τα σημερινά συστήματα Federated Learning που χρησιμοποιούν την περίπτωση των tiny systems εκτελούν την τεχνική του model Inference πάνω στην συσκευή την ίδια που βρίσκεται στο άκρο του υπολογιστικού νέφους[85]. Ένα από τα βασικότερα μειονεκτήματα για όλες σχεδόν τις περιπτώσεις και της σχετιζόμενης έρευνας είναι το γεγονός ότι υπάρχει έλλειψη ενός γενικού framework όσον αφορά την χρησιμοποιούμενη τεχνολογία το οποίο να είναι ανεξάρτητο νεφουπολογιστικού παρόχου (vendor-agnostic) και καθώς επίσης το γεγονός ότι υπάρχει ένας περιορισμός όσον αφορά την υπολογιστική διαθεσιμότητα και το μέγεθος αποθήκευσης δεδομένων στα συστήματα edge. Επομένως, η ανάπτυξη μιας τέτοια τεχνολογίας στο άκρο (edge) του υπολογιστικού νέφους με εφαρμογή σε πέραν του ενός παρόχους, παρουσιάζει μεγάλο ερευνητικό ενδιαφέρον και ισχυρές προκλήσεις.

3 Η Πρόταση της Διατριβής

3.1 Εισαγωγή

Στο παρόν κεφάλαιο, διατυπώνονται τα ερευνητικά ερωτήματα τα οποία στοχεύει να απαντήσει η συγκεκριμένη Διατριβή. Στη συνέχεια παρουσιάζεται η προτεινόμενη προσέγγιση σε αυτά.

3.2 Ερευνητικά Ερωτήματα και Προτάσεις

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο 2.3 σε σύγχρονα περιβάλλοντα νεφουπολογιστικών συστημάτων πολλών παρόχων η διαχείριση κατανεμημένων εφαρμογών είναι μια περιοχή που αφορά τεχνικές για λεπτομερειακή παρακολούθηση (monitoring) και βελτιστοποίηση της απόδοσης τους μέσω αναπροσαρμογής δυναμικά της υποδομής του υπολογιστικού νέφους που φιλοξενεί αυτές τις εφαρμογές. Επίσης αφορά φυσικά και την υψηλή διαθεσιμότητα των εφαρμογών ώστε να παρέχονται αδιάλειπτα οι υπηρεσίες στους τελικούς χρήστες.

Η παρούσα Διατριβή πραγματεύεται καινοτόμες αρχιτεκτονικές κατανεμημένων συστημάτων παρακολούθησης σε περιβάλλοντα πολλών παρόχων νεφουπολογιστικών συστημάτων. Επιπλέον, έμφαση δίνεται σε κατανεμημένες εφαρμογές διαχείρισης μεγάλων δεδομένων στα αντίστοιχα περιβάλλοντα υποδομών που αναφέρθηκαν.

Επιπλέον πολύ σημαντική παράμετρος είναι η δυνατότητα δυναμικής αναπροσαρμογής των πόρων νεφουπολογιστικού συστήματος με βάση κάποια συνάρτηση που λαμβάνει κύρια παράμετρο τον χρόνο ώστε να αποφεύγονται κρίσιμες καθυστερήσεις με καταστροφικά ίσως αποτελέσματα στην αναπροσαρμογή των υποδομών. Αυτό συμβαίνει σε περιπτώσεις που έχει παρατηρηθεί αυξημένο φορτίο σε κάποιες εικονικές μηχανές που φιλοξενούν κατανεμημένες εφαρμογές και πιθανόν υποβαθμίζουν την απόδοση και την απόκρισή τους. Σε τέτοιες περιπτώσεις το νεφουπολογιστικό σύστημα οφείλει να δρα «πυροσβεστικά» και ευεργετικά ώστε να αυξάνεται και η αποδοχή της υπηρεσίας από τον τελικό χρήστη.

Ωστόσο υπάρχουν και πεδία έρευνας που αφορούν και προληπτική δράση και αναπροσαρμογή πόρων που γίνεται προκειμένου να εξασφαλιστούν οι προδιαγεγραμμένες πολλές φορές απαιτήσεις των τελικών χρηστών μέσω Συμβολαίων που δίνουν οι πάροχοι (Service Level Agreements). Σε αυτό τον τομέα σημαντικό ρόλο παίζει η εφαρμογή κατανεμημένων αλγορίθμων επεξεργασίας δεδομένων τοπικά σε κάθε κόμβο του νεφουπολογιστικού συστήματος αλλά και κεντρικά. Οι νέες και εξελιγμένες τεχνικές federation learning και βαθιάς μηχανικής μάθησης είναι δυνατόν να πετύχουν εξαιρετικές ακρίβειες προβλέψεων στην ζήτηση πόρων. Σημαντικό σημείο αποτελεί ωστόσο και το γεγονός ότι χρειάζεται αρκετές φορές να χρησιμοποιηθούν τεχνικές εκμάθησης και αντίστοιχες τεχνολογίες στο άκρο ή αλλιώς edge cloud προκειμένου να γίνουν οι αναγκαίες προγνώσεις χρειαζούμενων πόρων. Σε τέτοιες περιπτώσεις τεχνολογίες τύπου Tiny Machine Learning στο Άκρο είναι αναγκαίες για χρήση τους από πολλούς νεφουπολογιστικούς παρόχους.

Με βάση τις παραπάνω σκέψεις και σε συνάρτηση με όσα παρουσιάστηκαν στην βιβλιογραφική αναζήτηση του Κεφαλαίου 2 προκύπτουν διάφορα ερευνητικά ερωτήματα που έχουν ήδη αναφερθεί στην Εισαγωγή:

- Ποιος ο καταλληλότερος τρόπος για ευέλικτη παρακολούθηση κατανεμημένων εφαρμογών σε περιβάλλοντα υπολογιστικού νέφους πολλών παρόχων?
- Πώς μπορούμε να πετύχουμε βέλτιστη προσαρμογή των πόρων σε ένα περιβάλλον υπολογιστικού νέφους πολλών παρόχων?
- Πώς θα επιτευχθεί μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων που φιλοξενεί κατανεμημένη εφαρμογή με βέλτιστη αξιοποίηση της υπάρχουσας υποδομής?
- Πως επηρεάζεται η ακρίβεια στην πρόβλεψη ζήτησης πόρων σε περιπτώσεις υπολογισμών σε μικρές συσκευές στο Άκρο (edge) του νεφουπολογιστικού συστήματος πολλών παρόχων?

Η καινοτομία της Διατριβής με βάση τα ερωτήματα που προκύπτουν από έρευνα στην βιβλιογραφία, έγκειται σε τέσσερα βασικά σημεία:

1. Προτείνεται μια καινοτόμος τεχνική σύνθετης επεξεργασίας συμβάντων με εφαρμογή σε παρακολούθηση νεφουπολογιστικών περιβαλλόντων πολλών παρόχων.
2. Παρουσιάζεται μια νέα μεθοδολογία σε ένα περιβάλλον πολλών παρόχων υπολογιστικού νέφους για αυτόματη λήψη απόφασης που οδηγεί σε βέλτιστη αναπροσαρμογή πόρων με γνώμονα τον χρόνο.
3. Προτείνεται μια καινοτόμος μέθοδος πρόβλεψης ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων με αλγορίθμους Federated Learning και βαθιάς μηχανικής μάθησης με ταυτόχρονη βελτιστοποίηση του χρόνου εκτέλεσης των αλγορίθμων.
4. Προτείνεται μια καινοτόμος μέθοδος πρόβλεψης ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων με αλγορίθμους Federated Learning αλλά και με χρήση της τεχνολογίας Tiny Machine Learning για το Άκρο(Edge) του νέφους όπου οι υπολογιστικοί πόροι είναι περιορισμένοι, με ταυτόχρονη βελτιστοποίηση του χρόνου εκτέλεσης των αλγορίθμων.

Στον ακόλουθο πίνακα δίνεται μια σύνοψη των ερευνητικών ερωτημάτων που στοχεύει να απαντήσει η Διατριβή, των προτεινόμενων λύσεων, του τρόπου αξιολόγησης καθώς επίσης και αναφορά της ενότητας που αναλύονται:

ΕΡΕΥΝΗΤΙΚΟ ΕΡΩΤΗΜΑ	ΛΥΣΗ	ΑΞΙΟΛΟΓΗΣΗ	ΕΝΟΤΗΤΑ
Ποιος ο καταλληλότερος τρόπος για ευέλικτη παρακολούθηση κατανεμημένων εφαρμογών σε περιβάλλοντα υπολογιστικού νέφους πολλών παρόχων?	**Καινοτόμος τεχνική σύνθετης επεξεργασίας συμβάντων με εφαρμογή σε παρακολούθηση νεφουπολογιστικών περιβαλλόντων πολλών παρόχων**	**Πειραματικές μετρήσεις σε 2 διαφορετικά Ιδιωτικά Υπολογιστικά Νέφη με διάφορες εικονικές μηχανές και υπολογιστές**	**4**
Πώς μπορούμε να πετύχουμε βέλτιστη προσαρμογή των πόρων σε ένα περιβάλλον υπολογιστικού νέφους πολλών παρόχων?	**Νέα μεθοδολογία σε ένα περιβάλλον πολλών παρόχων υπολογιστικού νέφους για απόφαση που οδηγεί σε βέλτιστη αναπροσαρμογή πόρων με βασική παράμετρο τον χρόνο**	**Πειραματικές μετρήσεις σε Ιδιωτικό και Δημόσιο Υπολογιστικό Νέφος με διάφορες εικονικές μηχανές**	**5**
Πώς θα επιτευχθεί μεγαλύτερη ακρίβεια στην πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων που φιλοξενεί κατανεμημένη εφαρμογή με βέλτιστη αξιοποίηση της υπάρχουσας υποδομής?	**Καινοτόμος μέθοδος πρόβλεψης ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων με αλγορίθμους Federated Learning, βαθιάς μηχανικής μάθησης και ταυτόχρονη βελτιστοποίηση του χρόνου εκτέλεσης της εκπαίδευσης των αλγορίθμων.**	**Πειραματικές μετρήσεις σε Ιδιωτικό Υπολογιστικό Νέφος με διάφορες εικονικές μηχανές με τεχνολογία docker**	**6**
Πως επηρεάζεται η ακρίβεια στην πρόβλεψη ζήτησης πόρων σε περιπτώσεις υπολογισμών σε μικρές συσκευές στο Άκρο (edge) του νεφουπολογιστικού συστήματος πολλών παρόχων?	**Καινοτόμος μέθοδος πρόβλεψης ζήτησης πόρων νεφουπολογιστικού συστήματος πολλών παρόχων με αλγορίθμους Federated Learning αλλά και με χρήση της τεχνολογίας Tiny Machine Learning για το Άκρο(Edge) του νέφους όπου οι υπολογιστικοί πόροι είναι περιορισμένοι, με ταυτόχρονη βελτιστοποίηση του χρόνου εκτέλεσης των αλγορίθμων.**	**Πειραματικές μετρήσεις σε Ιδιωτικό Υπολογιστικό Νέφος με διάφορες εικονικές μηχανές με τεχνολογία docker αλλά με περιορισμένους υπολογιστικούς πόρους**	**7**

Πίνακας 2: Ερωτήματα και Λύσεις της Διατριβής

4 Ευέλικτος τρόπος παρακολούθησης κατανεμημένων εφαρμογών

4.1 Εισαγωγή

Όπως αναφέρθηκε και στο κεφάλαιο 2.3.1 η παρακολούθηση νεφουπολογιστικών υποδομών και κατανεμημένων εφαρμογών παρουσιάζει πολλές προκλήσεις. Οι προκλήσεις αυτές είναι αυξημένες σήμερα και λόγω του γεγονότος ότι περισσότερα δεδομένα γεννούνται και επεξεργάζονται σε διάφορες κατανεμημένες εφαρμογές σε υπολογιστικά περιβάλλοντα πολλών παρόχων. Αυτό από πλευράς υποδομής σημαίνει μεγαλύτερες ανάγκες σε υπολογιστική ισχύ, δίκτυα και αποθηκευτικούς χώρους. Ειδικά παρουσιάζεται η περίπτωση αυξομειώσεων της ζήτησης πόρων σε εφαρμογές επεξεργασίας μεγάλων δεδομένων με δυναμικό τρόπο (scale up / scale down). Με την βοήθεια της παρακολούθησης συλλέγονται κατάλληλα δεδομένα προκειμένου να λαμβάνονται έγκαιρα αποφάσεις για αναδιαμόρφωση πόρων καθώς και του τρόπου «σερβιρίσματος» κατανεμημένων εφαρμογών και φιλοξενίας τους σε υπολογιστικό νέφος απαρτιζόμενο από πολλούς παρόχους. Η διαδικασία αυτή της παρακολούθησης περιλαμβάνει ανταπόκριση σε δύσκολες περιπτώσεις που απαιτούν εξελιγμένα εργαλεία και τεχνικές.

Μια από τις πιο αξιόλογες μεθόδους που ανταποκρίνεται με επιτυχία στις παραπάνω απαιτήσεις είναι η επεξεργασία συμβάντων. Η επεξεργασία συμβάντων αφορά την καταγραφή και ανάλυση ροών δεδομένων για θέματα σχετιζόμενα με εφαρμογές και συνακόλουθα παραγωγή ειδοποιήσεων σχετικά με αυτά. Η επεξεργασία σύνθετων συμβάντων (Complex Event Processing ή CEP) αναφέρεται σε επεξεργασία δεδομένων που συνδυάζει δεδομένα με βάση κάποια πρότυπα “data patterns” και με σκοπό όταν ανιχνεύονται τέτοιες δομές πληροφορίας να ενεργοποιούνται συγκεκριμένες διαδικασίες. Τα συστήματα επεξεργασίας σύνθετων συμβάντων είναι κατάλληλα για συγκερασμό πολλών ροών δεδομένων. Το μεγάλο τους πλεονέκτημα είναι η ικανότητα να συλλέγουν πληροφορίες από διάφορες ετερογενείς πηγές δεδομένων και να φιλτράρουν, να συνδυάζουν αυτά τα δεδομένα εντός συγκεκριμένων χρονικών περιόδων (τα λεγόμενα χρονικά παράθυρα). Η ιδέα της χρήσης συστημάτων επεξεργασίας σύνθετων συμβάντων για παρακολούθηση νεφουπολογιστικών συστημάτων εδράζεται σε δύο βασικές αρχιτεκτονικές:

- Κεντρικοποιημένη αρχιτεκτονική συστήματος επεξεργασίας σύνθετων συμβάντων
- Κατανεμημένη αρχιτεκτονική συστήματος επεξεργασίας σύνθετων συμβάντων

Η Κεντρικοποιημένη αρχιτεκτονική επεξεργασίας σύνθετων συμβάντων χρησιμοποιεί μια και μόνο μηχανή επεξεργασίας σύνθετων δεδομένων (CEP Engine) η οποία επεξεργάζεται όλα τα δεδομένα παρακολούθησης και εντοπίζει συγκεκριμένα πρότυπα δεδομένων με βάση κανόνες που μπορούν να οριστούν στην μηχανή. Από την άλλη μεριά, η κατανεμημένη αρχιτεκτονική επεξεργασίας σύνθετων συμβάντων περιλαμβάνει ένα σετ από μηχανές επεξεργασίας σύνθετων δεδομένων (CEP Engine) οι οποίες συνεργάζονται μεταξύ τους ανταλλάσσοντας μηνύματα και έχουν την δυνατότητα με μεγαλύτερη απόδοση να ανιχνεύουν πρότυπα (data patterns) συμβάντων χρησιμοποιώντας κανόνες οι οποίοι διαφέρουν αναλόγως της εγγύτητας των διαφόρων πηγών δεδομένων στην εκάστοτε μηχανή CEP Engine. Δίνεται με αυτό τον τρόπο η δυνατότητα ορισμού σε κάθε περίπτωση μηχανής CEP Engine διαφορετικών κανόνων επεξεργασίας σύνθετων συμβάντων και ανίχνευσης με μεγαλύτερη ακρίβεια τυχόν αξιόλογων γεγονότων προς παρακολούθηση.

Υπάρχουσες κεντρικοποιημένες προσεγγίσεις όπως οι εργασίες των [33],[34],[35] φανερώνουν ότι γίνεται χρήση μεγάλου εύρους ζώνης και υπολογιστικών πόρων επεξεργαστικής ισχύος το οποίο συχνά σημαίνει ζήτηση πόρων που δεν υπάρχουν αλλά και ρίσκου μιας και το μοναδικό CEP Engine γίνεται Single-Point-of-Failure όταν επεξεργάζεται τεράστιο όγκο δεδομένων που αφορούν την παρακολούθηση της κατάστασης των υποδομών. Από την άλλη, κατανεμημένες αρχιτεκτονικές επεξεργασίας σύνθετων συμβάντων όπως οι εργασίες των Hirzel[36] και του Ku[37] παρουσιάζουν καλύτερη απόδοση με όρους διακίνησης φορτίου (throughput) και επεξεργασίας δεδομένων εξαιτίας του διαμοιρασμού του φορτίου της επεξεργασίας των δεδομένων μεταξύ των διαφόρων μηχανών CEP Engines και επίσης παρουσιάζουν καλύτερα αποτελέσματα επεκτασιμότητας χωρίς το ρίσκο του single-point-of-failure. Παρόλα αυτά, και οι δύο μέχρι τώρα αρχιτεκτονικές επεξεργασίας σύνθετων συμβάντων παρουσιάζουν ένα βασικό περιορισμό και αυτό είναι η εφαρμογή τους σε ένα κάθε φορά πάροχο υπολογιστικού νέφους. Αυτό το γεγονός από μόνο του περιορίζει τις δυνατότητες που έχουν οι εφαρμογές διαχείρισης μεγάλων δεδομένων και είναι ένα ζήτημα που χρήζει περαιτέρω έρευνας.

4.2 Σχετικές Εργασίες

Η επεξεργασία σύνθετων συμβάντων (Complex Event Processing-CEP) συνεισφέρει πολλά στην ανίχνευση ειδικών γεγονότων δια μέσου της χρήσης προσαρμοσμένων προτύπων και κανόνων και με τελικό σκοπό την δημιουργία ειδοποιήσεων ειδικά σε περιπτώσεις όπου αυξανόμενες ροές δεδομένων παρατηρούνται και ασυνήθιστα-μη φυσιολογικά γεγονότα συμβαίνουν. Σχετικά με την κατανεμημένη αρχιτεκτονική επεξεργασίας σύνθετων συμβάντων λίγες πρόσφατες εργασίες εστιάζουν στην τεχνική της παράλληλης επεξεργασίας προτύπων σε διάφορες ροές δεδομένων.

Ειδικότερα, στην εργασία του Hirzel [38] χρησιμοποιώντας «κλειδιά» προκειμένου να κατατμηθούν τα εισερχόμενα γεγονότα προτείνεται μια συγκεκριμένη σύνταξη προσαρμοσμένων προτύπων και ένας συγκεκριμένος τρόπος «μετάφρασης» των γεγονότων. Με αυτό τον τρόπο γεγονότα με διαφορετικά «κλειδιά» επεξεργάζονται σε παράλληλη βάση. Ο Hirzel επωφελείται της συγκεκριμένης κατάτμησης των δεδομένων (εισερχόμενων συμβάντων) χρησιμοποιώντας κατάλληλή προγραμματιστική γλώσσα. Σε μια παρόμοια εργασία του Ku [39] προτείνεται μια κατανεμημένη αρχιτεκτονική σύνθετης επεξεργασίας συμβάντων (CEP) στην οποία διαμοιράζονται τα επιμέρους φορτία επεξεργασίας των σύνθετων συμβάντων μεταξύ των σταθμών/μηχανών που φιλοξενούν τις μηχανές CEP Engines. Η βασική αρχιτεκτονική επικοινωνιών επιτυγχάνεται χρησιμοποιώντας έναν κατανεμημένο message broker βασισμένο σε τεχνολογία Apache River, ένα δίκτυο από κατανεμημένα συστήματα στην μορφή συνεργαζόμενων υπηρεσιών. Οι συγγραφείς της εργασίας [39] χρησιμοποιούν έναν κατανεμημένο αλγόριθμο ανίχνευσης σύνθετων συμβάντων με μια λογική Masters-Workers. Η καινοτομία της συγκεκριμένης εργασίας εστιάζεται στο γεγονός ότι μπορεί να εφαρμοστεί σε γεωγραφική διασπορά των διαφόρων διαδικασιών για επιμέρους ανίχνευση γεγονότων χρησιμοποιώντας μηχανές CEP Engines σε λογική Master-Slave.

Συνοψίζοντας, στην πρώτη εργασία του Hirzel [38] η προτεινόμενη προσέγγιση είναι επαρκής να χρησιμοποιηθεί όταν η γλώσσα που διαχειρίζεται τα δεδομένα είναι συγκεκριμένης μορφής και ικανοποιεί συγκεκριμένα σενάρια επεξεργασίας δεδομένων. Η τεχνική παραλληλισμού που χρησιμοποιείται είναι είτε κεντρικοποιημένη χρησιμοποιώντας μια μηχανή CEP Engine είτε κατανεμημένη χρησιμοποιώντας διάφορες μηχανές CEP Engines, και

στις δύο περιπτώσεις όμως χρησιμοποιείται ένας πάροχος υπολογιστικού νέφους. Στην δεύτερη εργασία του Ku [39] η τεχνική που χρησιμοποιείται παρουσιάζει μια σημαντική αύξηση στην επικοινωνία (όγκο ανταλλαγής δεδομένων) όταν ο αριθμός των γεγονότων ανά δευτερόλεπτο είναι λιγότερος από 500. Και στις δύο εργασίες όμως δεν υιοθετείται η λογική των πολλών ταυτόχρονα παρόχων υπολογιστικού νέφους χωρίς να επωφελούνται των όποιων πλεονεκτημάτων παρουσιάζει αυτό. Επιπλέον, στην δεύτερη εργασία η δυναμική μεταβολή του αριθμού των κατανεμημένων μηχανών CEP Engines δεν χρησιμοποιείται. Θεωρούμε την χρήση μιας στατικής αρχιτεκτονικής προκαθορισμένου αριθμού μηχανών CEP.

Η εργασία του Paraiso [40] παρουσιάζει μια κατανεμημένη μηχανή CEP Engine την οποία ονομάζει DiCEPE και η οποία είναι μια πλατφόρμα η οποία εστιάζει στην ολοκλήρωση CEP Engines μηχανών σε κρίσιμα κατανεμημένα συστήματα. Κατάλληλα επικοινωνιακά πρωτόκολλα χρησιμοποιούνται προκειμένου να διασυνδέουν CEP μηχανές με ευκολία και μεταξύ μεγάλων γεωγραφικών περιοχών. Σε μια παρόμοια αρχιτεκτονική, η εργασία του Flouris [41] έχει αναπτύξει ένα πρωτότυπο πλατφόρμας που λέγεται FERARI και η οποία υλοποιεί επεξεργασία σύνθετων συμβάντων σε πραγματικό χρόνο για μεγάλου όγκου δεδομένα σε κατανεμημένη αρχιτεκτονική. Στην εργασία του Frascati [42] έχει αναπτυχθεί μια ανοιχτού κώδικα πλατφόρμα που ωστόσο δημιουργεί ένα μεγάλο όγκο μηνυμάτων που ανταλλάσσονται. Στην εργασία του Flouris [41] από την άλλη δεν παρέχεται οποιοδήποτε δυναμικό μοντέλο ανταλλαγής μηνυμάτων τύπου publish-subscribe μεταξύ των διαφόρων CEP Engines μηχανών. Αντί αυτού χρησιμοποιούνται τεχνικές push-pull που δεν υπακούνε σε λογική επεκτασιμότητας. Επιπλέον, χρησιμοποιούν μόνο μια μηχανή CEP Engine ανά νεφουπολογιστικό πάροχο με τεχνική παραλληλισμού (parallelism) σε αντίθεση με αυτό που προτείνεται σε αυτή την Διδακτορική διατριβή και αφορά πολλές CEP Engines μηχανές ανά πάροχο υπολογιστικού νέφους. Καμμιά από τις δύο προαναφερθείσες εργασίες: Frascati [42] & Flouris [41] δεν χρησιμοποιεί οποιοδήποτε δίκτυο επεξεργασίας συμβάντων (Event Processing Network – EPN) το οποίο μπορεί δυναμικά και με εύκολο τρόπο να αναπτυχθεί σε περιβάλλον πολλών παρόχων υπολογιστικού νέφους και να αναπτύσσει λειτουργίες διαφόρων επιπέδων πολυπλοκότητας στην επεξεργασία γεγονότων.

Μια άλλη εργασία που κάνει χρήση των δυνατοτήτων για κατανεμημένη ανίχνευση γεγονότων και η οποία ονομάζεται «CEP επόμενης γενεάς» είναι αυτή του Schultz-Moller [43]. Όπως αναφέρθηκε και στις προηγούμενες εργασίες, και εδώ χρησιμοποιείται μια συγκεκριμένη γλώσσα για επεξεργασία συμβάντων με κατανεμημένο τρόπο με βάση κάποια πρότυπα και σχετικούς κανόνες. Ωστόσο οι ίδιοι κανόνες εφαρμόζονται σε όλες τις CEP Engine μηχανές χωρίς την δυνατότητα να αναπροσαρμόζονται δυναμικά σε κάθε CEP μηχανή τοπικά και αναλόγως των αναγκών που εξυπηρετούν.

Η εργασία του Mdhaffar [44] εισάγει μια δυναμική αρχιτεκτονική για μέτρηση της απόδοσης του υπολογιστικού νέφους και ανάλυσης των διαφόρων σύνθετων γεγονότων βασισμένη είτε σε κεντρικοποιημένη είτε σε κατανεμημένη αρχιτεκτονική. Η εργασία παρουσιάζει ένα σύστημα το οποίο μπορεί δυναμικά να μεταπηδά από κεντρικοποιημένες σε κατανεμημένες CEP λογικές αναλόγως του υπολογιστικού φορτίου και της δικτυακής κίνησης. Παρόλα αυτά, καμμιά επεξεργασία γεγονότων (απλών ή σύνθετων) δεν λαμβάνει χώρα τοπικά σε οποιαδήποτε μηχανή CEP Engine εικονικής μηχανής. Η οποιαδήποτε τοπική επεξεργασία αφορά απλή ανίχνευση ανωμαλιών (anomaly/outlier detection) σε συγκεκριμένες ροές δεδομένων.

Μια αξιόλογη επίσης εργασία που έχει παρουσιαστεί πριν καιρό αφορά μια κεντρικοποιημένη αρχιτεκτονική επεξεργασίας σύνθετων συμβάντων η οποία συνδυάζει τεχνολογίες που προσφέρουν ενοποίηση (integration) όπως το Mule ESB και η ESPER μηχανή. Τα δεδομένα συλλέγονται σε αυτή την αρχιτεκτονική από μια πλατφόρμα που λέγεται Xively IoT platform και προέρχονται από διάφορες πηγές. Η εργασία αυτή προέρχεται από τον Boubeta [45]. Επιπλέον μια άλλη εργασία του Leitner [46] προτείνει μια κεντρικοποιημένη αρχιτεκτονική επεξεργασίας συμβάντων που εφαρμόζει μια τεχνική πολλών βημάτων συσχέτισης γεγονότων, η οποία παρακολουθεί νεφουπολογιστικές εφαρμογές που χρησιμοποιούν ένα μεγάλο αριθμό εικονικών μηχανών. Μέσω αυτής της τεχνικής η «ελαστικότητα» (elasticity) της εφαρμογής που φιλοξενείται στο υπολογιστικό νέφος αυξάνεται. Σε μια άλλη παρόμοια εργασία χρησιμοποίησης της πλατφόρμας Cloudscale [47], τα δεδομένα που προκύπτουν από την παρακολούθηση της νεφουπολογιστικής υποδομής χρησιμοποιούνται στην λήψη αποφάσεων για ενεργοποίηση ή απενεργοποίηση εικονικών μηχανών στο υπολογιστικό νέφος. Ωστόσο η συγκεκριμένη πλατφόρμα διαχείρισης υποδομής υπολογιστικού νέφους δεν λαμβάνει υπόψιν της μετρικές υποδομών όπως χρήση επεξεργαστικής ισχύος, κατανάλωση μνήμης αλλά ούτε παραμέτρους εφαρμογών όπως ο χρόνος απόκρισης μιας σημαντικής εφαρμογής. Και για τις τρεις προαναφερθείσες εργασίες μπορούμε να αναφέρουμε ένα βασικό μειονέκτημα: αυτό της θεώρησης ύπαρξης ενός μόνο παρόχου υπολογιστικού νέφους κάθε φορά πράγμα που δεν επωφελείται των πλεονεκτημάτων των υποδομών πολλών παρόχων και των δυναμικών πρωτοκόλλων επικοινωνίας που χρειάζονται στην αναπροσαρμογή των εικονικών μηχανών που χρησιμοποιούνται.

Τέλος, κάποιες ακόμα εργασίες έχουν αναπτυχθεί ερευνητικά οι οποίες χρησιμοποιούν δεδομένα από συμβάντα προερχόμενα από ετερογενή περιβάλλοντα σε νεφουπολογιστικό επίπεδο και είναι βασισμένες σε αρχιτεκτονικές υπηρεσιών (Service Oriented Architectures). Στην εργασία [48] οι συγγραφείς εισάγουν μια αρχιτεκτονική η οποία χρησιμοποιεί λογική υπηρεσιών νέφους ώστε με συγκεκριμένο αλγόριθμο να μπορούν να ανιχνεύουν πρότυπα που φανερώνουν παραβίαση Service Level Agreements πελατών πχ. παραβίαση SLA για χαρακτηριστικά παρεχόμενου αποθηκευτικού χώρου μέσω υπολογιστικού νέφους. Επιπλέον οι συγγραφείς των εργασιών [49] & [50] προτείνουν μια αρχιτεκτονική βασισμένη σε γεγονότα μορφής υπηρεσιών (Service Oriented Architectures). Η προτεινόμενη αρχιτεκτονική είναι βασισμένη σε περιεχόμενο δεδομένων και επεξεργασίας τους και μπορεί να εφαρμοστεί σε μια ευρεία γκάμα εφαρμογών, ωστόσο χρησιμοποιεί μια μόνο μηχανή CEP Engine κεντρικά χωρίς την δυνατότητα να επεξεργάζεται κατανεμημένα τα διάφορα συμβάντα. Και στις τρεις προαναφερθείσες εργασίες δεν λαμβάνεται υπόψιν η πιθανότητα της εμπλοκής πέραν του ενός νεφουπολογιστικού παρόχου καθώς και οι διευκολύνσεις που παρέχουν οι υπολογιστικές υποδομές νέφους πολλών παροχών. Επιπλέον, σε αυτές τις εργασίες χρησιμοποιείται μια προσέγγιση επεξεργασίας σύνθετων συμβάντων (CEP) ενός μόνο επιπέδου.

Σε αντίθεση με τις ανωτέρω ερευνητικές εργασίες και μελέτες, στην παρούσα Διδακτορική Διατριβή παρουσιάζεται μια κατανεμημένη αρχιτεκτονική παρακολούθησης της υποδομής υπολογιστικού νέφους που επεκτείνεται σε πολλά επίπεδα επεξεργασίας σύνθετων συμβάντων (CEP) μεταξύ πολλών παρόχων[A2]. Επιπλέον παρουσιάζεται μια τεχνική επικοινωνίας βασισμένη σε μοντέλο publish-subscribe το οποίο προφέρει δυνατότητες επεκτασιμότητας, ευελιξίας, κατάτμησης της υποδομής και δυνατότητας προσαρμογής σε ένα δυναμικά αναπροσαρμοζόμενο περιβάλλον όσον αφορά τους πόρους του υπολογιστικού

νέφους. Τέλος, παρουσιάζεται κι η περίπτωση τα διάφορα συμβάντα να παράγονται από διάφορα επίπεδα τα οποία μπορούν να συσχετίζονται μεταξύ τους χρησιμοποιώντας ειδικές συναρτήσεις και άλγεβρα, τεχνικές που δεν έχουν παρουσιαστεί σε άλλες ερευνητικές εργασίες μέχρι σήμερα.

4.3 Ερευνητική Προσέγγιση και Σχεδιασμός

Σε αυτή την παράγραφο παρουσιάζεται ερευνητικά η προσέγγιση και η φιλοσοφία αυτής της Διατριβής για την παρουσίαση ενός καινοτόμου Δικτύου Παρακολούθησης και Επεξεργασίας Συμβάντων (Event Processing Network – EPN) το οποίο μπορεί να κατανέμεται σε διαφορετικούς εικονικούς πόρους (μηχανές) πολλών παρόχων υπολογιστικού νέφους προκειμένου να παρακολουθούν και να δίνουν πληροφορίες για συγκεκριμένες εφαρμογές υπολογιστικού νέφους[A2]. Ο απώτερος σκοπός μιας τέτοιας ανάπτυξης ενός τέτοιου δικτύου παρακολούθησης είναι η ανίχνευση ευκαιριών αναπροσαρμογής της υποδομής γεγονός που βοηθά στην εξασφάλιση της επιθυμητής ποιότητας της παρεχόμενης υπηρεσίας νεφουπολογιστικής εφαρμογής.

4.3.1 Ερευνητικό Μοντέλο

Στην παρούσα Διατριβή για τις ανάγκες ενός συστήματος παρακολούθησης υποδομών υπολογιστικού νέφους πολλών παρόχων με χρήση κατανεμημένης αρχιτεκτονικής χρησιμοποιείται ένα Δίκτυο Επεξεργασίας Συμβάντων (EPN- Event Processing Network). Ένα τέτοιο δίκτυο είναι ένα αφαιρετικό μοντέλο που απαρτίζεται από την χρήση διαφόρων στοιχείων όπως Event Processing Agents (EPA), Event Producers και Event Consumers όλα διασυνδεδεμένα μέσω ενός συνόλου από Event Channels. Οι λειτουργικότητες των προαναφερθεισών στοιχείων περιγράφονται με λεπτομέρεια στην εργασία των Etzion και Niblett [51].

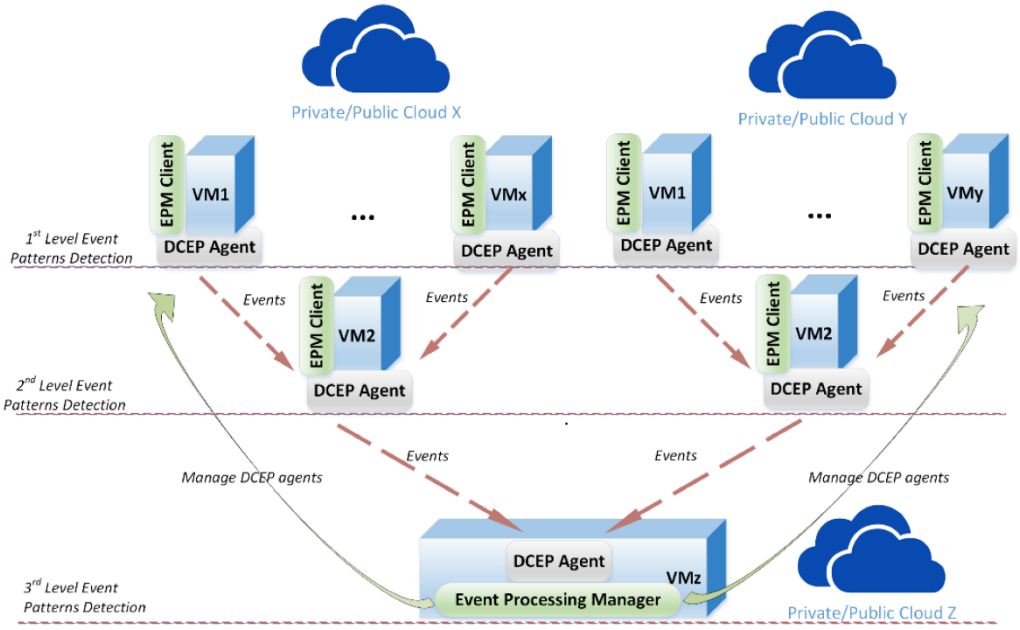
Πιο συγκεκριμένα, οι Event Producers είναι πόροι που γεννούν συμβάντα ενώ οι Event Consumers είναι πόροι που καταναλώνουν και δέχονται αυτά τα συμβάντα. Σε περιβάλλοντα υπολογιστικού νέφους πολλών παρόχων, οι Event Producers συμπεριλαμβάνουν εικονικές μηχανές που φιλοξενούν τμήματα των κατανεμημένων εφαρμογών και εκπέμπουν συμβάντα από παρακολούθηση της υποδομής και αφορούν τόσο την κατάσταση της εικονικής μηχανής όσο και την κατάσταση της εφαρμογής που φιλοξενείται σε αυτήν. Τα Event Processing Agents (EPAs) λειτουργούν τόσο ως Event Consumers που καταναλώνουν συμβάντα από την παρακολούθηση της υποδομής, όσο και ως Event Producers καθώς είναι σε θέση να ανιχνεύσουν σύνθετα πρότυπα συμβάντων και να τα αναμεταδώσουν σε άλλα τμήματα του Δικτύου Επεξεργασίας Συμβάντων (EPN- Event Processing Network). Κάθε Event Processing Agent (EPA) έχει την δυνατότητα να «φιλτράρει», να «ταιριάζει» και να «παράγει» σύνθετα συμβάντα σύμφωνα με συγκεκριμένους ορισμένους κανόνες με βάση κάποια πρότυπα που δίνουν χρήσιμη πληροφορία όσον αφορά την κατάσταση των κατανεμημένων εφαρμογών υπολογιστικού νέφους πολλών παρόχων.

4.3.2 Αρχιτεκτονικός Σχεδιασμός

Στον Αρχιτεκτονικό σχεδιασμό που προτείνεται στην συγκεκριμένη Διατριβή[A2], θεωρούμε ότι οι Event Processing Agents (EPAs) υλοποιούνται χρησιμοποιώντας διασυνδεδεμένες μηχανές σύνθετης επεξεργασίας συμβάντων (CEP Engines). Ο σκοπός της τεχνικής CEP είναι να ανιχνεύσει γεγονότα και πρότυπα με μεγάλη σπουδαιότητα όπως ευκαιρίες από άποψη

κόστους ή ρίσκα πχ. στον τομέα της ασφάλειας για την υπάρχουσα τοπολογία πολλών παρόχων υπολογιστικού νέφους και να απαντήσει σε αυτά άμεσα και αποτελεσματικά.

Η χρήση στην παρούσα Διατριβή πολλών Event Processing Agents σε μια κατανεμημένη λογική φέρνει στο προσκήνιο τα πλεονεκτήματα της πολύ-επίπεδης επεξεργασίας σύνθετων συμβάντων (CEP). Η Αρχιτεκτονική που ακολουθείται είναι η ακόλουθη:



Σχήμα 6: Αρχιτεκτονική τριών επιπέδων[A2]

Με βάση το ανωτέρω αρχιτεκτονικό διάγραμμα, παρουσιάζονται στην συγκεκριμένη τοπολογία τρία ξεχωριστά επίπεδα Σύνθετης Επεξεργασίας Συμβάντων (CEP) τα οποία ανιχνεύουν ιεραρχικά αξιόλογα συμβάντα όπως για παράδειγμα ακολούθως:

- Μέση κατανάλωση επεξεργαστικής ισχύος (CPU) > 80% για ένα instance application server (level 1) για συγκεκριμένη νεφουπολογιστική εφαρμογή.
- Μέση κατανάλωση επεξεργαστικής ισχύος (CPU) > 80% για όλα τα instance application servers του παρόχου X (level 2) για συγκεκριμένη νεφουπολογιστική εφαρμογή. Σε αυτή την περίπτωση έχουν καταγραφεί μέσω ενός configuration file τα χαρακτηριστικά της εφαρμογής και φυσικά οι ανάγκες σε υπολογιστικούς πόρους που απαιτούνται για την ορθή λειτουργία της και την άμεση και πρέπουσα χρονική απόκρισή της. Με αυτό τον τρόπο υπάρχει δυνατότητα αναπροσαρμογής αναλόγως της εκάστοτε παρακολουθούμενης εφαρμογής(agnostic) του κατωφλίου κατανάλωσης επεξεργαστικής ισχύος (CPU threshold parameter).
- Μέση κατανάλωση επεξεργαστικής ισχύος (CPU) > 80% για όλα τα instance application servers όλων των παρόχων (level 3) για συγκεκριμένη νεφουπολογιστική εφαρμογή. Το συγκεκριμένο κατώφλι αναπροσαρμόζεται εύκολα και μέσω ενός κεντρικού configuration file ευρισκόμενου στο κεντρικό server 3ου επιπέδου ανάλογα με τις ανάγκες της εκάστοτε παρακολοθούμενης εφαρμογής όπως ήδη αναφέρθηκε (EPM server όπως εξηγείται στην επομενη σελίδα).

Κάθε ένα από τα Event Processing Agents (EPAs) που είναι εγκατεστημένα στις διάφορες εικονικές μηχανές (Virtual Machines) επικοινωνούν και συνεργάζονται μεταξύ τους με πρωτόκολλα publish-subscribe βασισμένα στην θεωρία ουρών μηνυμάτων και μετάδοσης συμβάντων δια μέσου των τριών επιπέδων επεξεργασίας συμβάντων όπως φαίνονται στο διάγραμμα προηγουμένως. Πιο συγκεκριμένα στην συγκεκριμένη αρχιτεκτονική (Σχήμα 6), το δίκτυο των "DCEP" agents που είναι στην περίπτωση μας οι Event Processing Agents (EPAs) δομείται μεταξύ τριών επιπέδων:

1. Το επίπεδο της εικονικής μηχανής (instance VM layer) ή αλλιώς το 1ο επίπεδο ανίχνευσης προτύπων συμβάντων
2. Το επίπεδο του υπολογιστικού νέφους παρόχου (Cloud Provider layer) ή αλλιώς το 2ο επίπεδο ανίχνευσης προτύπων συμβάντων
3. Το καθολικό επίπεδο (Global layer) ή αλλιώς το 3ο επίπεδο ανίχνευσης προτύπων συμβάντων

Πιο αναλυτικά, το πρώτο επίπεδο αναφέρεται στην εγκατάσταση και ρύθμιση Event Processing Agents (EPAs) σε κάθε εικονική μηχανή με στόχο να εστιάζει στην συγκέντρωση πληροφορίας, στο φιλτράρισμά της και στην μετάδοση συμβάντων που αφορούν την κατάσταση της υποδομής (της συγκεκριμένης εικονικής μηχανής). Το δεύτερο επίπεδο περιλαμβάνει την χρήση ενός Event Processing Agent (EPA) ανά νεφουπολογιστικό πάροχο για εξαγωγή πληροφορίας και συμπερασμάτων σχετικά με τα δεδομένα του παρόχου και την εφαρμογή που φιλοξενεί στις υποδομές του. Γίνεται με αυτό τον τρόπο μια συγκέντρωση πληροφορίας σχετικά με την κατάσταση των πόρων και των εφαρμογών που φιλοξενούνται ανά πάροχο υπολογιστικού νέφους βασισμένο στα δεδομένα των τοπικών "DCEP" agents και τις αναφορές κατάστασης από κάθε εικονική μηχανή. Τελικά, το τρίτο επίπεδο περιλαμβάνει και συγκεντρώνει τα δεδομένα που προκύπτουν από τους "DCEP" agents του δευτέρου επιπέδου προκειμένου να προκύψει μια τελική συγκέντρωση όλων των δεδομένων όλης της νεφουπολογιστικής υποδομής από όλους τους παρόχους ώστε να καταλήξουμε σε μια συνολική τελική εικόνα παρακολούθησης του συνόλου της υποδομής (monitoring).

Επιπλέον, η συγκεκριμένη Αρχιτεκτονική φαντάζει ως πιο αναγκαία από ποτέ καθώς σε ένα δυναμικό περιβάλλον όπου οι πόροι των παρόχων υπολογιστικού νέφους που φιλοξενούν εφαρμογές δεν είναι προκαθορισμένοι και στατικοί από την αρχή, ένας συγκεκριμένος μηχανισμός για την διατήρηση και παρακολούθηση του δικτύου επεξεργασίας συμβάντων που περιγράφηκε είναι απαραίτητος. Σε αυτή την λογική στην Αρχιτεκτονική που έχει παρουσιαστεί στο Σχήμα 6 γίνεται χρήση του Event Processing Management (EPM) server και των αντιστοίχων EPM clients (λογική Server-Client) που έχει ως στόχο την αποτελεσματική διαχείριση των "DCEP" agents δηλαδή την αποτελεσματική εγκατάσταση, deployment, συγχρονισμό και επικοινωνία των "DCEP" agents που φιλοξενούνται σε διάφορες Εικονικές Μηχανές διαφόρων παρόχων υπολογιστικού νέφους. Μετά την επιτυχή εγκατάσταση αυτών των "DCEP" agents το Event Processing Management (EPM) αναλαμβάνει το «σετάρισμα» των διαφόρων κανόνων για σύνθετα πρότυπα επεξεργασίας συμβάντων που όμως λαμβάνουν χώρα στους Event Processing Agent (EPA) που εν προκειμένω είναι οι "DCEP" agents όπως έχει ήδη αναφερθεί. Αυτά τα συμβάντα συλλέγονται από τους "DCEP" agents από διαφόρους αισθητήρες εβρισκόμενοι πάνω στις διάφορες εικονικές μηχανές.

Με άλλα λόγια στην Αρχιτεκτονική παρακολούθησης του υπολογιστικού νέφους που έχουμε αναπτύξει στην παρούσα Διατριβή[A2], χρησιμοποιούμε ως βασικό Agent τον EPA και ένα

βοηθητικό του που είναι ο EPM client. Το EPM υποσύστημα είναι υπεύθυνο για την διαχείριση του δικτύου παρακολούθησης(monitoring) της νεφουπολογιστικής υποδομής που χρησιμοποιεί τους Event Processing Agents (EPA) ("DCEP" Agents) που έχουμε ήδη αναφέρει. Το συγκεκριμένο υποσύστημα EPM που είναι υπεύθυνο για την εγκατάσταση και διαχείριση του συστήματος παρακολούθησης(monitoring) ακολουθεί μια λογική πελάτη-εξυπηρετητή (client- server) και απαρτίζεται από τα δύο ακόλουθα βασικά στοιχεία:

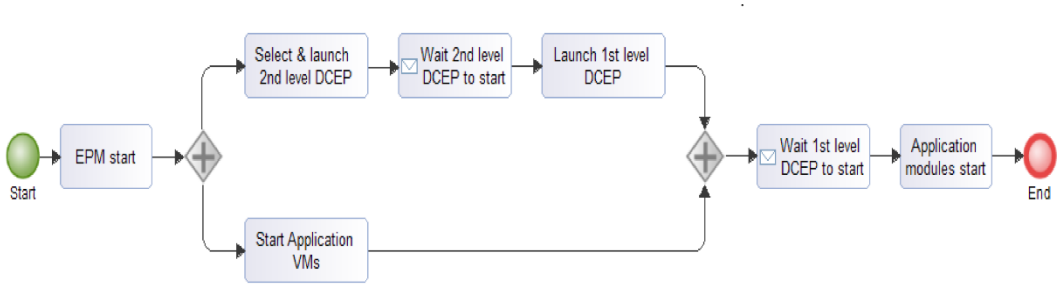
1. EPM Clients: Εγκαθίστανται και ελέγχουν τους EPA Agents ("DCEP" Agents) σε κάθε εικονική μηχανή στο πρώτο, δεύτερο ή τρίτο επίπεδο της κατανεμημένης αναγνώρισης προτύπων συμβάντων. Δεν ασχολούνται ούτε εμπλέκονται σε διαδικασίες που αφορούν παρακολούθηση (monitoring) της νεφουπολογιστικής υποδομής , όπως κάνουν οι EPA Agents, και είναι ξεχωριστές οντότητες. Αυτοί οι EPM Clients περιέχουν κώδικα (configuration scripts) για τον καθορισμό και την εκκίνηση του πρώτου ή δεύτερου επιπέδου των Event Processing Agents (EPA) σύμφωνα με οδηγίες που στέλνονται από τον Event Processing Management (EPM) server. Αυτοί οι EPM Clients περιέχουν επίσης πληροφορίες και κωδικούς διασύνδεσης με τον EPM server. Οι EPM Clients έρχονται συνήθως ήδη προ εγκατεστημένοι σε εικονική μηχανή και ενεργοποιούνται κατά την διάρκεια της εκκινήσεως μια εικονικής μηχανής. Μια εναλλακτική προσέγγιση είναι ο EPM server πρώτα να συνδέεται στις υπό εκκίνηση εικονικές μηχανές και να εγκαθιστά τους EPM Clients μετά την εκκίνηση των εικονικών μηχανών. Αυτό φυσικά προϋποθέτει ότι ο EPM server θα γνωρίζει εκ των προτέρων για τις εικονικές μηχανές που θα χρησιμοποιηθούν, τις IP διευθύνσεις τους και τους αντίστοιχους κωδικούς και επίσης θα είναι δυνατή η χρήση του SSH shell στον κεντρικό server.

2. Ο EPM server είναι ο ελεγκτής και διαχειριστής της λειτουργίας των EPM Clients. Είναι ο «εγκέφαλος» του EPM υποσυστήματος και φιλοξενείται στο τρίτο επίπεδο της κατανεμημένης αρχιτεκτονικής επεξεργασίας σύνθετων συμβάντων. Αυτό φαίνεται εμφανώς στο Σχήμα 6 της Αρχιτεκτονικής. Ο συγκεκριμένος server είναι όπως έχει αναφερθεί υπεύθυνος για την εγκατάσταση EPM Clients στις διάφορες εικονικές μηχανές – εάν δεν έχουν ήδη εγκατασταθεί – και μετά για την διαχείριση τους μέσω εντολών και διοχέτευσης προς αυτούς κατάλληλων κανόνων(rules) σύνθετης επεξεργασίας συμβάντων. Με αυτό τον τρόπο καθοδηγεί την ανάπτυξη του EPN δικτύου. Επιπλέον ένας ακόμα ρόλος του EPM server είναι να ελέγχει περιοδικά το status των EPM Clients καθώς και των εικονικών μηχανών στις οποίες φιλοξενούνται και αν διαπιστωθεί δυσλειτουργία ή λειτουργία εκτός, να μπορεί με κατάλληλες εντολές να αναπροσαρμόζει το EPN δίκτυο. Σε περίπτωση που διαπιστωθεί μια δυσλειτουργία ή ανεπάρκεια σε κάποιο κόμβο(Event Processing Agent (EPA)) η λειτουργία για μετάδοση δεδομένων monitoring αναπροσαρμόζεται και εξυπηρετείται από άλλον κόμβο. Αυτό οφείλεται στην δυνατότητα δυναμικής αναπροσαρμογής που προσφέρει το message queuing πρωτόκολλο. Περισσότερες λεπτομέρειες δίνονται ακολούθως. Με αυτό τον τρόπο ακόμα και αν χαθεί η λειτουργία κάποιου κόμβου ενδιαμέσου επιπέδου (2^{nd} layer) το routing των μηνυμάτων γίνεται μέσω άλλου κόμβου προς το $3^{ο}$ τελικό επίπεδο συγκέντρωσης πληροφορίας. Συνεπώς εξασφαλίζεται η υψηλή διαθεσιμότητα της υπηρεσίας παρακολούθησης (monitoring).

Στη συνέχεια παρουσιάζεται η διαδικασία που ακολουθείται βάση του σχεδιασμού που έχει γίνει σχετικά με την εκκίνηση μιας κατανεμημένης εφαρμογής σε περιβάλλον υπολογιστικού

νέφους πολλών παρόχων. Πρωτίστως και πριν η κατανεμημένη εφαρμογή υπολογιστικού νέφους πολλών παρόχων ξεκινήσει την λειτουργία της, θα πρέπει το δίκτυο Event Processing Network να έχει «σεταριστεί» και να είναι έτοιμο να συλλέξει και να επεξεργαστεί συμβάντα που έχουν να κάνουν με την παρακολούθηση της υποδομής και της εφαρμογής (monitoring tasks).

Αφού πληρούνται τα παραπάνω, η διαδικασία περιγράφεται ακολούθως στο σχήμα 7:



Σχήμα 7: Διαδικασία εκκίνησης κατανεμημένης εφαρμογής πολλών παροχών υπολογιστικού νέφους σύνθετης επεξεργασίας συμβάντων.

Πιο συγκεκριμένα, μετά την εκκίνηση μιας εικονικής μηχανής, ο εγκατεστημένος σε αυτήν EPM Client προσπαθεί να συνδεθεί στον EPM Server χρησιμοποιώντας SSH protocol. Εάν η σύνδεση είναι επιτυχής, ο EPM Client στέλνει την σχετική αναγνωριστική πληροφορία για την συγκεκριμένη εικονική μηχανή και μετά ο EPM Server καταχωρεί ένα συγκεκριμένο μοναδικό ID νούμερο για αυτήν. Αυτό αποθηκεύεται σε κάποιο αρχείο του EPM Server για πιθανές μελλοντικές διασυνδέσεις (sessions). Ο EPM Server ως κεντρικός διαχειριστής αποφασίζει χρησιμοποιώντας συγκεκριμένη στρατηγική, ποιες εικονικές μηχανές θα λειτουργούν ως πρώτου επιπέδου Event Processing Agents (EPA) και ποιες ως δεύτερου επιπέδου Event Processing Agents (EPA). Η ακόμα και για τις περιπτώσεις που μπορεί μια εικονική μηχανή να λειτουργεί ταυτόχρονα, για ειδικούς λόγους, τόσο σαν πρώτου επιπέδου Event Processing Agent (EPA) και σαν δεύτερου επιπέδου Event Processing Agent (EPA). Ακολούθως, μετά την λήψη των σχετικών αποφάσεων, ο EPM Server ενημερώνει τους EPM Clients για τους ρόλους τους, τους περνάει σχετικές πληροφορίες και οι EPM Clients εκτελούν σχετικές εντολές διαμόρφωσης των εικονικών μηχανών και προετοιμάζουν έτσι την σωστή λειτουργία των EPA Agents. Με αυτό τον τρόπο οι EPM Clients παρακολουθούν την εκκίνηση των EPA Agents και μόλις αυτή ολοκληρωθεί στην κάθε εικονική μηχανή αυτοί οι Clients ενημερώνουν τον EPM Server. Όταν το EPN δίκτυο ετοιμαστεί και είναι σε πλήρη λειτουργία, τότε ο EPM Server στέλνει σήματα προς την κατανεμημένη εφαρμογή να ξεκινήσει να λειτουργεί.

Είναι πολύ σημαντικό να αναφερθεί ότι στην διαδικασία που ενημερώνονται και ενεργοποιούνται τα EPA Agents, οι δεύτερου επιπέδου DCEP (EPA Agents) ενεργοποιούνται πριν ξεκινήσουν οι πρώτου επιπέδου DCEP (EPA Agents), καθότι είναι απαραίτητο να περαστεί η δικτυακή πληροφορία του δεύτερου επιπέδου στο πρώτο. Έτσι τελικά οι πρώτου επιπέδου DCEP (EPA Agents) προωθούν τα συμβάντα προς τους DCEP (EPA Agents) του δεύτερου επιπέδου καταλήγοντας σε μια ιεραρχική δεντρική συνεργαζόμενη δομή. Στην συνέχεια οι δεύτερου επιπέδου DCEP (EPA Agents) προωθούν τα συμβάντα προς το τρίτο επίπεδο όπου βρίσκεται ο EPM Server και είναι ο τελικός αποδέκτης. Αυτός αποφασίζει τελικά για τα περαιτέρω actions.

4.4 Υλοποίηση

Σε αυτή την παράγραφο χρησιμοποιώντας την Αρχιτεκτονική που παρουσιάστηκε προηγουμένως, παρουσιάζουμε τις τεχνολογίες που χρειάζονται για να υλοποιηθεί το κάθε τμήμα της. Με βάση την αρχιτεκτονική της κατανεμημένης επεξεργασίας συμβάντων δύο βασικές λειτουργίες πρέπει να υλοποιηθούν αποτελεσματικά. Η πρώτη αφορά την θεωρία ουράς και την μετάδοση στους subscribers των συμβάντων των σχετικών με την παρακολούθηση πόρων της υποδομής υπολογιστικού νέφους πολλών παρόχων. Η δεύτερη αφορά την σύνθετη επεξεργασία συμβάντων αυτών των γεγονότων. Για την πρώτη λειτουργία χρησιμοποιείται ένα Enterprise Service Bus (ESB) ενώ για την δεύτερη λειτουργία γίνεται χρήση διαφόρων μηχανών επεξεργασίας σύνθετων συμβάντων (CEP Engines) . Στην συγκεκριμένη υλοποίηση οι παραγωγοί συμβάντων εστιάζονται στις ακόλουθες περιπτώσεις:

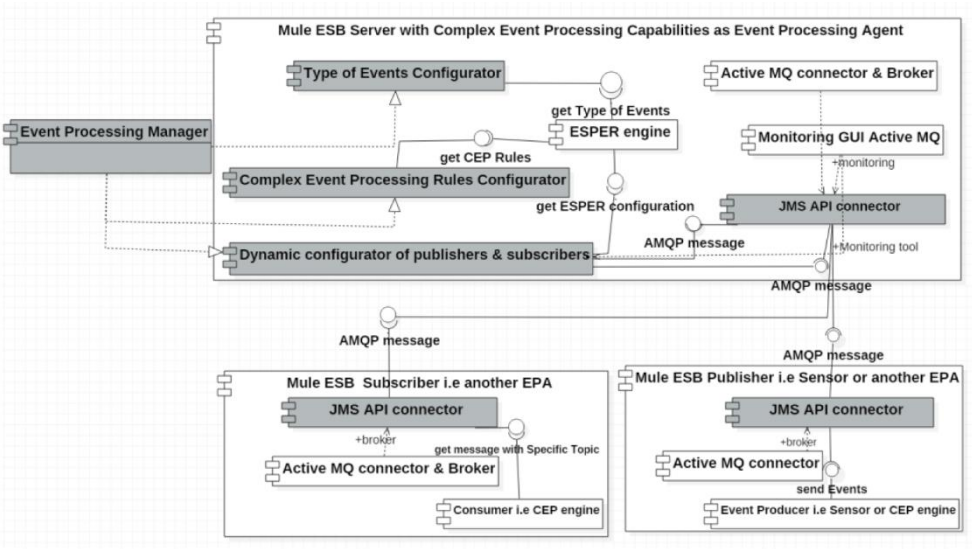
- Από εικονικούς πόρους του υπολογιστικού νέφους και σχετικούς αισθητήρες που παράγουν πληροφορία σχετική με την απόδοση της υποδομής όπως η χρήση της Μνήμης των εικονικών μηχανών, το φορτίο της επεξεργαστικής ισχύος κτλ.

- Από αισθητήρες που έχουν να κάνουν με την μέτρηση της απόδοσης της κατανεμημένης νεφουπολογιστικής εφαρμογής όπως πχ. η μέτρηση της χρονικής απόκρισης της εφαρμογής.

- Οι Agents EPAs που παράγουν σύνθετα συμβάντα βασισμένα στους τύπους των γεγονότων από τις προηγούμενες δύο περιπτώσεις.

Τα δεδομένα που παράγονται από τους παραπάνω παραγωγούς συμβάντων (event producers) μεταφέρονται μέσω της χρήσης ενός Enterprise Service Bus (ESB). Ένα instance αυτού του ESB είναι εγκατεστημένο σε κάθε εικονική μηχανή που φιλοξενεί στοιχεία κατανεμημένης εφαρμογής υπολογιστικού νέφους πολλών παρόχων. Αυτές οι εικονικές μηχανές μπορεί να εξυπηρετούν διάφορους ρόλους όπως Βάσεις δεδομένων της εφαρμογής, application servers της εφαρμογής κτλ. Παράλληλα, ένα instance μιας μηχανής επεξεργασίας σύνθετων συμβάντων (CEP) είναι επίσης εγκατεστημένη σε κάθε εικονική μηχανή που χρησιμοποιείται. Αυτή η μηχανή επεξεργασίας σύνθετων συμβάντων χρησιμοποιεί πρότυπα (κανόνες) για επεξεργασία συμβάντων που προσδιορίζουν τις συνθήκες με βάση τις οποίες συμβάντα για επαναπροσδιορισμό των πόρων της υποδομής θα παράγονται ή σύνθετα συμβάντα που έχουν συγκεντρώσει πληροφορίες από διάφορα συμβάντα χαμηλότερα στην ιεραρχία προωθούνται προς επεξεργασία σε ανώτερο ιεραρχικά αρχιτεκτονικό επίπεδο όπως έχουμε δείξει στο σχήμα 6.

Η τεχνολογία υλοποίησης του περιεγραμμένου συστήματος παρακολούθησης νεφουπολογιστικής υποδομής πολλών παρόχων φαίνεται ακολούθως στο σχήμα 8:



Σχήμα 8: Τεχνολογίες υλοποιήσης και επικοινωνιών μεταξύ των διαφόρων στοιχείων[A2]

Για την υλοποίηση του ESB έχουμε χρησιμοποιήσει και «σετάρει» το λογισμικό ανοιχτού κώδικα που αναφέρεται ως MuleSoft [52]. Για την επιλογή της συγκεκριμένης τεχνολογίας βασιστήκαμε στην παρούσα Διατριβή στην μελέτη και εκτενή αξιολόγηση που έχει γίνει από τους Rademakers & Dirksen [53] ως το “best-of-breed” προιόν διαθέσιμο ικανοποιώντας τα ακόλουθα κριτήρια:

- Βασική λειτουργία ενός ESB λογισμικού
- Ποιότητα διαθέσιμης πληροφορίας
- Από την άποψη της αγοράς λογισμικού μεγάλη διείσδυση και χρήση στον συγκεκριμένο τομέα των Enterprise Service Buses.
- Συνεχής ανάπτυξη και εξέλιξη του λογισμικού και εκτεταμένη υποστήριξη από το community
- Δυνατότητα ανάπτυξης custom λογικής
- Χρήση transport protocols και ευέλικτων τρόπων διασύνδεσης
- Δυνατότητα χρήσης ανοιχτού λογισμικού frameworks
- Πολύ καλές υπηρεσίες υποστήριξης της ESB πλατφόρμας για το enterprise version

Επιπλέον, λόγω της ανάγκης για δυναμική προσαρμογή της υποδομής του υπολογιστικού νέφους και της δέσμευσης και αποδέσμευσης ανάλογα των αναγκών διαφόρων εικονικών μηχανών, είναι προφανές ότι ένας ευέλικτος τύπος πρωτοκόλλου μηνυμάτων πρέπει να χρησιμοποιείται στην παρούσα υλοποίηση προκειμένου να μεταφέρονται πρωτογενή δεδομένα (raw data) από τις διάφορες πηγές δεδομένων (πχ. hardware + software sensors) προς τους Agents EPAs. Γι αυτό τον λόγο, υιοθετήθηκε η χρήση του Advanced Message Queing Protocol (AMQP) προκειμένου να μεταφέρονται τα διάφορα συμβάντα και με την βοήθεια του MuleSoft ESB, σύμφωνα με το publish-subscribe μηχανισμό. Πιο συγκεκριμένα, σε αυτό το messaging πρωτόκολλο χρησιμοποιούμε το Apache Active MQ το οποίο είναι από τα πιο γνωστά και αποτελεσματικά πρωτόκολλα σε μετάδοση μηνυμάτων μεταξύ

συστημάτων και εικονικών μηχανών[54]. Είναι ένα πρωτόκολλο που χρησιμοποιεί ένα Message Broker γραμμένο σε γλώσσα Java και χρησιμοποιείται με ένα πλήρη Java Message Service (JMS) client. Τα βασικά χαρακτηριστικά του καλύπτουν τις ανάγκες της παρούσας κατάστασης για μια κατανεμημένη αρχιτεκτονική σύνθετης επεξεργασίας συμβάντων (CEP) μεταξύ πολλών παρόχων υπολογιστικού νέφους. Πιο ειδικά παρέχει τα ακόλουθα χαρακτηριστικά:

- Active MQ: είναι ένα standards-based πρωτόκολλο το οποίο είναι συμβατό με JMS 1.1. Η τεχνολογία του JMS που χρησιμοποιεί προσφέρει πολλά προνόμια όπως παράδοση μυνημάτων με ασύγχρονο τρόπο, διατήρης των μυνημάτων για subscribers που είναι πολλοί σημαντικοί για την δυναμικά αναπτυσσόμενη/αναπροσαρμοζόμενη αρχιτεκτονική μεταξύ πολλών παρόχων.

- Το Active MQ παρέχει μια ευρεία γκάμα από επιλογές διασυνδέσεων με χρήση πρωτοκόλλων όπως HTTPS, multicast, TCP, SSL. Αυτές οι επιλογές δίνουν μια σημαντική ευελιξία στην πραγματοποίηση επικονωνιών μεταξύ publishers – subscribers.

- Εξαιτίας της γενικότερης προτεινόμενης αρχιτεκτονικής, μια λογική loosely coupled του Active MQ δίνει λιγότερες εξαρτήσεις και είναι πολύ περισσότερη χρήσιμη σε δυναμικά αναπροσαρμοζόμενη αρχιτεκτονική με συμβάντα [54]. Με αυτό τον τρόπο εξπυπηρτούναι και οι περιπτώσεις που υπάρξει απώλεια κάποιου πχ. κόμβου του 2^{ου} επιπεδου που συλλέγει raw data events. Λόγω της loosely coupled αρχιτεκτονικής τα συγκεκριμένα δεδομένα events μπορούν να ανακατευθυνθούν σε subscriber Active MQ ευρισκόμενου επίσης στο 2^{ο} επίπεδο και «ακούγωντας» στον συγκεκριμένο header/topic μυνημάτων, να αναπροωθούνται αυτά τα μυνήματα προς το τελικό 3^{ο} επίπεδο χωρίς να χάνεται η πληροφορία της παρακολούθησης.Με αυτό τον τρόπο εξασφαλίζεται η υψηλή διαθεσιμότητα της υπηρεσίας παρακαλούθησης. Αξίζει να σημειωθεί ότι σε αυτό βοηθά και ο συγχρονισμός μεταξυ των διαφόρων Brokers που επιτυγχάνεται μέσω του δικτύου EPM όπως παρουσιάστηκε προηγουμένως.

Επίσης για την ανάγκη της σύνθετης διαχείρισης συμβάντων η τεχνολογία που χρησιμοποιείται είναι η ESPER τεχνολογία [55]. Η τεχνολογία ESPER είναι μια ανοιχτού κώδικα μηχανή που συνδυάζει το Event Stream Processing και τις ιδιότητες της επεξεργασίας σύνθετων συμβάντων για επεξεργασία δεδομένων. Το ESPER χρησιμοποιεί το Event Processing Language (EPL) προκειμένου να δομήσει ένα υψηλού επιπέδου επεκτασιμότητας και αξιοποίησης μνήμης εργαλείο επεξεργασίας ροής δεδομένων προκειμένου να ανιχνεύει καταστάσεις και να δημιουργεί συμβάντα όταν παραβιάζονται κάποιες συνθήκες. Η συγκεριμένη EPL γλώσσα δίνει την δυνατότητα οι ροές δεδομένων να φιλτράρονται, να συγκεντρώνονται και να εκπέμπονται συγκεκριμένα συμβάντα υπό συγκεκριμένα πρότυπα και συνθήκες.

Στην υλοποίηση που αναπτύχθηκε στην συγκεκριμένη Διδακτορική εργασία[A2], παρουσιάζεται στο σχήμα 8 το Unified Model Language διάγραμμα στοιχείων και υποστοιχείων ανά επίπεδο (σε κάθε από τα 3 επίπεδα) και ανά EPA Agent. Με γκρι χρώμα αναφέρονται τα νέα στοιχεία που υλοποιήθηκαν με κώδικα προγραμματισμού σε αυτή την εργασία για το σύστημα παρακολούθησης (monitoring) και με τον τρόπο αυτό επαυξάνουν και ενδυναμώνουν τις λειτουργικότητες των στοχείων MuleSoft ESB & Esper που

χρησιμοποιούνται ήδη και δίνονται ως ανοιχτές τεχνολογίες. Συγκεκριμένα τα στοιχεία (modules) που υλοποιήθηκαν ανά επίπεδο είναι τα ακόλουθα:

- Type of Events Configurator: Αυτό είναι ένα υποτμήμα του συστήματος παρακολούθησης (monitoring) που παρέχει στην μηχανή ESPER πληροφορίες σχετικά με τον τύπο συμβάντων που πρόκειται να επεξεργαστεί.

- Complex Event Processing Rules Configurator: Αυτό είναι ένα υποτμήμα του συστήματος παρακολούθησης (monitoring) που εισάγει τα κατάλληλα πρότυπα και κανόνες γεγονότων εκπεφρασμένα σε γλώσσα EPL προκειμένου να δίνεται η ευκαιρία να αναγνωρίζονται σύνθετες καταστάσεις γεγονότων σε πραγματικό χρόνο.

- ESPER Engine: Αυτό το υποτμήμα αφορά στο βασικό ESPER engine το οποίο εμφανίζεται σε κατανεμημένη αρχιτεκτονική σε όλο το υπολογιστικό νέφος και στόχος είναι η βέλτιστη ανίχνευση σύνθετων συμβάντων.

- Dynamic Configurator of publishers & subscribers: Αυτό το υποτμήμα ορίζει τους διάφορους καταναλωτές των μυνημάτων με βάση τα topic τα οποία ορίζονται στο JMS API Active MQ service. Αυτό το functionality είναι που βοηθά στην παροχή υψηλής διαθεσιμότητας στην υπηρεσία αναταλλαγής μυνημάτων μεταξύ των κόμβων, ακόμα και εάν έχουμε απώλειες λειτουργίας κάποιων/κάποιου κόμβου.

- JMS API Connector: Αυτό το υποτμήμα χρησιμοποιείται ως ενδιάμεσο λογισμικό που έχει ως στόχο να στέλνει τα παραγόμενα συμβάντα προς κατανάλωση από τις διάφορες εικονικές μηχανές του υπολογιστικού νέφους.

- Monitoring GUI Active MQ: Αυτό το υποτμήμα έχει την χρησιμότητα ενός εργαλείου παρακολούθησης όπου κρίσιμη πληροφορία σχετικά με συμβάντα που μεταδίδονται μέσω publish-subscribe μοντέλου του Active MQ broker παρουσιάζονται.

- Active MQ Connector & Broker : Αυτό το υποτμήμα είναι ένας ανοιχτού κώδικα message broker γραμμένος σε JAVA (JMS).

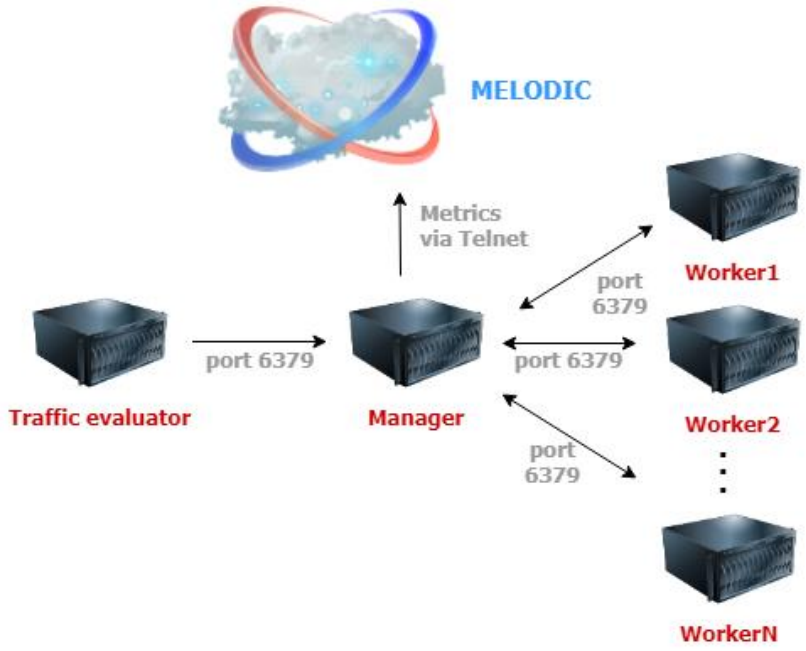
- Event Processing Manager: Αυτό είναι ένα υποτμήμα του συστήματος παρακολούθησης (monitoring) το οποίο είναι υπεύθυνο για τον συγχρονισμό και διαχείριση των διαφόρων EPAs και ESB instances τα οποία φιλοξενούνται σε όλες τις εικονικές μηχανές οι οποίες επίσης φιλοξενούν τμήματα κατανεμημένων εφαρμογών επεξεργασίας μεγάλων δεδομένων.

Χρησιμοποιώντας την παραπάνω τεχνολογία υλοποιούμε ένα σύστημα παρακολούθησης μας κατανεμημένης εφαρμογής για διαχείριση κυκλοφορίας οχημάτων η οποία χρησιμοποιεί πόρους υπολογιστικού νέφους και επεξεργάζεται τα διάφορα συμβάντα μέσω CEP Engine. Η συγκεκριμένη εφαρμογή είναι μια μεγάλων δεδομένων εφαρμογή η οποία λόγω της δυναμικής και απαιτητικής φύσης της είναι αναγκαία η φιλοξενία της σε νεφουπολογιστικούς πόρους πολλών παρόχων και είναι απαραίτητη γι αυτό τον λόγο η συνεχής και αποτελεσματική παρακολούθησή της (monitoring) για λόγους βελτιστοποίησης της χρήσης των πόρων του νέφους.

Στην συγκεκριμένη υλοποίηση το σύστημα ελέγχου κυκλοφορίας συλλέγει πληροφορίες από πολλές ετερογενείς πηγές (ανθρώπους, αυτοκίνητα, δημόσια μέσα μεταφοράς, σηματοδότες στους δρόμους κτλ) και συνδυαμορφώνεται από πολλούς παράγοντες όπως καιρός, μαζικά

γεγονότα ανθρώπων κτλ. Αυτή η κατάσταση περιλαμβάνει σενάρια διαχείρισης μεγάλων δεδομένων και άρα η δυνατότητα για ανίχνευση σύνθετων συμβάντων είναι αναγκαία για την αντίδραση και σωστή αντιμετώπιση περιπτώσεων που μπορούν να «κανιβαλίσουν» την λειτουργία της υποδομής και να επηρεάσουν την ποιότητα της παρεχόμενης υπηρεσίας της εφαρμογής. Η υπηρεσία αφορά την επεξεργασία σε πραγματικό χρόνο μεγάλου όγκου δεδομένων που προέρχονται από διάφορες πηγές και εξυπηρετούν εξομοιώσεις κίνησης οχημάτων. Πιο συγκεκριμένα η εφαρμογή που έχει υλοποιηθεί για τις ανάγκες της Διατριβής χρησιμοποιείται για να «τρέξει» εξομοιώσεις (simulations) που λαμβάνουν σαν δεδομένα εισόδου δεδομένα ελέγχου κίνησης. Σαν αποτέλεσμα των εξομοιώσεων παίρνουμε διάφορα μεγέθη congestion, travel times, average speeds, Total waiting times. Ετσι τα αποτελέσματα μπορούν να αξιοποιηθούν σε ένα σωρό περιπτώσεις πχ ρυθμίσεις για φωτεινούς σηματοδότες, κατασκευή νέων δρόμων κτλ.

Η υλοποιημένη γενική Αρχιτεκτονική του συγκεκριμένου συστήματος αποτελείται από τρία βασικά τμήματα: τον traffic evaluator, τον simulation manager και τους simulation workers. Τα δύο πρώτα στοιχεία (traffic evaluator && simulation manager) είναι σταθερά στοιχεία με συγκεκριμένα hardware(υλικό) χαρακτηριστικά και εκτελούν την ανάλυση των δεδομένων από τους workers (τα οποία έχουν αναπτυχθεί σε τεχνολογία python) και την συλλογή των σχετικών αποτελεσμάτων από την ανάλυση των δεδομένων της κίνησης από αυτούς. Ο κάθε worker τρέχει συγκεκριμένη εξομοίωση για τα σενάρια της κίνησης και μεταβιβάζει τα αποτελέσματα στα κεντρικά στοιχεία traffic evaluator && simulation manager. Ο κάθε worker μπορεί κάλλιστα να είναι μια εικονική μηχανή. Μια γενική εικόνα της αρχιτεκτονικής φαίνεται στο σχήμα 9:



Σχήμα 9: Αρχιτεκτονική εξομοίωσης κίνησης [A2]

Η συγκεκριμένη αρχιτεκτονική επιτρέπει την δυναμική αναπροσαρμογή της υπολογιστικής υποδομής μετά την επεξεργασία πρωτογενών δεδομένων (raw data) που λαμβάνονται στα πλαίσια των εξομοιώσεων από εξομοιούμενους αισθητήρες κίνησης που γεννούν σχετικά

δεδομένα. Οι συναρτήσεις σύνθετης επεξεργασίας συμβάντων που χρειάζονται για την παρακολούθηση (monitoring) της συγκεκριμένης υποδομής που «τρέχει» το λογισμικό για τις εξομοιώσεις και που τελικά αποφασίζουν για την πιθανή αναπροσαρμογή των πόρων πχ αριθμός από workers κτλ περιλαμβάνουν percentile & floor συναρτήσεις με συγκεκριμένα χρονικά παράθυρα. Η λογική των time batch windows buffers χρησιμοποιείται αντί των απλών sliding time windows. Σχετικές αναφορές των συναρτήσεων και των χρονικών περιθωρίων δίδεται εκτενώς στην βιβλιογραφία [79],[80].

Τα διάφορα κεντρικά στοιχεία αποτελούνται από το Esper CEP engine & ActiveMQ event broker όπως εξάλλου έχει περιγραφεί η συγκεκριμένη τεχνολογία στο σχήμα 8. Αυτά τα κεντρικά στοιχεία δέχονται τις διάφορες μετρικές από τα διάφορα workers και αναλόγως αποφασίζουν αν η συνολική εξομοίωση αναμένεται να τελειώσει στην ώρα της με βάση συγκεκριμένα Service Level Objectives.

Στα πλαίσια της σύνθετης επεξεργασίας συμβάντων οι μετρικές που συλλέγονται από τον simulation manager είναι οι ακόλουθες:

- TotalCores – αναφέρεται στον συνολικό αριθμό των cores που είναι διαθέσιμοι στους workers.
- RawExecutionTime – ο χρόνος που χρειάζεται να εκτελεστεί ένα απλό task σε έναν worker.
- SimulationLeftNumber – ο αριθμός των tasks (simulations) που αναμένουν να ολοκληρωθούν ακόμα.
- RemainingSimulationTimeMetric – ο υπολειπόμενος χρόνος μέσα στον οποίο το πρόγραμμα εξομοίωσης αναμένεται να ολοκληρωθεί.

Η μετρική ‘FinishTimeMargin’ είναι ένα άνω όριο για τον χρόνο που χρειάζεται να εκτελεστούν όλα τα tasks για τις εξομοιώσεις και υπολογίζεται ως ακολούθως:

$$\frac{(\text{SimulationLeftNumber} * \text{AverageExecutionTime})}{\text{TotalCoresMetric}} - \text{RemainingSimulationTime} \qquad (1)$$

Εάν τελικά η τιμή της ‘FinishTimeMargin’ μετρικής είναι μεγαλύτερη του μηδενός τότε εκπέμπεται ένα συμβάν ‘SimulationNotFinishOnTime’ το οποίο προσδιορίζει επαναπροσδιορισμό της υποδομής (τελικό reconfiguration) και έχει την ακόλουθη τιμή με βάση τον ακόλουθο υπολογισμό:

*(ceil(Simula-tionLeftNumber /TotalCores) * ETPercentile) - RemainingSimulatioTim-*eMetric (2)

4.5 Αξιολόγηση

Αναφερόμενοι στην ανωτέρω Αρχιτεκτονική και Τεχνολογία, στο συγκεκριμένο κεφάλαιο παρουσιάζεται η μέθοδος και τα αποτελέσματα των αξιολογήσεων της σχετικής τεχνολογίας. Χρησιμοποιούνται και στην παρούσα παράγραφο οι ίδιες έννοιες για τις συναρτήσεις και μετρικές όπως έχουν ήδη περιγραφεί προηγουμένως:

- TotalCores
- RawExecutionTime

- SimulationLeftNumber
- RemainingSimulationTimeMetric
- FinishTimeMargin

Η υποδομή που αναπτύχθηκε και χρησιμοποιήθηκε προκειμένου να αξιολογηθεί η λειτουργεία ενός σεναρίου που αφορά μια κατανεμημένη εφαρμογή Εξομοίωσης Κίνησης Οχημάτων περιλαμβάνει τα ακόλουθα τμήματα:

- 1 Microsoft Windows PC OS 10 (64 bit, RAM: 16 GB, CPU: 4 Cores, Hard Disk: 460 GB) το οποίο επιτελεί τον ρόλο των traffic evaluator & simulation manager που έχουν αναφερθεί προηγούμενα καθώς φιλοξενεί τις σχετικές εικονικές μηχανές (VMs) επάνω του.

- 3 Εικονικές Μηχανές (Virtual Machines) σε περιβάλλον Openstack Cloud που επιτελούν τον ρόλο των simulation workers. Αυτές οι εικονικές μηχανές έχουν OS Ubuntu 16 με τα ακόλουθα hardware χαρακτηριστικά: 64-bit CPUs , Hard Disk: 20 GB, RAM: 4 GB.

Επιπλέον, πέραν των ανωτέρω, χρησιμοποιήθηκε και ένας γεννήτορας συμβάντων (event generator) που είναι τοποθετημένος σε μια εικονική μηχανή στο φυσικό υπολογιστή (PC) και εξομοιώνει τα δεδομένα που γεννιόνται σε συνθήκες κίνησης οχημάτων πχ. από μετρητές οχημάτων στον δρόμο. Αυτός ο γεννήτορας είναι βασισμένος σε τεχνολογία κρυστάλλων (Quartz) που προσφέρει η Mule ESB τεχνολογία που έχουμε αναλύσει σε προηγούμενο κεφάλαιο. Επιπλέον προσφέρεται η δυνατότητα να καθορίζεται κάθε φορά ο ρυθμός εκπομπής αυτών των γεγονότων-συμβάντων ανά δευτερόλεπτο προκειμένου να προκύπτουν αξιόπιστα αποτελέσματα μετρήσεων. Τέλος σε όλες τις εικονικές μηχανές χρησιμοποιήθηκε DCEP Agent εγκατεστημένος προκειμένου να λειτουργήσει η κατανεμημένη εφαρμογή παρακολούθησης (monitoring) της υποδομής αλλά κ η δυνατότητα δυναμικής αναπροσαρμογής της με εκπομπή κατάλληλων γεγονότων πληροφόρησης (events).

Πιο συγκεκριμένα, για την αξιολόγηση μετρήθηκαν οι καταναλώσεις μνήμης και υπολογιστικής ισχύος (RAM & CPU) στις εικονικές μηχανές που φιλοξενούν τους DCEP Agents για διάφορους ρυθμούς εισερχομένων συμβάντων προς αυτές από τους γεννήτορες συμβάντων (εξομοίωση μέτρησης κίνησης). Ειδικότερα χρησιμοποιήθηκαν οι ακόλουθοι ρυθμοί συμβάντων (events):

- 500 events/sec

- 1000 events/sec

Και οι οποίοι αναφέρονται στα RawExecutionTime συμβάντα του σεναρίου εξομοίωσης της κίνησης. Τα υπόλοιπα συμβάντα όπως

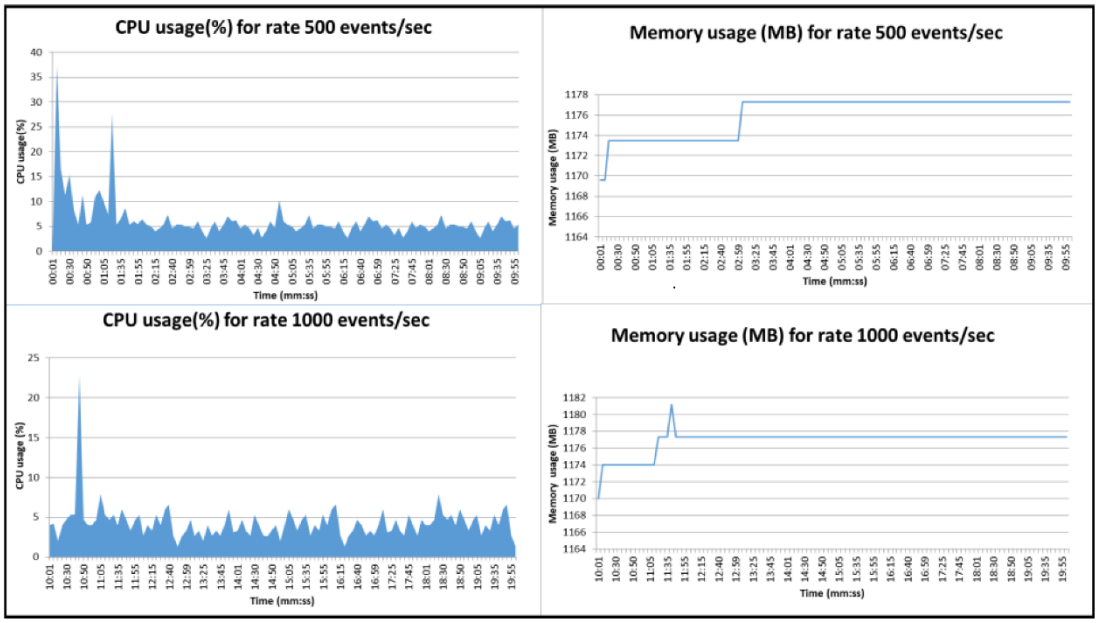
- TotalCores

- SimulationLeftNumber

- RemainingSimulationTimeMetric

Γεννιόνται από τον simulation manager (εικονική μηχανή) που φιλοξενείται στο φυσικό PC και που αποτελεί μια εφαρμογή χαρακτηριστικών μεγάλων δεδομένων και γεννά γεγονότα με ρυθμό χαμηλό της τάξης των 100 events/sec.

Οι μετρήσεις που προκύπτουν για επεξεργαστική ισχύ (CPU) και μνήμη (RAM) στις εικονικές μηχανές που φιλοξενούν τους DCEP Agents φαίνονται στο ακόλουθο σχήμα για μια χρονική περίοδο 10 λεπτών:



Σχήμα 10: Αξιολόγηση αποτελέσματων συστήματος παρακολούθησης (monitoring)[A2]

Με βάση τα παραπάνω αποτελέσματα παρατηρούμε ότι η αρχιτεκτονική DCEP (κατανεμημένη) παρουσιάζει μια σταθερή κατανάλωση μνήμης γύρω στο 30% με το οποίο συμφωνεί με ότι έχουμε πει για τις συναρτήσεις που χρησιμοποιούμε στην κατανεμημένη επεξεργασία συμβάντων μέσω της χρήσης χρονικών παραθύρων (time windows). Επίσης δεν παρουσιάζονται ιδιαίτερες διακυμάνσεις. Σχετικά με την κατανάλωση της επεξεργαστικής ισχύος παρατηρούμε ότι αυτή δεν υπερβαίνει το 36%. Καθώς μάλιστα ο ρυθμός εκπομπής των συμβάντων αυξάνει (events rate) τότε παρατηρούμε ότι η κατανάλωση επεξεργαστικής ισχύος είναι χαμηλότερη γεγονός που φανερώνει το μεγάλο πλεονέκτημα και δυνατότητα διαχείρισης που έχουν οι ουρές (message queues – Active MQ) που χρησιμοποιούνται για την μετάδοση των μηνυμάτων με τα δεδομένα από τους διάφορους Agents. Επίσης κατά την διάρκεια των δύο πρώτων λεπτών λειτουργίας του κατανεμημένου συστήματος επεξεργασίας παρατηρούμε κάποιες έντονες διακυμάνσεις (peaks), τόσο σε Μνήμη όσο και σε υπολογιστική ισχύ, οι οποίες είναι αναμενόμενες λόγω του warm up χρόνου που έχουν οι JVM DCEP Agents (υλοποιημένοι δηλαδή σε τεχνολογία java γεγονός που δημιουργεί αυτή την μικρή καθυστέρηση) προκειμένου να παρέχουν τα καλύτερα αποτελέσματα λειτουργίας. Στις δοκιμές που πραγματοποιήσαμε χρειάστηκε ένας χρόνος περίπου 5 λεπτών προκειμένου να ισορροπήσει το σύστημα σε μια απολύτως σταθερή κατάσταση καταναλώσεων.

Στην παραπάνω αξιολόγηση ένα σημαντικό πλεονέκτημα της υλοποίησης του κατανεμημένου συστήματος επεξεργασίας σύνθετων συμβάντων (DCEP) είναι η χρήση κατάλληλων τεχνολογιών που επιτρέπουν τον συνδυασμό δύο ή περισσότερων ροών δεδομένων και την χρήση σύνθετων μαθηματικών συναρτήσεων οι οποίες οδηγούν σε

σύνθετα πρότυπα συμβάντων (complex event pattern rules). Στο σενάριο εξομοίωσης κίνησης οχημάτων, διάφορα συμβάντα (events) συνενώνονται και επεξεργάζονται τα οποία φέρουν διαφορετικά topics ώστε τελικά αν οι συνθήκες το επιτρέπουν να εκπέμπονται scaling events που ονομάζονται, όπως αναφέρθηκαν πριν, SimulationNotFinishOnTime. Αυτά είναι τα γεγονότα (events) που θα παρέχουν χρήσιμη πληροφορία για πιθανή λήψη απόφασης σχετικά με την αναπροσαρμογή του υπολογιστικού νέφους.

4.6 Συμπεράσματα

Στο παρόν κεφάλαιο 4 της Διδακτορικής Διατριβής παρουσιάζεται μια τριών επιπέδων κατανεμημένη αρχιτεκτονική συστήματος παρακολούθησης μιας εφαρμογής μεγάλων δεδομένων (Big Data Intensive) η οποία φιλοξενείται σε πέραν το ενός παρόχους υπολογιστικού νέφους. Συγκεκριμένη εφαρμογή παρουσιάστηκε σχετικά η οποία δείχνει τα πλεονεκτήματα της συγκεκριμένης τεχνολογίας και αρχιτεκτονικής. Πιο συγκεκριμένα, η αξιολόγηση έδειξε ότι γίνεται χρήση επαρκών επιπέδων κατανάλωσης μνήμης και επεξεργαστικής ισχύος και με πολύ ικανοποιητικό επίπεδο επεξεργασίας σύνθετων συμβάντων. Σε σχέση με άλλες τεχνολογίες πχ. Siddhi CEP Engine, με την χρήση της Esper τεχνολογίας παρουσιάζεται το πλεονέκτημα ορισμού και χρήσης περισσότερων παραμετρικών συναρτήσεων που βοηθούν καλύτερα στην διαδικασία επεξεργασίας σύνθετων συμβάντων σε τέτοια συστήματα παρακολούθησης υποδομής υπολογιστικού νέφους(monitoring).

5 Προσαρμογή πόρων κατανεμημένων εφαρμογών με βασική παράμετρο τον χρόνο

5.1 Εισαγωγή

Η χρήση του υπολογιστικού νέφους αποτελεί στην εποχή μας ένα ανεκτίμητο πλεονέκτημα καθότι δίνει την δυνατότητα σε επιχειρήσεις και οργανισμούς να προμηθεύονται υπηρεσίες όταν και με όποιο τρόπο χρειαστεί με ένα εξαιρετικά οικονομικότερο τρόπο σε σχέση με τις κλασσικές μεθόδους των επιτόπιων φιλοξενούμενων κέντρων δεδομένων. Πιο συγκεκριμένα, η λογική του υπολογιστικού νέφους περιλαμβάνει την δυνατότητα παροχής ανά απαίτηση (on-demand) πόρων υπολογιστικής ισχύος, αποθηκευτικών και δικτυακών υπηρεσιών στον εκάστοτε αιτούντα πελάτη [58]. Αυτή η δυνατότητα επιτρέπει στους χρήστες να χρησιμοποιούν πόρους οι οποίοι αναπροσαρμόζονται αυτόματα (και οι οποίοι φιλοξενούν εφαρμογές) αναλόγως του συνεχώς μεταβαλλόμενου φορτίου και κίνησης στις υποδομές. Αυτή η αναπροσαρμογή των πόρων επιτυγχάνεται είτε με επέκταση είτε με συρρίκνωση της υποδομής (scaling out/scaling in) είτε με οριζόντιο είτε με κάθετο τρόπο αυτόματα σε στιγμές υψηλού ή χαμηλού φορτίου (ζήτησης) αντίστοιχα. Όπως έχει αναφερθεί και σε προηγούμενη παράγραφο, η ικανότητα δυναμικά να δεσμεύονται ή να αποδεσμεύονται υπολογιστικοί πόροι σύμφωνα με την ζήτηση από τον τελικό χρήστη ορίζεται στην επιστήμη των υπολογιστών ως ελαστότητα (elasticity) [59]. Η παροχή εικονικών μηχανών (Virtual Machines) σε συγκεκριμένο χρόνο και χωρίς καθυστερήσεις είναι πολύ σημαντική παράμετρος σύμφωνα με τις επιδιώξεις των χρηστών [60],[61].

Υιοθετώντας το υπολογιστικό νέφος σημαίνει την παροχή εικονικών πόρων το οποίο επιτρέπει στους χρήστες την πρόσβαση σε Terabytes αποθηκευτικών χώρων, υψηλή υπολογιστική ισχύ και υψηλή διαθεσιμότητα υπηρεσιών σε ένα μοντέλο κοστολόγησης ανά χρήση (pay-as-you-go) [62]. Καθώς πλέον όλο και περισσότερες επιχειρήσεις άρχισαν να υιοθετούν και να εμπιστεύονται το υπολογιστικό νέφος, ξεκίνησαν να εξάγουν το φορτίο τους σε υποδομές που προσφέρονται από απλούς παρόχους υπολογιστικού νέφους. Αυτό σε πρώτη φάση οδήγησε στο φαινόμενο της προσκόλλησης σε έναν μόνον πάροχο (vendor lock-in) γεγονός που δεν επιτρέπει την αξιοποίηση με βέλτιστο τρόπο μια υποδομής που μπορεί να προσφέρεται από περισσότερους από έναν παρόχους (με πλεονεκτήματα σε τόπο, επάρκεια, κόστος). Η διαθεσιμότητα στις μέρες μας πολλών παρόχων δημόσιων υποδομών με την μορφή υπηρεσίας (Infrastructure as a Service) όπως η Amazon, Google, HP, IBM, RackSpace αλλά και ιδιωτικών παρόχων όπως η Openstack και η VMware παραμένει ανεκμετάλλευτη από τον μέσο χρήστη του υπολογιστικού νέφους. Στις μέχρι τώρα περισσότερες εφαρμογές που φιλοξενούνται στο υπολογιστικό νέφος οι χρήστες θέτουν τα διάφορα υπολογιστικά task σε έναν πάροχο υπολογιστικού νέφους θεωρώντας μόνο την αναμενόμενη ή προβλεπόμενη συμπεριφορά. Ως αποτέλεσμα αυτού του γεγονότος, οι περισσότεροι χρήστες δεν λαμβάνουν υπόψιν τους την περίπτωση δυναμικών υποδομών που προέρχονται από ταυτόχρονα πέραν του ενός παρόχων και που είναι χαρακτηριστικό των περισσότερων εφαρμογών μεγάλων δεδομένων. Με άλλα λόγια οι χρήστες αυτοί δεν εκμεταλλεύονται τα πλεονεκτήματα που προσφέρουν οι αρχιτεκτονικές πολλών παροχών υπολογιστικού νέφους [62].

Το υπολογιστικό νέφος επιτρέπει σε επιχειρήσεις και οργανισμούς που το χρησιμοποιούν να αυξάνουν ή να ελαττώνουν την χρήση πόρων με βάση την ζήτηση και σε σχέση με τις αλλαγές στον επεξεργαστικό φόρτο που παρουσιάζεται. Η εικονική υποδομή (Virtualization) είναι ένα

από τα βασικά χαρακτηριστικά του υπολογιστικού νέφους που παρέχει μηχανισμούς για αναπροσαρμογή και μεταφορά των πόρων χρησιμοποιώντας και ανάλογα εργαλεία παρακολούθησης. Με την βοήθεια όλων αυτών καταλήγουμε σε κατάλληλη αναπροσαρμογή της υποδομής. Έτσι η ανάγκη για αναπροσαρμογή των πόρων του υπολογιστικού νέφους είναι πολύ σημαντική ειδικά σε περιβάλλοντα πολλών παρόχων. Αυτές οι αναπροσαρμογές αναφέρονται κυρίως σε αλλαγές στο μέγεθος της εικονικής μηχανής (VM), μεταφορά εικονικών μηχανών (VM migrations) και ενεργοποίηση και απενεργοποίηση στην υποδομή εικονικών μηχανών. Για το κατάλληλο reconfiguration (αναπροσαρμογή) διάφοροι παράμετροι κόστους λαμβάνονται υπόψιν. Στην παρούσα Διατριβή η παράμετρος κόστους σχετική με τον χρόνο χρησιμοποιείται, που είναι βασισμένη στον χρόνο εκκίνησης της εικονικής μηχανής (VM startup time) και στον χρόνο ανάπτυξης της εκάστοτε εφαρμογής (Application component deployment time)[A1]. Η απόφαση της αναπροσαρμογής λαμβάνεται με την βοήθεια μιας σχετικής συνάρτησης (Utility Function) η οποία αποτελεί μέρος της Melodic πλατφόρμας που έχει αναφερθεί σε προηγούμενες παραγράφους.

5.2 Σχετικές Εργασίες

Η δυναμική αναπροσαρμογή των πόρων ανά ζήτηση σύμφωνα και με το επεξεργαστικό φορτίο είναι ένα από τα μεγαλύτερα πλεονεκτήματα του υπολογιστικού νέφους. Τέτοια αναπροσαρμογή περιλαμβάνει αλλαγή μεγέθους εικονικής μηχανής (VM resizing), εκκίνηση νέας εικονικής μηχανής (VM spawning) και μετακίνηση εικονικής μηχανής. Αυτό σημαίνει και δυναμική αναπροσαρμογή στα κόστη όμως. Υπάρχουσες εργασίες εξετάζουν μεταξύ άλλων το κόστος σχετικά με τον χρόνο εκκίνησης διαφόρων τύπων εικονικών μηχανών και το κόστος για την επανεγκατάσταση διαφόρων τμημάτων κατανεμημένης εφαρμογής υπολογιστικού νέφους σε μια νέα τοπολογία που διαχειρίζεται με καταλληλότερο τρόπο τις διακυμάνσεις των φορτίων και της κίνησης. Στην παρούσα παράγραφο παρουσιάζονται κάποια κόστη σχετικά με αναπροσαρμογές υποδομών σε υπολογιστικά νέφη.

Πιο συγκεκριμένα, στην εργασία των Mao και Humphrey [63] έχουν παρουσιαστεί αποτελέσματα σχετικά με τους χρόνους εκκίνησης εικονικών μηχανών σε τρεις παρόχους υπολογιστικού νέφους και συγκεκριμένα στους Amazon EC2, Microsoft Azure & Rackspace. Σε αυτή την μελέτη, αναφέρονται σχετικές μετρήσεις των χρόνων εκκίνησης αλλά και αναλύσεις που δείχνουν την συσχέτιση μεταξύ του χρόνου εκκίνησης και παραγόντων όπως το μέγεθος του λειτουργικού συστήματος της εικονικής μηχανής, τον τύπο της εικονικής μηχανής, τον αριθμό των ταυτόχρονων εκκινήσεων ίδιων εικονικών μηχανών αλλά και του χρόνου μέσα στην ημέρα που γίνεται η εκκίνηση. Το αρνητικό της συγκεκριμένης εργασίας είναι ότι οι μετρήσεις πραγματοποιήθηκαν αρκετά παλιά, γύρω στο 2012 και χρειάζεται μια ανανέωση με νέα σειρά μετρήσεων. Επίσης τα δεδομένα που προκύπτουν δεν αξιοποιούνται κάπως σε λειτουργίες αποφάσεων για αναπροσαρμογή πόρων. Σε μια παρόμοια εργασία, αυτή των Salfner [64], οι συγγραφείς αναλύουν τον χρόνο downtime κατά την διάρκεια μεταφοράς εικονικών μηχανών για την αναπροσαρμογή πόρων του υπολογιστικού νέφους. Τα αποτελέσματα ύστερα από διάφορες δοκιμές έδειξαν ότι ο χρόνος για την μεταφορά εικονικής μηχανής (VM migration) και του downtime υπηρεσιών επηρεάζονται κυρίως από την χρήση μνήμης που γίνεται στις συγκεκριμένες εικονικές μηχανές. Ο χρόνος downtime μεταβάλλεται αρκετά και επηρεάζει πολύ την ποιότητα των προσφερόμενων υπηρεσιών (quality of service). Δυστυχώς στην συγκεκριμένη εργασία και παρόλα τα σημαντικά ευρήματα που αναφέρονται, δεν παρουσιάζεται κάποια μέθοδος που να λαμβάνει υπόψιν αυτήν την χρονική παράμετρο κόστους στην διαδικασία επαναπροσδιορισμού των πόρων

της νεφουπολογιστικής υποδομής ώστε να μπορεί να ελαχιστοποιήσει το αρνητικό αποτέλεσμά της (πχ. καθυστερήσεις κτλ.).

Σε μια διαφορετική εργασία σε σχέση με τις ανωτέρω, οι Izzah, Yusoh και Tang [65] προτείνουν μια μέθοδο βασισμένη σε μια συνάρτηση penalty ως τμήμα ενός Γενετικού Αλγορίθμου για την ανάπτυξη και λειτουργία (deployment) κατανεμημένων εφαρμογών ως υπηρεσία νέφους (Software as a Service) φιλοξενούμενων σε διάφορες εικονικές μηχανές που βρίσκονται σε πέραν του ενός παρόχων. Ο κύριος στόχος είναι να ελαχιστοποιούνται κάθε φορά οι πόροι που χρησιμοποιούνται από την κατανεμημένη εφαρμογή και την ίδια στιγμή να διατηρείται ένα επαρκές επίπεδο ποιότητας υπηρεσίας (Quality of Service) σεβόμενοι τους τυχόν περιορισμούς που τίθενται. Βασισμένοι στα πειραματικά αποτελέσματα της εργασίας παρατηρούμε ωστόσο ότι αν και ο προτεινόμενος αλγόριθμος παράγει πάντα μια υλοποιήσιμη και οικονομική λύση χρειάζεται ένα μεγάλο χρονικό διάστημα για να ολοκληρωθεί η όποια αναπροσαρμογή της υποδομής. Επίσης, καμμιά αναφορά στην συγκεκριμένη εργασία δεν λαμβάνεται υπόψιν προκειμένου να συμπεριληφθεί η χρονική παράμετρος που αφορά την εκκίνηση εικονικών μηχανών, μια παράμετρος που είναι πολύ σημαντική στην όλη διαδικασία επαναπροσδιορισμού της υποδομής και διαθεσιμότητας των κατανεμημένων εφαρμογών υπολογιστικού νέφους.

Μέρος της έρευνας που έγινε στην παρούσα Διατριβή εν σχέση με την συνάρτηση penalty για τον επαναπροσδιορισμό της υποδομής του υπολογιστικού νέφους, αφορά στο να εξεταστούν περιπτώσεις όπου δεν υπάρχουν δεδομένα προηγούμενων λειτουργιών σε πόρους υπολογιστικού νέφους. Μια τέτοια περίπτωση θα μπορούσε να αναφερθεί ως η χρήση μιας εξειδικευμένης (custom) εικονικής μηχανής (με custom hardware χαρακτηριστικά) όπου δεν υπάρχουν προγενέστερες μετρήσεις. Τέτοιες λοιπόν περιπτώσεις κρίθηκε σκόπιμο να ερευνηθούν.

Πιο συγκεκριμένα, στην εργασία των Saniee [66] παρουσιάζεται μια απλή φόρμουλα για κατασκευή μια πολυώνυμης συνάρτησης όταν η συνάρτηση είναι μοναδική. Η περίπτωση της συνάρτησης με πολλές μεταβλητές παρουσιάζεται. Με άλλα λόγια, ένα ανάλογο του Langrange interpolation polynomial προκύπτει. Περιγράφεται η παρεμβολή (interpolation) μιας m μεταβλητών πολυωνυμικής συνάρτησης βαθμού n δεδομένου του συνδυασμού m+n δεδομένου του n. Παρολαυτά, η συγκεκριμένη εργασία χρειάζεται να επεξεργαστεί και να εξετάσει και άλλες παραμέτρους περαιτέρω προκειμένου να οδηγήσει σε επιτυχές αποτέλεσμα. Σε μια άλλη εργασία, οι Uyanik και Guler [67] αναλύουν τις 5 ανεξάρτητες μεταβλητές του standard model και εξετάζουν αν πετυχαίνουν επαρκή πρόγνωση και KPSS score [68] στην εξαρτημένη τελική μεταβλητή βασισμένη στην στατιστική θεωρία που αφορά το ANOVA [69]. Ο βασικός στόχος τους είναι να εξηγήσουν την πολλαπλή γραμμική παλινδρόμηση σε στάδια. Οι υποθέσεις που πρέπει να χρησιμοποιηθούν, απαραίτητες για την ανάλυση, εξετάζονται και η γραμμική παλινδρόμηση που εκτελείται βασίζεται σε αυτές τις υποθέσεις χρησιμοποιώντας αντίστοιχα δεδομένα. Οι τιμές που προκύπτουν και δείχνουν την ακρίβεια προγνώσεων του μοντέλου που εξετάζεται είναι R=0.932 και της αντίστοιχης εξαρτημένης μεταβλητής που δίνει την τελική τιμή πρόγνωσης είναι R2=0.87. Έτσι προκύπτει ότι το μοντέλο δίνει καλή πρόγνωση στην εξαρτημένη μεταβλητή, αλλά ωστόσο δεν είναι τόσο ακριβές όσο η μέθοδος του κλασσικού αλγορίθμου ελαχίστων τετραγώνων γραμμικής παλινδρόμησης πολλών μεταβλητών (OLS- Multiple Linear Regression) [69]. Ειδικότερα στην περίπτωση του αλγορίθμου OLS- Multiple Linear Regression θεωρούμε ότι έχουμε μια από τις καλύτερες τεχνικές για να αναλύουμε δεδομένα. Πολλές νεότερες μέθοδοι και αλγόριθμοι προγνώσεων έχουν την βάση τους στον OLS- Multiple Linear Regression όπως είναι η μέθοδος

ANOVA και γενικότερες γραμμικές μέθοδοι. Η παρούσα Διατριβή λόγω ακριβώς αυτών των πλεονεκτημάτων χρησιμοποιεί τον OLS- Multiple Linear Regression σαν βάση στην ανάπτυξη μεθόδων επαναπροσδιορισμού της υποδομής υπολογιστικού νέφους[A1].

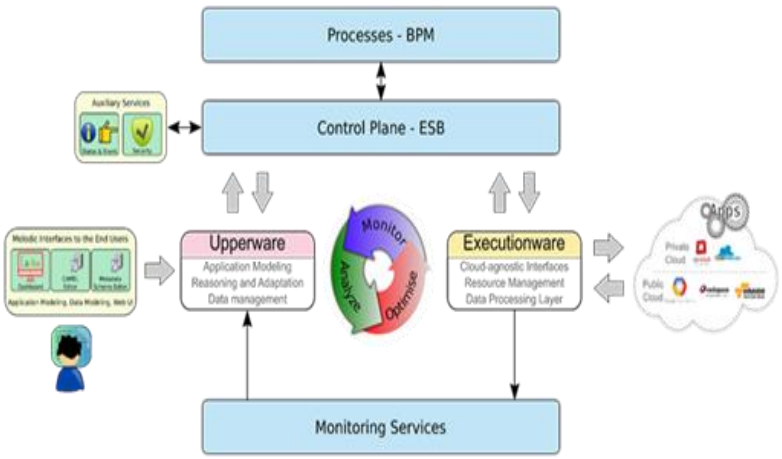
5.3 Ερευνητική Προσέγγιση

Στην παρούσα Διατριβή μια νέα και αρκετά καινοτόμος πλατφόρμα η οποία λέγεται MELODIC χρησιμοποιήθηκε σαν ένας αυτόματος (DevOp) μηχανισμός διαχείρισης υποδομής για την διαχείριση του κύκλου ζωής κατανεμημένων εφαρμογών που φιλοξενούνται σε πέραν του ενός παρόχου υπολογιστικού νέφους [70],[71]. Η βασική ιδέα λειτουργίας της συγκεκριμένης πλατφόρμας είναι η λειτουργία μαθηματικών μοντέλων σε πραγματικό χρόνο ροής δεδομένων, το επονομαζόμενο models@run.time. Με βάση αυτό, η αρχιτεκτονική της εφαρμογής, τα τμήματα που απαρτίζεται αλλά και τα δεδομένα που θα υποστούν επεξεργασία μπορούν όλα να περιγραφούν χρησιμοποιώντας μια κατάλληλη γλώσσα ανάπτυξης μοντέλων, την λεγόμενη Domain Specific Language (DSL). Πιο συγκεκριμένα, η περιγραφή της εφαρμογής σε αυτή την γλώσσα περιλαμβάνει τους στόχους για μια επιτυχή ανάπτυξή της στο υπολογιστικό νέφος (π.χ. μείωση του κόστους), τους περιορισμούς επίσης εν σχέση με αυτή την ανάπτυξη (π.χ. χρήση κέντρων δεδομένων που βρίσκονται σε συγκεκριμένες γεωγραφικές περιοχές), αλλά και την τρέχουσα κατάσταση της τοπολογίας της κατανεμημένης εφαρμογής με στόχο την βελτιστοποίηση της κατανομής για κάθε τμήμα της εφαρμογής στο μέλλον.

Η πλατφόρμα Melodic που είναι μια πλατφόρμα η οποία παρέχεται υπό την μορφή υπηρεσίας (platform as a service) απαρτίζεται από 3 βασικά μέρη:

- Τις διεπαφές (interfaces) του Melodic με τους τελικούς χρήστες
- Το τμήμα του Upperware
- Το τμήμα του Executionware

Ένα σχήμα που περιγράφει βασικά την αρχιτεκτονική της πλατφόρμας Melodic φαίνεται ακολούθως:



Σχήμα 11: Αρχιτεκτονική MELODIC[A1]

Το πρώτο κομμάτι που αφορά εργαλεία και διεπαφές χρησιμοποιείται για να περιγράψει και να υλοποιήσει εφαρμογές χρηστών και αντίστοιχα δεδομένα που αλληλοεπιδρούνε με την platform as a service (Melodic). Ο καθορισμός και σχεδιασμός των διαφόρων διεπαφών επιτυγχάνεται με την βοήθεια της χρήσης της γλώσσας CAMEL [72] η οποία παρέχει ένα πλούσιο σύνολο από μοντέλα σχετικά με σχεδιασμό, με ανάπτυξη εφαρμογών υπολογιστικού νέφους αλλά και με μοντέλα για επεξεργασία και καθορισμό δεδομένων. Αφού δημιουργηθούν τα αναγκαία μοντέλα (εφαρμογών και δεδομένων) με την βοήθεια των διαφόρων εργαλείων, μετά αυτά τροφοδοτούνται ως είσοδος στο κομμάτι του Upperware.

Το δεύτερο κομμάτι (Upperware) είναι υπεύθυνο για τον καθορισμό του βέλτιστου τρόπου ανάπτυξης εφαρμογών στο υπολογιστικό νέφος όπως επίσης και κατάλληλης φιλοξενίας δεδομένων σε δυναμικά αναπροσαρμοζόμενο περιβάλλον υπολογιστικού νέφους όσον αφορά τους πόρους πολλών παρόχων. Αυτό όπως ήδη αναφέρθηκε επιτυγχάνεται με την βοήθεια του μοντέλου από την CAMEL αλλά και θεωρώντας επίσης την τρέχουσα απόδοση του υπολογιστικού νέφους όσον αφορά το υπολογιστικό φόρτο, τα κόστη κτλ. Το να εξευρεθεί μια βέλτιστη τοπολογία για την ανάπτυξη κατανεμημένης εφαρμογής σε περιβάλλον πολλών παρόχων είναι αποτέλεσμα της αξιολόγησης μιας συνάρτησης ωφελείας (utility function). Σε αυτή την συνάρτηση ένα σημαντικό κομμάτι της αξιολόγησης αφορά τον χρόνο εκκίνησης των εικονικών μηχανών (Virtual Machines) καθώς και τον χρόνο ανάπτυξης (deployment) συγκεκριμένων τμημάτων της εφαρμογής που χρειάζεται να επαναπροσδιοριστούν (to be reconfigured). Ειδικότερα, μια εφαρμογή λογισμικού η οποία αναφέρεται ως Penalty Calculator αναπτύχθηκε στα πλαίσια της παρούσας Διατριβής για τον προσδιορισμό του χρονικού κόστους κάθε υποψήφιας λύσης για επαναπροσδιορισμό (reconfiguration) της υποδομής και που εξετάζεται σαν βέλτιστη από τα εργαλεία του Upperware Melodic platform.

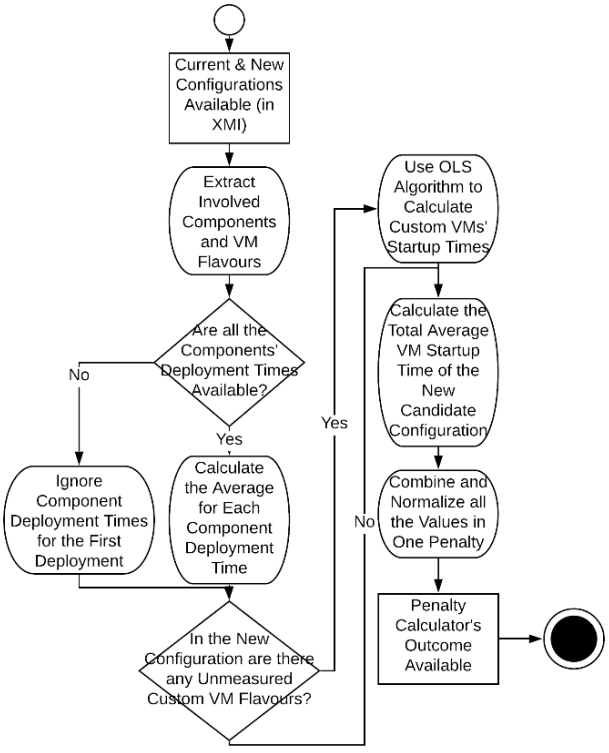
Το τρίτο κομμάτι της πλατφόρμας του Melodic περιλαμβάνει το Executionware το οποίο είναι υπεύθυνο να εκτελεί τις ενέργειες που αφορούν τις αναπτύξεις εφαρμογών σε παρόχους υπολογιστικού νέφους αλλά και αναπροσαρμογές (reconfigurations) της υποδομής τους με το να στέλνει κατάλληλα αιτήματα (requests) μέσω Application Programming Interfaces (APIs) δηλαδή ειδικές διεπαφές των παρόχων. Το Executionware γενικότερα είναι υπεύθυνο για να διαχειρίζεται και να οργανώνει διαφορετικού τύπου πόρους υπολογιστικού νέφους ενώ παράλληλα είναι υπεύθυνο για την παρακολούθηση των κατανεμημένων μεταξύ νεφουπολογιστικών παροχών εφαρμογών.

5.3.1 Η προσέγγιση του Penalty Calculator Αλγορίθμου

Ο αντικειμενικός στόχος του Penalty Calculator αλγορίθμου είναι να υπολογίζει μια κανονικοποιημένη τιμή συγκρίνοντας την τρέχουσα με την υποψήφια λύση επαναπροσδιορισμού της υποδομής (reconfiguration) και η οποία προέρχεται από το τμήμα που λέγεται Solver του Upperware. Με άλλα λόγια, ο Solver είναι ένα τμήμα του Upperware της πλατφόρμας του Melodic και είναι υπεύθυνο να γεννά μια σειρά υποψηφίων λύσεων επαναπροσδιορισμού της υποδομής του υπολογιστικού νέφους με βάση συγκεκριμένους περιορισμούς και στόχους βελτιστοποίησης πχ. ελάττωση του κόστους και αύξηση του χρόνου απόκρισης της υπηρεσίας, οι οποίοι αυξάνουν σαφώς την ποιότητα των παρεχόμενων υπηρεσιών μέσω υπολογιστικού νέφους. Ο Penalty Calculator επηρεάζει την απόφαση στο να γίνει ή όχι αποδεκτή μια νέα υποψήφια λύση εφαρμογής κατανεμημένης τοπολογίας μεταξύ παρόχων υπολογιστικού νέφους βασιζόμενη στην τιμή της συνάρτησης που θα προκύψει. Όσο μικρότερη είναι η τιμή της συνάρτησης penalty τόσο καλύτερη

θεωρείται η υποψήφια λύση καθώς υπονοεί ένα μικρότερο χρόνο για την υλοποίηση του προτεινόμενου επαναπροσδιορισμού της υποδομής (reconfiguration time).

Με άλλα λόγια, με βάση την πληροφορία που δίνει ο Utility Generator του Melodic στον Penalty Calculator Algorithm, ο τελευταίος αλγόριθμος χρησιμοποιείται για να συγκρίνει την παλιά και την νέα προτεινόμενη λύση δίδοντας μια τιμή penalty η οποία τελικά καθορίζει εάν η νέα λύση θα εφαρμοστεί ή όχι. Ο Αλγόριθμος του Penalty Calculator χρησιμοποιεί μετρημένους χρόνους εκκίνησης εικονικών μηχανών (Virtual Machines startup times) και μετρημένους χρόνους ανάπτυξης στην υποδομή εφαρμογών (average values of deployment times) προκειμένου να υπολογίσει το χρονικό κόστος για την αλλαγή από την τρέχουσα τοπολογία της κατανεμημένης εφαρμογής σε μια νέα τοπολογία. Εάν δεν υπάρχει ιστορικό από προηγούμενες μετρήσεις για χρόνους ανάπτυξης εφαρμογής ο αλγόριθμος λαμβάνει υπόψιν του μόνο τους χρόνους εκκίνησης των εικονικών μηχανών (Virtual Machines startup times). Σε περίπτωση που κομμάτι της προτεινόμενης λύσης περιλαμβάνει νέου τύπου εικονική μηχανή για να εγκατασταθεί στην υποδομή, χρησιμοποιείται ο αλγόριθμος του Ordinary Least Squares Regression (OLSR) [73] από τον Penalty Calculator ώστε να εκτιμηθεί ο χρόνος εκκίνησης της συγκεκριμένης εικονικής μηχανής χρησιμοποιώντας ως βάση τις μετρήσεις από σχετικού κοντινού τύπου εικονικές μηχανές των διαφόρων παρόχων υπολογιστικού νέφους. Η γενική ροή και βήματα του αλγορίθμου Penalty Calculator φαίνεται στο ακόλουθο σχήμα 12:



Σχήμα 12: Βήματα του Αλγορίθμου Penalty Calculator[A1]

Ειδικότερα, στον αλγόριθμο Ordinary Least Squares Regression (OLSR) [73] μια απλή εξηρτημένη μεταβλητή χρησιμοποιείται ώστε να μοντελοποιήσει τον χρόνο εκκίνησης της εικονικής μηχανής με βάση μια σειρά μετρήσεων και τιμών ως αναφορά. Οι ανεξάρτητες

μεταβλητές που χρησιμοποιούνται στον συγκεκριμένο αλγόριθμο (explanatory variables) αφορούν χρήση Μνήμης (RAM), αριθμό πυρήνων CPU και χρήση Σκληρού Δίσκου. Σε συνάρτηση δηλαδή των τιμών που υπάρχουν για αυτά τα μεγέθη για διαφόρων τύπων εικονικών μηχανών εξάγεται (γίνεται πρόγνωση) μια τιμή για τον χρόνο εκκίνησης. Η γενική εικόνα του μοντέλου OLS περιλαμβάνει την σχέση μεταξύ μιας συνεχούς εξαρτημένης μεταβλητής Y και κάποιων συνεχών ανεξάρτητων συνεχών μεταβλητών X χρησιμοποιώντας μια γραμμή μέγιστης προσαρμογής (best-fit line) ως ακολούθως:

$$Y = a + b_1 * X_1 + b_2 * X_2 + b_3 * X_3 \quad (3)$$

Στην παραπάνω εξίσωση, το α δείχνει την τιμή της μεταβλητής Y όταν όλες οι ανεξάρτητες μεταβλητές (X1,X2,X3..) είναι μηδέν. Κάθε παράμετρος b δείχνει την μέση αλλαγή στην μεταβλητή Y που είναι σχετιζόμενη με μια μοναδιαία αλλαγή στην μεταβλητή X, ενώ ελέγχουμε τις υπόλοιπες ανεξάρτητες μεταβλητές του μοντέλου.

Επιπλέον των παραπάνω επεξηγήσεων των παραμέτρων του OLS μοντέλου που χρησιμοποιείται στον Penalty Calculator Αλγόριθμο, το R-squared μέγεθος χρησιμοποιείται συχνά και δίνει μια μέτρηση της ποσοστιαίας διακύμανσης (variation) εν σχέση με την μεταβλητή που επεξηγείται από το μοντέλο. Ο ορισμός του R-squared δίνεται από την ακόλουθη εξίσωση:

$$R^2 = \frac{RSS\ after\ regression}{total\ RSS} \quad (4)$$

Το συγκεκριμένο μέγεθος δηλαδή δίνει το ποσοστό της απόκλισης της εξαρτημένης μεταβλητής που προκύπτει όταν προσθέτουμε μια ανεξάρτητη μεταβλητή στο μοντέλο.

Ολοκληρώνοντας την περιγραφή του αλγορίθμου, σαν τελευταίο βήμα πριν πάρουμε την τιμή του Penalty Calculator η μέθοδος min-max normalization εφαρμόζεται ως ακολούθως προκειμένου η απόκριση του Penalty Calculator τελικά να είναι στο διάστημα [0…1] και άρα κανονικοποιημένη:

$$\hat{T}^i_{reconfig} = \frac{T^i_{reconfig} - min(T_{reconfig})}{max(T_{reconfig}) - min(T_{reconfig})} \quad (5)$$

Στην παραπάνω εξίσωση, η Treconfig τιμή στον αριθμητή είναι το άθροισμα των μέσων τιμών χρόνου εκκίνησης των εικονικών μηχανών της νέας λύσης συν την μέση τιμή του χρόνου ανάπτυξης των νέων τμημάτων της εφαρμογής στην υποδομή.

5.4 Υλοποίηση

Όσον αφορά την υλοποίηση του Penalty Calculator αλγορίθμου, αυτός αποτελεί μέρος της πλατφόρμας του Melodic και συγκεκριμένα του τμήματος του Upperware και χρησιμοποιείται σαν βιβλιοθήκη (library) από τον Utility Generator. Ο Penalty Calculator όπως έχει ήδη διατυπωθεί, είναι ένα τμήμα που υπολογίζει μια απλή τιμή για κάθε πιθανή λύση η οποία δίδεται μετά στο utility function που εκφράζει όλους τους στόχους της εφαρμογής. Ο συγκεκριμένος αλγόριθμος είναι υλοποιημένος σε γλώσσα προγραμματισμού java.

Πιο συγκεκριμένα, ο αλγόριθμος του Penalty Calculator λαμβάνει σαν «εισόδους» στην συνάρτηση που τον υλοποιεί XMI(XML Metadata Interchange) αρχεία που περιγράφουν όλα τα στοιχεία που απαρτίζουν την υπάρχουσα και την νέα προτεινόμενη υποδομή

υπολογιστικού νέφους. Αυτά τα στοιχεία περιέχουν πληροφορία που αφορά το λειτουργικό σύστημα (Operating System), hardware πληροφορία που θα χρησιμοποιηθεί αλλά και τοπολογία των εικονικών πόρων που πρόκειται να φιλοξενήσουν τα στοιχεία της κατανεμημένης εφαρμογής. Ειδικότερα, ένα XMI αρχείο περιέχει πληροφορίες σχετικά με διάφορα στοιχεία που περιέχονται στην νέα υποδομή που προτείνεται όπως:

- Το id του στοιχείου της υποδομής (configuration element)
- Η τιμή (price) αυτού του στοιχείου
- Πληροφορία σχετική με το υπολογιστικό νέφος:
    - Cloud type
    - API
    - Credentials

Ένα παράδειγμα πληροφορίας για μια εικονική μηχανή (VM) όπως περιέχεται στο XMI αρχείο φαίνεται ακολούθως:

```
"id":"1a79a4d60de6718e8e5b326e338ae533/RegionOne/1",
"name": "Ubuntu 16.04 LTS AMD 64",
"operatingSystemType": "LINUX",
"operatingSystemFamily": "UBUNTU",
"operatingSystemArchitecture": "AMD64",
"operatingSystemVersion":"16.04"
```

Τμήμα Κώδικα 1

Επιπλέον, σχετικά με την τοποθεσία της κάθε εικονικής μηχανής στο υπολογιστικό νέφος καταγράφεται στο XMI αρχείο ως ακολούθως:

```
"id":"1a79a4d60de6718e8e5b326e338ae533/RegionOne",
"name": "RegionOne",
"providerId": "RegionOne",
"locationScope":"ZONE",
"city": "string",
"country": "string",
"latitude": 51.1657,
"longitude":10.4515
```

Τμήμα Κώδικα 2

Επίσης, πληροφορία σχετικά με τα hardware(υλικού) χαρακτηριστικά των στοιχείων των προτεινόμενων στην νέα λύση παρουσιάζονται στο XMI αρχείο με την ακόλουθη μορφή

```
"id":"1a79a4d60de6718e8e5b326e338ae533/RegionOne/1""name":
"t1.microcustom",

"providerId": "1",

"cores": 1,

"ram": 1024,

"disk":100,

"cardinality":2
```

Τμήμα Κώδικα 3

Τέλος, η τιμή cardinality παρέχεται για κάθε στοιχείο της προτεινόμενης λύσης και δείχνει πόσα ίδια στοιχεία χρειαζόμαστε πχ. στην νέα λύση. Στη συνέχεια, ο Penalty Calculator παρέχει κανονικοποιημένες τιμές μεταξύ 0 και 1 με χρήση της min-max μεθόδου κανονικοποίησης. Η τιμή 0 δείχνει το χαμηλότερο δυνατό penalty δηλαδή την πιο επιθυμητή λύση και η τιμή 1 δείχνει το πιο υψηλό penalty δηλαδή την λιγότερο επιθυμητή λύση. Προκειμένου να χρησιμοποιηθεί μια υψηλής απόδοσης κατανεμημένη μνήμη που θα επιταχύνει τους υπολογισμούς που κάνει ο αλγόριθμος του Penalty Calculator η λύση της Memcached έχει επιλεγεί. Η Memcached είναι ένας in memory τρόπος αποθήκευσης υπό την μορφή key-values pair και επιτρέπει να γίνεται πιο αποδοτική και γρήγορη χρήση της μνήμης ενός συστήματος [74].

Στην περίπτωσή μας, η Memcached χρησιμοποιείται για να διατηρεί και να ανακτάται όποτε χρειάζεται μέσω ειδικών API calls οι χρόνοι εκκίνησης των διαφόρων εικονικών μηχανών από τους διαφόρους παρόχους υπολογιστικού νέφους για τους διάφορους υπολογισμούς του Penalty Calculator. Η κατηγοριοποίηση των χρόνων εκκίνησης των διαφόρων εικονικών μηχανών γίνεται με την βοήθεια ανεξάρτητων μεταβλητών που αφορούν την μνήμη, την επεξεργαστική ισχύ, τον σκληρό δίσκο, τους τύπους των εικονικών μηχανών κτλ. Αυτό σημαίνει ότι υπάρχουν διάφοροι προκαθορισμένοι χρόνοι εκκίνησης εικονικών μηχανών που αντιστοιχούν σε συγκεκριμένους συνδυασμούς πόρων υπολογιστικού νέφους από διαφόρους παρόχους. Όσον αφορά τους χρόνους ανάπτυξης εφαρμογών στην υπολογιστική υποδομή αυτοί είναι σταθεροί και αποθηκεύονται σε μια βάση δεδομένων που είναι time series based. Για να επιτυγχάνεται μια γρήγορη ανάκτηση δεδομένων η λύση της Influx Db βάσης δεδομένων έχει επιλεγεί [75]. Η Influx DB είναι μια time series βάση δεδομένων καθώς τα δεδομένα που παράγονται από τους διάφορους sensors της υποδομής είναι σε άμεση συσχέτιση με τον χρόνο (time). Είναι open source με πολύ ενισχυμένο community και έχει αναπτυχθεί από την εταιρία InfluxData. Αποτελεί γενικότερα κατάλληλη επιλογή για δεδομένα τύπου χρονικά (time series) για περιπτώσεις operations monitoring, για application metrics, Internet of Things sensor data και για real-time analytics.

Βασικά πλεονεκτήματά της είναι τα ακόλουθα που χρειάζονται για την περίπτωση χρήσης της στο Penalty Calculator:

- Χρησιμοποιεί γρηγορότερη τεχνολογία αποθήκευσης δεδομένων σε βάση δεδομένων από τις σχεσιακές βάσεις. Στην περίπτωσή μας δεν χρησιμοποιείται πολύπλοκο indexing(δεικτοδότηση) πράγμα που θα καθυστερούσε τόσο την ανάγνωση όσο και την γραφή σε συγκεκριμένου τύπου time series δεδομένα. Πιο πολύπλοκο indexing χρησιμοποιείται στις σχεσιακές βάσεις.

- Λόγω του απλού indexing είναι πολύ γρήγορες οι εγγραφές επίσης στην βάση πράγμα που επιταχύνει την λειτουργία του Penalty Calculator σε σχέση με την χρήση κλασσικών σχεσιακών βάσεων.

5.5 Αξιολόγηση

5.5.1 Αξιολόγηση χρόνου εκκίνησης εικονικών μηχανών

Μια βασική παράμετρος όπως φάνηκε στην δημιουργία και υλοποίηση του Penalty Calculator είναι δεδομένα που υπάρχουν σχετικά με τους χρόνους εκκίνησης εικονικών μηχανών. Στην παρούσα Διατριβή έγιναν πλήθος μετρήσεις σχετικά με την συγκεκριμένη παράμετρο. Παρόμοια με την εργασία των Mao και Humphrey [63] που πηγαίνει πίσω στο 2012, στην παρούσα Διατριβή έχουμε επιχειρήσει και επιτύχει νέες μετρήσεις μεταξύ τριών ιδιωτικών και δημόσιων παρόχων υπολογιστικού νέφους και συγκεκριμένα:

- Ένα εσωτερικό πειραματικό περιβάλλον υπολογιστικού νέφους που έχει αναπτυχθεί στο πανεπιστήμιο ULM της Γερμανίας και αφορά εγκατάσταση Openstack
- Ένας δημόσιος πάροχος υπολογιστικού νέφους Amazon AWS
- Ένας δεύτερος δημόσιος πάροχος υπολογιστικού νέφους Google.

Για την διαδικασία της ανάλυσης, αξιολόγησης και μετρήσεων διάφορες περιοχές (regions) με κέντρα δεδομένων των δημόσιων παρόχων χρησιμοποιήθηκαν και διάφοροι τύποι εικονικών μηχανών ελέγχθηκαν και μετρήθηκαν σε ότι αφορά τον χρόνο εκκίνησης. Πιο συγκεκριμένα, 3500 μετρήσεις έλαβαν χώρα με διάφορους τύπους εικονικών μηχανών (VM flavours) φιλοξενούμενων σε διάφορα datacenters (κέντρα δεδομένων) σε όλο τον κόσμο ακολουθώντας κάθε αφορά αυξανόμενη λογική (ρυθμό) στην ταυτόχρονη εκκίνηση των εικονικών μηχανών.

Για όλες τις εικονικές μηχανές που χρησιμοποιήθηκαν στις μετρήσεις το λειτουργικό σύστημα που ήταν εγκατεστημένο σε κάθε μια από αυτές είναι το Linux Ubuntu. Προκειμένου να είναι δυνατή η εκκίνηση διαφόρων εικονικών μηχανών μεταξύ διαφόρων παρόχων υπολογιστικού νέφους έχουμε βασιστεί σε μια υλοποίηση ενός εργαλείου των Baur και Domaschka [76] που αφορά μια βιβλιοθήκη Java language για πολλούς παρόχους υπολογιστικού νέφους βασισμένη στην τεχνολογία Apache jClouds [77]. Σημαντικό στην μέτρηση του χρόνου εκκίνησης είναι ο σαφής ορισμός του. "Έτσι σαν χρόνος εκκίνησης ορίζεται ο χρόνος από την στιγμή που στέλνεται η εντολή για εκκίνηση μέχρι την πρώτη επιτυχή SSH σύνδεση στην εκκινούμενη εικονική μηχανή. Σε αυτό το χρονικό διάστημα περιλαμβάνονται οι ακόλουθες ενέργειες:

- Δημιουργία ενός SSH keypair
- Δημιουργία ενός security group
- Στην περίπτωση που χρησιμοποιείται το Openstack Cloud, ο χρόνος δημιουργίας μιας floating IP address.

Στο ιδιωτικό Openstack Cloud περιβάλλον, ο τύπος «flavour» ορίζει την επεξεργαστική ισχύ, την μνήμη και τον αποθηκευτικό χώρο μιας εικονικής μηχανής. Στην διαδικασία των πειραμάτων χρησιμοποιήθηκαν για την περίπτωση του ιδιωτικού cloud περιβάλλοντος οι ακόλουθοι τύποι ή flavours των εικονικών μηχανών:

Openstack Flavours	VCPUs	RAM (in MB)
m1.small2	2	1024
m1.medium2	4	4096
m1.large2	8	8192
m1.xlarge	8	16384

Πίνακας 3: Openstack flavours [A1]

Στην περίπτωση της υποδομής Amazon AWS, οι εικονικές μηχανές τύπου T2 είναι χαμηλού κόστους, γενικού σκοπού και μπορούν να προσφέρουν μια βασική απόδοση επεξεργαστικής ισχύος και να διαχειρίζονται επίσης εκτινάξεις (burst-peak) φορτίων όταν είναι αναγκαίο. Στις μετρήσεις τα είδη T2 εικονικών μηχανών που χρησιμοποιήθηκαν φαίνονται στον ακόλουθο πίνακα:

EC2 Flavours	VCPUs	RAM (in MB)
t2.micro	1	1024
t2.small	1	2048
t2.medium	2	4096
t2.large	2	8192
t2.xlarge	4	16384
t2.2xlarge	8	32768

Πίνακας 4: Amazon EC2 flavours [A1]

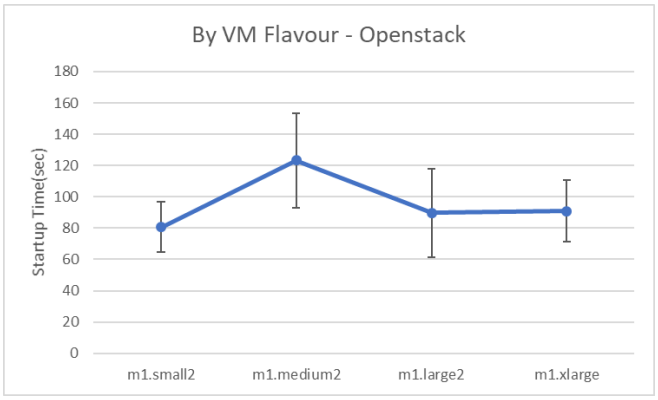
Τέλος, για την περίπτωση της Google Cloud υποδομής οι πρώτης γενεάς τύπου N1 εικονικές μηχανές χρησιμοποιήθηκαν στις μετρήσεις όπως φαίνεται στον ακόλουθο πίνακα:

Google Cloud	VCPUs	RAM (in MB)
n1-standard-1	1	3840
n1-standard-2	2	7680
n1-standard-4	4	15360
n1-standard-8	8	30720

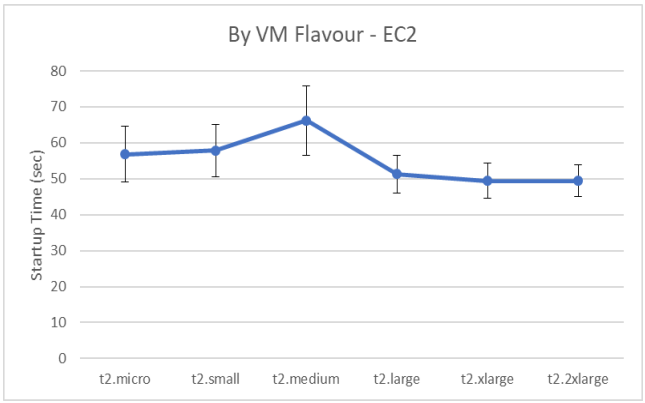
Πίνακας 5: Google Cloud Machine Types [A1]

Όπως αναφέρθηκε είναι σημαντικό να οριστεί σωστά ο χρόνος εκκίνησης εικονικών μηχανών ανεξαρτήτως παρόχου υπολογιστικού νέφους. Γενικά, οι διάφοροι πάροχοι θέλοντας να περιγράψουν τον κύκλο ζωής των εικονικών μηχανών παρέχουν μια λειτουργικότητα που λέγεται “status tags” και στην ουσία περιγράφει την κατάσταση στην οποία βρίσκεται η κάθε εικονική μηχανή (σε λειτουργία, υπό εκκίνηση, σταματημένη κτλ.). Ωστόσο οι διάφοροι πάροχοι υπολογιστικού νέφους χρησιμοποιούν διαφορετικές καταστάσεις- ορισμούς για τις εικονικές μηχανές που διαθέτουν γεγονός που προκαλεί παρερμηνείες και δυσκολίες. Για να αποφευχθεί αυτή η κατάσταση, υιοθετήθηκε η στρατηγική να αγνοηθούν τα “status tags” και να χρησιμοποιείται ο ορισμός του χρόνου εκκίνησης σαν ο χρόνος από την στιγμή που στέλνεται ένα αίτημα (request) για εκκίνηση μιας εικονικής μηχανής μέχρι την στιγμή που μπορεί κάποιος/ένα σύστημα να συνδεθεί απομακρυσμένα (first successful SSH login time).

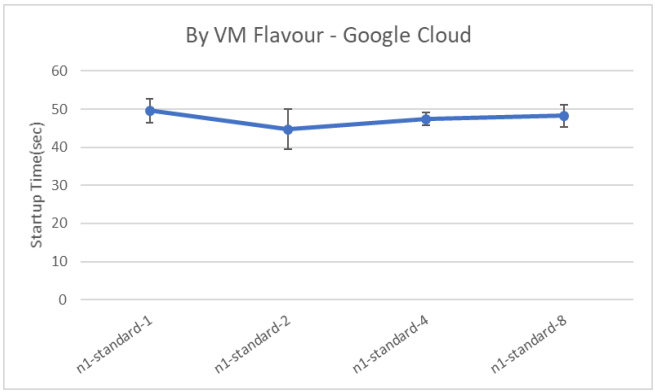
Όσον αφορά τις μετρήσεις έχουν κατηγοριοποιηθεί σε τρεις κατηγορίες. Η πρώτη αφορά μια ομάδα μετρήσεων σε τρεις παρόχους υπηρεσιών υπολογιστικού νέφους εστιάζοντας στην σχέση μεταξύ του χρόνου εκκίνησης και του τύπου (flavour) της εικονικής μηχανής που χρησιμοποιείται. Για κάθε ομάδα μετρήσεων και για κάθε τύπο εικονικής μηχανής η ανάπτυξη ενός έως 20 instances ίδιου τύπου εικονικών μηχανών είτε σειριακά είτε παράλληλα παρουσιάζονται ακολούθως. Το όριο των 20 instances τίθεται από τους περιορισμούς του Application Programming Interface (API) που παρέχουν για δια-λειτουργικότητα οι διάφοροι πάροχοι υπολογιστικού νέφους:



Σχήμα 13: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών ιδιωτικού παρόχου Openstack ανάλογα του τύπου VM[A1]



Σχήμα 14: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Amazon EC2 ανάλογα του τύπου VM[A1]



Σχήμα 15: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Google ανάλογα του τύπου VM[A1]

Σε όλες τις παραπάνω απεικονισμένες τιμές εφαρμόζουμε την κανονική διακύμανση (standard deviation). Σύμφωνα με τα αποτελέσματα των παραπάνω μετρήσεων (σχήματα 13-14-15) είναι εμφανές ότι ο χρόνος εκκίνησης εικονικών μηχανών (virtual machines) για το ιδιωτικό πάροχο υπηρεσιών υπολογιστικού νέφους (Openstack ULM Germany) είναι μεγαλύτερος από αυτόν στους δημόσιους παρόχους (Amazon, Google).

Πιο συγκεκριμένα, οι εικονικές μηχανές του ιδιωτικού παρόχου Openstack παρέχονται με ένα χρόνο εκκίνησης που κυμαίνεται από 81 έως 123 δευτερόλεπτα και εξαρτάται από τον τύπο

«flavour» της εικονικής μηχανής κάθε φορά. Για τους υπόλοιπους παρόχους, οι χρόνοι εκκίνησης για τις εικονικές μηχανές ποικίλουν από 49 έως 66 δευτερόλεπτα για την Amazon και 45 έως 50 δευτερόλεπτα για την Google. Οι μετρημένες μηχανές από την Openstack παρουσιάζουν μεγαλύτερους χρόνους προκειμένου να δεσμεύσουν και να ρυθμίσουν τους πόρους τους από ότι γίνεται στις μηχανές των δημοσίων παρόχων υπολογιστικού νέφους. Με βάση επίσης τις δοκιμές και μετρήσεις που έγιναν, βρέθηκε ότι η κανονική διακύμανση στα νούμερα των χρόνων εκκίνησης του παρόχου Openstack είναι σημαντικά μεγαλύτερη από τους δημόσιους παρόχους. Αυτό δείχνει ότι ο ιδιωτικός πάροχος παρουσιάζει ασταθή γενικά υποδομή τόσο όσον αφορά τους πόρους όσο και τους μηχανισμούς των διαφόρων task που τρέχουν.

Η δεύτερη σειρά δοκιμών και μετρήσεων αφορά την διασπορά των κέντρων δεδομένων (data centers) σε διαφορετικές περιοχές ανά τον κόσμο για τους δύο μεγάλους δημόσιους παρόχους Amazon και Google και πως αυτή η διασπορά επιδρά στην μεταβολή του χρόνου εκκίνησης των εικονικών μηχανών. Η περίπτωση του ιδιωτικού Openstack αφορά μόνο ένα κέντρο δεδομένων στην περιοχή Ulm της Γερμανίας. Γενικά οι δημόσιοι πάροχοι υπηρεσιών υπολογιστικού νέφους προκειμένου να διασφαλίσουν την υψηλή διαθεσιμότητα των υπηρεσιών τους φιλοξενούν τους υπολογιστικούς πόρους τους σε διάφορες περιοχές παγκοσμίως. Με αυτό τον τρόπο επιτυγχάνουν ελαχιστοποίηση του κινδύνου να χαθεί τελείως η δυνατότητα παροχής αυτών των υπηρεσιών (mitigation of risk with high availability). Αυτές οι περιοχές χωρίζονται σε Regions και σε Availability Zones. Ο ορισμός της περιοχής Region περιλαμβάνει μια ξεχωριστή γεωγραφική περιοχή η οποία περιλαμβάνει διάφορες απομονωμένες περιοχές γνωστές ως Availability Zones που αντιστοιχούν σε φυσικές οντότητες πχ κέντρα δεδομένων (data centers).

Ο ακόλουθος πίνακας παρουσιάζει τις περιοχές των κέντρων δεδομένων που χρησιμοποιήθηκαν στις μετρήσεις μας για την Amazon και την Google:

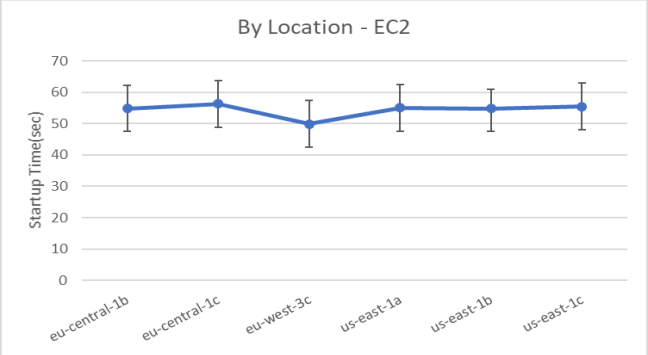
Availability Zones	Locations
AWS EC2	
eu-central-1b	Frankfurt (Germany)
eu-central-1c	Frankfurt (Germany)
eu-west-3c	Paris (France)
us-east-1a	N. Virginia (US)
us-east-1c	N. Virginia (US)
Google Cloud	
europe-west1-b	St. Ghislain (Belgium)
europe-west2-b	London (UK)

Πίνακας 6: Availability Zones και Locations[A1]

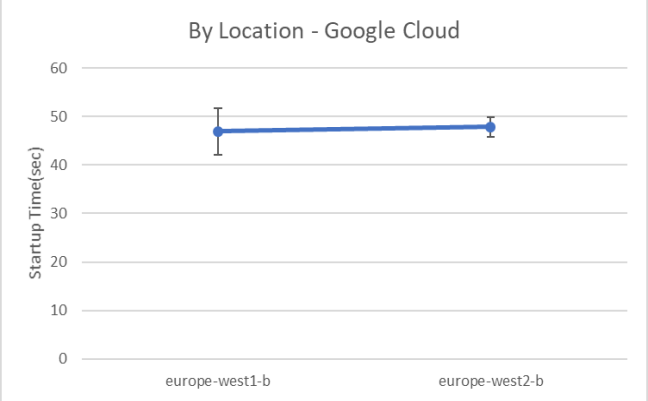
Στα ακόλουθα σχήματα 16 και 17 παρουσιάζονται τα αποτελέσματα των μετρήσεων των χρόνων εκκίνησης ως προς περιοχές ανά πάροχο. Και για τους δύο δημόσιους παρόχους, δεν διαπιστώνεται σημαντική μεταβολή των χρόνων εκκίνησης των διαφόρων εικονικών μηχανών καθώς ο αριθμός των αιτημάτων για παροχή νέων σε εκκίνηση εικονικών μηχανών μεταβάλλεται από Region σε Region και από Availability Zone σε Availability Zone.

Ειδικότερα, για την Amazon ο μέσος χρόνος εκκίνησης ήταν 55 δευτερόλεπτα ακόμα και για εικονικές μηχανές που εκκινούσαν στις Ηνωμένες Πολιτείες της Αμερικής. Μια μικρή βελτίωση στον χρόνο κατά 5 δευτερόλεπτα παρατηρήθηκε σε όλες τις εικονικές μηχανές που εκκίνησαν στο κέντρο δεδομένων του Παρισιού, ενώ η κανονική διακύμανση όλων αυτών των μετρήσεων δεν ξεπέρασε τα 15 δευτερόλεπτα παρουσιάζοντας μια σχετικά σταθερή

συμπεριφορά. Σχετικά με τον πάροχο της Google, οι εικονικές μηχανές που δοκιμάστηκαν χρειάστηκαν περίπου κατά μέσο όρο 48 δευτερόλεπτα για εκκίνηση με 1 δευτερόλεπτο μεταβλητότητα μεταξύ των περιοχών κέντρων δεδομένων (Availability Zones): West Europe 1 & West Europe 2. Στον συγκεκριμένο πάροχο η κανονική διακύμανση (standard deviation) ήταν σημαντικά χαμηλότερη από την προηγούμενη περίπτωση της Amazon καθώς δεν ξεπέρασε τα 9 δευτερόλεπτα ένα γεγονός που μαρτυρά ακόμα πιο σταθερή συμπεριφορά για τον πάροχο της Google.

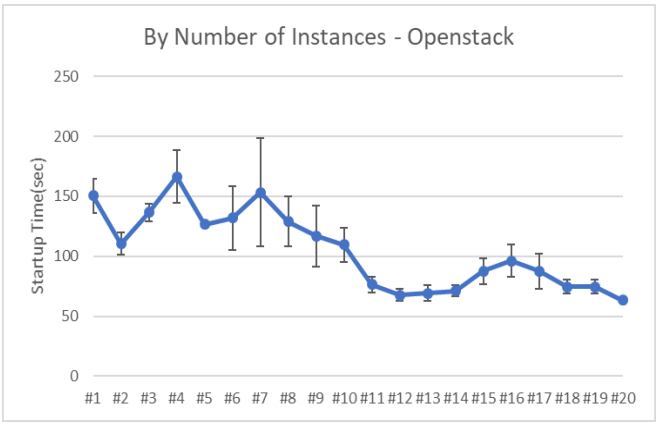


Σχήμα 16: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Amazon ανάλογα του Availability Zone[A1]

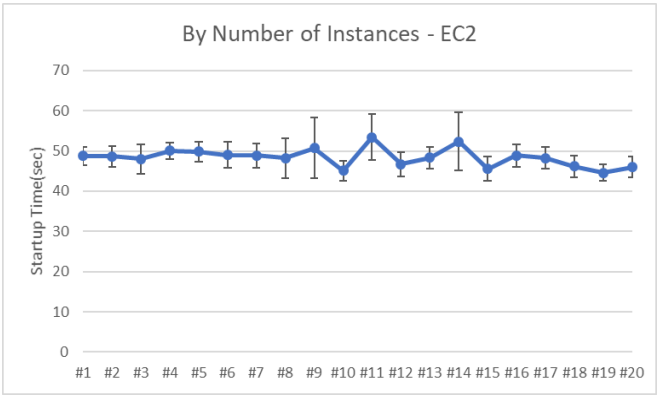


Σχήμα 17: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Google ανάλογα του Availability Zone[A1]

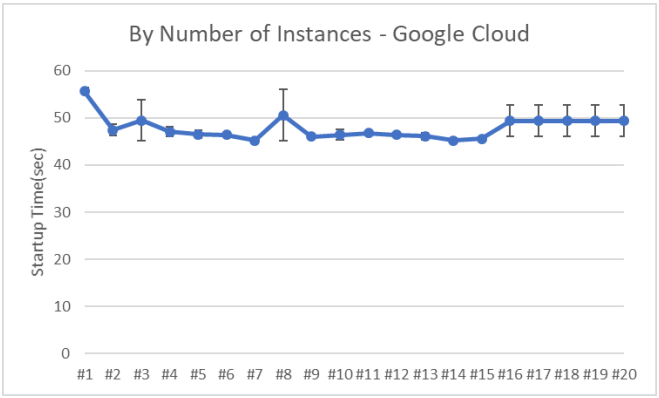
Στην Τρίτη κατηγορία μετρήσεων, συμπεριλάβαμε την επίδραση που έχει στον χρόνο εκκίνησης εικονικών μηχανών ο αριθμός των εικονικών μηχανών που εκκινούν ταυτόχρονα (Virtual Machines requested simultaneously) φτάνοντας τον αριθμό των 20. Ο συγκεκριμένος αριθμός αποτελεί το άνω όριο που τίθεται από τους δημόσιους παρόχους. Τα αποτελέσματα των μετρήσεων παρουσιάζονται στα σχήματα ακολούθως 18-20:



Σχήμα 18: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών ιδιωτικού παρόχου Openstack σε συνάρτηση των ταυτόχρονων εκκινούμενων VM[A1]



Σχήμα 19: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Amazon EC2 σε συνάρτηση των ταυτόχρονων εκκινούμενων VM[A1]



Σχήμα 20: Μέση τιμή χρόνου εκκίνησης εικονικών μηχανών δημοσίου παρόχου Google σε συνάρτηση των ταυτόχρονων εκκινούμενων VM[A1]

Με βάση τα ανωτέρω αποτελέσματα, οι εικονικές μηχανές του Openstack παρουσιάζουν μια μεγαλύτερη διακύμανση όπως αναμενόταν εξάλλου, και η οποία ελαττώνεται όσο ο αριθμός των ζητούμενων εικονικών μηχανών αυξάνει. Επιπλέον, παρατηρήσαμε σημαντικές διακυμάνσεις μεταξύ ίδιου αριθμού instances εικονικών μηχανών που έφτασε στην τιμή των 89 δευτερολέπτων όταν εκκινούν 7 εικονικές μηχανές παράλληλα. Αυτό το γεγονός αποκαλύπτει μια σημαντικά μη σταθερή συμπεριφορά για την περίπτωση του ιδιωτικού παρόχου υπολογιστικού νέφους.

Στην περίπτωση ωστόσο των δημοσίων παρόχων παρατηρούμε μια πιο σταθερή συμπεριφορά με ελάχιστες διακυμάνσεις στις τιμές εκκίνησης εικονικών μηχανών. Ειδικότερα, παρατηρήσαμε μέσες τιμές εκκίνησης μεταξύ 45 δευτερολέπτων (για 10 instances) και 53 δευτερολέπτων (για 11 instances) για την Amazon και 45 δευτερολέπτων (για 7 instances) και 56 δευτερολέπτων (για 1 instance) για την Google καθώς διαφορετικά αιτήματα για εκκίνηση VM υποβλήθηκαν. Αυτή η συμπεριφορά των δημοσίων παρόχων είναι αναμενόμενη καθώς λόγω του μεγέθους αυτών των παρόχων πάντα υπάρχουν διαθέσιμοι πόροι που όταν ζητηθούν ad-hoc να μπορούν να διατεθούν άμεσα. Επίσης, διαπιστώσαμε παρόμοια διακύμανση στις μετρήσεις για τις τιμές εκκίνησης εικονικών μηχανών όταν ζητείται ίδιος αριθμός instances τόσο στο υπολογιστικό νέφος της Google όσο και στην Amazon. Συγκεκριμένα στην πρώτη περίπτωση η κανονική διακύμανση (standard deviation) έφτασε τα 10 δευτερόλεπτα ενώ στην δεύτερη περίπτωση της Amazon τα 15 δευτερόλεπτα. Τέλος, μια τελευταία παρατήρηση αφορά την σύγκριση με την προηγούμενη εργασία των Mao & Humphrey[63] όπου το 2012 είχαν μετρήσει μια μέση τιμή εκκίνησης εικονικών μηχανών στην Amazon γύρω στα 100 δευτερόλεπτα ενώ στις τωρινές μετρήσεις παρατηρούμε μια μείωση της τάξης του 48%. Αυτό το γεγονός αποδεικνύει τις σημαντικές επενδύσεις και την σημασία που έχουν δώσει οι δημόσιοι πάροχοι υπολογιστικού νέφους πλέον στις υπολογιστικές τους δομές τα τελευταία χρόνια.

5.5.2 Αξιολόγηση της λειτουργίας του Penalty Calculator

Με βάση το παράδειγμα που αναπτύχθηκε στο σχήμα 9 και το κεφάλαιο 4.4, δείχνουμε στην συνέχεια αυτού του κεφαλαίου πως ο Penalty Calculator μέσα στο Melodic platform μπορεί να χρησιμοποιηθεί στην συγκεκριμένη εφαρμογή εξομοίωσης κίνησης οχημάτων σε μια Ευρωπαϊκή πόλη. Η αρχική ανάπτυξη της εφαρμογής, όπως έχει ήδη παρουσιαστεί, αποτελείται από 5 βασικά στοιχεία-τμήματα όπως φαίνονται στο σχήμα 9:

- Traffic evaluation component (1 στοιχείο)
- Simulation manager (1 στοιχείο)
- Simulation workers (3 στοιχεία)

Ο traffic evaluator είναι υπεύθυνος για την ανάλυση της κίνησης και στέλνει στον simulation manager πληροφορία για την στιγμή που είναι αναγκαία να εκτελεστεί μια εξομοίωση. Από την άλλη, οι Simulation workers είναι τμήματα αναπτυγμένα σε python γλώσσα υπεύθυνα για την εκτέλεση εργασιών όπως αξιολόγηση ρυθμίσεων για εξομοίωση κίνησης όπως αυτές οι εργασίες έχουν ανατεθεί από τον simulation manager τρέχοντας εξομοιώσεις και με την χρήση του λογισμικού Traffic Simulation Framework (C#).

Για την αξιολόγηση του Penalty Calculator δύο configurations χρησιμοποιήθηκαν ως είσοδοι: η τρέχουσα κατάσταση και η προτεινόμενη (μελλοντική) κατάσταση. Οι δύο είσοδοι δίνονται με την μορφή XMI (XML Metadata Interchange) αρχείων εισόδου από το Melodic platform στην συνάρτηση του Penalty calculator. Η προτεινόμενη νέα κατάσταση υποδομής περιλαμβάνει στις δοκιμές μας τόσο νέες προκαθορισμένου τύπου εικονικές μηχανές από τους 2 δημόσιους φορείς (VM flavours) όσο και κάποιους τύπους εικονικών μηχανών μη προκαθορισμένους (custom VMs) όπως πχ. t1.microcustom VM. Ειδικότερα οι προκαθορισμένοι τύποι εικονικών μηχανών στο συγκεκριμένο παράδειγμα προέρχονται από 2 παρόχους υπολογιστικού νέφους: Amazon EC2 & Openstack. Στον ακόλουθο πίνακα 7 παρουσιάζουμε τις τιμές εκκίνησης προκαθορισμένων εικονικών μηχανών για τους δύο

παρόχους όπως έχουν αποθηκευτεί στην Memcached για γρήγορη ανάκτηση και επεξεργασία. Αξίζει να σημειωθεί ότι το πρόθεμα μηχανής t2 αναφέρεται σε τύπο(flavour) της Amazon ενώ το m1 αναφέρεται σε τύπο(flavour) μηχανής του Openstack.

VM flavour	VM startup time(sec)
t2.nano	*50*
t2.small	*100*
t2.medium	*110*
t2.large	*120*
t1.xlarge	*130*
t1.2xlarge	*130*
m1.tiny	*55*
m1.small	*79*
m1.medium	*88*
m1.large	*132*
m1.xlarge	*140*

Πίνακας 7: Προκαθορισμένοι τύποι VMs (flavours) και αντίστοιχοι χρόνοι εκκίνησης (startup times)[A1]

Με βάση τα ανωτέρω είδη εικονικών μηχανών, παρουσιάζεται στον ακόλουθο πίνακα 8 η αντιστοίχιση των ανωτέρω χρόνων εκκίνησης με τα χαρακτηριστικά των τύπων των εικονικών μηχανών:

VM startup time (sec)	*Number of cores for vCPU*	*RAM (GB)*	*Disk (GB)*
50	*1*	*0.6*	*0.5*
100	*1*	*1.7*	*160*
110	*4*	*7.5*	*850*
120	*8*	*15*	*1690*
130	*7*	*17.1*	*420*
130	*5*	*2*	*350*
55	*1*	*0.5*	*0.5*
79	*1*	*2.048*	*10*
88	*2*	*4.096*	*10*
132	*4*	*8.192*	*20*
140	*8*	*16.384*	*40*

Πίνακας 8: Αντιστοίχιση χρόνων εκκίνησης εικονικών μηχανών και στοιχείων πόρων[A1]

Οι τιμές του πίνακα 8 χρησιμοποιούνται σαν ιστορικά δεδομένα για να εκπαιδεύσουν τον αλγόριθμο Ordinary Least Squares Regression (OLSR) ο οποίος όπως έχει αναλυθεί, αποτελεί τμήμα της συνάρτησης του Penalty Calculator και βοηθάει στο να προβλεφθούν οι χρόνοι εκκίνησης μη προκαθορισμένων τύπων εικονικών μηχανών. Στην περίπτωσή μας το νέο μη προκαθορισμένο εικονικό μηχάνημα (custom VM) που προτείνεται μεταξύ άλλων στην νέα λύση, φαίνεται στο ακόλουθο σχήμα XMI που περιλαμβάνει τα (hardware) χαρακτηριστικά υλικού του:

```
},
"hardware": {
    "id": "1a79a4d60de6718e8e5b326e338ae533/RegionOne/1",
    "name": "t1.microcustom",
    "providerId": "1",
    "cores": 1,
    "ram": 1,
    "disk": 100,
    "location": {
        "id": "1a79a4d60de6718e8e5b326e338ae533/RegionOne",
        "name": "RegionOne",
        "providerId": "RegionOne",
        "locationScope": "ZONE",
        "isAssignable": true,
        "geoLocation": {
            "city": "string",
            "country": "string",
            "latitude": 0,
            "longitude": 0
        }
    }
},
```


Σχήμα 21: XMI της νέας προτεινόμενης custom εικονικής μηχανής

Έτσι με βάση τον πίνακα 8 και τα στοιχεία του σχήματος 21 όπου χρησιμοποιούμε την t1.microcustom εικονική μηχανή με στοιχεία CPUcores=1 , RAM=1, Disk=100 και με βάση την εκπαίδευση του OLSR αλγορίθμου, η τιμή εκκίνησης που προκύπτει είναι 65 δευτερόλεπτα.

Επίσης στον Penalty Calculator λαμβάνονται υπόψιν και οι χρόνοι εγκατάστασης των διαφόρων εφαρμογών(Components deployment times) . Στον πίνακα 8 παρουσιάζονται οι μετρημένοι χρόνοι για τα διάφορα στοιχεία (Components) και οι οποίοι αποθηκεύονται στην Influx DB όπως έχουμε ήδη αναφέρει. Όλα τα στοιχεία των εφαρμογών εγκαταστάθηκαν (deployed) με την βοήθεια της πλατφόρμας Melodic και οι αντίστοιχοι χρόνοι για ανάπτυξη τους είναι όπως αναφέρονται στον ακόλουθο πίνακα 9:

Component Name	**Deployment Time (seconds)**
Traffic evaluation component	319.599
Simulation manager component	399.625
Simulation Worker 1	272.550
Simulation Worker 2	254.526
Simulation Worker 3	265.752

Πίνακας 9: Χρόνοι ανάπτυξης των στοιχείων[A1]

Χρησιμοποιώντας σαν δεδομένα τους χρόνους εκκίνησης του Πίνακα 7 & 8 και τους χρόνους ανάπτυξης των στοιχείων του Πίνακα 9 υπολογίζονται οι παράμετροι της εξίσωσης (3) ως ακολούθως:

A=122.267

B1= 0.260

B2=-0.024

B3=-0.006

With a r-Squared parameter: 0.981

Σχήμα 22: Παράμετροι της εξίσωσης (1) [A1]

Βασισμένοι σε αυτά τα αποτελέσματα, ο αλγόριθμος Ordinary Least Squares Regression (OLSR) φαίνεται αρκετά ακριβής και όπως βλέπουμε βασίζεται στις ανεξάρτητες μεταβλητές κατά ένα ποσοστό 98,15% και κατά 2.6% στην σταθερά α. Αυτός ο αλγόριθμος όπως ειπώθηκε χρησιμοποιείται για να δώσει μια ακριβή πρόβλεψη για οποιοδήποτε τύπου μη-προκαθορισμένη εικονική μηχανή (custom VM) που θα μπορούσε να αποτελεί κομμάτι μιας νέας υποδομής υπολογιστικού νέφους. Τελευταίο και σημαντικό, κατά την διάρκεια των μετρήσεων και πειραμάτων της νέας λύσης για την υποδομή, η Min-Max κανονικοποιημένη τιμή υπολογίστηκε που αφορά την κανονικοποιημένη μέση τιμή του χρόνου εκκίνησης εικονικής μηχανής (VM startup time) και του χρόνου ανάπτυξης κατανεμημένης εφαρμογής (Component Deployment Time) και προκύπτει ίση με 0.415. Με βάση αυτή την τιμή καλείται μετά ο Utility Generator της πλατφόρμας Melodic για να αποφασίσει την καταλληλότερη λύση από όσες έχουν προταθεί.

5.6 Συμπεράσματα

Στην συγκεκριμένη Διατριβή εστιάσαμε σε μια από τις πιο κρίσιμες αποφάσεις όσον αφορά την βέλτιστη λήψη αποφάσεων για την αναπροσαρμογή υποδομής υπολογιστικού νέφους σε ένα δυναμικό περιβάλλον πολλών παρόχων που αποτελεί και το καινοτόμο σημείο. Ειδικότερα, παρουσιάζεται ένα σύστημα που υπολογίζει penalty συναρτήσεις με παράμετρο κόστους τον χρόνο. Συγκρίνει και προτείνει νέες λύσεις σε σχέση με τωρινές που όμως δεν μπορούν πχ. να διαχειριστούν έντονα εισερχόμενα σε κατανεμημένες εφαρμογές φορτία κίνησης και δημιουργούν κορυφές φορτίου (spikes). Ο αλγόριθμος που έχει φτιαχτεί λαμβάνει υπόψιν του τόσο χρόνους εκκίνησης εικονικών μηχανών αλλά και χρόνους ανάπτυξης κατανεμημένων εφαρμογών προκειμένου τελικά να υπολογιστεί μια κανονικοποιημένη τιμή απόφασης. Τέλος, σημαντικό σε όλα αυτά αποτελούν και οι μέθοδοι και αριθμός μετρήσεων που έλαβαν χώρα και οι οποίοι συνεισφέρουν στην διαδικασία μάθησης (train process) των αλγορίθμων που αναπτύχθηκαν στα πλαίσια του Penalty Calculator για μη προκαθορισμένους χρόνους εκκίνησης (custom VMs).

6 Ακριβής πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος που φιλοξενεί κατανεμημένη εφαρμογή

6.1 Εισαγωγή

Η ύπαρξη του υπολογιστικού νέφους αλλά και άλλες κρίσιμες τεχνολογίες έχουν παρουσιάσει ως συνηθισμένη τακτική σε πολλές περιπτώσεις την κεντρική επεξεργασία δεδομένων που παράγονται στο «άκρο» του υπολογιστικού νέφους. Στις μέρες μας, υπάρχει μια μεγάλη τάση για μαζική παραγωγή δεδομένων από μεγάλο αριθμό συνεχώς αυξανόμενων συσκευών ευρισκόμενες στο «άκρο» ή “edge” του υπολογιστικού νέφους όπως λέγεται. Τέτοιες συσκευές μπορεί να είναι οι ακόλουθες:

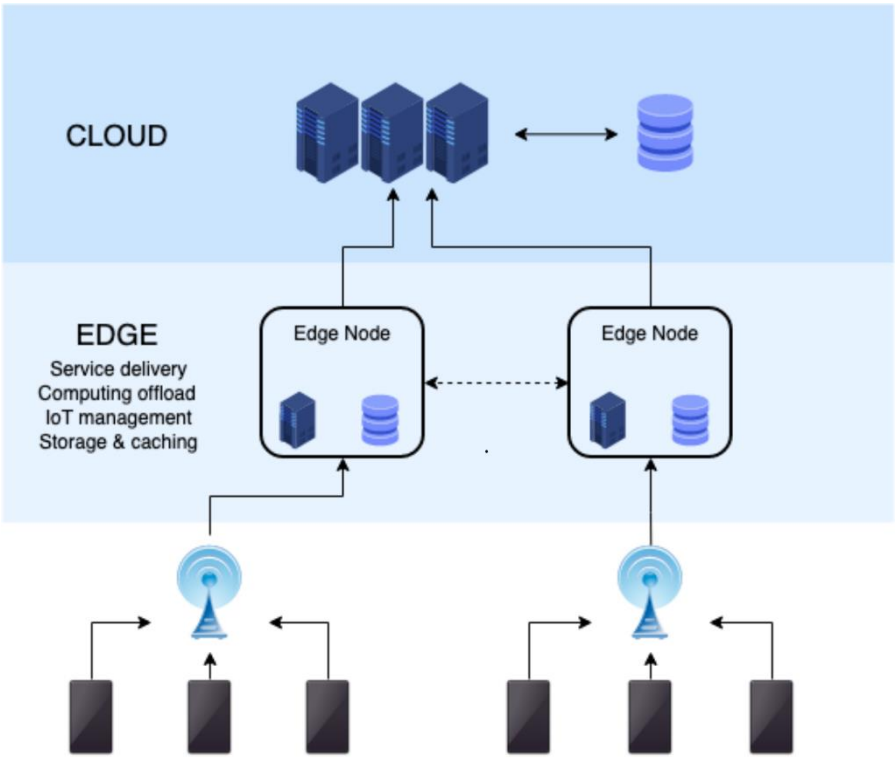
- Wearables
- Έξυπνα τηλέφωνα
- Έξυπνες κάρτες
- Αισθητήρες παντός είδους
- Συσκευές GPS
- Κινητά τηλέφωνα
- Άλλες Internet of Things συσκευές

Στην αναφορά του Report: “IDC Data Age 2025” σχετικά με έργα ψηφιοποιήσης “The Digitization of the World: From Edge to Core” αναφέρεται ότι τα συνολικά δεδομένα τα οποία θα έχουν γεννηθεί μέχρι τον χρόνο 2025 αναμένεται να έχουν φτάσει το αστρονομικό ποσό των 175 zettabytes, περίπου 10 φορές επάνω από τα επίπεδα του έτους 2016. Οι συσκευές στο άκρο του υπολογιστικού νέφους ή αλλιώς οι συσκευές του διαδικτύου των πραγμάτων (Internet of Things Devices) εκτιμάται ότι θα γεννήσουν πάνω από 90 zettabytes δεδομένων. Τέτοιοι μαζικοί όγκοι δεδομένων χρειάζονται επαρκή υπολογιστική επεξεργασία δεδομένων και δυνατότητες μετασχηματισμών. Στην περίοδο που διανύουμε υπάρχουν περίπου 7 δισεκατομμύρια συσκευές διαδικτύου των πραγμάτων οι οποίες είναι διασυνδεδεμένες και 3 δισεκατομμύρια έξυπνα τηλέφωνα διασπαρμένα σε όλο τον κόσμο. Αυτές οι συσκευές είναι εφοδιασμένες με εξελιγμένους αισθητήρες και με δυνατότητες υπολογιστικής ισχύος και επικοινωνιών. Νέες εφαρμογές εστιασμένες σε δεδομένα, υπηρεσίες δεδομένων και κατανεμημένα φορτία συνεχώς εγείρουν την ανάγκη για νέες αρχιτεκτονικές οι οποίες επαρκώς θα υποστηρίζουν προκαταβολικά συντήρηση και διαχείριση αυτών των δυναμικών περιβαλλόντων. Τεχνικές προδιαγραφές χρειάζονται ώστε να υποστηρίζουν υψηλής διαθεσιμότητας εφαρμογές με επίκεντρο τα δεδομένα οι οποίες θα παρέχονται διαμέσου περιβαλλόντων πέραν του ενός νεφο-υπολογιστικών παροχών σε απομονωμένα sites. Αυτά τα χαρακτηριστικά εστιάζουν τόσο στις τρέχουσες ανάγκες όσο και σε μελλοντικές καινοτομίες οι οποίες τελικά οδηγούν στην υιοθέτηση αρχιτεκτονικών “Edge Cloud Computing”.

Όπως αναφέρεται στις εργασίες των Wang[81] και Alahakoon[82] ο μεγάλος όγκων δεδομένων που γεννάται σε αυτές τις συσκευές χρειάζονται αντιστοίχως μεγάλες υπολογιστικές δυνατότητες υποδομών για φιλτράρισμα και επεξεργασία αυτών των δεδομένων. Νέες εφαρμογές δημιουργίας μεγάλων δεδομένων (Big-data), υπηρεσιών δεδομένων, ολοένα και αυξανόμενων επεξεργαστικών φορτίων και φορτίων κίνησης δημιουργούν την ανάγκη για διαφόρους τύπους αρχιτεκτονικές συστημάτων ώστε να διασφαλίζεται επαρκής και πλήρης υποστήριξη των κατανεμημένων εφαρμογών και υπηρεσιών. Υπάρχουν πολλές εργασίες που επεξηγούν ακριβώς αυτές τις αρχιτεκτονικές με βασική αυτή των Sitton-Candanedo [83] αλλά επίσης ένας αξιόπιστος ορισμός δίδεται ως ακολούθως από το Wikipedia [84]:

“Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data”

Μια βασική αρχιτεκτονική αυτού του τύπου φαίνεται στο ακόλουθο σχήμα 23:



Σχήμα 23 – Βασική Αρχιτεκτονική Edge Cloud Computing

Με την ολοένα αυξανόμενη χρήση των συσκευών στα τελικά σημεία των χρηστών (endpoints) και τις εξελίξεις στην υπολογιστική τεχνολογία η τεχνική του Federated Learning όπως αυτό περιγράφεται στις εργασίες των Konency και McMahan[85] κρίνεται απαραίτητο ως λύση να εφαρμοστεί σε σημεία άκρης ή «edge» του υπολογιστικού νέφους συνεισφέροντας έτσι σε ένα συνεργατικό σχήμα γνώσης. Αποφεύγεται έτσι η λιγότερο αποδοτική μεθοδολογία της χρήσης τοπικών απομονωμένων από τους άλλους κόμβους αλγορίθμων που εκπαιδεύονται ξεχωριστά ο καθένας.

Σύμφωνα με τον ορισμό που δίνει η Wikipedia[86]:

Το «Federated Learning» είναι μια τεχνική Μηχανικής Μάθησης (Machine Learning) η οποία εκπαιδεύει έναν αλγόριθμο μεταξύ διαφόρων αποκεντρικοποιημένων συσκευών στο “edge” ή μεταξύ διαφόρων διακομιστών (servers) οι οποίοι διατηρούν τοπικά δεδομένα χωρίς να ανταλλάσσονται αυτά τα τοπικά και ιδιωτικά δεδομένα μεταξύ των διακομιστών. Αυτή η προσέγγιση είναι αντίθετη με τις παραδοσιακές κεντρικοποιημένες τεχνικές Μηχανικής Μάθησης όπου όλα τα τοπικά ανά κόμβο δεδομένα συγκεντρώνονται σε ένα διακομιστή κεντρικά (aggregator server) και που συχνά τυγχάνει αυτά τα γεωγραφικά κατανεμημένα δεδομένα να ακολουθούν ίδια κατανομή. Το «Federated Learning» επιτρέπει στους διάφορους συνεργαζόμενους κόμβους/τμήματα να εφαρμόζουν ένα κοινό, στιβαρό μοντέλο μηχανικής μάθησης χωρίς ωστόσο να ανταλλάσσουν δεδομένα παρέχοντας με αυτό τον τρόπο την δυνατότητα να ικανοποιούνται κρίσιμες συνθήκες ασφάλειας όπως ασφαλή διατήρηση δεδομένων, ασφαλής πρόσβαση σε αυτά και αποφυγή επίσης άσκοπης μεταφοράς δεδομένων με αποτέλεσμα την επιβάρυνση δικτυακών πόρων όπως εύρος ζώνης κτλ. Με το «Federated Learning» οι παραπάνω τεχνικές εξελίσσονται ώστε να χρησιμοποιούν πιο επαρκή τρόπο βελτιώνοντας με αυτόν τον τρόπο της πρόγνωσης πάνω σε κατανεμημένα δεδομένα ροών. Η χρήση διαφόρων πηγών δεδομένων οι οποίες φιλοξενούνται σε συσκευές το άκρο αυξάνει τη διαθεσιμότητα δεδομένων και συνεισφέρουν σε βελτιωμένους αλγόριθμους εκπαίδευσης «Federated Learning» μαζί με την απαιτούμενη ακρίβεια προγνώσεων.

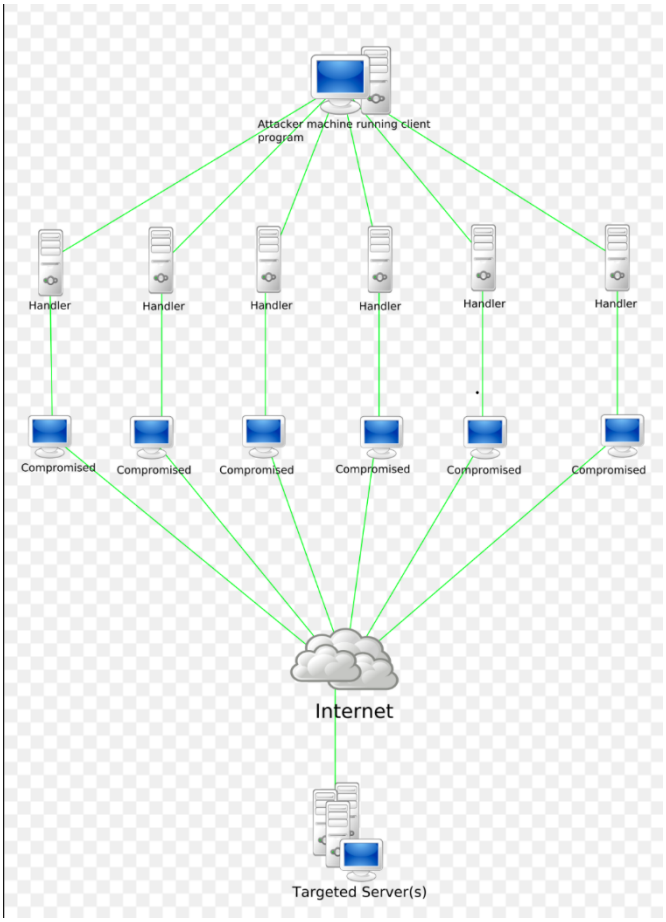
Στην συνηθισμένη και παραδοσιακή διαδικασία της βαθιάς μηχανικής μάθησης, όπως ήδη περιεγράφηκε, τα δεδομένα συγκεντρώνονται σε ένα συγκεκριμένο σημείο για επεξεργασία. Αυτή η επεξεργασία περιλαμβάνει ανίχνευση, κατηγοριοποίηση των δεδομένων (σε δεδομένα εκμάθησης και πειράματος) και πρόγνωση μελλοντικών γεγονότων και τιμών. Παραδείγματα ανωτέρω δεδομένων, αποτελούν δεδομένα από αισθητήρες θερμοκρασίας οι οποίοι βρίσκονται στην «άκρη» ή “edge” του δικτύου και μια τοπολογία εφαρμογής που επεξεργάζεται τα πρωτογενή παραγόμενα δεδομένα (raw data) από αυτούς. Σε μια τέτοια περίπτωση η μηχανική μάθηση μπορεί να βοηθήσει στην πρόβλεψη μελλοντικών συμβάντων σχετικών επί παραδείγματι με θερμοκρασιακές διακυμάνσεις ή στην πρόβλεψη στο «άκρο» και για μετρικές που αφορούν πόρους υπολογιστικού νέφους όπως πχ. η κατανάλωση επεξεργαστικής ισχύος (CPU) που μπορεί να φανερώσει την ανάγκη για επανασχεδιασμό της υποδομής προκειμένου να διατηρείται σε επιθυμητό επίπεδο η ποιότητα της παρεχόμενης υπηρεσίας (Quality of Service). Όπως αναφέρθηκε όλα αυτά τα πρωτογενή δεδομένα (raw data) που έχουν προκύψει από τους αισθητήρες συγκεντρώνονται σε ένα σημείο για επεξεργασία. Μια τέτοια προσέγγιση όμως παρουσιάζει αρκετά προβλήματα που έχουν να κάνουν με περιορισμένους πόρους στο εύρος ζώνης (bandwidth) και με μη σταθερές δικτυακές συνδέσεις. Αυτά τα προβλήματα μπορεί να προκαλέσουν καθυστερήσεις στην διαδικασία της μηχανικής μάθησης και να μειώσουν την ακρίβεια στην τελική τιμή πρόβλεψης. Επιπλέον, με βάση και την εργασία του Lim[87], άλλοι περιορισμοί σε πόρους που αφορά τεράστια αποθηκευτική χωρητικότητα και υπερβολικές καταναλώσεις επεξεργαστικής ισχύος (CPU) σε ένα κεντρικό διακομιστή (server) μπορεί να προκαλέσουν πρόβλημα όσον αφορά καθυστερήσεις στο αποτέλεσμα της διαδικασίας μάθησης ή ακόμα να οδηγήσουν και σε μη αξιόπιστα αποτελέσματα προγνώσεων.

Επιπρόσθετα, το γεγονός ότι σε μια κεντρικοποιημένη αρχιτεκτονική όλα τα ευαίσθητα και μη δεδομένα των τελικών χρηστών στέλνονται για επεξεργασία σε έναν απομακρυσμένο διακομιστή του υπολογιστικού νέφους (cloud server) εισάγει μια σειρά από ανησυχίες όσον

αφορά την ασφάλεια και πιθανές περιπτώσεις διαρροής κρίσιμης πληροφορίας αναλόγως την περίπτωση. Αυτό συμβαίνει γιατί τα δεδομένα που παράγονται στις συσκευές ή διακομιστές που βρίσκονται στο άκρο ή “edge” αφήνουν την αρχική τους τοποθεσία και μεταφέρονται και συγκεντρώνονται σε ένα κεντρικό server διατρέχοντας ένα δαιδαλώδες δίκτυο πολλές φορές. Κατά την διάρκεια μεταφοράς δεδομένων πολλά ρίσκα ασφάλειας μπορούν να εμφανιστούν:

- Man-in-the-middle attacks: Ο επιτιθέμενος παρεμποδίζει τη νόμιμη επικοινωνία μεταξύ δύο μερών, τα οποία είναι φιλικά μεταξύ τους. Στη συνέχεια, ο κακόβουλος ελέγχει τη ροή επικοινωνίας και μπορεί να αποσπάσει ή να αλλάξει πληροφορίες που στέλνονται από έναν από τους αρχικούς συμμετέχοντες. Οι επιθέσεις man-in-the-middle εφαρμόζονται ιδιαίτερα στο πρωτόκολλο Diffie-Hellman, όταν η συμφωνία ανταλλαγής κλειδιών γίνεται χωρίς επικύρωση «authentication». Οι επιθέσεις man-in-the-middle έχουν δύο κοινές μορφές: ο επιτιθέμενος είτε υποκλέπτει (κρυφακούει) «eavesdropping», είτε και αλλοιώνει κατάλληλα το μήνυμα.

- Distributed denial-of-service attack (DDoS attack): Είναι η κατάσταση κατά την οποία ο επιτιθέμενος προσπαθεί να καταστήσει μια μηχανή ή έναν δικτυακό πόρο μη διαθέσιμο στους εν δυνάμει χρήστες με προσωρινή ή μόνιμη «καταστροφή» των υπηρεσιών προερχόμενη από ένα host συνδεδεμένο στο σχετικό δίκτυο. Στην κλασσική περίπτωση του Denial of Service έχουμε το γεγονός του «μπουκώματος» της υπό εξέτασης μηχανής με τεράστιο αριθμό αιτημάτων (i.e. http requests or other) που υπερφορτώνουν την μηχανή στόχο και αποτρέπουν έτσι τους κανονικούς χρήστες να την προσπελάσουν. Στην περίπτωση μας που αποτελεί τον μεγαλύτερο κίνδυνο μιας κεντρικοποιημένης υποδομής Μηχανικής Μάθησης είναι το εισερχόμενο μεγάλο φορτίο προερχόμενο από πολλές πηγές με αποτέλεσμα να είναι δύσκολο να σταματήσει κάποιος την επίθεση μπλοκάροντας μια μόνο απλή πηγή επίθεσης. Μια γενική περιγραφή δίδεται στο κατωτέρω σχήμα 24:



Σχήμα 24 - Distributed denial-of-service attack

- Data and machine learning model poisoning: Η περίπτωση του data poisoning σε συστήματα μηχανικής μάθησης αφορά την έγχυση κάποιων «διορθωμένων» δεδομένων στο dataset που χρησιμοποιείται για εκπαίδευση των αλγορίθμων επηρεάζοντας έτσι την τελική ακρίβεια και αποτελέσματα των προβλέψεων. Με βάση την βιβλιογραφία και τις σχετικές περιγραφές όπως για παράδειγμα στην αναφορά του Bdtechalks [88] μια μόλυνση στα δεδομένα εκπαίδευσης κατά 3% οδηγεί σε μια πτώση κατά 11% της ακρίβειας των αλγορίθμων.

Με βάση τις ανωτέρω αναφορές και προβλήματα, η λύση του Federated Learning και του Edge Cloud έρχεται να δώσει διέξοδο καθώς αναφέρονται σε περιπτώσεις ενός framework όπου τα δεδομένα παραμένουν δίπλα στην πηγή που τα δημιούργησε. Σε αυτό το framework και όπως παρουσιάζεται στην εργασία των Chen[89][90] και Lu[91] γίνεται ολοκλήρωση του δικτύου, της υπολογιστικής ισχύος, της αποθήκευσης δεδομένων μέσω ενός κατανεμημένου αλγορίθμου ο οποίος παρέχει υπηρεσίες στις συσκευές που βρίσκονται στο άκρο του δικτύου. Όπως περιγράφεται λεπτομερειακά στην εργασία του Imakura [92] η τεχνική του Federated Learning δίνει την δυνατότητα αυξανόμενης εκπαίδευσης μηχανικής μάθησης των μοντέλων με διαμοιραζόμενα δεδομένα από το υπολογιστικό νέφος και τις άκρες του δικτύου γενικότερα χωρίς να χρειάζεται η ανταλλαγή οποιουδήποτε ευαίσθητου ή ιδιωτικού δεδομένου. Με αυτό τον τρόπο, η συγκεκριμένη τεχνική υποστηρίζει την ιδιωτικότητα και την προστασία της ασφάλειας των δεδομένων εξ ορισμού και αρχικού σχεδιασμού του

αλγορίθμου ενώ την ίδια στιγμή μπορεί και πετυχαίνει χαμηλότερο φορτίο κίνησης, χαμηλότερες καθυστερήσεις και χαμηλότερη κατανάλωση ισχύος. Όπως αναφέρεται στην εργασία των Fantacci και Picano [93], το Federated Learning δίνει την δυνατότητα στους τοπικούς κόμβους να εκπαιδεύσουν ένα κοινό μοντέλο που κρατάνε τοπικά κάθε φορά ανταλλάσσοντας μόνο τα βάρη που προκύπτουν από κάθε κύκλο εκπαίδευσης καθώς και τα hyper-parameters χωρίς να χρειάζεται να μεταφέρουν τα πρωτογενή ιδιωτικά δεδομένα. Αυτό είναι αποτέλεσμα της λογικής της συγκεκριμένης τεχνικής να φέρνει την διαδικασία μηχανικής μάθησης κοντά στις πηγές παραγωγής δεδομένων.

Ένα από τα βασικά σημεία του Federated Learning, όπως έχει ήδη αναφερθεί, είναι να εγγυάται την ασφάλεια των συμμετεχόντων μερών-κόμβων ανταλλάσσοντας μόνο παραμέτρους του εκπαιδευόμενου μοντέλου αντί να ανταλλάσσει ιδιωτικά και ευαίσθητα δεδομένα. Ωστόσο αυτή η διαδικασία μπορεί επίσης να αποβεί επικίνδυνη και να είναι υποκείμενη σε πληθώρα «επιθέσεων» που έχουν να κάνουν με «δηλητηρίαση» ή “poisoning” της όλης διαδικασίας μάθησης Federated Learning. Για παράδειγμα, ένας κακόβουλος που συμμετέχει ως κόμβος στην διαδικασία μπορεί να στέλνει λανθασμένα παραμέτρους μοντέλου ή κατεστραμμένες παραμέτρους ώστε να «διαβάλει» την διαδικασία μάθησης σε global aggregation βήμα που λαμβάνει χώρα στον κεντρικό διακομιστή (server) της υποδομής νέφους. Σαν αποτέλεσμα το global μοντέλο της όλης διαδικασίας θα ανανεώνεται με λανθασμένες και μη ακριβείς παραμέτρους και η federated learning διαδικασία θα δίνει είτε κατεστραμμένα είτε ανακριβή αποτελέσματα προγνώσεων. Λεπτομερειακή περιγραφή δίνεται στην εργασία του Lim [94]. Εκτός βέβαια από τις στοχευμένες επιθέσεις υπάρχουν και οι περιπτώσεις γεγονότων όπου μπορούν να προκαλούνται χωρίς πρόθεση αλλά λόγω κάποιου λάθους στο υλικό ή στο λογισμικό σχετικά με τον αλγόριθμο.

Στην εργασία του Chandola [95] επεξηγείται λεπτομερώς ο ορισμός της ανίχνευσης μη ομαλής συμπεριφοράς ή αλλιώς “anomaly detection” ως η περίπτωση της ανίχνευσης εξαιρετικά σπάνιων συμβάντων ή μη κανονικών γεγονότων και «εχθρικών» δραστηριοτήτων σε συστήματα cyber-security και safety-critical. Ως βασική μέθοδος εκπαίδευσης στα νευρωνικά και βαθιά νευρωνικά δίκτυα είναι αυτή του back-propagation. Πρόκειται με άλλα λόγια, για μια πρακτική βέλτιστης προσαρμογής των βαρών ενός δικτύου (μοντέλου) εκμάθησης βασισμένη στους ρυθμούς λάθους που προέρχονται από τις συναρτήσεις απωλειών (loss functions) που έχουν προκύψει από την αμέσως προηγούμενη επαναληπτική διαδικασία εκμάθησης. Όταν ανωμαλίες όσον αφορά τα δεδομένα ή περίεργα πρότυπα δεδομένων παρουσιάζονται κατά την διάρκεια αυτών των επαναληπτικών διαδικασιών (ολοκλήρωση αλγορίθμου) τότε προκύπτουν υψηλές τιμές συναρτήσεων απωλειών (loss functions) όπως λέγονται, και οι οποίες προκαλούν χρονικές καθυστερήσεις αλλά και δίνουν υποβαθμισμένες τελικές παραμέτρους των βαρών των μοντέλων που χρησιμοποιούνται. Αυτό το γεγονός οδηγεί σε ανακρίβεια προβλέψεων για μελλοντικές τιμές ή γεγονότα που αφορούν διάφορες μετρούμενες μετρικές όπως για παράδειγμα η κατανάλωση επεξεργαστικής ισχύος (CPU Consumption) σε εικονικές μηχανές σε ένα περιβάλλον πολλών παροχών υπολογιστικού νέφους. Τέτοιες ανακρίβειες μπορούν να οδηγήσουν σε λανθασμένο επαναπροσδιορισμό των πόρων και άρα πιθανές υποβαθμίσεις των παρεχόμενων υπηρεσιών και εφαρμογών στους τελικούς χρήστες.

Με βάση τα ανωτέρω προβλήματα, στην παρούσα Διατριβή παρουσιάζεται μια λύση αποκεντρικοποιημένη και λειτουργικά ανεξάρτητη από ένα συγκεκριμένο πάροχο υπολογιστικού νέφους. Πιο συγκεκριμένα, προτείνεται μια Αρχιτεκτονική πολλών παρόχων ενός μοντέλου Federated Learning που χρησιμοποιείται για να κάνει προβλέψεις σε τιμές

που σχετίζονται με μετρικές σχετιζόμενες με τον χρόνο και που παρέχει χαμηλότερες καθυστερήσεις στους υπολογισμούς, χαμηλότερη κατανάλωση εύρους ζώνης, λιγότερους κινδύνους έλλειψης ιδιωτικότητας δεδομένων αλλά που ταυτόχρονα χρησιμοποιεί μια προσέγγιση επιλογής συμμετεχόντων μερών-κόμβων. Με αυτό τον τρόπο αυξάνεται η ακρίβεια στην πρόβλεψη χρησιμοποιώντας τόσο την ανίχνευση και απόρριψη των ανωμαλιών δεδομένων (data abnormalities) όσο την εξεύρεση για κάθε συμμετέχοντα κόμβο του επαρκούς μεγέθους δεδομένων προκειμένου να αναπτυχθεί ένα υψηλής ποιότητας μοντέλο.

Κατά τη διάρκεια της διαδικασίας Federated Learning, μοντέλα βαθιάς μηχανικής μάθησης χρησιμοποιούνται ένα γεγονός το οποίο αυξάνει ακόμα περισσότερο την ακρίβεια του συνολικού τελικού μοντέλου. Επιπλέον σε αυτή τη διδακτορική εργασία ένας αρκετά ανεπτυγμένος αλγόριθμος και ένα συγκεκριμένο setup πειραμάτων λαμβάνει χώρα σε συνάρτηση με την κατανομή δεδομένων των διαφόρων κόμβων στο Federated Learning cluster. Στον μισό αριθμό κόμβων τα δεδομένα είναι κατανεμημένα κατανεμημένα τυχαία ανά κόμβο και έχουν uniform κατανομή. Με άλλα λόγια τα δεδομένα είναι με ίδιο τρόπο και ανεξάρτητα κατανεμημένα (identically and independent distributed (IID)) σημαίνοντας ότι δεν υπάρχουν συνολικά trends, η κατανομή των δεδομένων δεν μεταβάλλεται πολύ και όλα τα κομμάτια των δεδομένων λαμβάνονται με την ίδια κατανομή συνάρτησης πιθανότητας. Χρησιμοποιώντας IDD δεδομένα στη μεριά του client σημαίνει ότι κάθε mini-batch δεδομένων το οποίο χρησιμοποιείται για κάθε local update είναι στατιστικά ίδιο με κάθε άλλο δείγμα δεδομένων από το συνολικό dataset, το οποίο είναι η ένωση όλων των τοπικών datasets των clients του Federated Learning συστήματος. Πρακτικά, είναι μη ρεαλιστικό να θεωρούμε ότι τα τοπικά δεδομένα σε κάθε κόμβο-client είναι πάντα με τον ίδιο τρόπο και ανεξάρτητα κατανεμημένα δηλαδή IID (identically and independent distributed). Για αυτό το λόγο στον υπολειπόμενο μισό αριθμό των κόμβων, όλα τα δεδομένα έχουν το ίδιο label αντιπροσωπεύοντας τις περιπτώσεις όπου οι κόμβοι δεν έχουν κατανεμημένο με τον ίδιο τρόπο τα δεδομένα τους (non-uniform information). Αυτό συμβαίνει επειδή το συνολικό σετ δεδομένων έχει δείγματα με πολλά διαφορετικά label που ανήκουν στην περίπτωση του σεναρίου non-IID δεδομένων. Ως αποτέλεσμα μπορούμε να υποστηρίξουμε ότι η εργασία στην παρούσα διατριβή, καινοτομεί στον τρόπο κατανομής των δεδομένων που μπορεί να λάβει χώρα στους διάφορους κόμβους ενός Federated Learning cluster (συνδυασμός iid και non-iid περίπτωσης) επειδή η κατανομή αυτή δεν είναι γνωστή εκ των προτέρων στα περισσότερα από τα πραγματικά σενάρια.

Έτσι με βάση την προτεινόμενη λύση, ένα επιλεκτικό Federated Learning μοντέλο συγκερασμού χρησιμοποιείται και βασίζεται σε δύο συνθήκες που θα πρέπει ταυτόχρονα να ικανοποιούνται. Πρώτον, η συνθήκη για το μέγεθος των δεδομένων διασφαλίζει επαρκές μέγεθος δεδομένων για χρήση τους ως δεδομένων εκπαίδευσης τοπικά στους κόμβους. Και αυτό γιατί πολλές φορές το μέγεθος των τοπικών δεδομένων εκπαίδευσης είναι διαφορετικό από κόμβο σε κόμβο. Δεύτερον, η τιμή για την συνθήκη της τοπικής συνάρτησης απωλειών (local loss function) βοηθά στην ανίχνευση ανωμαλιών στα δεδομένα κάθε κόμβου. Εάν μια υψηλή τιμή απωλειών υπάρχει σε ένα κόμβο αυτό δείχνει χαμηλή ποιότητα δεδομένων (anomalous data) που μπορεί να οδηγήσει σε μη ακριβείς παραμέτρους του τοπικού μοντέλου πρόβλεψης και για αυτό τον λόγο να πρέπει να εξαιρεθεί από την διαδικασία εκπαίδευσης του global μοντέλου. Αυτή η δεύτερη συνθήκη χρησιμοποιείται καθόσον ένα συγκεκριμένο τοπικό σετ δεδομένων δεν είναι αντιπροσωπευτικό της ολικής κατανομής δεδομένων όλου του συστήματος. Χρησιμοποιώντας αυτές τις δύο βασικές συνθήκες μαζί με

το γεγονός ότι non-convex βαθιάς μηχανικής μάθησης τοπικές συναρτήσεις βελτιστοποίησης (optimization local loss functions) εφαρμόζονται σε περιβάλλοντα πολλών παρόχων υπολογιστικού νέφους σε Federated Learning αρχιτεκτονικές, διασφαλίζει μια υψηλότερη ακρίβεια προβλέψεων σε όλο το σύστημα για μετρικές που αναφέρθηκαν ήδη. Τέτοιες αναφορές παρουσιάζουν οι εργασίες των Srivastava [96] αλλά και των Martin-Donas [97]. Οι non-convex συναρτήσεις βελτιστοποίησης απωλειών (non-convex local loss functions) σε κάθε κόμβο έχουν τα ίδια πλεονεκτήματα με τις απλές convex συναρτήσεις και την ίδια στιγμή διαφέρουν στο ότι συγκλίνουν και βελτιστοποιούνται πολύ γρηγορότερα σε ένα τοπικό ελάχιστο από ότι σε ένα global βέλτιστο χρησιμοποιώντας την Stochastic Gradient Descent μέθοδο και το mini-batching [98][99].

Σε συνδυασμό με τα παραπάνω, ωστόσο αξίζει να αναφερθεί και η δυσκολία πολλές φορές στην εύρεση πόρων αρκετών πχ. υπολογιστική ισχύς ή μέγεθος αποθήκευσης σε συστήματα που βρίσκονται στο άκρο ή “edge”. Επακόλουθο αυτού είναι να μειώνουν την απόδοση των συστημάτων μάθησης που πιθανά φιλοξενούνται στο “edge”. Είναι πολύ σημαντικό να υπάρχει ένα επαρκές σύστημα μηχανικής μάθησης αξιόπιστο και που να μπορεί να προσπερνά αυτά τα προβλήματα. Γι’ αυτό το λόγο στην ανωτέρω Federated Learning λογική επιπρόσθετα θεωρούμε την εκπαίδευση των κυρίως δεδομένων για την εκπαίδευση του καθολικού μοντέλου FL να λαμβάνει χώρα σε σύστημα υψηλής διαθεσιμότητας πόρων. Η εκτέλεση ωστόσο του Inference κομματιού του μοντέλου πρόβλεψης μπορεί να εκτελείται σε συστήματα ή συσκευές στο άκρο του υπολογιστικού νέφους αντικείμενο που εξετάζεται ενδελεχώς στην επόμενη παράγραφο 7 της παρούσας Διατριβής.

6.2 Σχετικές Εργασίες

Τα τελευταία χρόνια αρκετή έρευνα έχει διεξαχθεί στο πεδίο του Federated Learning (FL) ή κατανεμημένης μάθησης σε περιβάλλοντα πολλών νεφουπολογιστικών παρόχων προκειμένου να χρησιμοποιείται για αναπροσαρμογή και πρόβλεψη για την ζήτηση πόρων. Παράμετροι που έχουν αξία και ερευνώνται στην διεθνή βιβλιογραφία για τις παραπάνω τεχνολογίες είναι ο τρόπος ανταλλαγής παραμέτρων στους προαναφερόμενους αλγορίθμους πχ. σύγχρονος ή ασύγχρονος, η χρήση convex ή non-convex τοπικών συναρτήσεων απωλειών, μέθοδοι γενικής μόχλευσης ή global aggregation methods, τεχνικής επιλογής κόμβων για συμμετοχή στην Federated Learning διαδικασία μάθησης και μέθοδοι αύξησης των προβλέψεων και της καλύτερης χρήσης των πόρων σε κόμβους(nodes) που βρίσκονται στο edge ή αλλιώς στο «άκρο». Σε προηγούμενες ερευνητικές προσεγγίσεις του Federated Learning χρησιμοποιούνται πρωτόκολλα σύγχρονων επικοινωνιών στα οποία ο κεντρικός σέρβερ κατανέμει το κεντρικό μοντέλο σε έναν επιλεγμένο αριθμό κόμβων και παράλληλα συγκεντρώνει τις τιμές των τοπικών μοντέλων εφαρμόζοντας σταθμισμένο μέσο όρο, αφού δεχθεί όλα τα updates από τους client κόμβους. Αυτή η μέθοδος είναι αρκετά κοστοβόρα λόγω των καθυστερήσεων συγχρονισμού καθώς ο κεντρικός σέρβερ χρειάζεται να περιμένει για τις αποκρίσεις από τα local updates όλων των clients πριν το global aggregation. Η θεώρηση ύπαρξης συσκευών που καθυστερούν είναι μια επιθυμητή, ενώ ένα αναξιόπιστο δίκτυο μπορεί να προκαλέσει προβλήματα στη διαδικασία του Federated Learning. Από την άλλη μεριά, αναφορές ασύγχρονης επικοινωνίας στο Federated Learning παρουσιάζονται ως εναλλακτικές όπου κεντρικός σέρβερ μπορεί να μαζεύει τις παραμέτρους των μοντέλων χωρίς να περιμένει τους κόμβους που καθυστερούν αρκετά. Παρόλα αυτά αυτή η μέθοδος υποθέτει ότι υπάρχει ένας συγκεκριμένος αριθμός δεδομένων σε κάθε κόμβο κατά τη διάρκεια της διαδικασίας εκπαίδευσης το οποίο δεν είναι μια περίπτωση στην πραγματική ζωή. Συγκεντρωτικά και συνοπτικά παρουσιάζονται στην παρούσα ενότητα οι διάφορες εργασίες που μελετήθηκαν καθώς και βασικά χαρακτηριστικά τους.

Με την μεγάλη αύξηση του αριθμού των συστημάτων στο άκρο «edge systems» του υπολογιστικού νέφους και τις μεγάλες εξελίξεις που έχουν λάβει χώρα στην τεχνολογία υλικού και λογισμικού των υπολογιστών, τεχνικές Κατανεμημένης Μάθησης ή αλλιώς Federated Learning [85] [102] μπορούν να εφαρμοστούν τόσο σε κανονικούς διακομιστές (Servers) του υπολογιστικού νέφους όσο και σε διακομιστές ευρισκόμενοι στο άκρο του υπολογιστικού νέφους (edge) συνεισφέροντας με αυτό τον τρόπο σε ένα συνεργατικό σχήμα γνώσης. Ξεπερνάει με αυτό τον τρόπο την κλασσική μέθοδο των απομονωμένων κόμβων όπου τρέχουν αλγορίθμους σε τοπικά μόνο δεδομένα και όχι συνεργατικά. Δυστυχώς στην προαναφερθήσα εργασία και προκειμένου να οδηγηθούμε σε καλύτερο συνολικά εκπαιδευόμενο γενικό συνεργατικό μοντέλο θα ήταν προτιμότερο να εφαρμόζονταν και τεχνικές επιλογής clients (Αρχιτεκτονική Server – Client) γεγονός που θα αύξανε και την ακρίβεια προβλέψεων του αλγορίθμου. Γενικά μιλώντας, το Federated Learning πετυχαίνει την συνεργατική εκμάθηση μοντέλων μηχανικής μάθησης με δεδομένα ευρισκόμενα τόσο στο υπολογιστικό νέφος αλλά και στο άκρο του αποφεύγωντας ωστόσο να ανταλλάσει ιδιωτικές πληροφορίες [92]. Έτσι θεωρείται μια ιδιαίτερα ασφαλής τεχνική και την ίδια στιγμή εξασφαλίζει την ελλάτωση του δικτυακού φόρτου, την ελλάτωση των καθυστερήσεων απόκρισης του συνολικού αλγορίθμου και της κατανάλωση ισχύος [93].

Στην εργασία που παρουσιάζει ο Chen [103], ένα ασύγχρονο σύστημα πραγματικού χρόνου βασισμένο σε ένα framework κατανεμημένης μάθησης (federated learning) είναι αυτό που εξετάζεται και όπου βλέπουμε κάποιες συσκευές στο άκρο (edge devices) να συνδέονται σε ένα κεντρικό διακομιστή (server). Αυτός ο διακομιστής (server) συλλέγει παραμέτρους από τοπικά μοντέλα των επιμέρους συσκευών και οι οποίες παράμετροι προέρχονται από convex και deep learning συναρτήσεις απώλειας (loss functions) που εκτελούνται τοπικά σε κάθε συσκευή. Στην συγκεκριμένη εργασία αν και επιτυγχάνεται καλή ακρίβεια δεν χρησιμοποιείται καθόλου μεθοδολογία επιλογής client βάσει του μεγέθους των δεδομένων τοπικά στον κάθε κόμβο ούτε κάποιος διαμοιρασμός των βαρών για το συνολικά διαμορφούμενο μοντέλο πρόγνωσης κατά την διάρκεια του aggregation (συλλογή) των παραμέτρων από τους διάφορους clients του federated συστήματος. Επιπλέον η περίπτωση των περιορισμών των πόρων σε συσκευές που βρίσκονται στο άκρο «edge» του υπολογιστικού νέφους δεν λαμβάνεται υπόψιν αλλά ούτε και η δυνατότητα σε τέτοιες περιπτώσεις να εκτελεστεί το inference κομμάτι του εκπαιδευμένου αλγορίθμου πρόγνωσης με αποδοτικότερο τρόπο. Η ίδια ασύγχρονη μέθοδος επικοινωνιών ακολουθείται και στην ερευνητική εργασία των Xie [104] όπου η προτεινόμενη προσέγγιση δείχνει πολύ καλά αποτελέσματα με μια σχεδόν γραμμική σύγκλιση σε ένα γενικό βέλτιστο σημείο σύγκλισης για συναρτήσεις convex. Τοπικές συναρτήσεις deep learning δεν εξετάζονται καθόλου στην συγκεκριμένη εργασία. Εάν εμφανιστούν data abnormalities ή μεγάλη ετερογένεια στα δεδομένα, αυτό θα επηρεάσει αρνητικά την όλη διαδικασία προβλέψεων με αμφίβολα αποτελέσματα.

Μελετώντας δίκτυα υπολογιστών εστιασμένα στο άκρο του υπολογιστικού νέφους συναντάμε χαρακτηριστικά την εργασία των Fantacci and Picano [93] που χρησιμοποιεί το Federated Learning για να κάνει πρόγνωση των αιτημάτων για πόρους που θα χρειαστούν οι πιο δημοφιλείς τύποι εφαρμογών στο διαδίκτυο. Χρησιμοποιώντας σύγχρονο τρόπο επικοινωνίας μεταξύ των κινητών συσκευών που βρίσκονται στο «άκρο» του υπολογιστικού νέφους και εφαρμόζοντας convex τοπικές συναρτήσεις, το προτεινόμενο μοντέλο επιτυγχάνει υψηλό βαθμό ακρίβειας σχετικά με προγνώσεις σε ζήτηση πόρων από σχετικές εφαρμογές. Η συνολική συλλογή των παραμέτρων του μοντέλου πρόγνωσης σε global επίπεδο (aggregation method) είναι ένας weighted τρόπος με βάση το μέγεθος των δεδομένων κάθε κόμβου / συσκευής που συμμετέχει στην διαδικασία μάθησης. Παρολαυτά, καμμιά μέθοδος επιλογής κόμβων δεν εφαρμόζεται στην συγκεκριμένη εργασία και με αυτό τον τρόπο δεν μπορεί να αποφευχθεί η ύπαρξη ανωμαλιών δεδομένων ή «μόλυνση» δεδομένων σε κάποιο πιθανά συμμετέχοντα κόμβο ή συσκευής στην μάθηση. Σε μια παρόμοια εργασία, αυτή των Liu et al. [105], ένα σύστημα πρόβλεψης της ροής δεδομένων παρουσιάζεται βασισμένο στην τεχνολογία του Federated Learning. Στην συγκεκριμένη παρουσίαση, ένα ακριβές μοντέλο πρόγνωσης κίνησης δεδομένων διατηρώντας ταυτόχρονα την ιδιωτικότητα των δεδομένων υλοποιείται. Η συγκεκριμένη υλοποίηση δίνει υψηλό βαθμό ακρίβειας, μεγαλύτερο και από αυτόν των μοντέλων deep learning. Δεν παρουσιάζεται καμμία έκπτωση ασφάλειας ή ιδιωτικότητας δεδομένων της συσκευής στο «άκρο» του νέφους. Σε αυτην την εργασία επίσης δεν αναφέρεται καμμιά μεθοδος global aggregation βασισμένη σε επιλογή κόμβων συμμετοχής ή χρονικού παραθύρου global aggregation. Όλα τα ανωτέρω είναι τα βασικά μειονεκτήματα της εργασίας των Liu et al. [105].

Ακολουθώντας την παραδοσιακή λογική της κατανεμημένης μάθησης όπως της Federated Learning μεθόδου, μια καινοτόμος μεθοδολογία που επαυξάνει την μέθοδο της Federated

Learning παρουσιάζεται στην εργασία των Lu et al. [106]. Στην συγκεκριμένη εργασία μια μεθοδολογία συμπίεσης των δεδομένων μεταφοράς αλλά και των τοπικών παραμέτρων μεταξύ των διαφόρων κόμβων (clients) και του κεντρικού server εφαρμόζεται με στόχο την επαυξημένη ασφάλεια μετάδοσης και χωρίς να επηρεάζεται ιδιαίτερα η ακρίβεια του συνολικού μοντέλου σε σχέση με τις τεχνικές μετάδοσης μη-συμπιεσμένων δεδομένων. Βέβαια ούτε σε αυτή την εργασία βλέπουμε χρήση Deep Learning τοπικών συναρτήσεων απωλειών χωρίς μάλιστα καμμία τεχνική για επιλογή των κατάλληλων κόμβων που θα συμμετέχουν στο Federated Learning με αποτέλεσμα να παρουσιάζεται μια απόκλιση μεταξύ των πραγματικών και αναμενόμενων αποτελεσμάτων όσον αφορά την ακρίβεια προγνώσεων. Γενικότερα, επειδή η αξιοπιστία σε όρους περιεχομένου των δεδομένων μπορεί να ποικίλει και να μεταβάλλεται από κόμβο σε κόμβο ενός Federated Learning συστήματος, η δημοσίευση των Qin et al. [107] προτείνει ένα μοντέλο επιλογής κόμβων στο γενικό aggregation (συσσωμάτωση) σε λογική Federated Learning σε περιπτώσεις ανίχνευσης ανωμαλιών στα δεδομένα. Επειδή αρκετοί κόμβοι στην μεθοδολογία του Federated Learning είναι δυνατόν να περιέχουν τοπικά δεδομένα με «ανωμαλίες και ασυνέχειες» τα τοπικά μοντέλα που προκύποτυν από αυτά τα δεδομένα εξαιρούνται του τελικού συνολικού μοντέλου γεγονός που αυξάνει την συνολική ακρίβεια προγνώσεων. Δυστυχώς στη συγκεκριμένη εργασία δεν εξετάζεται το ικανό μέγεθος των τοπικών δεδομένων ώστε να μπορούν να εκπαιδεύσουν επαρκώς τα τοπικά μοντέλα σε κάθε κόμβο, γεγονός που θα μπορούσε να αυξήσει ακόμα περισσότερο την ακρίβεια προγνώσεων.

Ένας σύγχρονος τρόπος επικοινωνίας μεταξύ κομβου client και κεντρικού διακομιστή(server) σε ένα σύστημα Federated Learning εξετάζεται στην εργασία των Chen [146], όπου ανταλλάσσονται παράμετροι του Gradient Descent και τελικά χρησιμοποιούνται κατά τη διαδικασία του global aggregation σε κάθε βήμα εκμάθησης στον διακομιστή aggregation server. Στην συγκεκριμένη μεθοδολογία δεν χρησιμοποιούνται ούτε συναρτήσεις βαθιάς μηχανικής μάθησης όσον αφορά τις απώλειες, ούτε μεθοδολογίες αποφάσεων για τη συμμετοχή των κόμβων-clients στο global aggregation κάθε φορά. Τα αποτελέσματα δείχνουν μια σημαντική μείωση στο φόρτο της επικοινωνίας σε σύγκριση με τα παραδοσιακά συστήματα Federated Learning αλλά όχι σημαντική βελτίωση στην ακρίβεια προγνώσεων. Μια άλλη εργασία σχετικά με μη φυσιολογική συμπεριφορά κόμβου (abnormal client behaviour) παρουσιάζεται στην αναφορά του Li [147], και αφορά περιπτώσεις όπου είτε με σκοπό είτε χωρίς σκοπό αποκλίνουν από την κλασική διαδικασία μάθησης ενός συστήματος federated learning έχοντας ως αποτέλεσμα μη φυσιολογικές συμπεριφορές. Σε αυτή την εργασία, μια σύγχρονη μέθοδος επικοινωνίας εφαρμόζεται χρησιμοποιώντας όλους τους τύπους συναρτήσουν απωλειών όπως convex, non-convex και βαθιά μηχανική μάθηση σε συσχέτιση με μια μέθοδο global aggregation βασισμένη μόνο σε τιμές συναρτήσεων τοπικών απωλειών.

Στην εργασία του Ek et al. [108] αναλύεται και παρουσιάζονται συγκρίσεις και αξιολογήσεις κάποιων αλγορίθμων που αφορούν αρχιτεκτονικές aggregation με χρήση Federation Learning κυρίως με εφαρμογή σε αναγνώριση ανθρώπινης δραστηριότητας. Οι αλγόριθμοι που εξετάζονται με τεχνική aggregation, δηλαδή συλλογή τοπικών επιμέρους εκπαιδευμένων παραμέτρων και εξαγωγή τελικών παραμέτρων του μοντέλου, είναι οι FedAvg, FedPer και FedMA. Τα πειραματικά αποτελέσματα δείχνουν μια μέτρια ακρίβεια πρόγνωσης σε προβλήματα classfication πχ. εικόνες. Δυστυχώς και σε αυτή την μελέτη, δεν εξετάζοναι τυχόν ανωμαλίες ή ασυνήθιστα patterns σε δεδομένα. Ωστόσο μια μελέτη που λαμβάνει υπόψιν τις περιπτώσεις ετερογένειας των δεδομένων στους διάφορους κόμβους, είναι αυτή των Arivazhagan et al. [109]. Ειδικότερα, ο αλγόριθμος FedPer που παρουσιάζεται προσφέρει μια βασική και προσωποποιημένη προσέγγιση για Federated Learning σε deep learning νευρωνικά δίκτυα προκειμένου να καταπολεμήσει τα φαινόμενα στατιστικής ετερογένειας δεδομένων. Βέβαια και πάλι η περίπτωση ανωμαλιών σε δεδομένα δεν μπορεί να εξεταστεί

με τον συγκεκριμένο αλγόριθμο ούτε να αντιμετωπιστεί. Τέλος, μια τελευταία παράμετρος που στην συγκεκριμένη εργασία μένει αναπάντητη είναι οι όποιοι τυχόν περιορισμοί σε περιπτώσεις περιορισμένων υπολογιστικών πόρων σε συστήματα που βρίσκονται στο άκρο «edge» του υπολογιστικού νέφους και πως αυτοί μπορούν να επηρεάζουν την ακρίβεια προγνώσεων.

Σε μια άλλη εργασία, αυτή των Ye et al. [110], και προκειμένου να αντιμετωπιστεί το πρόβλημα που έχει ήδη ανεφερθεί σχετικά με τους περιορισμούς σε πόρους υλικού (hardware) σε χρήση μνήμης και δίσκου των συμμετεχόντων στον αλγόριθμο κινητών συσκευών κατά την εκτέλεση του αλγορίθμου FedAvg, νέες τεχνικές βελτιστοποίησης της εκτέλεσης της μεθόδου Federated Learning παρουσιάζονται. Πιο συγκεκριμένα, στην λύση περιλαμβάνεται ένας διαχωρισμός των διαδικασιών που αφορούν την ολοκλήρωση της εκμάθησης των τοπικών αλγορίθμων στις κινητές συσκευές και τον edge server και της διαδικασίας του global aggregation που εκτελείται στον edge server και στον κεντρικό server. Ωστόσο σε αυτή την εργασία δεν λαμβάνει χώρα οποιαδήποτε διαδικασία λήψης αποφάσεων σχετικά με την επιλογή clients στην μέθοδο του Federated Learning ώστε να αντιμετωπίζεται η ύπαρξη τυχόν ανωμαλιών στα δεδομένα.

Μια αρκετά αξιόπιστη τεχνική επιλογής κόμβων-clients στην διαδικασία του Federated Learning για περιπτώσεις και κινητών συστημάτων ευρισκόμενων στο άκρο του υπολογιστικού νέφους και αντιμετωπίζοντας τα θέματα των ελάχιστων διαθέσιμων πόρων και δικτύου παρουσιάζεται στην εργασία των Nishio και Yonetani [111]. Ακολουθώντας την ανωτέρω λογική η νέα μέθοδος επιτρέπει στον κεντρικό server να λαμβάνει (aggregate) όσο πιο πολλές παραμέτρους τοπικών μοντέλων δημιουργώντας μια μεγάλη αρχιτεκτονική Federated Learning. Επιταχύνει την βελτίωση των μοντέλων Machine Learning και των τοπικών Deep Learning functions. Παρολαυτά, η επιλογή των clients λαμβάνει χώρα με τυχαίο τρόπο και χωρίς να υπακούει σε συγκεκριμένο κανόνα και όρους το οποίο σημαίνει ότι δεν βελτιώνεται πολύ η ακρίβεια προγνώσεων ούτε αντιμετωπίζεται η ύπαρξη πιθανών ανωμαλιών στα δεδομένα σε κάποιον πχ. client.

Σύμφωνα με την εργασία του Huang et al. [112], προτείνεται ένα Federated Learning σύστημα για διαχείριση ετερογενών δεδομένων σε ένα κατανεμημένο νεφουπολογιστικό σύστημα με ένα πολύ αποδοτικό τρόπο. Ο αλγόριθμος που χρησιμοποιείται στο Federated Learning σύστημα είναι ο LoAdaBoost ο οποίος αυξάνει την ακρίβεια στις προγνώσεις. Παρολαυτά, η κρίσιμη παράμετρος που αφορά την σύγκλιση του αλγορίθμου δεν είναι τόσο αποδοτική. Επίσης το κλασσικό πρόβλημα με την περιορισμένη διαθεσιμότητα πόρων αλλά και η διαδικασία επιλογής clients επίσης είναι παράμετροι που δεν εξετάζονται σε αυτή την εργασία. Από την άλλη μεριά, η εργασία των Li et al. [113] περιλαμβάνει ένα σύστημα βελτιστοποίησης Federated Learning αρχιτεκτονικής το οποίο επιλύει θέματα ετερογένειας δεδομένων σε διαμοιρασμένα (federated) δίκτυα. Αυτό που λείπει στην συγκεκριμένη εργασία είναι η θεώρηση και επίλυση των ασυνεχειών και ανωμαλιών στα δεδομένα των κόμβων (clients) που επηρεάζουν αρνητικά την ακρίβεια στην διαδικασία πρόγνωσης.

Τελικά, και εστιάζοντας στην τεχνολογία Federated Learning, η εργασία των Wang et al. [99] παρουσιάζει ένα gradient-descent based Federated Learning σύστημα με έναν προσαρμοζόμενο αλγόριθμο ελέγχου που υπολογίζει την καλύτερη συχνότητα με βάση την οποία λαμβάνουν χώρα περιοδικά τοπικά και συνολικά aggregation λειτουργίες ώστε να ευρεθεί η ελάχιστη συνολική τιμή της γενικής συνάρτησης απώλειας και η βέλτιστη χρήση των αντίστοιχων πορών. Στην συγκεκριμένη εργασία μιλάμε για την παράμετρο του χρόνου. Τα αποτελέσματα δείχνουν αρκετά καλή ακρίβεια προγνώσεων όσον αφορά τους μελλοντικούς πόρους που χρειάζονται για την λειτουργία μιας εφαρμογής, αλλά δεν λαμβάνουν υπόψιν τιμές των τοπικών συναρτήσεων απωλειών (deep learning local loss

functions) αλλά ούτε θεωρούν περιπτώσεις επιλογής κόμβων (clients) όταν θέλουμε να αποφύγουμε data abnormalities, λύσεις που σίγουρα θα μπορούσαν να αυξήσουν την ακρίβεια προγνώσεων κάποιων μετρικών στο cloud πχ. CPU consumption σε διακομιστές (servers) σε Multi-Cloud περιβάλλοντα.

Σε αντίθεση με τις σχετικές εργασίες state-of-the-art όπως παρουσιάστηκαν, η εργασία στην παρούσα Διδακτορική Διατριβή αντιμετωπίζει τόσο τα θέματα δυναμικού υπολογισμού της συχνότητας του global aggregation σε ένα federated learning σύστημα με δεδομένο χρονικό περιορισμό πόρων όσο και της βελτιώσης της ακρίβειας πρόγνωσης εξαιρώντας δεδομένα που εμφανίζουν abnormalities. Αυτή η βελτίωση λαμβάνει χώρα με τους ακόλουθους 2 τρόπους:

- Χρήση τοπικής συνάντησης απώλειας βαθιάς μηχανικής μάθησης σε κάθε Federated Learning κόμβο λαμβάνοντας υπόψη τις data abnormalities σε κάθε τοπικό κόμβο κατά τη διάρκεια του global aggregaton.
- Θεωρώντας επαρκή όγκο δεδομένων κατά τη διάρκεια της διαδικασίας επιλογής κόμβων στη φάση της global aggregation διαδικασίας του συστήματος Multicloud Federated Learning.

Επιπλέον, η εργασία της παρούσας Διδακτορικής Διατριβής[A5] αναπτύσσει μια καινοτόμο προσέγγιση στον τρόπο που οι κατανομές των δεδομένων μπορούν να λάβουν χώρα όσον αφορά τους διάφορους κόμβους του Federated Learning cluster. Επί της ουσίας γίνεται ένας συνδυασμός σεναρίων iid και non-iid κατανομής δεδομένων. Τελικά τοπικές συναρτήσεις βαθιάς μηχανικής μάθησης χρησιμοποιούνται σε κάθε συμμετέχοντα κόμβο γεγονός το οποίο δίνει ένα πολύ καλό αποτέλεσμα σύγκλισης σε σύγκριση με άλλες συναρτήσεις [A5]. Μια βασική σύγκριση των ανωτέρω παρουσιασθέντα ερευνητικών εργασιών και των χαρακτηριστικώς τους παρουσιάζεται στον ακόλουθο πίνακα 10:

ΕΡΓΑΣΙΑ	***Σύγχρονη /Ασύγχρονη επικοινωνία***	***Συνάρτηση Απωλειών Convex***	***Συνάρτηση Απωλειών Non-Convex***	***Συνάρτηση Απωλειών Deep Learning***	***Aggregation βασισμένο σε σταθμισμένο (Weighted) όγκο δεδομένων ανά συμμετέχοντα κόμβο***	***Επιλογή συμμετέχοντα κόμβου αναλόγως του τοπικού όγκου δεδομένων***	***Επιλογή συμμετέχοντα κόμβου αναλόγως της τιμής της τοπικής συνάρτησης απωλειών***	***Αλγόριθμος ελέγχου για προσαρμογή χρονικού παραθύρου του global aggregation***
Asynchronous Online Federated Learning [103]	***Ασύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***
Asynchronous Federated Optimization [104]	***Ασύγχρονη***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***
A Federated Learning for Mobile Edge Computing Networks [93]	***Σύγχρονη***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***
Privacy preserving traffic flow prediction: A federated learning approach [105]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***
Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing [106]	***Ασύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***
A Selective Model Aggregation Approach in Federated Learning for Online Anomaly Detection [107]	***Σύγχρονη***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΟΧΙ***
LAG: Lazily Aggregated Gradient for Communication Efficient Distributed Learning [19]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	-	***ΟΧΙ***	***ΟΧΙ***	***ΝΑΙ***	***ΟΧΙ***
Abnormal Client Behavior Detection in Federated Learning [20]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***
Evaluation of Federated Learning Aggregation Algorithms	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***	***ΟΧΙ***

Application to Human Activity Recognition [21]								
Federated Learning with Personalization Layers [22]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***
EdgeFed: Optimized Federated Learning Based on Edge Computing 2020 [23]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***
Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge[24]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***
LoAdaBoost: loss based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data[25]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***
Federated Optimization in Heterogeneous Networks [26]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΌΧΙ***
Adaptive Federated Learning in Resource Constrained Edge Computing Systems [6]	***Σύγχρονη***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΝΑΙ***	***ΌΧΙ***	***ΌΧΙ***	***ΝΑΙ***
Τρέχουσα Εργασία Διδακτορικής Διατριβής[Α5]	***Σύγχρονη***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***	***ΝΑΙ***

Πίνακας 10: Σύγκριση διδακτορικής εργασίας με άλλες ερευνητικές εργασίες

6.3 Ερευνητική Προσέγγιση και Αρχιτεκτονική Αλγορίθμου

6.3.1 Γενική Προσέγγιση

Στα πλαίσια της εργασίας του διδακτορικού, αναπτύχθηκε ένας αλγόριθμος και με βάση την αναφορά της εργασίας των (Wang et al.) [99] έχει ως στόχο την επίλυση με χρήση μηχανικής μάθησης του θέματος της ελαχιστοποίησης της γενικής συνάρτησης απωλειών του Federated Learning συστήματος που περιλαμβάνει νεφο-υπολογιστικούς κόμβους αλλά και κόμβους ευρισκόμενους στο άκρο "edge":

$$w^* \triangleq argminF(w) \ (6)$$

Στην παραπάνω συνάρτηση w^* είναι η διανυσματική παράμετρος του γενικού μοντέλου όταν φτάνει το ελάχιστο. F(w) είναι η γενική (καθολική) συνάρτηση απωλειών που προκύπτει ως συνισταμένη από όλα τα τοπικά δεδομένα και σχετικές συναρτήσεις απωλειών των κατανεμημένων client κόμβων (mulit-cloud nodes /edge nodes):

$$F(w) = \frac{\sum_{i=1}^{N} D_i F_i(w)}{D} \ (7)$$

Στην παραπάνω συνάρτηση N είναι η παράμετρος που προσδιορίζει τον αριθμό των client κόμβων, D_i είναι η παράμετρος που προσδιορίζει το μέγεθος των τοπικών δεδομένων του κάθε client και D είναι η παράμετρος που προσδιορίζει το συνολικό μέγεθος των δεδομένων όλου του συστήματος Federated Learning (edge και multi-cloud) και το οποίο δίδεται με την ακόλουθη εξίσωση:

$$D \triangleq \sum_{i=1}^{N} D_i \quad (8)$$

Επίσης η παράμετρος $F_i(w)$ αναπαριστά την τοπική συνάρτηση απώλειας (loss function) του εκάστοτε client και δίνει την απόκλιση λάθους του τοπικού εκπαιδευμένου μοντέλου με βάση τα τοπικά δεδομένα. Έτσι κάθε client κόμβος εκπαιδευόμενος αναπτύσει τις τοπικές παραμέτρους του μοντέλου και οι οποίες παράμετροι αποθηκεύονται στον πίνακα $w_i(t)$ όπου t=0,1,2,… αντιπροσωπεύει τον δείκτη ολοκλήρωσης του μοντέλου και επανάληψης εκμάθησης. Στην στιγμή t=0 ο aggregator αρχικοποιεί την επικοινωνία προς όλους τους clients και στέλνει τα αρχικά βάρη ή παραμέτρους του μοντέλου w(0) μαζί με το αρχικό χρονικό βήμα $\tau^* = 1$ που ορίζει το κάθε πότε συμβαίνει ένα global aggregation. Με αυτό τον τρόπο οι τοπικές παράμετροι όλων των κόμβων (clients) που συμμετέχουν στην αρχιτεκτονική Federated Learning αρχικοποιούνται όλες στην ίδια τιμή πριν ξεκινήσει η διαδικασία. Μετά από μια ή περισσότερες τοπικές ολοκληρώσεις του τοπικού αλγόριθμου μια global aggregation διαδικασία λαμβάνει χώρα στον κεντρικό aggregator server όπως έχουμε προαναφέρει μόνο αν ο χρόνος εκτέλεσης του global aggregation είναι πολλαπλάσιο βήμα του βασικού χρονικού βήματος δηλαδή του τ^* δηλαδή χρονική στιγμή t=Kτ^* όπου το K=ακέραιος αριθμός.

Για την χρονική στιγμή t>0, νέες τιμές των τοπικών παραμέτρων των μοντέλων $w_i(t)$ υπολογίζονται με βάση τον αλγόριθμο Gradient Descent που τρέχει τοπικά και γίνεται update η τιμή της τοπικής loss συνάρτησης για κάθε κόμβο (client) i. Για κάθε τοπικό update του τοπικού αλγορίθμου χρησιμοποιεί σαν αρχική τιμή της τρέχουσας ολοκλήρωσης (current iteration t) την τιμή της προηγούμενης χρονικά ολοκλήρωσης t-1:

$$w_i(t) = \widetilde{w_i}\,(\mathrm{t}-1) - \mathrm{n}\,\nabla F_i\,(\widetilde{w_i}\,(t-1)) \ (9)$$

όπου n είναι ο βαθμός μάθησης. Η διαδικασία global aggregation συλλέγει πολλαπλές αλλαγές που συμβαίνουν στα τοπικά μοντέλα των client nodes ώστε τελικά να βελτιώνεται το FL μοντέλο. Ήδη έχει αναφερθεί ότι το σύστημα εκτελεί βήματα χρονικής διάρκειας τ* των local updates που λαμβάνουν χώρα σε κάθε client node μεταξύ δύο global aggregations. Μετά από κάθε global aggregation η τοπική παράμετρος $w_i(t)$ σε κάθε client node συνήθως αλλάζει. Πιο συγκεκριμένα, χρησιμοποιούμε την αναπαράσταση $\widetilde{w_i}$(t) σε κάθε node client i όταν αλλάζει μετά από global aggregation. Εάν κανένα aggregation δεν έχει λάβει χώρα την χρονική στιγμή ολοκλήρωσης t, τότε έχουμε $\widetilde{w_i}$(t) = $w_i(t)$. Εάν global aggregation έχει λάβει χώρα την χρονική στιγμή ολοκλήρωσης t τότε $\widetilde{w_i(t)} \neq w_i(t)$ και γι αυτό το λόγο θέτουμε $\widetilde{w_i(t)} = w_i(t)$, όπου ο συμβολισμός $w_i(t)$ είναι ο μέσος όρος των βαρών σύμφωνα με το μέγεθος των δεδομένων που βρίσκονται σε κάθε κόμβο:

$$w(t) = \frac{\sum_{i=1}^{N} D_i w_i(t)}{D} \quad (10)$$

Η συγκεκριμένη διδακτορική διατριβή κινείται σε σχετικό θέμα με αυτό που έχει παρουσιαστεί στην εργασία των (Wang et al.) [99] όπου ένα σύστημα Federated Learning με χρήση Gradient Descent τεχνικής σε συνδυασμό με ένα αλγόριθμο που κάνει προσαρμογή του έλεγχου των πόρων, έχει υλοποιηθεί. Εν σχέση με την εργασία των (Wang et al.) [99], η παρούσα διατριβή παρουσιάζει μια καινοτόμο μεθοδολογία επιλογής εκείνων των client nodes-κόμβων που θα συμμετέχουν τελικά σε κάθε global aggregation. Η προτεινόμενη μεθοδολογία επιλογής clients είναι βασισμένη σε ένα δυναμικά προσαρμοζόμενο κατώφλι που προκύπτει από την τιμή του μεγέθους των τοπικών δεδομένων σε κάθε client και από ένα δυναμικά προσαρμοζόμενο κατώφλι σχετικό με κατάλληλες τιμές τοπικών συναρτήσεων απωλειών από κάθε client. Τόσο το προσαρμοζόμενο κατώφλι των μεγεθών δεδομένων όσο και το προαναφερθέν προσαρμοζόμενο κατώφλι σχετικά με τις τιμές των συναρτήσεων απωλειών υπολογίζονται από την μέση τιμή των τοπικών (client) μεγεθών των δεδομένων και των τοπικών (client) συναρτήσεων απωλειών αντίστοιχα μείον την τιμή της τυπικής απόκλισης για κάθε περίπτωση. Προκειμένου λοιπόν σε ένα global aggregation να βρέθει η βέλτιστη τιμή του κατωφλίου τόσο για το επαρκές μέγεθος δεδομένων που πρέπει να έχει ένας client όσο και της τιμής της τοπικής συνάρτησης απωλειών, δύο βασικές συνθήκες λαμβάνονται υπόψιν:

1. Το μέγεθος δεδομένων κάθε κόμβου client θα πρέπει να είναι μεγαλύτερο από ένα δυναμικά προσαρμοζόμενο κατώφλι μεγέθους δεδομένων όπως αναφέρθηκε πριν ώστε να μπορούν οι clients που συμμετέχουν στο global aggregation να συνεισφέρουν με αποτελεσματικό τρόπο.
2. Η αρκούντος μικρή τιμή κάθε τοπικής συνάρτησης απωλειών για κάθε client και η οποία θα πρέπει να είναι μικρότερη από το κατώφλι που συζητήθηκε πριν για τις τοπικές συναρτήσεις απωλειών ώστε να αποφευχθεί η παρουσία «ανώμαλων» δεδομένων (data abnormalities) ή καταστροφικού θορύβου στην διαδικασία εκμάθησης.

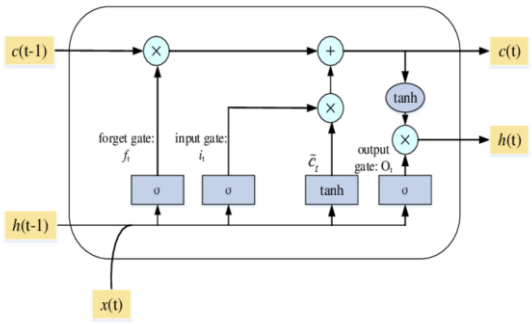
Επίσης στην παρούσα διατριβή, Αλγόριθμοι βαθιάς μηχανικής μάθησης και συγκεκριμένα LSTM Algorithms [120] με τις αντίστοιχες συναρτήσεις απωλειών εφαρμόζονται στις τοπικές επαναλήψεις-ολοκληρώσεις των αλγορίθμων ως νέος τύπος τοπικών συναρτήσεων απωλειών σε σχέση με την προηγούμενη εργασία των (Wang et al.) [99]. Το γεγονός αυτό επίσης κατατείνει και αυξάνει ακόμα περισσότερο την ακρίβεια προγνώσεων από το σύστημα MulticloudFL που έχει τώρα αναπτυχθεί.

Το σημαντικό των δικτύων μακρών βραχείας μνήμης (LSTM – Long Short Term Memory Networks) είναι ότι αν και παραλλαγή των RNN δικτύων δηλαδή των δικτύων με αντροφοδότηση, χρησιμοποιούνται θεωρώντας ότι τα δεδομένα εισόδου δεν μπορεί να είναι πλήρως ανεξάρτητα μεταξύ τους. Προτάθηκαν από τους Hochreiter και Schmdhuber το 1997 για την αντμετώπιση του προβλήματος της εξασθενούσης παραγώγου που εμπόδιζε την εκπαίδευση στα RNN. Αυτό το είδος δικτύων έμελε να γίνει το κυριάρχο μοντέλο προβλέψεων με πλήθος εφαρμογών αργότερα.

Τα δίκτυα LSTM, σε μακροσκοπικό επίπεδο, δεν έχουν ουσιαστικές αρχιτεκτονικές διαφορές από τα RNN όπως αναφέρθηκαν. Είναι αναδρομικά, έχουν εσωτερική κατάσταση και υπάρχει και σε αυτά η έννοια του ξεδιπλώματος της λειτουργίας τους στο χρόνο. Αυτό που είναι διαφορετικό είναι ο τρόπος με τον οποίο υπολογίζεται η εσωτερική τους κατάσταση.

Ένα δίκτυο LSTM εμπεριέχει υποσυστήματα (νευρωνικά δίκτυα) που ονομάζονται πύλες (gates) και ελέγχουν το ποιες πληροφορίες θα προστίθενται η θα αφαιρούνται στην εσωτερική κατάσταση του δικτύου, ή πιο απλά τι θα θυμάται και τι θα ξεχνά το δίκτυο. Κάνοντας την μνήμη του δικτύου πιο επιλεκτική, καθίσταται εφικτό να θυμάται το δίκτυο και μακροχρόνιες συσχετίσεις. Ένας νευρώνας LSTM δέχεται στην είσοδό του ένα διάνυσμα μιας ακολουθίας, συλλέγει πληροφορία για αυτό και προωθεί την πληροφορία προς τα εμπρός. Διαθέτει ένα διάνυσμα μνήμης, το cell και μαζί με τα δείγματα εισόδου δέχεται και ένα διάνυσμα κρυφής κατάστασης. Η ροή των δεδομένων στο εσωτερικό του νευρώνα ρυθμίζεται από τρεις πύλες, την πύλη εισόδου (input gate), την πύλη εξόδου (output gate) και την forget gate. Κάθε πύλη διαθέτει εκπαιδεύσιμα βάρη σε μορφή πινάκων, που ρυθμίζουν τη συμπεριφορά της.

Μια βασική δομή εσωτερική μιας μονάδας δικτύου LSTM φαίνεται στο ακόλουθο σχήμα:



Σχήμα 25 – Εσωτερική δομή μιας μονάδας (cell) δικτύου LSTM

Στο ανωτέρω σχήμα:

X = element-wise πολλαπλασιασμός

+ = element-wise πρόσθεση

σ = επίπεδο NN λογιστικής συνάρτησης

tanh = επίπεδο NN συνάρτησης tanh – element wise υπολογισμός

Βασική έννοια στα δίκτυα LSTM είναι η εσωτερική κατάσταση της μονάδας (cell state) c. Αυτή παίζει το ρόλο μνήμης και επιτρέπει σε πληροφορία από παλαιότερους κύκλους λειτουργίας

να διατηρηθεί και σε πιο πρόσφατους περιορίζοντας έτσι το πρόβλημα της βραχυχρόνιας μνήμης. Καθώς προχωρούν οι κύκλοι στον αναδρομικό τρόπο λειτουργίας του δικτύου LSTM, πληροφορίες προστίθενται και αφαιρούνται στο cell state δια μέσου των πυλών και προωθούνται παρακάτω στην ροή επεξεργασίας. Κάθε πύλη είναι ένα επίπεδο νευρωνικού δικτύου με λογιστική συνάρτηση ενεργοποίησης ακολουθούμενο από element-wise πολλαπλασιασμό δλδ. πολλαπλασιάζονται τα αντίστοιχης θέσης στοιχεία των διανυσμάτων. Καθώς η έξοδος τέτοιων νευρώνων έχει τιμή μεταξύ 0 και 1, ο element-wise πολλαπλασιασμός της εξόδου μιας πύλης με κάποιο διάνυσμα, έστω u (ίδιας διάστασης), ελέγχει τη διέλευση της πληροφορίας του διανύσματος u. Τιμές 0 στην έξοδο το νευρωνικού δικτύου καταστέλλουν(μηδενίζουν) την πληροφορία αντίστοιχης θέσης στο διάνυσμα u ενώ τιμές ίσες με 1 διατηρούν την τιμή και άρα επιτρέπουν την διέλευση της. Ενδιάμεσες τιμές επιτρέπουν τη διέλευση σε κάποιο βαθμό, δηλαδή αδυνατίζοντας λίγο ή πολύ τις αρχικές τιμές.

Στη συνέχεια παρατίθενται οι εσωτερικές δομές μιας μονάδας LSTM καθώς και οι υπολογισμοί του ορθού περάσματος (forward pass). Οι σχέσεις που παρατίθενται χρησιμοποιούν την συμβολογραφία που υπάρχει στο σχήμα 25. Τονίζεται ότι τα στοιχεία LSTM, εκτός από εσωτερική κατάσταση c, έχουν και μια κρυφή κατάσταση h (hidden state) που επίσης μεταφέρουν στους διαδοχικούς κύκλους επεξεργασίας ως έξοδο (πρόβλεψη) του LSTM. Η κρυφή κατάσταση h παρέχει επιπλέον ευελιξία καθώς αποσυνδέει την εσωτερική κατάσταση c από το τι είναι επιθμητό να παραχθεί ως έξοδος h του δικτύου.

Με βάση το σχήμα 25α οι λειτουργίες των τριών πυλών, σε λεπτομέρεια, είναι οι εξής:

- **Πύλη εισόδου:** από την πύλη εισόδου διέρχεται το δείγμα x(t) και ένα διάνυσμα κρυφής κατάστασης h(t) και αποφασίζεται σε τι βαθμό θα χρησιμοποιηθούν για να τροποποιήσουν το cell. Διαθέτει μια σιγμοειδή συνάρτηση, η οποία αποφασίζει αν τα διανύσματα θα διέλθουν από την πύλη. Επιπλέον, αυτά διέρχονται από μια συνάρτηση υπερβολικής εφαπτομένης που αποφασίζει τη βαρύτητά τους στο διάστημα (−1, 1):

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{11}$$

$$\widetilde{C_t} = \tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{12}$$

- **Forget gate:** η πύλη αυτή αποφασίζει αν θα χαθεί η πληροφορία στη μνήμη του cell ή όχι. Όπως η πύλη εισόδου, έτσι και η forget gate ορίζεται από μια σιγμοειδή συνάρτηση:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{13}$$

  Ο συνδυασμός των τιμών της πύλης εισόδου και της forget gate χρησιμοποιείται για ανανέωση της μνήμης του cell:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \tag{14}$$

- **Πύλη εξόδου:** η πύλη εξόδου είναι αυτή που αποφασίζει αν η πληροφορία εισόδου θα είναι ορατή στην έξοδο. Όπως και στις άλλες δύο πύλες, έτσι και εδώ, αυτό ρυθμίζεται μέσω μιας σιγμοειδούς συνάρτησης. Το αποτέλεσμα

της πύλης εξόδου συνδυάζεται με την πληροφορία του cell ώστε να παραχθεί η νέα κρυφή κατάσταση στην έξοδο του νευρώνα.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad (15)$$

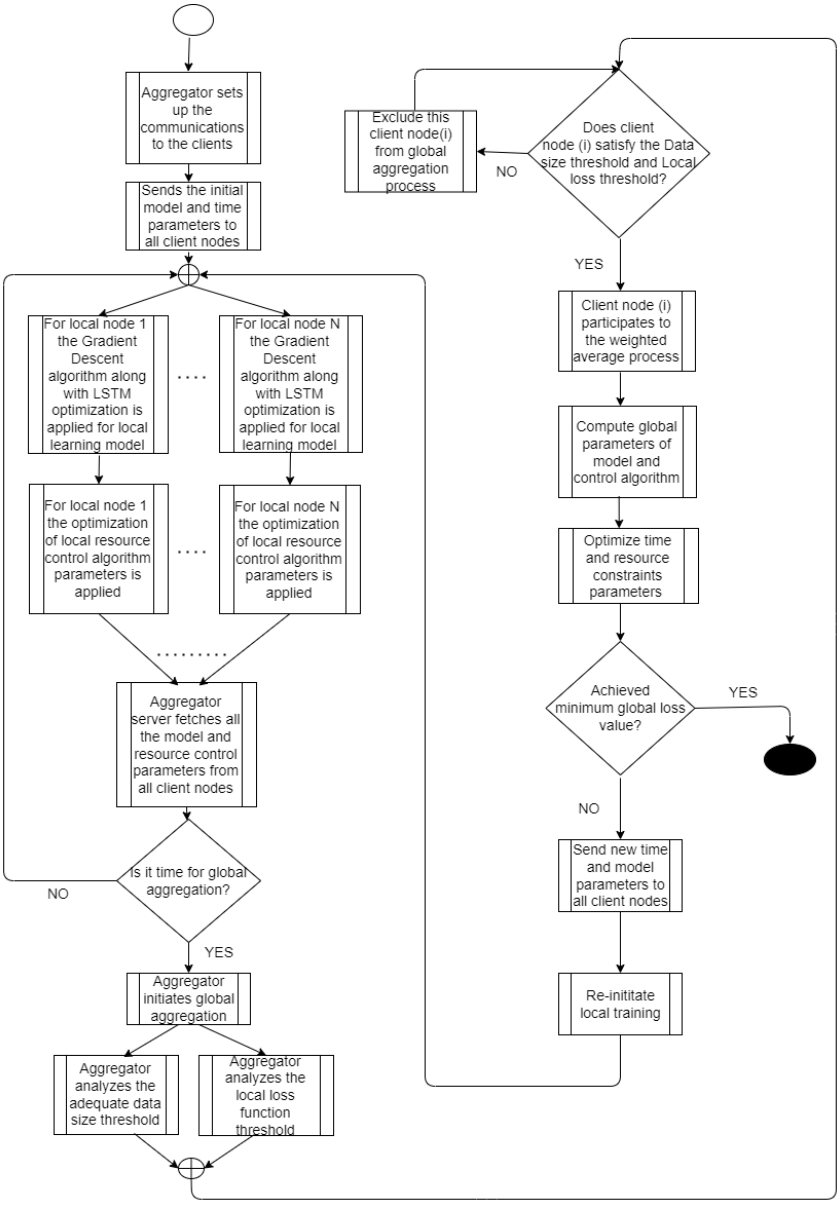
$$h_t = o_t * \tanh(C_t) \quad (16)$$

Συνοψίζοντας, η πύλη λήθης αποφασίζει τι είναι χρήσιμο να κρατηθεί στο cell state από τους προηγούμενους κύκλους λειτουργίας, ενώ η πύλη εισόδου τι πληροφορία να προστεθεί στο cell state από τον τρέχοντα κύκλο. Η πύλη εξόδου ελέγχει τι από το cell state θα βγει ως πρόβλεψη του τρέχοντος κύκλου. Λόγω της ιδιότητας να θυμάται εισόδους για αυθαίρετο χρόνο, το LSTM μας επιτρέπει να διαχειριστούμε ακολουθιακά δεδομένα, χωρίς η ανατροφοδότηση να είναι ”πολωμένη” στα γειτονικά διανύσματα μιας ακολουθίας.

Επιπλέον, στο σύστημα MulticloudFL που έχουμε αναπτύξει θεωρούμε όχι πανομοιότυπο τύπο δεδομένων από client node σε client node δηλαδή όχι πανομοιότυπη κατανομή δεδομένων από client σε client καθώς μια τέτοια περίπτωση δεν μπορεί να εκφράσει και πραγματικό σενάριο. Συνεπώς το μοντέλο ππυ εκπαιδεύεται σε κάθε client είναι βασισμένο σε non-independent and identically distributed (non i.i.d) δεδομένα και η σύγκλιση επέρχεται με κατάλληλη διαμόρφωση του step size εκπαίδευσής του.

Επίσης, μια βασική λειτουργία του συστήματος MulticloudFL που έχει αναπτυχθεί στην παρούσα διατριβή είναι το γεγονός ότι δεδομένου ενός συγκεκριμένου προβλήματος περιορισμών σε πόρους πχ. μέγιστος διαθέσιμος χρόνος στον οποίο το FL σύστημα θα πρέπει να έχει ολοκληρώσει την εκμάθηση, θα πρέπει να βρει τις βέλτιστες τιμές τ^* and T=K*τ^* , όπως έχει ήδη αναφερθεί, ώστε η συνολική συνάρτηση απώλειας του συστήματος Multicloud FL να ελαχιστιποιηθεί στο τέλος στην μικρότερη δυνατή τιμή (εξίσωση 2). Γι αυτό τον λόγο του χρονικού περιορισμού ένας αλγόριθμος ελέγχου επίσης εφαρμόζεται. Ο χρόνος βήματος τ^* για κάθε τοπική ολοκλήρωση και οι παράμετροι του αλγορίθμου ελέγχου βελτιστοποιούνται αρχικά τοπικά σε κάθε κόμβο client και τελικά μετά από κάθε συνολική διαδικασία global aggregation. Γενικά και ως μέρος της όλης διαδικασίας FL το κρίσιμο σημείο είναι η εξέταση του εάν ή όχι η ελάχιστη τιμή της συνολικής συνάρτησης απώλειας του συστήματος έχει επιτευχθεί. Εάν όχι, τότε οι νέες συνολικές παράμετροι του μοντέλου και το νέο χρονικά βήμα τ^* στέλνεται πίσω στους client κόμβους, η τοπική εκμάθηση στους κόμβους επανεκιννεί και η όλη διαδικασία συνολικής εκμάθησης επαναλαμβάνεται. Στο συγκεκριμένο σύστημα αυτό που πρέπει να αναφερθεί είναι ότι είναι πλήρως επεκτάσιμο και δεν υπάρχει θεωρητικά περιορισμός στον αριθμό των συμμετεχόντων client nodes. Φυσικά με την βοήθεια του control algorithm όπως προαναφέρθηκε γίνεται κάθε φορά βελτιστοποίηση στον χρόνο εκτέλεσης του αλγορίθμου με τους διαφορετικούς αριθμούς συμμετεχόντων client nodes.

Στο ακόλουθο σχήμα παρουσιάζεται ένα διάγραμμα ροής όπου παρουσιάζονται τα βασικά βήματα του MulticloudFL συστήματος αλγορίθμων.



Σχήμα 26 – Βασικά βήματα του MulticloudFL συστήματος αλγορίθμων

6.3.2 Τεχνική Επιλογής των Clients-Κόμβων στον Αλγόριθμο Adaptive Federated Learning

Στο σύστημα Federated Learning που έχει ήδη αναφερθεί στις προηγούμενες παραγράφους, μια καινοτόμος μέθοδος παρουσιάζεται στην συνέχεια με στόχο την βελτίωση της ακρίβειας προγνώσεων. Σε προηγούμενη εργασία όπως αυτή των (Wang et al.) [99], (Nishio and Yonetani) [111] έχει γίνει χρήση μιας μεθόδου μέσης τιμής στα βάρη των τοπικών μοντέλων από τους clients κατά την διαδικασία του global aggregation όλων των συνδεδεμένων clients (που μπορεί να είναι είτε clients πέραν του ενός νεφουπολογιστικού παρόχου ή κόμβοι στο «άκρο» του νέφους). Η διεθνής βιβλιογραφία όπως αναφέρθηκε σε προηγούμενη παράγραφο δεν παρέχει κάποια λύση ή υλοποίηση που να αντιμετωπίζει με επάρκεια χαμηλές ποιότητες τοπικών δεδομένων δηλαδή abnormalities σε clients που έχουν αρκετό θόρυβο, ούτε φαίνεται να υπάρχει λύση που να αντιμετωπίζει με αποτελεσματικότητα την ύπαρξη μικρής ποσότητας τοπικών δεδομένων σε client μη ικανών να εκπαιδεύσουν σωστά τους σχετικούς τοπικούς αλλά και γενικούς αλγορίθμους Federated Learning. Με την υλοποίηση που παρουσιάζεται στην παρούσα διδακτορική διατριβή με το νέο σύστημα Multi-cloud Federated Learning τα δύο προαναφερθέντα προβλήματα αντιμετωπίζονται επαρκώς με πολύ καλά αποτελέσματα όπως θα φανεί στην συνέχεια.

Ειδικότερα, το πρώτο τμήμα της τεχνικής επιλογής Clients του αλγορίθμου που παρουσιάζεται στην παρούσα διδακτορική διατριβή τρέχει στον aggregator server και συμπεριλαμβάνει την διαδικασία εύρεσης των δεδομένων που φιλοξενούνται σε κάθε client ξεχωριστά καθώς και την εύρεση των τιμών των σχετικών συναρτήσεων απωλειών πριν η διαδικασία του global aggregation λάβει χώρα. Με το επιτυχές πέρας της διαδικασίας εύρεσης, μια προσαρμοζόμενη τιμή κατωφλίου υπολογίζεται καθώς έχει λάβει υπόψιν την ύπαρξη της συγκεντρωθήσας πληροφορίας με όλες τις πιθανές ανωμαλίες στα δεδομένα με την έννοια της ύπαρξης σημείων δεδομένων εξαιρετικά πέρα και πάνω από τα συνήθη. Αυτού του είδους τα εξωπραγματικά δεδομένα καλούνται “outliers” [121] και οφείλονται είτε σε μεγάλη μεταβλητότητα κάποιας αναξιόπιστης μέτρησης είτε σε κάποιο λάθος πειραματικής μέτρησης και όχι κατ’ ανάγκην σε κάποια συνθήκη παραβίασης κάποιας κατάστασης. Σε κάθε περίπτωση αυτού του είδους τα δεδομένα θα πρέπει να εξαιρούνται από την διαδικασία μάθησης γιατί μπορούν τα συγκεκριμένα να οδηγήσουν σε ανακριβή τελικά μοντέλα πρόβλεψης.

Για τους παραπάνω λόγους, υιοθετούμε μια προσαρμοζόμενη τιμή κατωφλίου (thr) για ένα κατανεμημένο σετ δεδομένων, απαλλαγμένο από εξωπραγματικά δεδομένα ή αλλιώς “outliers” που μπορεί να υπάρχουν είτε στα τοπικά δεδομένα είτε να εμφανίζονται ως αποτέλεσμα σε τοπικές συναρτήσεις απωλειών στους clients, οριζόμενο (το κατώφλι) ως ακολούθως:

- Για την περίπτωση του μεγέθους των τοπικών δεδομένων ($thr_{DataSize}$):

$$thr_{DataSize} = \begin{cases} MEAN_{(Total\ Data\ Size)} - 0.5 * std_{(Total\ Data\ Size)}, \\ if\ MEAN_{(Total\ Data\ Size)} < std_{(Total\ Data\ Size)} \\ MEAN_{(Total\ Data\ Size)} - 1.0 * std_{(Total\ Data\ Size)}, \\ if\ MEAN_{(Total\ Data\ Size)} > std_{(Total\ Data\ Size)} \end{cases} \quad (17)$$

- Για την περίπτωση της τοπικής συνάρτησης απώλειας ($thr_{Loss\ Value}$):

$$thr_{Loss\,Value} = \begin{cases} MEAN_{(local\ loss\ value)} - 0.5 * std_{(local\ loss\ value)}, \\ \quad if\ MEAN_{(local\ loss\ value)} < std_{(local\ loss\ value)} \\ MEAN_{(local\ loss\ value)} - 1.0 * std_{(local\ loss\ value)}, \\ \quad if\ MEAN_{(local\ loss\ value)} mean\ value \\ \qquad\qquad of\ local\ loss > std_{(local\ loss\ value)} \end{cases} (18)$$

Όπου χρησιμοποιούμε την τυπική απόκλιση στις εξισώσεις μας (*std*).

Σαν δεύτερο βασικό τμήμα του client selection αλγορίθμου, αποτελεί η διαδικασία συνολικής συλλογής και επεξεργασίας των τοπικών παραμέτρων των επιμέρους clients ή αλλιώς global aggregation με τελικό αποτέλεσμα την εύρεση σταθμισμένου μέσου όρου των συνολικών παραμέτρων όλου του τελικού μοντέλου Federated Learning. Σε αυτή την διαδικασία, μόνο οι clients που τόσο το μέγεθος των δεδομένων τους ικανοποιεί το ανωτέρω κατώφλι μεγέθους τοπικών δεδομένων και την ίδια στιγμή που η τοπική τους συνάρτηση απωλειών είναι μικρότερη από το ανωτέρω ορισμένο κατώφλι λάθους, μπορούν να λάβουν μέρος στο global aggregation στον κεντρικό server. Με αυτό τον τρόπο διασφαλίζουμε ότι επαρκείς ποσότητες δεδομένων σε κάθε client συνεισφέρουν ενισχυτικά στην μαθησιακή διαδικασία του Federated Learning χωρίς να προσθέτουν θόρυβο ή παράλογη συμπεριφορά που θα μπορούσαν να οδηγήσουν σε λανθασμένα ή ανακριβή αποτελέσματα προγνώσεων.

Η διαδικασία που μόλις περιεγράφηκε παρουσιάζεται στην ακόλουθη εξίσωση και αναφέρεται λεπτομερειακά υπό μορφή ψευδό-κώδικα στο ακόλουθο σχήμα 25.

$$w(t) = \begin{cases} \dfrac{\sum_{i\varepsilon\{1,..,N\}} D_i w_i}{\sum_{i\varepsilon\{1,..,N\}} D_i}, data\ size_{(i)} > thr_{DataSize} \\ \quad and\ loss_{(i)} < thr_{Loss\,Value} \\ w(t)\ ,\ \ data\ size_{(i)} < thr_{DataSize}\ or\ \ loss_{(i)} > thr_{Loss\,Value} \end{cases} (19)$$

Συνεπώς, στην εξίσωση 11 θεωρούμε ότι το σύστημα Federated Learning έχει N clients nodes: 1,2,…,N με τοπικά datasets για καθένα από αυτούς τους κόμβους. D_i απεικονίζει το μέγεθος των δεδομένων για κάθε τοπικό dataset για κάθε client node i όπου θα πρέπει να ικανοποιεί και τις δύο συνθήκες για το μέγεθος των τοπικών δεδομένων και την τοπική συνάρτηση απωλειών. Επίσης, το διάνυσμα w_i συμβολίζει τις τοπικές εκπαιδευμένες παραμέτρους του μοντέλου για τον client node i που συμμετέχει στο global aggregation δίνοντας τελικά το σταθμισμένο συνολικό global model με τις αντίστοιχες τελικές παραμέτρους.

Σχετικά με τα ανωτέρω κατώφλια λάθους και ακριβώς επειδή αυτά προέρχονται από τις μέσες τιμές των συμμετεχόντων client nodes του συστήματος FL cluster, η πιθανότητα να μην ικανοποιείται καμμιά από τις ανωτέρω συνθήκες για μέγεθος δεδομένων ή τιμή της τοπικής συνάρτησης λάθους δεν μπορεί να ικανοποιηθεί. Τουλάχιστον ένας κόμβος (client node) θα συμμετέχει σε κάθε global aggregation step.

Algorithm 1 Client Selection FL Algorithm

Client Selection FL Algorithm

Require: Local minibatch size B, number of client participants m
per global aggregation, number of local iterations τ, learning rate n,
N total number of client nodes

Ensure: Global model w(t)

[Participant i]
receive w global model parameter for i participant from Server
Local Training (i,w):
Split local dataset Di to minibatches of size B minimum
  For each local epoch j from 1 to τ DO:
    For each b ε B DO:
      wi <-- w - n*ΔL(w;b) (n is the learning rate and ΔL is the gradient
                of local Loss on b.)
    end For
  end For
send the wi model parameter to Server

[Server]
Initialize w(0):
  For each iteration τ from 1 to T(=κ*τ) DO:
  Receive from Client Participants all wi local model parameters
    For each Client Participant (1,..,N) DO in parallel:
      wi(t+1) <-- LocalTraining (i,w(t))
    end For
    Discover from all Client participants N :
        - Data Size Threshold T1
        - Local Loss Threshold T2
    Choose a subset of S of m participants from N when
    the 2 conditions are achieved:
    participant Data Size > T1 and Loss value <T2

    For each participant i ε S (1,...,m):
      Calculate the Average Weighted Aggregation
      based on Data Size of each client (Equation (8))
    end For
  end For
send to Client Participants the new w global model parameter values

Σχήμα 27 – Περιγραφή Αλγορίθμου επιλογής Clients

6.3.3 Τεχνική Ελέγχου Κατανάλωσης Πόρων στον Αλγόριθμο Adaptive Federated Learning

Σε αυτό το κεφάλαιο παρουσιάζεται το έτερο μεγάλο τμήμα του Αλγορίθμου που αναπτύξαμε στα πλαίσια της Διδακτορικής Διατριβής και λέγεται Adaptive Federated Learning όπου αφορά τον έλεγχο της χρήσης των υπολογιστικών πόρων (αλγόριθμος ελέγχου) και συγκεκριμένα περιγράφει τα βήματα που ακολουθούνται για τον επανυπολογισμό του χρόνου τ* (time window) μετά από κάθε εκτέλεση global aggregation βασισμένου στις παραμέτρους από την τελευταία κατάσταση του συστήματος.

Ειδικότερα, το σημαντικό τμήμα στον Αλγόριθμο είναι η βέλιστη χρήση των διαθέσιμων πόρων με στόχο την ελαχιστοποίησητων σφαλμάτων πρόβλεψης και έτσι της συνάρτησης απωλειών κατά την διάρκεια της διαδικασίας του Federated Learning. Σαν επακόλουθο, και με βάση τον κατανεμημένο κανόνα της επικλινής καθόδου που χρησιμοποιείται στην περίπτωσή μας [122], το πρόβλημα καταλήγει σε θέμα βελτιστοποίησης των τιμών Τ και τ και συνεπώς τ*. Η βελτιστοποίηση αυτών των παραμέτρων καταφέρνουν να ελαχιστοποιήσουν την τιμή της τελικής συνολικής (global) συνάρτησης απωλειών του συστήματος δεδομένων των περιορισμών στους πόρους. Ορίζεται η συνάρτηση που συσχετίζει το Τ με το τ σαν Τ=Κ*τ και βοηθά στην διαδικασία της ελαχιστοποίησης των τοπικών και συνολικών συναρτήσεων απωλειών.

Γενικά, θεωρούμε στον αλγόριθμό μας Μ διαφορετικούς τύπους πόρων που περιλαμβάνουν περιπτώσεις όπως παραμέτρους : χρόνο, εύρος ζώνης επικοινωνίας, ενέργεια κτλ. Για κάθε τύπο πόρου m: m∈{1,2,..., M} και για κάθε τοπικό βήμα ανανέωσης των παραμέτρων λόγω εκπαίδευσης, όλοι οι κόμβοι του Federated Learning συστήματος αποτελούμενου από πολλούς παρόχους καταναλώνουν c_m μονάδες υπολογιστικού πόρου m τύπου και μετά από κάθε βήμα συνολικού (global) aggregation καταναλώνονται b_m μονάδες υπολογιστικού πόρου τύπου m όπου και ισχύει $c_m > 0$ και b_m > 0 (finite numbers). Για δεδομένες παραμέτρους Τ και τ οι συνολικά καταναλισκώμενοι πόροι δίδονται από τον ακόλουθο μαθηματικό τύπο:

$$\text{Total}_{\text{ResConsumption}} = (T+1)\, c_m + (K+1)\, b_m \quad (20)$$

Για την εξίσωση (20), είναι σημαντικό να σημειώσουμε ότι κατά την διάρκεια της διαδικασίας Federated Learning σε κάθε client node i πρώτα υπολογίζεται η τιμή της τοπικής συνάρτησης απώλειας και εν συνεχεία στέλνονται τα αποτελέσματα από όλους τους κόμβους στον server aggregator προκειμένου να υπολογιστεί η σταθμισμένη τιμή της συνολικής συνάρτησης απώλειας. Καθώς κάθε client node γνωρίζει την Kth συνολική παράμετρο του μοντέλου μόνο μετά την Kth συνολική ολοκλήρωση (iteration & global aggregation) καθότι οι συνολικές καθολικές παράμετροι του μοντέλου είναι γνωστές σε κάθε κόμβο μόνο μετα την επιτυχή ολοκλήρωση του global aggregation στον κεντρικό aggregator server, η τοπική τιμή της τοπικής συνάρτησης απώλειας στον Kth γύρο θα σταλεί στον aggregator server την χρονική στιγμή Κ+1 του βήματος aggregation και ακολούθως αφού υπολογιστεί η Κ -th συνολική τιμή απώλειας της συνολικής συνάρτησης απώλειας (global loss function) θα σταλεί πίσω σε όλους τους client nodes. Προκειμένου να υπολογιστεί η τελευταία συνολική τιμή της συνάρτησης απώλειας (Τ=Κ*τ) μια επιπρόσθετη τοπική και συνολική ολοκλήρωση

(επανάληψη) χρειάζεται να λάβει χώρα στο τέλος όλης της διαδικασίας εκμάθησης και αυτός είναι ο λόγος που στην εξίσωση (20) εμφανίζονται οι παράμετροι (T+1) & (K+1).

Θεωρώντας ότι R_m είναι η συνολική κατανάλωση πόρων τύπου m κατά την διαδικασία εκπαίδευσης του όλου συστήματος Federated Learning, ο αλγόριθμος ελέγχου προσπαθεί να βρει μια βέλτιστη ελάχιστη λύση με βάση το κατωτέρο:

$$min_{\tau,K\in\{1,2,3,\dots\}}^{Global\ Loss\ Function}$$

$$\text{with the condition: (T+1) } c_m + \text{(K+1) } b_m \leq R_m,$$

$$\forall m \in \{1, \dots., M\} \quad (21)$$

Μιλώντας γενικότερα, είναι πολύ δύσκολο να χρησιμοποιηθεί μια αναλυτική έκφραση που να συσχετίζει το μέγεθος τ με το K και την επίδρασή τους στην συνολική συνάρτηση απωλειών όλου του συστήματος Federated Learning πολλών νεφουπολογιστικών παρόχων. Αυτή η δυσκολία μπορεί να εστιαστεί στην συσχέτιση της ιδιότητας σύγκλισης της επικλινής καθόδου [122] με την επίδραση την ίδια στιγμή της συχνότητας εκτελέσεως global aggregation σητην ιδιότητα συγκλίσεως. Οι παράμετροι C_m και b_m μπορούν να είναι χρονικά μεταβαλλόμενοι. Γι αυτό τον λόγο προτείνεται ο προαναφερθείς αλγόριθμος ελέγχου Federated Learning.

Σύμφωνα με θεωρητική ανάλυση και ορισμούς που έχουν παρουσιαστεί σε σχετική εργασία (Wang et al.) [99], ο βασικός στόχος του αλγορίθμου ελέγχου είναι η ελαχιστοποίηση της ακόλουθης έκφρασης:

$$G(\tau) \triangleq \frac{max_m \frac{c_m\tau + b_m}{R'_m\tau}}{2\eta\varphi} + \sqrt{\frac{\left(max_m \frac{c_m\tau + b_m}{R'_m\tau}\right)^2}{4n^2\varphi^2} + \frac{\rho h(\tau)}{n\varphi\tau} + \rho h(\tau)} \quad (22)$$

Όπου $R'_m = R_m - b_m - c_m$ και R_m είναι η συνολική κατανάλωση πόρων τύπου m. Επιπλέον, n είναι ο ρυθμός μάθησης, φ είναι η σταθερά που έχει οριστεί σαν παράμετρος ελέγχου στο Lemma 2 της εργασίας των (Wang et al.) [99], h(τ) είναι η συνάρτηση που εξηγείται στην εργασία των (Wang et al.) [99] σαν η διαφορά μεταξύ των παραμέτρων του μοντέλου που έχουν προκύψει μεταξύ κατανεμημένων και κεντρικοποιημένων επικλινών καθόδων, ρ είναι η παράμετρος Lipschitz που έχει ήδη οριστεί από την εργασία των (Wang et al.) [99] και επίσης β είναι η παράμετρος Smoothness που χρησιμοποιείται και αυτή από την εργασία των (Wang et al.) [99].

Βασισμένοι στην παραπάνω έκφραση (22), η ελαχιστοποίηση του G(τ) χρησιμοποιώντας μια βέλτιστη τιμή του τ (η οποία αντιπροσωπεύει τον αριθμό των επαναλήψεων εκμάθησης των τοπικών αλγορίθμων σε κάθε κόμβο που εκτελούνται μεταξύ δύο global aggregations) ορίζεται ως τ* και δίνεται από την ακόλουθη εξίσωση:

$$\tau^* = \arg min_{\tau \in \{1,2,3,\dots\}} \; G(\tau) \quad (23)$$

και μετά την βέλτιστη εύρεση της παραπάνω παραμέτρου τ*, υπολογίζεται ο βέλτιστος αριθμός βημάτων "global aggregation" ως K* με βάση την ακόλουθη εξίσωση:

$$K^* = \; min_m \; \frac{R'_m}{c_m \tau^* + b_m} \quad (24)$$

Η έκφραση του G(τ) έχει παραμέτρους που σχετίζονται με την κατανάλωση πόρων όπως οι $c_m \; and \; b_m \; (για \; οποιοδήποτε \; τύπο \; δεδομένων)$ και άλλες παραμέτρους όπως οι ρ, β και δ που σχετίζονται με τα χαρακτηριστικά των συναρτήσεων απωλειών. Αυτός ο υπολογισμός λαμβάνει χώρα σε πραγματικό χρόνο κατά την διάρκεια της διαδικασίας εκπαίδευσης Federated Learning.

Οι τιμές του $c_m \; and \; b_m$ προκύπτουν από μετρήσεις των μετρικών των πόρων που έχουν να κάνουν με καταναλώσεις στην μεριά των client nodes και του aggregator αντίστοιχα. Στην παρούσα διδακτορική διατριβή ο πόρος που εξετάζεται όπως ειπώθηκε είναι ο χρόνος, όπου ο μέγιστος χρόνος ανά τοπική εκτέλεση του τοπικού αλγορίθμου σε κάθε node και συνολικά θεωρείται ως c_m. Επιπλέον, για aggregation υπάρχει και η παράμερος τύπου m (i.e. time) όπου η σχετική κατανάλωση υποδηλώνεται από b_m για κάθε πόρο και συγκρίνεται με την συνολική κατανάλωση πόρων εν σχέση με το resource budget R_m . Με άλλα λόγια, $c_m \; και \; b_m$ ανταποκρίνονται στον πραγματικό χρόνο που χρειάζεται για κάθε τοπικό update και για κάθε global aggregation. Εάν οι καταναλισκόμενοι πόροι προσεγγίζουν το όριο (budget limit) ενός συγκεκριμένου τύπου πόρου m (στην περίπτωσή μας είναι ο χρόνος) τότε σταματά η διαδικασία εκμάθησης του αλγορίθμου και επιστρέφεται το τελικό αποτέλεσμα.

Οι τιμές των παραμέτρων ρ,β,δ υπολογίζονται από τις τοπικές και ολικές συναρτήσεις απωλειών και τις επικλινείς καθοδικές μεθόδους υπολογισμένες τόσο σε κάθε client χωριστά όσο και συνολικά στον aggregator server κατά την διάρκεια της διαδικασίας Federated Learning. Για την επάρκεια αυτού του υπολογισμού, κάθε κόμβος client χρειάζεται να έχει πρόσβαση τόσο στις τοπικές παραμέτρους όσο και στις σχετιζόμενες συνολικές παραμέτρους του global μοντέλου («same iteration at timestamp t») που αναπτύσεται στον aggregator κάθε φορά μετα από εκτέλεση ενός global aggregation στην ολοκλήρωση της χρονικής στιγμής t. Επειδή οι συνολικές παράμετροι του μοντέλου είναι γνωστές σε κάθε κόμβο client αφού η διαδικασία του global aggregation ολοκληρωθεί κάθε φορά, οι υπολογισμένες τιμές ρ,β,δ είναι διαθέσιμες για επανυπολογισμό τ* ξεκινώντας από το δεύτερο global aggregation βήμα μετά την αρχικοποιήση των παραμέτρων στην εκκίνηση της διαδικασίας. Με άλλα λόγια χρησιμοποιούνται τιμές από το προηγούμενο βήμα του global aggregation. Αυτή η περιγραφή είναι εμφανής στην περιγραφή του αλγορίθμου ελέγχου Κατανάλωσης Πόρων στον Αλγόριθμο Adaptive Federated Learning στο σχήμα 26.

Επιπλέον, κατά την διάρκεια του επανυπολογισμού στο aggregation μετά από κάθε καθολικό βήμα aggregation, αναζητούναι νέες τιμές του τ* με άνω όριο το γ threshold επί τόσες φορές το τ* . Τελικά βρίσκεται εκείνη η τιμή του τ* που όπως έχει ήδη αναφερθεί ελαχιστοποιεί το G(τ) με γ>0 μια συγκεκριμένη παράμετρος. Η τιμή του γ με άλλα λόγια είναι σεταρισμένη με βάση την δική μας επιλογή. Ο ρόλος της παραμέτρου γ είναι να περιορίζει τον χώρο αναζήτησης και επίσης να αποφέυγεται η περίπτωση η τιμή της παραμέτρου τ* να αυξάνει πολύ γρήγορα και να προκαλεί έτσι ανακρίβειες στην τελική πρόγνωση. Επιπλέον, μια μέγιστη τιμή της τ* χρησιμοποιείται γιατί εάν το τ* φτάσει πολύ μεγάλες τιμές θα παραβιάσει το συνολικό επιτρέπομενο budget πόρων και επακόλουθα θα μειώσει την αποδοτικότητα του

συστήματος. Η νέα υπολογισμένη τιμή τ* στέλνεται πίσω σε κάθε κόμβο μετά από ένα global aggregation μαζί με τις αντίστοιχες παραμέτρους του global μοντέλου που έχουν υπολογιστεί. Τα βασικά βήματα του αλγορίθμου ελέγχου Κατανάλωσης Πόρων στον Αλγόριθμο Adaptive Federated Learning παρουσιάζονται στο σχήμα 26.

Είναι σημαντικό να παρατηρήσουμε ότι οι μέχρι τώρα Αλγόριθμοι που παρουσιάστηκαν σχετικά με την εκπαίδευση του Federated Learning συστήματος πλεονεκτούν σε διάφορα σημεία έναντι άλλων εργασιών όπως πχ. των (Wang et al.) [99]. Τα σημεία που πλεονεκτούν εδράζονται κυρίως στην μέθοδο επιλογής clients κατά την διάρκεια της διαδικασίας εκμάθησης του Federated Learning καθώς και στην μέθοδο Regressive (δλδ συνεχών μεταβλητών μεγέθους ως προς τον χρόνο) Deep Learning για την ελαχιστοποίηση της τοπικής συνάρτησης απωλειών. Επίσης, ο αλγόριθμος ελέγχου όταν εκπαιδεύεται λαμβάνει υπόψιν του μόνο τις παραμέτρους των πόρων των clients που έχουν επιλεγεί για το global aggregation. Πιο αναλυτικά, ο aggregator server αφού επλέξει τους κατάλληλους clients που θα συμμετέχουν στο global aggregation, υπολογίζει τις καθολικές τιμές των παραμέτρων ρ,β,δ σαν μέση τιμή μόνο των επιλεγέντων clients. Αυτές οι νέες τιμές είναι που συνεισφέρουν τελικά στον επανυπολογισμό του τ* σύμφωνα με την εξίσωση (15). Αυτές οι καινοτόμες μέθοδοι εισάγουν υπολογισμούς που τελικά οδηγούν σε ακριβέστερα αποτελέσματα προγνώσεων εν σχέση με προηγούμενες εργασίες όπως των (Wang et al.) [99].

Τα βασικά βήματα που έχουν ήδη αναλυθεί, παρουσιάζονται αναλυτικά στο ακόλουθο σχήμα 28 και αφορούν τον αλγόριθμο Ελέγχου Κατανάλωσης Πόρων κατά την διαδικασία του Adaptive Federated Learning:

Algorithm 2 Resources Control Algorithm

Control Algorithm

Require: Resource budget R, control parameter φ,
search range parameter γ, maximum τ value τmax

Define: global model parameter vector w(t)
local model parameter vector wi(t)
local resource parameter ci
local resource parameter bi

[Server]
Initialize $\tau^* <-- 1$;
Initialize t <-- 0 ;
Initialize s <-- 0 //s resource counter;
Initialize w(0) as a constant;
Initialize w(global) <--- w(0);
REPEAT:
send w(t) parameters and τ^* to all client nodes;

save iteration index of last transmission of w(t): t0<-t ;

next global aggregation is after τ iterations : $t <-- t + \tau^*$;

Receive wi(t), ci from each client node ;
compute the weighted average with client selection of w(t) global model;

Receive ϱi, βi, Local Loss function, Anadelta of Local Loss fuction from each client selected
node i;

Estimate weighted average ϱ based on Data Size of each client selected node i;

Estimate weighted average β based on Data Size of each client selected node i;
Compute Anadelta of loss function and from difference of Anadeltas of local
and global loss functions estimate finally the weighted average δ ; [see ref [6]]

Compute new value of τ* according to Equation (9) via linear search on integer values of τ
within [1, min {γτ*;τmax};

Estimate resource consumptions cm, bm using cι received from all nodes ι and local parameters at the aggregator;
sm <-- sm +cm * τ+bm;
IF sm>Rm then decrease τ* to the maximum possible value such that the estimated resource consumption for remaining iterations is within Resource budget Rm.

[Participant]

Receive w(t) and new τ* from Server;

Estimate local ϱi,βi

Estimate type-m (time) resource consumption cm,i for one local update at node i;

Send all parameters wi(t) (updated) , ϱi, βi, local loss function of wi(t) , anadelta of loss function of wi(t), to Server

Σχήμα 28 – Περιγραφή Αλγορίθμου Adaptive Federated Learning (Control)

6.4 Αρχιτεκτονική Υλοποίησης

Όπως έχει γενικά αναφερθεί, το σύστημα που υλοποιήθηκε στην παρούσα διδακτορική διατριβή αφορά ένα αποκεντρικοποιημένο σύστημα με διάφορους client nodes ευρισκόμενοι σε διάφορους νεφουπολογιστικούς παρόχους όπου υλοποιούν συνεργατικά ένα Federated Learning model (multi-cloud & edge περιβάλλον). Ένας απομακρυσμένος aggregator server τοποθετημένος σε κάποιον από τους νεφο-υπολογιστικούς παρόχους χρησιμοποιείται επίσης. Το όλο σύστημα επιτελεί τόσο διαδικασίες εκμάθησης τοπικών αλγορίθμων σε κάθε client node ξεχωριστά όσο και απομακρυσμένες ενέργειες με το να στέλνονται σε ένα κεντρικό server βάρη τοπικά εκπαιδευμένων αλγορίθμων με τελικό στόχο το global aggregation, ώστε τελικά να υποστεί εκμάθηση (training) το Federated Learning σύστημα. Η τελική πρόγνωση του συστήματος πάνω σε δεδομένα αξιολόγησης δηλαδή πραγματικά ή με άλλα λόγια το Inference κομμάτι του αλγορίθμου μπορεί να εκτελείται σε κάποιο άκρο του υπολογιστικού νέφους ή αλλιώς edge server or device.

Όπως έχει ήδη παρουσιαστεί στην γραφική παράσταση σχήμα 24 με τα βασικά βήματα του MulticloudFL συστήματος αλγορίθμων, στο MulticloudFL η διαδικασία εκμάθησης περιλαμβάνει βήματα ανανέωσης των τοπικών παραμέτρων και βαρών των μοντέλων, όπου ο κάθε τοπικός client node κόμβος εκτελεί gradient descent εκμάθηση για να προσαρμόσει τις τοπικές παραμέτρους του μοντέλου. Όπως έχει ήδη αναφερθεί στην εξίσωση (7) όπου παρουσιάζεται ο τρόπος με τον οποίο μεταβάλλονται οι παράμετροι (τα βάρη) από ένα γύρο εκπαίδευσης στον επόμενο γύρο εκπαίδευσης, η τιμή του βήματος εκμάθησης ή ρυθμός μάθησης (learning rate) ή αλλιώς step size παίζει καθοριστικό ρόλο στην εκπαίδευση και επηρεάζει την ταχύτητα εκπαίδευσης. Συνήθως συνδυάζεται με έναν επιπλέον όρο στον υπολογισμό της μεταβολής των βαρών που είναι ανάλογος της μεταβολής στον προηγούμενο κύκλο εκπαίδευσης (Δwn). Ο συντελεστής αναλογίας ονομάζεται ορμή (momentum) και παρουσία της ορμής μ, η σχέση μεταβολής των βαρών γίνεται:

$$\Delta w_{N+1} = \Delta w + \mu * \Delta w_N \qquad (25)$$

όπου Δw είναι η μεταβολή των βαρών που υπολογίζονται από κύκλο σε κύκλο εκπαίδευσης (epoch) και $\mu * \Delta w_N$ είναι ο παράγοντας της ορμής που κρατά την κατεύθυνση της επικλινούς καθόδου περισσότερο σταθερή κάτι χρήσιμο σε πολύπλοκες επιφάνειες σφάλματος με πολλά τοπικά ελάχιστα. Συμπερασματικά, η ορμή δεν είναι κάτι απαραίτητο αλλά υπό προϋποθέσεις μπορεί να συμβάλλει θετικά στην εκπαίδευση ενός τεχνητού νευρωνικού δικτύου (ΤΝΔ).

Μεγάλες τιμές του ρυθμού μάθησης (learning rate) επιταχύνουν τη σύγκλιση προς το ελάχιστο σφάλμα αλλά σε συνδυασμό με μεγάλη ορμή αυξάνουν και τον κίνδυνο να προσπεραστεί το ολικό ελάχιστο Ε με πιθανό αποτέλεσμα την παλινδρόμηση ή αλλιώς ταλάντωση γύρω από τις βέλτιστες τιμές βαρών. Αντίθετα, μικρές τιμές του d αποτρέπουν το φαινόμενο της παλινδρόμησης αλλά απαιτούν περισσότερο χρόνο εκπαίδευσης. Η ιδανική τιμή για το ρυθμό μάθησης (learning rate) και την ορμή μ, εφόσον χρησιμοποιείται, καθορίζεται μέσω πειραματισμού.

Ο ορισμός της τιμής του ρυθμού μάθησης(learning rate) γίνεται στο configuration αρχείο της εφαρμογής του συστήματος MulticloudFL μέσω της παραμέτρου step size. Αυτή η τιμή προκύπτει από πολλές δοκιμές μέχρι να βρεθεί η βέλτιστη τιμή που θα εξασφαλίζει το

μικρότερο σφάλμα πρόβλεψης και την καλύτερη σύγκλιση. Η εκπαίδευση των αλγορίθμων τρέχει με επαναλαμβανόμενους κύκλους που στην γλώσσα των αλγορίθμων και της τεχνητής νοημοσύνης λέγεται epoch. Σε κάθε epoch διαβάζονται όλα τα δεδομένα εκπαίδευσης με διάφορους τρόπους (τυχαία ή συγκεκριμένα mini-batches κτλ.) και γίνεται η ανανέωση των τιμών των παραμέτρων του μοντέλου. Ο ρυθμός μάθησης είναι σημαντικό να επιλεγεί με τον πιο σωστό τρόπο ούτως ώστε να μην επηρεάσει την διαδικασία εκπαίδευσης του αλγορίθμου. Τοπικά σε κάθε κόμβο τρέχει μια διαδικασία gradient descent για να ελαχιστοποιεί τα λάθη σε κάθε κύκλο εκπαίδευσης (training epoch) και να ανανεώνει πιο σωστά τις παραμέτρους του μοντέλου.

Ο ρυθμός εκπαίδευσης (learning rate) έχει ένα μεγάλο αποτέλεσμα στην εκπαίδευση νευρωνικών δικτύων και κατανεμημένων (federated) αλγορίθμων. Ένας σωστός ρυθμός εκπαίδευσης μπορεί να επιταχύνει τον χρόνο εκπαίδευσης γενικά όπως ήδη αναφέρθηκε, να δώσει την ευκαιρία στον αλγόριθμο να ψάχνει επίπεδες επιφάνειες καλύτερα και να προσπερνά δυσκολίες τοπικών ελαχίστων ή μεγίστων που προκαλούν συναρτήσεις ενεργοποίησης non-convex τύπου. Επίσης υπάρχει και η επιλογή της χρήσης μεταβαλλόμενου ρυθμού εκπαίδευσης όπου μπορούμε να αλλάζουμε την τιμή ανάλογα του momentum του αλγορίθμου Federated Learning.

Στην υλοποίηση της παρούσας Διδακτορικής Διατριβής, έλαβε χώρα η δημιουργία ενός τοπικού μοντέλου εκμάθησης που συνοδεύθηκε από την χρήση της μεθόδου βελτιστοποίησης Adam [135]. Adam, που συνοδεύεται από τα αρχικά «adaptive moment estimation», είναι η τελευταία εξέλιξη της αρχιτεκτονικής των αλγορίθμων όπου παρουσιάζονται προσαρμοσμένες τιμές ρυθμών μάθησης ανά βάρος στο μοντέλο. Η συγκεκριμένη τεχνική ενσωματώνει ιδέες από τους αλγορίθμους AdaGrad, RMSProp και την έννοια του momentum [136,137]. Όπως ακριβώς οι αλγόριθμοι AdaGrad και RMSProp, ο αλγόριθμος Adam παρέχει εξατομικευμένους ρυθμούς μάθησης ανά βάρος ή αλλιώς παράμετρο μοντέλου. Έχει σχεδιαστεί να εξυπηρετεί την εκπαίδευση βαθιών νευρωνικών δικτύων, χρησιμοποιεί λιγότερο ποσοστό μνήμης και μικρότερο ποσοστό επεξεργαστικής ισχύος σε σχέση με τους άλλους δύο προαναφερθέντες αλγορίθμους. Η Adam χρησιμοποιεί το πρώτο μέσο όρο του RMSProp optimizer σαν αρχή και στην συνέχεια τον εκθετικό κινούμενο μέσο όρο της τετραγωνικής ρίζας του gradient αλγορίθμου. Με άλλα λόγια, ο μηχανισμός του Adam χρησιμοποιείται για να υπολογίσει τη προσαρμοζόμενη γραμμική παλινδρόμηση για κάθε παράμετρο του μοντέλου.

Το πέρασμα από το Τεχνητό Νευρωνικό Δίκτυο (ΤΝΔ) όλων των δεδομένων υπό μορφή διανυσμάτων εκπαίδευσης μια φορά, συνιστά ένα κύκλο εκπαίδευσης ή αλλιώς εποχή (epoch). Οι μεταβολές που προκαλούνται στα βάρη και τις πολώσεις σε μια εποχή μπορεί να υπολογιστούν με δύο βασικούς τρόπους. Στον πρώτο τρόπο χρησιμοποιείται η μάθηση δέσμης (batch learning) όπου το δίκτυο δέχεται σαν είσοδο, ένα-ένα, όλα τα διανύσματα εκπαίδευσης, συσσωρεύει (αθροίζει) τη μεταβολή στην τιμή των βαρών που προκύπτει από κάθε διάνυσμα και αναπροσαρμόζει τα βάρη στο τέλος κάθε εποχής, χρησιμοποιώντας τη συσσωρευμένη (accumulated) μεταβολή. Στον δεύτερο τρόπο έχουμε την επαυξητική μάθηση (incremental ή online learning) όπου η αναπροσαρμογή των βαρών γίνεται αμέσως μετά την χρήση ενός από τα διανύσματα εκπαίδευσης. Μια ενδιάμεση και προτιμητέα προσέγγιση στη χρήση δεδομένων εκπαίδευσης είναι η mini-batch. Σύμφωνα με αυτή το σύνολο των δεδομένων εκπαίδευσης χωρίζεται σε μικρότερες ομάδες ίδιου μεγέθους (32 εγγραφών, 64 εγγραφών, 128 εγγραφών κτλ) και σε κάθε μια από αυτές εφαρμόζεται η μάθηση δέσμης. Η mini-batch προσέγγιση είναι ιδιαίτερα δημοφιλής στην εκπαίδευση

βαθέων ΤΝΔ για πολύπλοκα προβλήματα, καθώς εκεί ο όγκος των δεδομένων εκπαίδευσης είναι συνήθως πολύ μεγάλος.

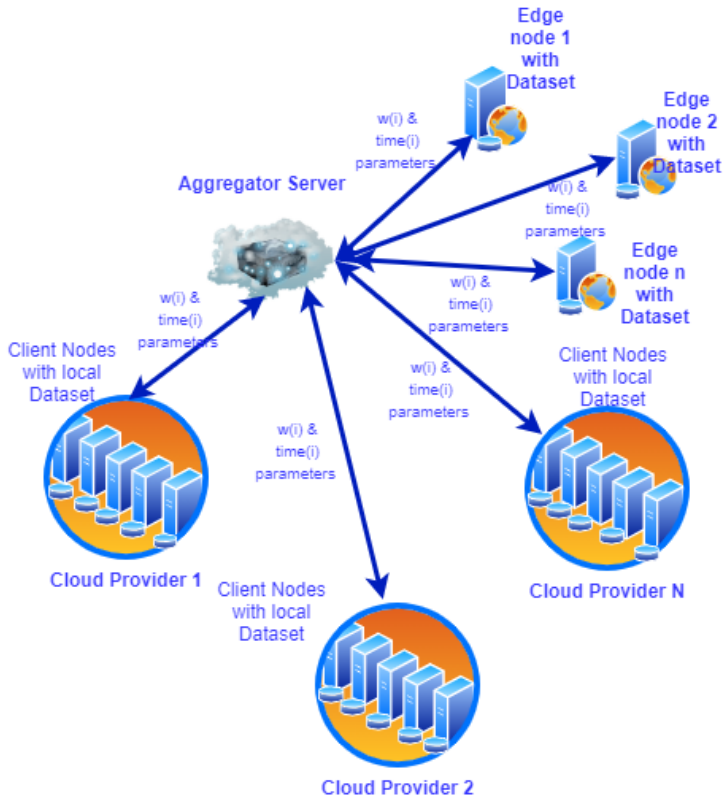
Ακριβώς για τον παραπάνω λόγο, είναι απαγορευτικό να υπολογίζεται ο gradient descent αλγόριθμος της τοπικής συνάρτησης κόστους (απωλειών) σε όλο το εύρος των δεδομένων εκπαίδευσης (training data) που ευρίσκονται στον κάθε τοπικό κόμβο (client). Σε περιπτώσεις όπως η παρούσα διδακτορική διατριβή που περιλαμβάνει την εκπαίδευση του συστήματος Federated Learning, η μεθοδολογία του στοχαστικού gradient descent χρησιμοποιείται όσον αφορά το διάβασμα των δεδομένων [138,139]. Σε αυτή την μέθοδο, χρησιμοποιείται ο υπολογισμός του gradient με βάση την συνάρτηση απωλειών σε ένα τμήμα των δεδομένων εκπαίδευσης επιλεγμένο με τυχαίο τρόπο «randomly sampled dataset» επί του συνόλου των δεδομένων το οποίο καλείται mini-batch. Κάθε mini-batch μπορεί να θεωρηθεί ένα υποσύνολο δεδομένων του συνόλου των δεδομένων εκπαίδευσης χωρίς να υπάρχει αλληλοεπικάλυψη μεταξύ των mini-batches. Κάθε τοπικό βήμα επανάληψης (iteration step) αντιπροσωπεύει ένα βήμα υπολογισμού gradient descent όπου το gradient υπολογίζεται σε ένα mini-batch των τοπικών δεδομένων εκπαίδευσης. Το είδος των δεδομένων για κάθε mini-batch αλλάζει από βήμα σε βήμα (step iteration), ένα νέο mini-batch δεδομένων δεδομένου μεγέθους επιλέγεται κάθε φορά τυχαία χωρίς να είναι ίδιο με το προηγούμενο.

Κατά την διάρκεια της εκπαίδευσης τοπικών μοντέλων σε κάθε κόμβο (client) μια μέθοδος ελάττωσης του ρίσκου να δημιουργηθεί υπερ-προσαρμογή καθώς και επιτάχυνσης της διαδικασίας εκπαίδευσης εφαρμόζεται στο MulticloudFL συστήματος αυτής της διατριβής. Αυτή η μέθοδος ονομάζεται «dropout» [140] ή αλλιώς προσωρινή απόρριψη νευρώνων. Στη συγκεκριμένη μέθοδο, επιλέγουμε τυχαία κάποιους νευρώνες και τα αντίστοιχα βάρη τους και μηδενίζουμε τις τιμές τους. Με αυτό τον τρόπο κάνουμε να μην συνεισφέρουν καθόλου κατά την διάρκεια της εκπαίδευσης με μέθοδο forward pass ή back propagation. Με άλλα λόγια, ένα ποσοστό νευρώνων στα κρυφά επίπεδα ή στο επίπεδο εισόδου «αποσυνδέονται» από το ΤΝΔ για ένα κύκλο εκπαίδευσης και οι υπολογισμοί σε αυτό τον κύκλο γίνονται χωρίς να λαμβάνονται υπόψη αυτοί οι νευρώνες όπως και τα βάρη που καταλήγουν σε ή ξεκινούν από αυτούς. Κατά κάποιο τρόπο, «πιέζεται» το υπόλοιπο δίκτυο να εξισορροπήσει την απώλειά τους και αυτό βοηθά στην αποφυγή παγίδευσης σε τοπικά ελάχιστα της συνάρτησης κόστους και υπερ-προσαρμογής. Μόλις γίνει η διόρθωση βαρών των νευρώνων που εξακολουθούν να συμμετέχουν, οι «αποσυνδεδεμένοι» νευρώνες επανασυνδέονται και κάποιοι άλλοι παίρνουν την θέση τους για τον επόμενο κύκλο εκπαίδευσης. Το ποσοστό dropout αποτελεί μία από τις υπέρ-παραμέτρους της εκπαίδευσης ΤΝΔ. Συνήθως η πιθανότητα να αφαιρεθεί ένας κόμβος δεν υπερβαίνει το 0.5 για τα κρυφά επίπεδα Αντίθετα, όταν τρέχουμε την διαδικασία προγνώσεων σε δεδομένα τεστ (αφού έχει ολοκληρωθεί η εκπαίδευση του δικτύου), το πλήρες δίκτυο με όλους τους νευρώνες και όλα τα βάρη χρησιμοποιείται.

Σύμφωνα με την παραπάνω διαδικασία, οι ανανεώσεις των τοπικών παραμέτρων των μοντέλων, ελαχιστοποιούν την τιμή της συνάρτησης απωλειών ή κόστους όπως λέγεται οριζόμενη σύμφωνα με το περιεχόμενο των δεδομένων του τοπικού dataset στον κάθε client κόμβο. Η διαδικασία του federated learning συνοδεύεται επίσης γενικά-global aggregation βήματα κάθε φορά που ολοκληρώνεται η τοπική εκπαίδευση όλων των παραμέτρων σε κάθε κόμβο-client. Αυτές οι τοπικές παράμετροι των κόμβων στέλνονται σε ένα server που λέγεται aggregator mode στο cloud. Εκεί λαμβάνει χώρα μια διαδικασία επιλογής των κατάλληλων client κόμβων ώστε να επιτευχθεί η βέλτιστη ακρίβεια πρόγνωσης λαμβάνοντας τελικά τον μέσο όρο των παραμέτρων των μοντέλων με βάση το μέγεθος των τοπικών δεδομένων του

κάθε client-κόμβου. Μετά την ολοκλήρωση της διαδικασίας aggregation, οι ανανεωμένες παράμεροι στέλνονται πίσω σε όλους τους κόμβους-client που ανήκουν στο σύστημα Federated Learning.

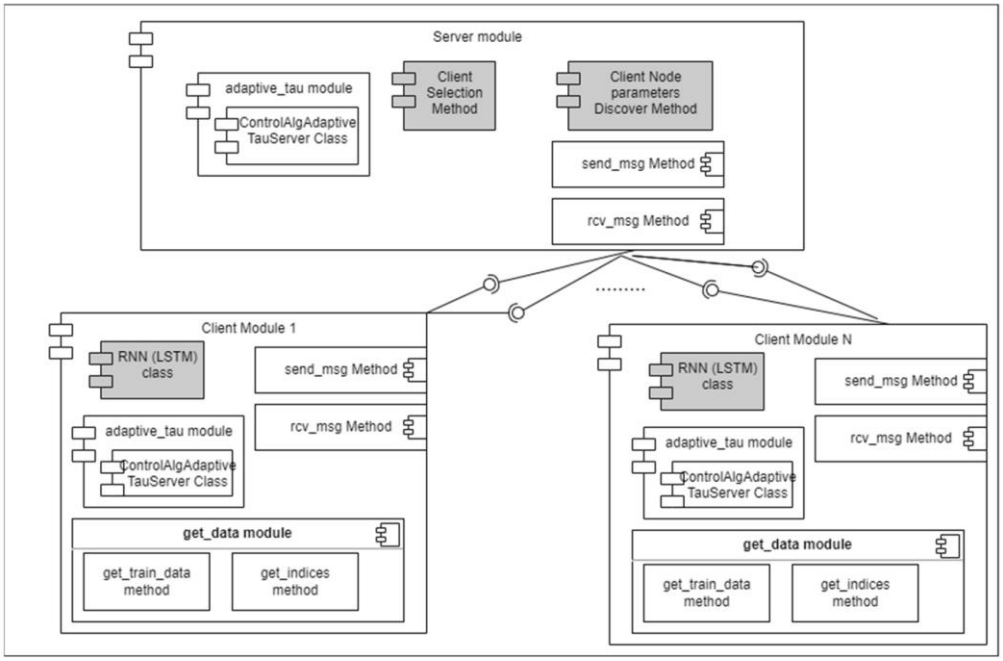
Εξαιτίας των προσαρμοζόμενων υπολογιστικών πόρων που χρησιμοποιούνται κατά την εκτέλεση του MulticloudFL, η συχνότητα που λαμβάνει χώρα το global aggregation μπορεί να επαναυπολογιστεί κατά την διάρκεια της διαδικασίας εκμάθησης. Στόχος είναι να επιτευχθεί το ελάχιστο της γενικής συνάρτησης απωλειών του Federated Learning συστήματος (εξίσωση 4). Με αυτό τον τρόπο, η υπολογισμένη συχνότητα για global aggregation επιτελεί βέλτιστη χρήση διαθέσιμων υπολογιστικών πόρων του Federated Learning συστήματος χωρίς υπο-χρησιμοποίηση(under-utilzation) ή υπερ-χρησιμοποίηση (over-utilization) τους. Στην υλοποίηση της παρούσας διατριβής, ο κύριος αναπροσαρμοζόμενος πόρος είναι η παράμετρος του χρόνου στον γενικό αλγόριθμο. Μια σε υψηλό επίπεδο (όχι λεπτομερειακή) Αρχιτεκτονική του Συστήματος σε γενικές γραμμές είναι η ακόλουθη:



Σχήμα 29 – Περιγραφή Γενικής Αρχιτεκτονικής Συστήματος

Επιπρόσθετα, στο σχήμα 30 ακολούθως, παρουσιάζουμε την υλοποίηση του γενικού διαγράμματος αλγορίθμου που παρουσιάστηκε στην παράγραφο 6.3.1 και συζητάμε κυρίως την τεχνολογία που έχε χρησιμοποιηθεί σε κάθε μέρος του Federated Learning συστήματος. Στο σχήμα 28 παρουσιάζουμε νέα modules και στοιχεία σε σχέση με μια παλιότερη σχετική εργασία παρόμοιας τεχνολογίας (Wang et al.) [99]. Οι δύο κύριες νέες λειτουργίες στο νέο σύστημα MulticloudFL είναι η χρήση βαθέων νευρωνικών δικτύων (deep learning) για αξιποίηση τους σε τοπικές συναρτήσεις απωλειών(LSTM algorithm) για καλύτερη

βελτιστοποίηση, η χρήση του module ανακάλυψης client καθώς και η διαδικασία επιλογής των κατάλληλων clients για να επιτυγχάνεται υψηλότερη ακρίβεια προγνώσεων.



Σχήμα 30 – Περιγραφή Αρχιτεκτονικής Υλοποίησης Λογισμικού του Multi Cloud FL system

Από άποψη υλοποίησης, το μοντέλο που χρησιμοποιείται τοπικά είναι ένα βαθύ νευρωνικό δίκτυο που ακολουθεί την μάθηση με επίβλεψη (supervised learning) με χρήση δεδομένων με ετικέτα (labeled data) όπως συνήθως συμβαίνει σε αυτές τις περιπτώσεις. Πιο συγκεκριμένα, δεδομένα με μορφή χρονοσειράς που αναφέρονται σε κατανάλωση επεξεργαστικής ισχύος (CPU) ως προς τον χρόνο χρησιμοποιούνται σαν δεδομένα εκπαίδευσης(training data) και σαν δεδομένα επικύρωσης ή ελέγχου (test/validation data). Στην συγκεκριμένη υλοποίηση έχουμε μια περίπτωση προβλήματος παλινδρόμησης. Το τοπικό μοντέλο νευρωνικού δικτύου με ανατροφοδότηση (Recurrent Neural Networks) χρησιμοποιεί την περίπτωση του δικτύου βραχείας μνήμης (LSTM – Long Short Term Memory Networks) όπως ήδη έχει επεξηγηθεί η λειτουργία του ανωτέρω. Ακριβώς επειδή μιλάμε για περίπτωση αλληλουχίας δεδομένων (sequences of data) η χρήση του LSTM δικτύου είναι η ενδεδειγμένη κυρίως και επειδή πρόκειται για προγνώσεις χρονοσειρών δεδομένων (παρουσιάζουν ευρεία ερευνητική δραστηριότητα ανά τον κόσμο). Η καταλληλόλητα των LSTMs δικτύων προκύπτει από το γεγονός ότι μπορούν να μαθαίνουν τις εξαρτήσεις μεταξύ των διαφόρων data points. Τα LSTM μοντέλα αναπτύσσουν μια συνάρτηση ώστε να μαθαίνουν από προηγούμενα ιστορικά σημεία (data) και να προβλέπουν επιτυχώς τα μελλοντικά σημεία. Στην υλοποίηση στην παρούσα διατριβή, ένα κρυφό επίπεδο περιέχει στο μοντέλο μας και ένα επίπεδο dropout για να αποφεύγεται όπως προαναφέρθηκε το φαινόμενο της υπερ-προσαρμογής (overfitting). Τέλος, μια σημαντική παράμετρος αποτελεί και η αρχικοποίηση των βαρών πριν από την εκπαίδευση ενός μοντέλου. Στην συγκεκριμένη περίπτωση, η αρχικοποίηση έγινε στο μηδέν. Αυτό πρακτικά επιτυγχάνεται στον κώδικα

υλοποίησης θέτοντας μηδενικά στους πίνακες global_grad_global_weight της κλάσσης ControlAlgAdaptiveTauServer.

Όπως μπορούμε να δούμε στο διάγραμμα του σχήματος 30, η βασική αρχιτεκτονική του server-client σχήματος υλοποιείται από τα αντίστοιχα modules. Πιο συγκεκριμένα, ο server είναι το κεντρικό module όπου η διαδικασία του global aggregation, η διαδικασία της ανακάλυψης των παραμέτρων των Clients στο Federated Learning σύστημα καθώς και η επιλογή κάθε φορά των κατάλληλων Clients με βάση τον αλγόριθμο Client Selection Adaptive Federated Learning, όπως παρουσιάστηκε στην παράγραφο 6.3.2, λαμβάνουν χώρα. Επιπλέον, το module στον κώδικα που λέγεται adaptivetau χρησιμοποιείται στο server μαζί με την κλάση ControlAlgAdaptiveTauServer ως μέρος της διαδικασίας του αλγορίθμου ελέγχου για την βελτιστοποίηση των πόρων (χρόνος) με την χρήση της παραμέτρου (_ value) χρησιμοποιώντας τους επιλεγέντες κόμβους-clients και πόρους. Τελευταία, στο μέρος του server χρησιμοποιούνται δύο μέθοδοι στον κώδικα για τις επικοινωνίες με τους client nodes (multi-cloud and edge nodes) και λέγονται send_msg & recv_msg για την αποστολή και λήψη δεδομένων αντίστοιχα.

Από την μεριά του Client, η βασική μονάδα (module) που χρησιμοποιείται σε κάθε κόμβο (client) είναι το Client module. Με τη βοήθεια αυτού του module τοπικά βήματα εκμάθησης εκτελούνται σε κάθε κόμβο-client του Federated Learning συστήματος μέχρι να επιτευχθεί μια τοπική σύγκλιση χρησιμοποιώντας τον αλγόριθμο Gradient Descent και τη συνάρτηση απωλειών της LSTM που περιλαμβάνεται στην κλάση RNN. Επιπλέον, η ήδη αναφερθείσα διαδικασία τοπικής εκμάθησης εκτελείται βασισμένη στα τοπικά δεδομένα του κάθε κόμβου-client ανακτώντας τα δεδομένα εκπαίδευσης με την χρήση των μεθόδων get_data() και get_train_data () από το τοπικό data storage του κάθε κόμβου. Επίσης, η μέθοδος get_indices() χρησιμοποιείται για την ανάγνωση δεδομένων με τρόπο Stochastic Gradient Descent όπως έχει ήδη εξηγηθεί. Όπως στην πλευρά του server, έτσι και εδώ οι δύο μέθοδοι: send_msg και recv_msg χρησιμοποιούνται για την βοήθεια στις επικοινωνίες και την ανταλλαγή των δεδομένων του μοντέλου μεταξύ των client κόμβων και του aggregator server. Τέλος, η μέθοδος ControlAlgAdaptiveTauClient χρησιμοποιείται σε κάθε client-κόμβο σαν μέρος του αλγορίθμου ελέγχου διαχείρισης υπολογιστικών πόρων.

Είναι σημαντικό να αναφερθεί ότι για τα διάφορα modules, κλάσεις και μεθόδους η τεχνολογία που έχει χρησιμοποιηθεί είναι η γλώσσα python έκδοση 3. Οι ανοιχτού-κώδικα βιβλιοθήκες που έχουν αναπτυχθεί από την Google τεχνολογία Tensorflow [141] και έχουν χρησιμοποιηθεί στο FL σύστημα που έχουμε αναπτύξει εστιάζουν κυρίως στην Keras Τεχνολογία [142]. Μια γενική άποψη της Αρχιτεκτονικής με τα νέα αναπτυγμένα modules με έμφαση στο γκρι χρώμα όπως φαίνεται στο παραπάνω σχήμα 30.

Σχετικά με την διαδικασία deployment του μοντέλου του Federated System, ακολουθούμε τα επόμενα βήματα για να το ολοκληρώσουμε επιτυχώς:

- Εκπαιδεύουμε και χτίζουμε το τελικό μοντέλο όπου η τελική συνάρτηση συνολικών απωλειών θα έχει την ελάχιστη τιμή σύμφωνα με το σχήμα συνολικού flow (26). Το module server.py είναι τοποθετημένο στον κεντρικό server ο οποίος συγκεντρώνει τα διάφορα τοπικά εκπαιδευμένα βάρη και το module client.py το οποίο είναι τοποθετημένο σε καθένα από τους clients-κόμβους (τοπικά βάρη).
- Μετά την εκπαίδευση του συνολικού μοντέλου FL, η παράμετροι του σώζονται με την χρήση της συνάρτησης model.save της βιβλιοθήκης keras του tensorflow

(https://www.tensorflow.org/guide/keras) στον κεντρικό aggregator server. Με αυτό τον τρόπο το μοντέλο σώζεται με ένα format ώστε να μπορεί να φορτωθεί σε διάφορους κόμβους-server nodes όπου τα δεδομένα inference (real data) βρίσκονται. Το παραπάνω format αφορά εκπαιδευμένο μοντέλο σε μορφή HDF5[143] file format, το οποίο περιλαμβάνει τόσο την αρχιτεκτονική του μοντέλου όσο και τα εκπαιδευμένα βάρη του.

- Προκειμένου να κάνουμε deploy το εκπαιδευμένο μοντέλο σε διάφορους κόμβους (nodes), χρησιμοποιούμε το Tensorflow serving και ακολούθως το τοπικό inferenceστα τοπικά δεδομένα τα πραγματικά του κάθε κόμβου για το federated σύστημα μπορεί να λάβει χώρα.
- Μετά το deployment του προαναφερθέντος μοντέλου προγνώσεις βασισμένες σε πραγματικά δεδομένα μπορούν να λάβουν χώρα.

Η υλοποίηση της περιεγραμμένης της αρχιτεκτονικής του MultiCloudFL συστήματος μπορεί να βρεθεί για περισσότερη λεπτομερειακή ανάλυση στο GitHub: https://github.com/vasilaros/Multicloud-Federated-Learning-FL.git .

6.5 Αξιολόγηση

Αξίζει να σημειωθεί ότι το Client Participation Adaptive Federated Learning System (MulticloudFL), που έχει υλοποιηθεί στην παρούσα διδακτορική διατριβή, έχει τεσταριστεί και αξιολογηθεί σε διάφορα σενάρια πραγματικών καταστάσεων. Σε αυτή την παράγραφο παρουσιάζουμε ένα από αυτά τα σενάρια ως ενδεικτικό παράδειγμα προκειμένου να αξιολογήσουμε την αξία της προσέγγισής μας. Πιο συγκεκριμένα αναφερόμαστε σε μια εφαρμογή εξομοίωσης η οποία λέγεται Computational Fluid Dynamic και η οποία έχει εμπνευστεί από ένα από τα πιλοτικά που χρησιμοποίησε η εταιρεία ICON για την ανάλυση των υδροδυναμικών μοντέλων και των σχετικών υδροδυναμικών δεδομένων κατά τη διάρκεια του Nebulous έργου (https://www.iconcfd.com/). Αυτή η εφαρμογή παράγει χρήσιμα αποτελέσματα αναλύσεων που αναφέρονται σε εξομοιώσεις που εξετάζουν σχεδιασμό μοντέλων και την αποτελεσματικότητα της υδροδυναμικής. Ειδικότερα η εφαρμογή εξομοίωσης αφορά μια περίπτωση υγρού 2 διαστάσεων. Αρχικά το υγρό ρέει από αριστερά προς τα δεξιά και ένα γραμμικό διαχωριστικό διαχωρίζει το υγρό και δημιουργεί στροβίλους. Τα χρώματα που εμφανίζονται στην εφαρμογή είναι παρόμοια με την περίπτωση της εφαρμογής της εταιρείας ICON και δείχνουν τον στροβιλισμό ή την τοπική κίνηση περιστροφής του υγρού. Χρησιμοποιώντας ελέγχους για να προσαρμόσουμε την ταχύτητα της ροής και του ιξώδους, η εφαρμογή σχεδιάζει διαφορετικά διαχωριστικά, τοποθετεί τριγύρω το υγρό, σχεδιάζει άλλες ποσότητες πίσω από το στροβιλισμό, δείχνει την εξερχόμενη δύναμη από το υγρό προς τα διαχωριστικά, και τελικά μετράει την πυκνότητα του υγρού και την ταχύτητα του σε κάθε σημείο. Η εφαρμογή Computational Fluid Dynamic είναι βασισμένη σε εξομοιώσεις που χρειάζονται έναν μεταβαλλόμενο ρυθμό από υπολογιστικούς πόρους. Ειδικότερα στην παρούσα διατριβή σχεδιάσαμε και υλοποιήσαμε 3 κατηγορίες εξομοιώσεων: hi-fi (high fidelity), low-fi (low-fidelity), και ένας υβριδικός συνδυασμός τους. Η πρώτη εξομοίωση low-fi or low-fidelity που έδωσε χαμηλή ευκρίνεια εικόνων σε pixels, χρειάστηκε λίγους πόρους υπολογιστικής ισχύος καθώς το low-fi χρησιμοποιείται για να χτιστεί γνώση από τις εξομοιώσεις. Οι εξομοιώσεις σε αυτή την κατηγορία έγιναν οι λιγότερες πραγματικές σε αυτόν που μαθαίνει. Συγκεκριμένα, αυτές περιλαμβάνουν στατικά μοντέλα και αναπαραστάσεις 2 διαστάσεων που έδωσαν μια γενική και όχι σε λεπτομέρεια εικόνα της πραγματικότητας. Από την άλλη μεριά, οι hi-fi ή high-fidelity εξομοιώσεις, οι οποίες τυπικά αναπαρίστανται από μια υψηλής ευκρίνειας εικόνα, έτοιμη για παραγωγή, αλληλοεπιδρώντας με πρωτότυπα χρειάζονται υψηλής απόδοσης υπολογιστική κυρίως ισχύ. Αυτό συμβαίνει γιατί η hi-fi εξομοίωση είναι ένα τεστ το οποίο εξομοιώνει πολύ κοντινά μια πραγματικού κόσμου κατάσταση και στο οποίο οι συμμετέχοντες στο τεστ λειτουργούν σαν να ήταν στην πραγματική ζωή. Υπάρχει και μια τρίτη κατηγορία εξομοίωσης συνδυασμός των 2 προηγούμενων, και η οποία δίνει τα δεδομένα τα οποία ονομάζονται polarized workload data.

Το MulticloudFL σύστημα αντιμετωπίζει το παραπάνω αρκετά δυναμικό φαινόμενο της δυναμικής ζήτησης πόρων. Έτσι βοηθάει την εφαρμογή εξομοίωσης Computational Fluid Dynamic ώστε να αποκτήσει τους απαιτούμενους πόρους προθύστερα για να ελαχιστοποιήσει το κόστος και να βελτιστοποιήσει τον λόγο κόστος ως προς χρόνο εξομοίωσης, βασισμένο σε περιορισμούς του τελικού χρήστη και σε συγκεκριμένα Service Level Agreements (SLAs) όπως και με βάση τον αριθμό και τον τύπο των εξομοιώσεων που ζητούνται. Υπολογιστικοί πόροι βελτιστοποιούνται δυναμικά σε παραγωγική λειτουργία της

εφαρμογής με βάση το τρέχον φορτίο της αλλά και προβλεπόμενα στο μέλλον φορτία εξομοίωσης.

Το αρχικό deployment της συγκεκριμένης εφαρμογής αποτελείται από 2 βασικά στοιχεία (εξαιρούμε το στοιχείο του aggregator) και στη συνέχεια το deployment της εφαρμογής εξετάζεται με τη χρήση 3, 5 ή 20 βασικών στοιχείων.

Για να αξιολογήσουμε την απόδοση του προτεινόμενου συστήματος MulticloudFL, χρησιμοποιήσαμε ένα πειραματικό περιβάλλον από 3 τύπους δεδομένων σχετικά με την κατανάλωση υπολογιστικής ισχύος χρησιμοποιώντας από 2 μέχρι 20 κόμβους-clients. Τα δεδομένα προκύπτουν σαν αποτελέσματα από 3 περιπτώσεις εξομοίωσης της κατανεμημένης εφαρμογής Fluid Dynamic Simulation (Σχήματα 31,32,33). Η κατανάλωση επεξεργαστικής ισχύος για κάθε κόμβο (node) ή αλλιώς εικονικής μηχανής αναφέρεται σε συνολική κατανάλωση της CPU ανά core ανά κόμβο του συστήματος. Ειδικές μετρικές ακρίβειας πρόβλεψης για CPU workload χρησιμοποιήθηκαν για την αξιολόγηση της κάθε περίπτωσης εξομοίωσης. Επιπλέον παρουσιάζεται η ανάγκη να αξιολογήσουμε και τη σταθερότητα στην απόδοση προβλέψεων του συνολικού μοντέλου Federated Learning. Έτσι δεν μπορούμε απλά να προσαρμόζουμε το μοντέλο στα δεδομένα εκπαίδευσης και να περιμένουμε ότι θα λειτουργεί επακριβώς και με την ίδια ακρίβεια πρόγνωσης σε πραγματικά δεδομένα που δεν έχει δει προηγουμένως. Υπάρχει ανάγκη για διαβεβαίωση ότι το federated model έχει λαβή όλα τα πιθανά πρότυπα από τα πραγματικά δεδομένα χωρίς να συλλέγει πολύ θόρυβο ή με άλλα λόγια να είναι χαμηλό το bias και η διακύμανση των δεδομένων. Στα πειράματά μας, χρησιμοποίησα ένα συνολικό ποσό από 2000 τοπικά data points που αφορούν επί τοις 100 κατανάλωση επεξεργαστικής ισχύος (% CPU consumption) υπό μορφή χρονοσειρών (κατανάλωση ανά δευτερόλεπτο) με μετρήσεις ανά κόμβο συστήματος. Από αυτό το ποσό των δεδομένων, το 75% χρησιμοποιήθηκε για εκπαίδευση του τοπικού μοντέλου gradient descent και το υπόλοιπο 25% για το validation-test του τοπικού μοντέλου στον κάθε κόμβο. Χρησιμοποιήσαμε το μεγαλύτερο ποσοστό των δεδομένων για σκοπούς εκπαίδευσης, διότι διαφορετικά θα είχαμε υψηλό το ρίσκο να χάσουμε σημαντικά πρότυπα και τάσεις δεδομένων, το οποίο με τη σειρά του θα αύξανε το σφάλμα που εισάγεται από την πόλωση τους (bias-induced error). Επιπλέον σε κάθε κόμβο χρησιμοποιήσαμε τον υπολογισμό gradient όσον αφορά τη συνάρτηση απωλειών (loss function), πάνω σε ένα δείγμα τυχαία επιλεγμένων δεδομένων (mini-batch) από το σύνολο όλων των πραγματικών δεδομένων προκειμένου να προσεγγίσουμε το πραγματικό gradient descent. Με άλλα λόγια, το stochastic gradient descent χρησιμοποιείται σε κάθε κόμβο του federated system και έτσι τα δεδομένα που χρησιμοποιούνται τοπικά για τοπική εκπαίδευση σε κάθε επανάληψη(iteration) είναι διαφορετικά από κόμβο σε κόμβο. Κάθε εποχή (epoch) αναφέρεται στο βήμα-step επανάληψης του gradient descent με την χρήση mini-batch στα τοπικά δεδομένα εκπαίδευσης. Αυτό το mini-batch αλλάζει από βήμα σε βήμα (σε κάθε εποχή) τυχαία.

Η χρήση των τεστ δεδομένων ή δεδομένων επικύρωσης (inference process) είναι μια πολύ χρήσιμη τεχνική για να αξιολογηθεί η αποτελεσματικότητα του μοντέλου federated learning, ειδικότερα σε περιπτώσεις όπου χρειάζεται να περιορίσουμε την υπερ-προσαρμογή (overfitting). Αυτά τα είδη των δεδομένων αντιπροσωπεύουν το κομμάτι των δεδομένων όπου το μοντέλο δεν έχει δει ποτέ προηγούμενα.

Το περιβάλλον για την αξιολόγηση βασίστηκε σε τεχνολογία Docker όπως και τεχνολογία εικονικών μηχανών[145]. Ο aggregator server και κάθε κόμβος client υλοποιήθηκαν

τοποθετώντας μια εικονική μηχανή σε περιβάλλον εικονικών μηχανών VMware με χαρακτηριστικά 4 cores VCPU, 8 GB RAM, και 30 GB Hard Disk Drives.

6.5.1 Πειραματικό Περιβάλλον-Setup

Για την διαδικασία της πειραματικής αξιολόγησης του συνολικού αλγορίθμου, χρησιμοποιήσαμε ένα περιβάλλον βασισμένο σε υποδομή multi-cloud και edge computing, με μεταβαλλόμενο αριθμό κόμβων από 2 μέχρι 20 και ετερογενή datasets για κάθε κόμβο, όσον αφορά το περιεχόμενο των τοπικών δεδομένων αλλά και του μεγέθους των τοπικών δεδομένων. Αυτό σημαίνει ότι κάθε κόμβος χρησιμοποιεί για κάθε τοπικό iteration ένα διαφορετικό υποσύνολο δεδομένων (διαφορετικό mini-batch of data). Ο aggregator της διαδικασίας του federated learning είναι τοποθετημένος σε ένα server (εικονική μηχανή) ξεχωριστός από τους υπολοίπους κόμβους του συστήματος. Κατά τη διαδικασία των πειραμάτων στα πλαίσια της συγκεκριμένης διατριβής χρησιμοποιήσαμε σαν τύπο πόρων μόνο τη χρονική παράμετρο. Έτσι θέσαμε M = 1. Για τις τιμές που χρησιμοποιούνται στον αλγόριθμο ελέγχου που έχουμε αναπτύξει, οι παράμετροι cm και bm αναφέρονται στον πραγματικό χρόνο που χρειάζεται για κάθε τοπική ανανέωση των παραμέτρων του τοπικού μοντέλου και επίσης στο χρόνο που χρειάζεται για το global aggregation, αντίστοιχα. Συγκρίνουμε τα αποτελέσματα των πειραμάτων μας με τα πειραματικά αποτελέσματα ενός παρόμοιου τύπου δεδομένων υπό μορφή χρονοσειρών όπου έχει χρησιμοποιηθεί η μέθοδος Adaptive Federated Learning η οποία ήδη αναφέρθηκε στην ερευνητική εργασία των Wang[99].

Χρησιμοποιήσαμε 3 διαφορετικούς τύπους datasets και την μέθοδο Stochastic Gradient Descent για την ανάγνωση των δεδομένων σαν τοπικός solver (optimizer). Τα 3 είδη των datasets προήλθαν από τις μετρημένες καταναλώσεις επεξεργαστικής ισχύος (CPU consumptions) για τις 3 περιπτώσεις εξομοιώσεως της κατανεμημένης εφαρμογής Computational Fluid Dynamic Simulation. Αυτές οι μετρήσεις προέρχονται από έναν από τους κόμβους του κατανεμημένου συστήματος για κάθε περίπτωση. Η πρώτη περίπτωση εξομοίωσης έδωσε τα αποτελέσματα “CPU Increasing Load with Spikes”, τα οποία είναι αρκετά ανώμαλες διακυμάνσεις του φορτίου της επεξεργαστικής ισχύος με συνεχή αυξανόμενα spikes (σχήμα 31). Οι σχετικές μετρικές με τα αποτελέσματα φαίνονται στον πίνακα 11 για επιπλέον αξιολόγηση. Η δεύτερη περίπτωση εξομοίωσης παρουσιάζει την περίπτωση “CPU Periodically Increasing Load with Fluctuations”, η οποία αναφέρεται σε πολύ απότομες μεταβολές του φορτίου της επεξεργαστικής ισχύος (σχήμα 32) και τα αποτελέσματα των μετρικών παρουσιάζονται στον πίνακα 13. Τελικά, η τελευταία περίπτωση εξομοίωσης παρουσιάζει δεδομένα από φορτίο επεξεργαστικής ισχύος τα οποία παρουσιάζονται στο σχήμα 33 ως “CPU PolarizedWorkload”, και τα σχετικά αποτελέσματα των μετρικών φαίνονται στον πίνακα 15. Για τη συγκεκριμένη περίπτωση τύπου δεδομένων, η κατανάλωση επεξεργαστικής ισχύος αναφέρεται σε περιοδικές μεταβολές του φορτίου της επεξεργαστικής ισχύος μεταξύ μιας ελάχιστης τιμής 0% και μιας μέγιστης τιμής 100%.

Για όλα τα datasets, η δημιουργία των mini-batches χρησιμοποιεί την ίδια αρχική τυχαία πηγή δεδομένων σε όλους τους κόμβους, το οποίο σημαίνει ότι όταν τα datasets σε όλους τους κόμβους είναι του ίδιου τύπου δεδομένων, τα mini-batches σε όλους τους κόμβους είναι σχεδόν ίδια σε μέγεθος στον ίδιο αριθμό επανάληψης (same iteration). Το τοπικό μοντέλο το οποίο χρησιμοποιείται σαν συνάρτηση απωλειών στο SGD προέρχεται από τον αλγόριθμο LSTM [120]. Η διασπορά των δεδομένων στους clients είναι τυχαία και με διάφορα μεγέθη καθώς και διαφορετικό περιεχόμενο δεδομένων.

Οι παράμετροι ελέγχου στο configuration file του συστήματος μας (multi-cloud edge FL system) είναι ως ακολούθως:

- Χρήση search range parameter γ=10
- Χρήση τmax = 100 (maximum τ value)
- Χρήση control parameter ϕ=0.01 για την περίπτωση του LSTM algorithm σαν locl loss function [120].
- Χρήση learning rate για τον Gradient Descent n=0.1. Αυτό προκύπτει ύστερα από πολλές δοκιμές δίνοντας την βέλτιστη τελική απόδοση.
- Resource (i.e. time) budget για τα πειράμτά μας τέθηκε σε R=15 seconds (μέγιστη τιμή εκτέλεσης).
- Χρήση stochastic gradient descent για την ανάγνωση των δεδομένων
- Αριθμός Data Points στο MulticloudFL ίσο με 2000 από όπου τα διάφορα τυχαία mini-batches προέρχονται.

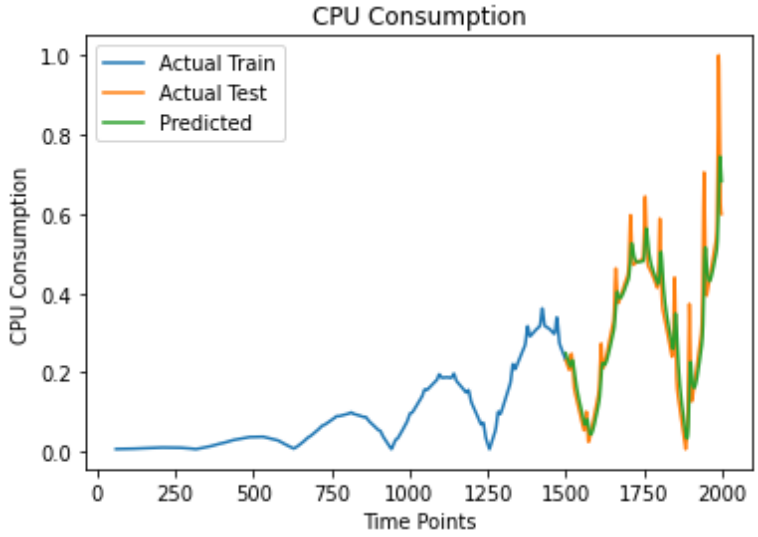
Αξίζει να αναφερθεί ότι η τιμή του learning rate προσδιορίσθηκε μετά από πολλά trial & error πειράματα.

Μια βασική παράμετρος στο τρόπο που δομούμε το περιβάλλον των πειραμάτων, είναι ο τρόπος κατανομής των δεδομένων στους διαφορετικούς κόμβους του Federated Learning συστήματος μας. Σχετικά με τις παραμέτρους λοιπόν κατανομής, ακολουθήσαμε ένα συνδυασμό μεθόδων uniform και non-uniform περιπτώσεων κατανομής δεδομένων. Αυτό σημαίνει ότι δείγματα δεδομένων με τα μισά labels είναι κατανεμημένα στους μισούς κόμβους όπου κάθε κόμβος έχει μοναδικά κατανεμημένη πληροφορία. Τα υπόλοιπα δείγματα δεδομένων είναι κατανεμημένο στο έτερο ήμισυ των κόμβων με δεδομένα τα οποία έχουν το ίδιο label σε κάθε κόμβο. Αυτό αντιπροσωπεύει την περίπτωση όπου κάθε κόμβος έχει μη μοναδικά κατανεμημένη πληροφορία (non-uniform information) επειδή το όλο dataset έχει δεδομένα με πολλαπλά διαφορετικά labels. Πιο συγκεκριμένα, κάθε φορά που υπάρχουν περισσότερα labels από αριθμό κόμβων, κάθε κόμβος μπορεί να έχει δεδομένα με περισσότερα από ένα label. Σίγουρα όμως ο αριθμός των labels σε κάθε κόμβο δεν είναι περισσότερος από το συνολικό αριθμό των labels διαιρεμένο με το συνολικό αριθμό των κόμβων στρογγυλοποιημένο στον επόμενο ακέραιο.

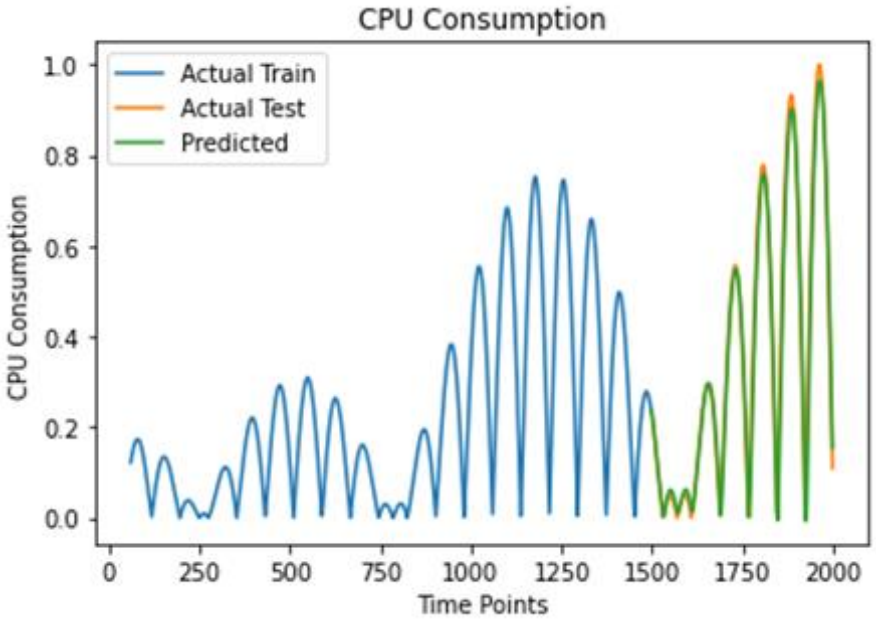
Στη γενική αρχιτεκτονική του αλγορίθμου LSTM που χρησιμοποιείται τοπικά σε κάθε κόμβο αλλά και όσον αφορά τον τρόπο ανάγνωσης των δεδομένων χρησιμοποιήσαμε τα ακόλουθα hyperparameters:

- Ένα hidden layer (dense regression deep learning network) αποτελούμενο από 50 νευρώνες (1 LSTM hidden layer)

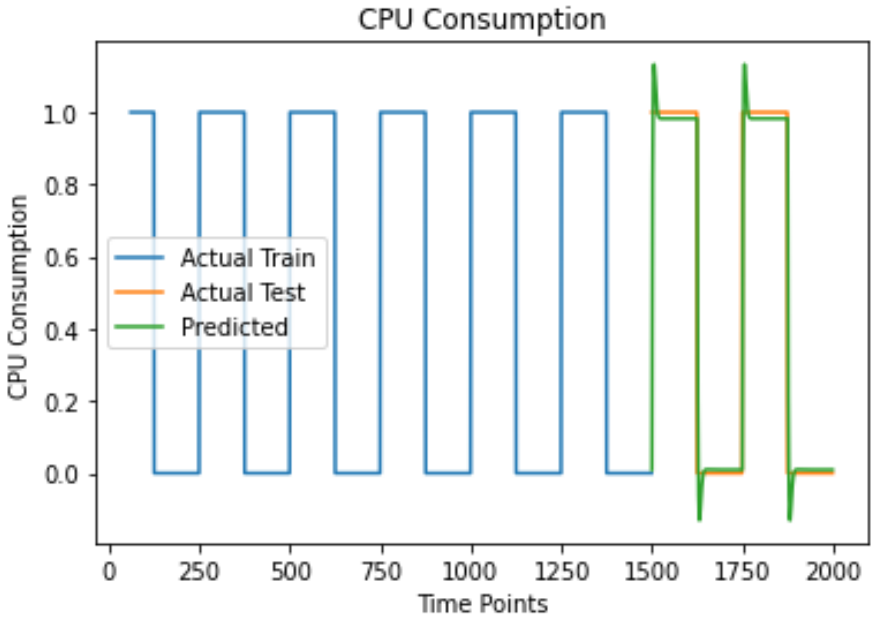
- Ένα dropout layer στην έξοδο με ένα rate ίσο με 0.2
- Time window επεξεργασίας = 60 iterations(updates) για τα δεδομένα εισόδου (sliding window) για κάθε epoch εκμάθησης.
- Το learning rate=0.1 (η βέλτιστη τιμή για να επιτευχθεί το χαμηλότερο global loss function value και η απαραίτητη σύγκλιση του αλγορίθμου συνολικά).
- Η μέθοδος βελτιστοποίησης που χρησιμοποιήθηκε είναι το adaptive moment estimation.



Σχήμα 31 – Δεδομένα για την περίπτωση «CPU Increasing Load with Spikes»



Σχήμα 32 – Δεδομένα για την περίπτωση «CPU Periodically Increasing Load with Fluctuations»



Σχήμα 33 – Δεδομένα για την περίπτωση « CPU PolarizedWorkload»

6.5.2 Αποτελέσματα Πειραμάτων και Ανάλυσή τους

Στα αποτελέσματα των πειραμάτων που φαίνονται σε αυτή την παράγραφο, οι μετρικές κατανάλωσης επεξεργαστικής ισχύος (CPU consumption) χρησιμοποιούνται σε διάφορους κόμβους ώστε τελικά να δώσουν μια ακριβή όσο το δύνατο πρόβλεψη της μελλοντικής κατανάλωσης επεξεργαστικής ισχύος (φορτίο-workload). Έτσι μετά την εκπαίδευση του συνολικού FL αλγορίθμου, γίνεται μια σύγκριση μεταξύ των τελικών αποτελεσμάτων Inference (πρόγνωση πράσινη γραμμή) και των πραγματικών αποτελεσμάτων (πραγματικά πορτοκαλί γραμμές). Οι ορισμοί που με βάση την βιβλιογραφία χρησιμοποιήθηκαν για την αξιολόγηση είναι οι ακόλουθοι:

- Mean Absolute Error (MAE) [124]
- Mean Squared Error or Loss (MSE) [125]
- Root Mean Squared Error (RMSE) [126]
- Mean Absolute Percentage Error (MAPE) [127]
- Symmetric Mean Absolute Percentage Error (SMAPE) [128]

Όπου για να δωθεί μια εικόνα της ακρίβειας που επιτυγχάνεται για κάθε είδος δεδομένων εφαρμόζονται οι ακόλουθοι με βάση τα ανωτέρω τύποι (n=αριθμός nodes, y_i είναι η πραγματική τιμή , $\widehat{y_p}$ είναι η προβλεπόμενη):

$$MAE = \frac{1}{n}\, x \sum_{i=1}^{n} |y_i - \widehat{y_p}| \qquad (26)$$

$$MSE = \frac{1}{n}\, x \sum_{i=1}^{n} \left(y_i - \widehat{y_p}\right)^2 \qquad (27)$$

$$RMSE = \left[\frac{1}{n} x \sum_{i=1}^{n} \left(y_i - \widehat{y_p}\right)^2\right]^{1/2} \quad (28)$$

$$MAPE = \frac{100\%}{n}\, x \sum_{i=1}^{n} \left|\frac{\widehat{y_p} - y_i}{y_i}\right| \qquad (29)$$

$$SMAPE = \frac{100\%}{n}\, x \sum_{i=1}^{n} \frac{|\widehat{y_p} - y_i|}{(|\widehat{y_p}| + |y_i|)/2} \qquad (30)$$

Εξετάζουμε 3 είδη τύπων δεδομένων που αναφέρονται σε κατανάλωση επεξεργαστικής ισχύος (CPU consumption). Το πρώτο αφορά πρότυπα δεδομένων που αφορούν αυξανόμενα spikes, ένα άλλο σύνολο δεδομένων (dataset) που αφορά πρότυπα με περιοδικά fluctuations της επεξεργαστικής ισχύος και μια τρίτη περίπτωση συνόλου δεδομένων (dataset) όπου πρότυπα δεδομένων τύπου polarized εξετάζονται. Όλα τα σύνολα δεδομένων αναφέρονται σε μετρικές κατανάλωσης επεξεργαστικής ισχύος (workload) που έχουν συλλεχθεί από κάθε κόμβο του Multicloud FL Computing system. Για κάθε σύνολο-είδος δεδομένων έχουμε μετρήσει την ακρίβεια πρόγνωσης με βάση τους τύπους που έχουν ήδη παρουσιαστεί προηγούμενα μέσω ένος οπτικοποιημένου διαγράμματος. Για κάθε πείραμα που έχει λάβει χώρα κάνουμε μια σύγκριση με την περίπτωση[99] όπου δεν εφαμόζονται οι τεχνικές επιλογής Client και χωρίς χρήση Deep Learning local loss function και παρατηρούμε την διαφορά.

Επιπλέον, όπως ήδη περιγράφηκε στο κεφάλαιο για τον αλγόριθμο ελέγχου πόρων (Control Algorithm) και σχετικά με την πολυπλοκότητα του χρόνου και της διαχείρισης μπορούμε να ισχυριστούμε ότι όταν έχουμε ένα unlimited χρονικό ορίζοντα είναι βέλτιστο να θέτουμε set_=1 και να εκτελείται το global aggregation μετά από κάθε βήμα τοπικού update των παραμέτρων-βαρών του τοπικού μοντέλου στον κόμβο. Παρόλαυτα, όταν έχουμε χρονικό περιορισμό, η εκπαίδευση μπορεί να σταματά μετά από ένα ορισμένο αριθμό επαναλήψεων, έτσι η τιμή του T είναι περιορισμένη. Φυσικά, ο αριθμός των κόμβων που αποκλείονται από την διαδικασία του global aggregation επηρεάζει τον συνολικό χρόνο εκτέλεσης του FL algorithm και αναμένεται φυσικά να τον ελλατώνει. Συγκρίσεις της παρούσας διδακτορικής διατριβλης με προηγούμενες εργασίες παρουσιάζονται και στην συνέχεια [99].

Ο πρώτος τύπος δεδομένων μας δίνει τα ακόλουθα αποτελέσματα εφαρμόζοντας την τεχνική επιλογής Client στον αλγόριθμο Federated Learning σε συνδυασμό με εφαρμογή της συνάτησης απωλειών Deep Learning Regressive(LSTM) loss function. Χρησιμοποιούμε 2 έως 20 κόμβους και 1 aggregator server για δεδομένα CPU Increasing Load with Spikes. Το συγκεκριμένο dataset όπως εξηγήθηκε αφορά την περίπτωση του low-phi Computational Fluid Dynamic Simulation:

Πίνακας 11: Μετρήσεις Ακρίβειας για δεδομένα CPU Increasing Load with Spikes

No of Nodes	N = 2 [16]	N = 2 MulticloudFL	N = 3 [16]	N = 3 MulticloudFL	N = 5 [16]	N = 5 MulticloudFL	N = 20 [16]	N = 20 MulticloudFL
MAE	0.047440	0.042696	0.043762	0.041137	0.043922	0.041287	0.0449	0.0429
MSE	0.005175	0.004197	0.004580	0.004031	0.004571	0.003977	0.0047	0.0041
RMSE	0.047440	0.042696	0.043763	0.041138	0.043922	0.041287	0.04477	0.042138
MAPE	25.946501	21.795061	22.8357	20.55213	22.827179	20.31619	21.827179	20.33
SMAPE	21.018166	18.706168	19.267344	17.91863	19.170655	17.82871	19.12	17.9

Στον ακόλουθο Πίνακα 12 τα ανωτέρα παρουσιασμένα αποτελέσματα συγκρίνονται και παρουσιάζονται βελτιωμένα (ελαττωμένες τιμές συναρτήσεων απώλειας) εν σχέση με τα αποτελέσματα της εργασίας των (Wang et al.) [99] όπου δεν υπάρχει καθόλου η έννοια του Client Selection Adaptive Federated Learning Algorithm ούτε της τοπικής συνάρτησης απώλειας βασισμένης σε Deep Learning.

Πίνακας 12: Σύγκριση % για Μετρήσεις Ακρίβειας για δεδομένα CPU Increasing Load with Spikes

Number of Nodes	N = 2	N = 3	N = 5	N = 20
MAE	−10%	−6%	−6%	−4%
MSE (loss)	−18.9%	−12%	−13%	−12%
RMSE	−10%	−6%	−6%	−5%
MAPE	−16%	−10%	−11%	−6%
SMAPE	−11%	−7%	−7%	−6%

Ο δεύτερος τύπος δεδομένων που χρησιμοποιήσαμε στα πειράματά μας αφορά την κατανάλωση επεξεργασικής ισχύος με αυξανόμενο φορτίο και συγκεκριμένα CPU Periodically Increasing Load with Fluctuations όπου λάβαμε τα ακόλουθα αποτελέσματα:

Πίνακας 13: Μετρήσεις Ακρίβειας για δεδομένα CPU Increasing Load with Fluctuations

No of Nodes	N = 2 [16]	N = 2 MulticloudFL	N = 3 [16]	N = 3 MulticloudFL	N = 5 [16]	N = 5 MulticloudFL	N = 20 [16]	N = 20 MulticloudFL
MAE	0.034889	0.029656	0.031341	0.029931	0.033381	0.032714	0.0337	0.0323
MSE	0.001949	0.001735	0.001749	0.001723	0.001978	0.001969	0.00199	0.001969
RMSE	0.034889	0.029656	0.031178	0.029931	0.040524	0.039714	0.041	0.0399
MAPE	56.295043	52.35439	62.452406	59.95431	51.151822	46.03664	51.22	47.037
SMAPE	25.119926	23.86393	23.688836	23.21506	26.845978	25.50368	26.9876	26.0001

Τα ανωτέρω αποτελέσματα συγκρινόμενα με την εργασία των Wang et al. (2019) παρουσιάζονται αρκετά βελτιωμένα (μειωμένες συναρτήσεις απωλειών με λιγότερα λάθη). Συγκεκριμένα οι διαφορές με την εργασία των (Wang et al.) [99] παρουσιάζονται ακολούθως:

Πίνακας 14: Σύγκριση % για Μετρήσεις Ακρίβειας για δεδομένα CPU Increasing Load with Fluctuations

Number of Nodes	**N = 2**	**N = 3**	**N = 5**	**N = 20**
MAE	−15%	−4.5%	−2%	−4.1%
MSE (loss)	−11%	−1.5%	−0.5%	−1%
RMSE	−15%	−4%	−2%	−2.6%
MAPE	−7%	−4%	−10%	−8%
SMAPE	−5%	−2%	−5%	−3.6%

Ο τρίτος και τελυταίος τύπος δεδομένων αναφέρεται στο CPU Polarized Workload δίνοντας τα ακόλουθα αποτελέσματα:

Πίνακας 15: Μετρήσεις Ακρίβειας για δεδομένα CPU Polarized

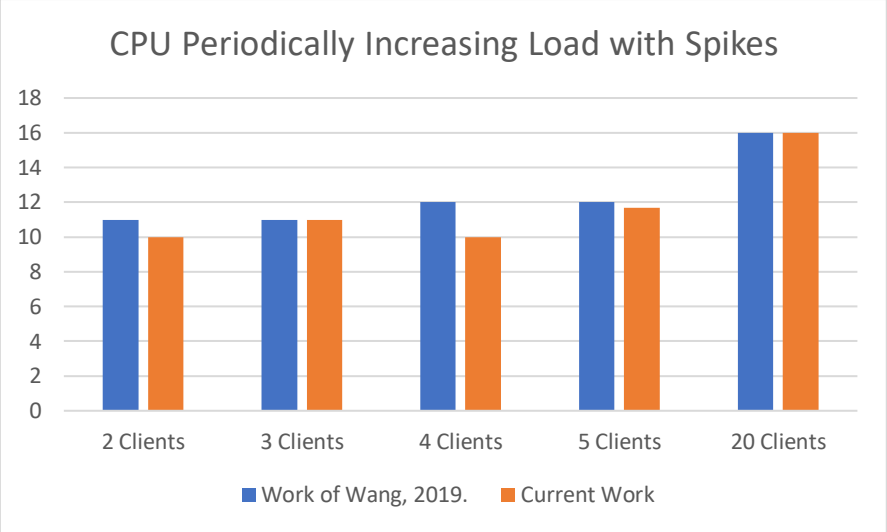
No of Nodes	**N = 2 [16]**	**N = 2 MulticloudFL**	**N = 3 [16]**	**N = 3 MulticloudFL**	**N = 5 [16]**	**N = 5 MulticloudFL**	**N = 20 [16]**	**N = 20 MulticloudFL**
MAE	0.033437	0.032434	0.035118	0.028446	0.034642	0.027714	0.035	0.028713
MSE	0.011076	0.010966	0.012115	0.011389	0.011651	0.010952	0.0117	0.01123
RMSE	0.034141	0.032434	0.034690	0.028446	0.034643	0.027715	0.03567	0.028815
MAPE	16181355	14725033	18406291	13859375	19496125	12087598	19556125	12197598
SMAPE	103.2559	102.2234	104.0340	101.9534	105.251958	102.0944	106.251958	103.0944

Τα παραπάνω αποτελέσματα σε σύγκριση με την περίπτωση της μη εφαρμογής Client Selection Adaptive Federated Learning Algorithm χωρίς συναρτήσεις απωλειών Deep Learning βελτιώνονται κατά τα ακόλουθα % ποσά:

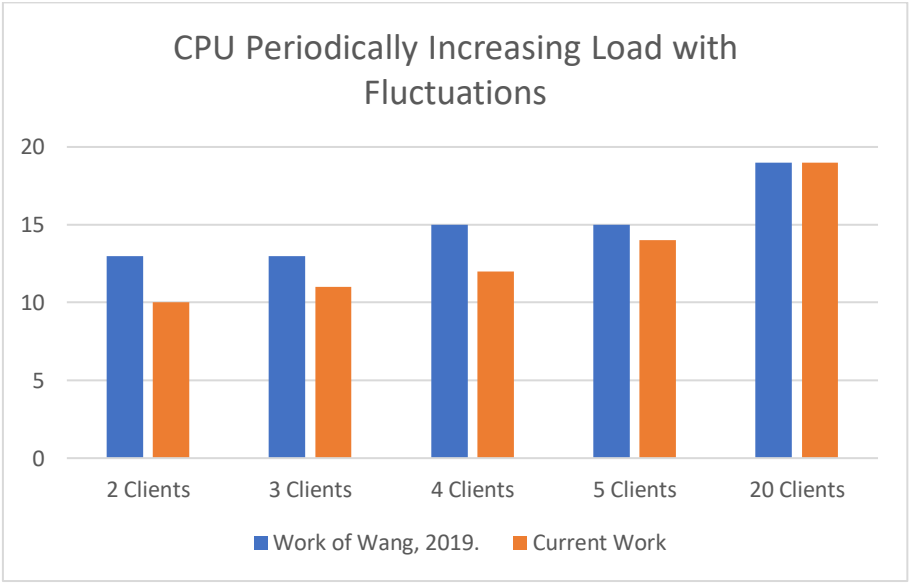
Πίνακας 16: Σύγκριση % για Μετρήσεις Ακρίβειας για δεδομένα CPU Polarized

Number of Nodes	**N = 2**	**N = 3**	**N = 5**	**N = 20**
MAE	−3%	−19%	−20%	−17%
MSE (loss)	−1%	−6%	−6%	−4%
RMSE	−5%	−18%	−20%	−19%
MAPE	−9%	−20%	−38%	−37%
SMAPE	−1%	−2%	−3%	−2.9%

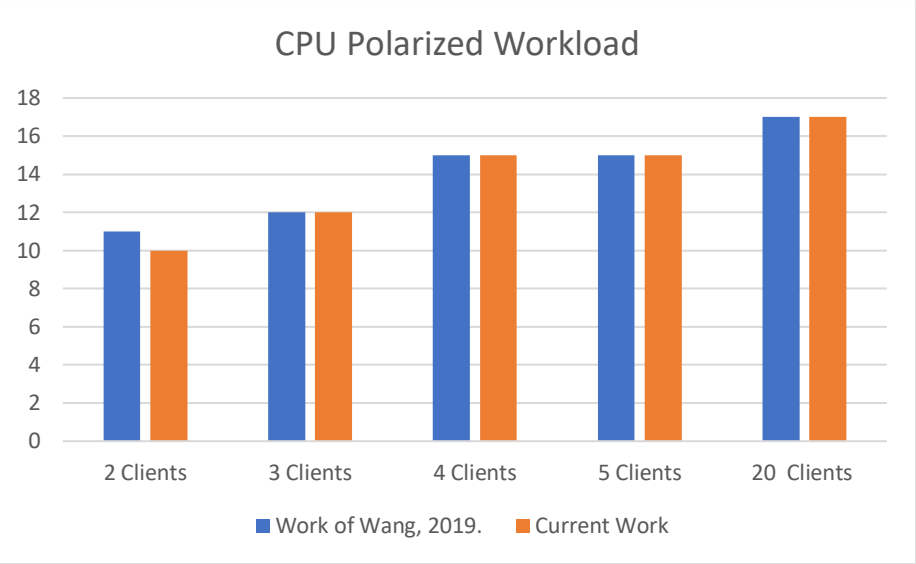
Σχετικά με τον χρόνο εκτέλεσης της εκπαίδευσης του αλγορίθμου και προκειμένου να αξιολογήσουμε τα τελικά βάρη του τελικού μοντέλου, συγκρίνουμε τις διάφορες μετρικές του χρόνου μεταξύ των δύο προσεγγίσεων (MultiCloudFL και [99]). Ο αριθμός των nodes-κόμβων μεταβάλλεται ως γνωστόν από 2 έως 20 για κάθε μια περίπτωση από τα τρία datasets που έχουμε αναφέρει ήδη προηγούμενα. Συνεπώς έχουμε τα ακόλουθα 3 γραφήματα (Σχήματα 34-35-36), όπου ο χρόνος δίδεται σε λεπτά στον κάθετο άξονα (Y) και ο αριθμός των συμμετεχόντων clients στον οριζόντιο άξονα (X):



Σχήμα 34 – Δεδομένα για «CPU Periodically Increasing Load with Spikes» όσον αφορα τον χρόνο εκτέλεσης εκπαίδευσης (εργασία παρούσας διδακτορικής διατριβής και εργασία [99]).



Σχήμα 35 – Δεδομένα για «CPU Periodically Increasing Load with Fluctuations» όσον αφορα τον χρόνο εκτέλεσης εκπαίδευσης (εργασία παρούσας διδακτορικής διατριβής και εργασία [99]).



Σχήμα 36 – Δεδομένα για «CPU Polarized Workload» όσον αφορα τον χρόνο εκτέλεσης εκπαίδευσης (εργασία παρούσας διδακτορικής διατριβής και εργασία [99]).

Σχετικά με τα παραπάνω αποτελέσματα, μπορούμε να πούμε ότι η παρούσα διδακτορική εργασία για όλους τους τύπους των δεδομένων και όλους τους αριθμούς των κόμβων δίνει ελαφρώς καλύτερους χρόνους εκτέλεσης εκπαίδευσης(μικρότερους) και άρα καλύτερη διαχείριση χρόνου efficency.

6.5.3 Συμπεράσματα της τεχνικής Επιλογής των Clients

Συμπερασματικά για τα παραπάνω αποτελέσματα για όλους τους τύπους δεδομένων, φαίνεται ότι σχετικά με τις τιμές του μεγέθους MAE η τρέχουσα δουλειά στην διδακτορική διατριβή παρουσιάζει βελτίωση από 3% έως 20% συγκρινόμενη με την εργασία των (Wang et al.) [99]. Το συγκεκριμένο μέγεθος εκφράζει τα αποτελέσματα της μέτρησης της διαφοράς μεταξύ 2 συνεχών χρονικών μεταβλητών : της πραγματικής κατανάλωσης της επεξεργαστικής ισχύος και της τιμής πρόγνωσής της. Παρατηρούμε ότι τα δεδομένα που αφορούν CPU polarised workload δίνουν καλύτερες MAE τιμές από αυτές των δεδομένων CPU increasing load και των CPU increasing load with spikes, επειδή στα δεδομένα CPU increasing load with spikes οι προβλεπόμενες με τις πραγματικές τιμές είναι εξαιρετικά πλησίον η μια της άλλης χωρίς παρουσία περίεργων συμπεριφορών ή outliers. Με άλλα λόγια το μέγεθος MAE δίνει λιγότερη σημασία σε δεδομένα με περιέργες συμπεριφορές (outliers) και παρουσιάζει καλύτερες τιμές όταν αυτά τα outliers απουσιάζουν [129].

Για το MSE (loss definition) και την μετρική RMSE, υπάρχει μια βελτίωση συγκρινόμενα με την εργασία των (Wang et al.) [99] η οποία κυμαίνεται από 1% έως σχεδόν 20% για όλα τα datasets. Αυτό είναι ένα αναμενόμενο αποτέλεσμα, καθότι ο αλγόριθμος επιλογής client αντιμετωπίζει επιτυχώς τα δεδομένα των clients που προκαλούν υψηλές τοπικές απώλειες κατά την διάρκεια της εκμάθησης όπως πχ. τα δεδομένα που παρουσιάζουν spikes. Αυτές οι τιμές απωλειών εξαιρούναι κατά την διάρκεια του global aggregation στο FL σύστημα. Γι αυτό τον λόγο, ειδικά για τα δεδομένα που αφορούν «CPU Increasing load with spikes» έχουμε καλύτερα αποτελέσματα από τα άλλα δύο είδη δεδομένων. Η ίδια περίπτωση ισχύει και για την μετρική RMSE η οποία είναι επίσης ευαίσθητη σε ακραίες τιμές δεδομένων.

Ακολουθώντας την ίδια λογική και η μετρική MAPE είναι αρκετά ευαίσθητη σε ακραία δεδομένα (outliers) και σε περιέργα μοτίβα δεδομένων και φυσικά συμπεριφέρεται καλύτερα όταν αυτά τα δεδομένα δεν υπάρχουν στην μελέτη [129],[130],[131],[132]. Βλέπουμε από τα αποτελέσματα ότι και τα 3 είδη δεδομένων δίνουν βελτιωμένα αποτελέσματα που ποικίλουν από 1% έως 38% εν σχέση με τα απολέσματα της εργασίας των (Wang et al.) [99] με τα δεδομένα των CPU Polarized Workload dataset να δίνουν τα καλύτερα MAPE αποτελέσματα. Αυτό συμβαίνει γιατί τα συγκεκριμένα δεδομένα δεν παρουσιάζουν ακραία αποτελέσματα πχ. outliers. Παρόλαυτα, είναι ευρέως γνωστό ότι [129],[131],[132] η μετρική SMAPE είναι πολύ ευαίσθητη και εξαρτάται κυρίως από από το επίπεδο των χρονικών δεδομένων και την ύπαρξη μικρών ιμών κοντινών στο μηδέν. Αυτός είναι ο λόγος που βλέπουμε χειρότερη συμπεριφορά τόσο για το CPU Polarized Workload όσο και CPU Increasing Load with Fluctuations σχετικά με την ακρίβεια SMAPE. Ο πρώτος τύπος δεδομένων CPU Increasing Load with Spikes δεν περιλαμβάνει πολλές τιμές κατανάλωσης επεξεργαστικής ισχύος (CPU consumption) κονά στο μηδέν και γι αυτό το λόγο παρουσιάζει καλύτερα αποτελέσμαρα όσον αφορά την μετρική SMAPE.

Γενικότερα μιλώντας, τα αποτελέσματα στην τρέχουσα παράγραφο της Διδακτορικής Διατριβής δείχνουν μια άυξηση σην ακρίβεια προβλέψεων σε σχέση με προηγούμενες ερευνητικές δουλειές [99], [113],[105], [133], [111]. Η χρήση τοπικών συναρτήσεων μεταφοράς/απωλειών τεχνολογίας Deep Learning σε συνδυασμό με χρήση μεθόδων επιλογής όπως αυτές έχουν ήδη παρουσιαστεί, αποτελεί ένα ιδιαίτερο καινοτόμο σχήμα μεθοδολογίας που δεν συναντάται σε προηγούμενες εργασίες [112], [113], [111], [107]. Είναι σημαντικό να πούμε ότι ο χρόνος σύγκλισης φαίνεται να βελτιώνεται συγκρινόμενος με την εργασία των (Wang et al.) [99] καθώς μόνο οι επιλεγέντες clients λαμβάνονται υπόψιν σε κάθε global iteration. Τελικά θα πρέπει να πούμε ότι οι προτεινόμενες μέθοδοι για τύπο προβλημάτων time regression που έχουν αναλυθεί σε αυτήν την εργασία είναι καινοτόμες δίνοντας πολύ ακριβείς λύσεις που μόνο μέχρι στιγμής εργασίες σε data classification προβλήματα έχουν παρουσίασει στο ερευνητικό πεδίο τέτοια πρόοδο [111], [107], [108], [93],[113], [99]. Τελευταίο αλλά όχι λιγότερο σημαντικό, είναι το γεγονός ότι η τρέχουσα εργασία στην παρούσα διδακτορική διατριβή δίνει καλύτερο χρόνο εκτέλεσης και εκπαίδευσης αλγορίθμου το οποίο σημαίνει αρτιότερη διαχείριση των χρονικών πόρων του αλγορίθμου. Αυτό πιστοποιείται με τα σχετικά πειραματικά αποτελέσματα που παρουσιάστηκαν προηγούμενα.

Παρόλαυτά, και και παρά τα πολλά πλεονεκτήματα της τρέχουσας εργασίας της παρούσας διδακτορικής διατριβής, μπορούμε να ανιχνεύσουμε κάποιους περιορισμούς ειδικότερα όσον αφορά τη διαδικασία της σύγχρονης εκπαίδευσης του συστήματος multi-cloud federated learning. Ένας σημαντικός περιορισμός είναι ότι το συνολικό μοντέλο του federated learning δεν λαμβάνει υπόψη του τους συμμετέχοντες που μπορούν να συμβάλλουν στη διαδικασία εκπαίδευσης κατά την διάρκεια που αυτή έχει ήδη αρχίσει (δηλαδή αν η διαδικασία εκπαίδευσης είναι ήδη σε εξέλιξη). Μη σύγχρονη (asynchronous) επικοινωνία προτείνεται σε αυτή την περίπτωση ούτως ώστε να επιλυθεί το πρόβλημα και να βελτιωθεί το scalability (επεκτασιμότητα) και η επάρκεια της προτεινόμενης προσέγγισης federated learning. Επιπλέον, ένας άλλος περιορισμός φαίνεται να είναι η σύγκλιση του μοντέλου με βάση τον σωστό καθορισμό των hyperparameters (fine-tuning of its hyperparameters). Αυτή η διαδικασία βασίζεται κατά πολύ στην εμπειρία του μηχανικού που σχεδιάζει το μοντέλο και που λαμβάνει υπόψη του trade-off ρυθμίσεις όπως η σύγκλιση του μαζί με τον χρόνο εκτέλεσης του αλγορίθμου.

6.6 Συμπεράσματα

Μέθοδοι Federated learning προτάθηκαν αρχικά ώστε να ενεργοποιούν τους κόμβους-clients συνεργατικά να εκπαιδεύουν ένα μοντέλο μηχανικής μάθησης και την ίδια ώρα να διασφαλίζουν την ιδιωτικότητα για κάθε κόμβο στο νεφο-υπολογιστικό περιβάλλον. Σε αυτή την παράγραφο της παρούσας διδακτορικής διατριβής εστιάζουμε στον αλγόριθμο federated learning τύπου gradient-descent-based ο οποίος συμπεριλαμβάνει τόσο τοπικά updates των βαρών, χρησιμοποιώντας τοπικές συναρτήσεις απωλειών βαθιάς μηχανικής μάθησης και τεχνικές global aggregation. Αφού εφαρμοστούν τεχνικές global aggregation παραβλέπουμε κόμβους του υπολογιστικού με νέφους που θα μείωναν την ακρίβεια πρόγνωσης και την απόδοση του συστήματος είτε λόγω τοπικών data abnormalities είτε λόγω ελλιπούς όγκου τοπικών δεδομένων. Ένας αλγόριθμος ελέγχου επίσης χρησιμοποιείται ώστε να επιτευχθεί η βέλτιστη συχνότητα μεταξύ των τοπικών ανανεώσεων και των global aggregations ώστε τελικά να πετύχουμε ένα συνολικό ελάχιστο κόστος υπό συγκεκριμένους περιορισμούς πόρων.

Χρησιμοποιώντας διάφορους τύπους πειραματικών δεδομένων (datasets), παρουσιάσαμε μια βελτίωση από 3% έως 38% όσον αφορά την ακρίβεια πρόβλεψης σε σύγκριση με σχετικές μεθοδολογίες οι οποίες έχουν παρουσιαστεί σε προηγούμενες εργασίες [99]. Τόσο iid και non-iid dataset σενάρια έχουν ερευνηθεί στα συγκεκριμένα πειραματικά αποτελέσματα.

Παρόλα αυτά κάποιοι περιορισμοί έχουν ήδη αναφερθεί στην προηγούμενη παράγραφο και έχουν να κάνουν σχετικά με τα ακόλουθα:

- Δυσκολίες με εν δυνάμει συμμετέχοντες κόμβους που θέλουν να συμμετάσχουν στη διαδικασία μάθησης ενώ ήδη αυτή έχει ξεκινήσει.
- Πιθανά θέματα επεκτασιμότητας τα οποία μπορεί να προέλθουν εξ αιτίας του σύγχρονου τρόπου επικοινωνιών μεταξύ των συμμετεχόντων κόμβων. Ένας ασύγχρονος τρόπος επικοινωνιών μπορεί να εξεταστεί στο μέλλον σαν πιθανή βελτίωση της υπάρχουσας εργασίας. Επιπλέον στο μέλλον μεθοδολογίες κρυπτογράφησης μπορούν να εφαρμοστούν στο εκπαιδευμένο τοπικό μοντέλο και και αυτά τα κρυπτογραφικά στοιχεία να ανταλλάσσονται μεταξύ των διαφόρων κόμβων και του κεντρικού συστήματός του συστήματος FL system ναι από με αποτέλεσμα να αποφεύγονται επιθέσεις τύπου man-in-the-middle attacks, data leakages κτλ.

7 Ακριβής πρόβλεψη ζήτησης πόρων νεφουπολογιστικού συστήματος σε συσκευές περιορισμένων πόρων στην Άκρη(Edge) του συστήματος

7.1 Εισαγωγή

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο 6, η ύπαρξη του υπολογιστικού νέφους αλλά και κρίσιμες τεχνολογίες έχουν παρουσιάσει ως συνηθισμένη τακτική σε πολλές περιπτώσεις την κεντρική επεξεργασία δεδομένων που παράγονται στο άκρο του υπολογιστικού νέφους. Στις μέρες μας, υπάρχει μια μεγάλη τάση για μαζική παραγωγή δεδομένων από μεγάλο αριθμό συνεχώς αυξανόμενων συσκευών ευρισκόμενες στο «άκρο» ή “edge” του υπολογιστικού νέφους όπως λέγεται. Τέτοιες συσκευές μπορεί να είναι Wearables, Έξυπνα τηλέφωνα, Έξυπνες κάρτες, Αισθητήρες παντός είδους, Συσκευές GPS, Κινητά τηλέφωνα, Άλλες Internet of Things συσκευές.

Τα συνολικά δεδομένα τα οποία θα έχουν γεννηθεί μέχρι τον χρόνο 2025 αναμένεται να έχουν φτάσει το αστρονομικό ποσό των 175 zettabytes, περίπου 10 φορές επάνω από τα επίπεδα του έτους 2016. Όπως αναφέρθηκε και στην προηγούμενη παράγραφο 6, για την περίπτωση του Edge Cloud Computing, η τεχνική του «Federated Learning» επιτρέπει στους διάφορους συνεργαζόμενους κόμβους/τμήματα του υπολογιστικού νέφους να εφαρμόζουν ένα κοινό, στιβαρό μοντέλο μηχανικής μάθησης χωρίς ωστόσο να ανταλλάσσουν δεδομένα παρέχοντας με αυτό τον τρόπο την δυνατότητα να ικανοποιούνται κρίσιμες συνθήκες ασφάλειας όπως ασφαλή διατήρηση δεδομένων, ασφαλής πρόσβαση σε αυτά και αποφυγή επίσης άσκοπης μεταφοράς δεδομένων με αποτέλεσμα την επιβάρυνση δικτυακών πόρων όπως εύρος ζώνης κτλ. Με το «Federated Learning» οι παραπάνω τεχνικές εξελίσσονται ώστε να χρησιμοποιούν πιο επαρκή τρόπο βελτιώνοντας με αυτόν τον τρόπο την πρόγνωσης πάνω σε κατανεμημένη ροή δεδομένων.

Στη συγκεκριμένη μεθοδολογία του παρόντος κεφαλαίου, επαναλαμβάνεται όπως και στο κεφάλαιο 6, ο απόκεντρικοποιημένος τρόπος διαχείρισης δεδομένων και εκτέλεσης του αλγορίθμου Federated Learning σε πολλούς κόμβους του νεφο-υπολογιστικού συστήματος. Στο συγκεκριμένο αλγόριθμο εφαρμόζεται μια καινοτόμος μεθοδολογία κατανομής των δεδομένων μεταξύ των διαφόρων κόμβων με ποικίλους τρόπους. Η συγκεκριμένη μεθοδολογία θεωρεί πανομοιότυπη και ανεξάρτητη κατανομή τύπου δεδομένων (Identically and Independent Distributed (IID)) για τον μισό αριθμό των κόμβων που συμμετέχουν στη διαδικασία του Federated Learning χρησιμοποιώντας την ίδια κατανομή πιθανότητας. Για το έτερο ήμισυ των κόμβων του Federated Learning Cluster, χρησιμοποιούμε μη πανομοιότυπη κατανομή τύπου δεδομένων. Η συγκεκριμένη μεθοδολογία καινοτομεί στο ότι ο συνολικός αλγόριθμος Federated Learning μπορεί να συμπεριλάβει όλα τα δυνατά σενάρια καθώς η κατανομή δεδομένων δεν είναι γνωστή εκ των προτέρων.

Επιπλέον, η περιορισμένη υπολογιστική δυνατότητα αλλά και το μέγεθος της αποθήκευσης των συστημάτων που βρίσκονται στο άκρο του νεφουπολογιστικού συστήματος είναι ένα μεγάλο πρόβλημα προκειμένου να αυξηθεί η απόδοση των συστημάτων μάθησης και η δυνατότητά τους να τρέχουν σε περιορισμένους πόρους των συστημάτων στο άκρο του cloud περιβάλλοντος [148]. Προκειμένου να υπερπηδήσουμε το προαναφερθέν πρόβλημα, η υλοποίηση ενός επαρκούς federated learning συστήματος σε περιβάλλον που βρίσκεται στο άκρο του υπολογιστικού χώρου (edge-cloud environment) κρίνεται εξαιρετικά σημαντικό

[148]. Με βάση την παραπάνω θεώρηση, τα συστήματα μάθησης στο άκρο του υπολογιστικού νέφους (edge cloud systems) κυρίως δημιουργήθηκαν προκειμένου να εκτελείται το inference τμήμα του μοντέλου στο άκρο του υπολογιστικού νέφους. Με βάση τις προαναφερθείσες ερευνητικές εξελίξεις, τα συστήματα μάθησης στο άκρο (edge cloud systems) παρουσιάζουν πολλά πλεονεκτήματα όπως η κατανομή της υπολογιστικής ισχύος μεταξύ και κόμβων που βρίσκονται στο άκρο του υπολογιστικού νέφους ένα γεγονός που αυξάνει ακόμα περισσότερο την συνολική υπολογιστική ισχύ.

Παρά την μεγάλη και ευρεία έρευνα στο χώρο του Federated Learning υπάρχει ένα κενό μεταξύ του Federated Learning και του Edge Learning όπως προαναφέρθηκε και ειδικότερα μεταξύ του Tiny Machine Learning case σε συνδυασμό με το Federated Learning. Ειδικότερα το Tiny Machine Learning or Tiny ML υπόσχεται μια νέα ευκαιρία εκμάθησης μοντέλων με αρκετή ευχέρεια ειδικότερα όταν υπάρχει ένα cluster από διακομιστές (Servers) Federated Learning με χαμηλό αριθμό πόρων [149],[101]. Τα σημερινά πιο επιτυχημένα συστήματα Federated Learning που χρησιμοποιούν την περίπτωση των tiny systems εκτελούν την τεχνική του model Inference πάνω στην συσκευή την ίδια που βρίσκεται στο άκρο του υπολογιστικού νέφους[85]. Ένα από τα βασικότερα μειονεκτήματα της σχετιζόμενης έρευνας είναι το γεγονός ότι υπάρχει έλλειψη ενός γενικού framework όσον αφορά το TinyML το οποίο να είναι ανεξάρτητο παρόχου (vendor-agnostic) και καθώς επίσης το γεγονός ότι υπάρχει ένας περιορισμός όσον αφορά την υπολογιστική διαθεσιμότητα και το μέγεθος αποθήκευσης δεδομένων στα συστήματα edge.

Η τρέχουσα ενότητα 7, παρουσιάζει το TensorFlow Lite Micro (TFLM)[1] σε συνδυασμό με την τεχνική του γνωστού από την παράγραφο 6 Federated Learning προκειμένου να απαντήσει στις παραπάνω ερωτήσεις [A6]. Το TFLM κάνει εύκολη την εφαρμογή τύπου Tiny ML οι οποίες τρέχουν σε περιβάλλοντα πολλών νεφό-υπολογιστικών παροχών και δίνει στους διάφορους παρόχους μια ανεξάρτητη πλατφόρμα προκειμένου να την χρησιμοποιήσουν προς όφελός τους. Σχετικά με το κομμάτι των προβλέψεων πάνω σε δεδομένα πραγματικού χρόνου όσον αφορά τον αλγόριθμο inference, στη συγκεκριμένη παράγραφο εφαρμόζουμε μια μέθοδο δέντρου αποφάσεων η οποία χρησιμοποιείται μετά την ολοκλήρωση της διαδικασίας εκπαίδευσης του αλγορίθμου Federated Learning[A6].

[1] https://www.tensorflow.org/lite

7.2 Σχετικές Εργασίες

Σε αυτή την παράγραφο παρουσιάζουμε σχετιζόμενες ερευνητικές εργασίες με τους μηχανισμούς Federated Learning (FL) σε συστήματα πολλών παροχών σε συνεργασία με τεχνολογία Tiny ML και τις συγκρίνουμε με την τρέχουσα εργασία. Με την μεγάλη αύξηση των συσκευών ευρισκομένων στο άκρο του υπολογιστικού νέφους όπως συστήματα, αισθητήρες IoT σε συσχέτιση με τις καινοτομίες στις τεχνικές του Federated Learning [85], [102], [150] μπορούν να εφαρμοστούν στο άκρο και στους διακομιστές των υπολογιστικών παρόχων συμφέρουσες τεχνικές ώστε να προκύψει ένα συνεργατικό σχήμα σε εκπαίδευση και εκμάθηση αλγόριθμων. Σημαντικές παράμετροι όπως σύγχρονες ή ασύγχρονες επικοινωνίες, συναρτήσεις βαθιάς μηχανικής μάθησης τοπικών απωλειών τύπου convex/ non-convex, μεθοδολογίες aggregation αλλά και τεχνικές επιλογών ερευνώνται και εξετάζονται σε συνδυασμό με τη χρήση τεχνολογίας Tiny ML για τους κόμβους που βρίσκονται στο άκρο του υπολογιστικού νέφους (edge). Δυστυχώς περιορισμοί λόγω λιγοστών διαθέσιμων πόρων σε συσκευές ευρισκόμενες στο άκρο του υπολογιστικού νέφους μαζί με χρήση τεχνολογίας Tiny ML δεν εξετάζονται ή δεν έχουν ερευνηθεί καθόλου στις ακόλουθες αναφορές.

Στην εργασία[90] μια συγκεκριμένη μέθοδος επικοινωνίας ασύγχρονου τρόπου χρησιμοποιείται σε ένα σχήμα εκμάθησης federated learning όπου οι συσκευές στο άκρο συνδέονται κεντρικά προς έναν διακομιστή(server) στο cloud. Συγκεκριμένες συναρτήσεις απωλειών οι οποίες χρησιμοποιούν εξελιγμένα βαθιά Νευρωνικά δίκτυα σε συνεργασία με συναρτήσεις convex λειτουργιών εφαρμόζονται μαζί με μηχανισμό aggregation σαν τμήμα ενός συστήματος FL system. Τα αποτελέσματα δείχνουν να είναι αρκετά καλά. Στο συγκεκριμένο σύστημα ωστόσο δεν υπάρχει τεχνική global aggregation σύμφωνα με το μέγεθος των τοπικών δεδομένων σε κάθε κόμβο ούτε κάποια τεχνική επιλογής κόμβων βασισμένη σε ένα κατώφλι τιμής της τοπικής συνάρτησης απωλειών (loss threshold) ανά κόμβο. Επιπλέον, η παράμετρος περιορισμένων πόρων η οποία υφίσταται σε συσκευές(servers) ή διακομιστές ευρισκόμενους στο άκρο του υπολογιστικού νέφους προκειμένου το κομμάτι του inference να μπορεί να τρέξει, δεν εξετάζεται καθόλου στη συγκεκριμένη έρευνα. Η ίδια ασύγχρονη μέθοδος επικοινωνίας παρουσιάζεται στην εργασία [104], όπου επιτυγχάνονται πολύ καλά αποτελέσματα χρησιμοποιώντας τεχνικές γραμμικής σύγκλισης σε ένα ολικό βέλτιστο σημείο (global optimum). Συναρτήσεις απωλειών τύπου Convex εφαρμόζονται αλλά όχι τοπικοί αλγόριθμοι νευρωνικών δικτύων βαθιάς μηχανικής μάθησης. Επιπλέον, η περίπτωση χρήσης συστημάτων λειτουργίας στο άκρο ενός περιβάλλοντος υπολογιστικού νέφους πολλών παροχών (edge systems) με περιορισμούς πόρων και εφαρμογή inference με χρήση τεχνολογίας tiny ML δεν ερευνάται καθόλου.

Σχετικά με τον τομέα του υπολογιστικού νέφους για κινητές συσκευές στο άκρο (mobile edge computing) η εργασία [93] χρησιμοποιεί federated learning ώστε να κάνει ακριβείς προβλέψεις για αιτήματα χρήσης πόρων σχετικά με γνωστούς τύπους εφαρμογών στο διαδίκτυο. Μια πολύ καλή ακρίβεια πρόγνωσης επιτυγχάνεται εφαρμόζοντας ένα σύγχρονο τρόπο επικοινωνίας μεταξύ των κινητών συσκευών και των διακομιστών (servers) στο υπολογιστικό νέφος. Επιπροσθέτως και προκειμένου να αυξηθεί η ακρίβεια πρόγνωσης των μοντέλων χρησιμοποιήθηκε ο τύπος των τοπικών συναρτήσεων απωλειών που λέγεται convex local loss functions. Κατά τη διάρκεια του aggregation του FL, ένα μοντέλο αποτελούμενο από βάρη και εκτιμώμενο με βάση τα εκάστοτε μεγέθη των τοπικών

δεδομένων των κόμβων χρησιμοποιείται χωρίς όμως επιλογή των κατάλληλων κόμβων για το global aggregation. Η τεχνολογία Tiny ML στο Inference δεν εξετάζεται καθόλου.

Θεωρώντας την γενική παραδοσιακή τεχνική κατανεμημένης μηχανικής μάθησης, η εργασία [91] παρουσιάζει μια καινοτόμο μεθοδολογία Federated Learning όπου συμπιεσμένα δεδομένα ανταλλάσσονται μεταξύ των κόμβων και του νεφο-υπολογιστικού διακομιστή (cloud server) λειτουργώντας ως ένα πολύ καλό security control χωρίς να επηρεάζεται αρνητικά η ακρίβεια προβλέψεων. Παρ' όλα αυτά, δεν χρησιμοποιούνται στη συγκεκριμένη εργασία καθόλου βαθιάς μηχανικής μάθησης Νευρωνικά δίκτυα ούτε γίνεται κάποια επιλογή κόμβων, όπως επίσης δεν αναφέρεται καμία χρήση τεχνικής για Tiny ML inference.

Στην ερευνητική εργασία[108] γίνεται μια αναζήτηση και ανάλυση κάποιων αλγορίθμων Federated Learning Aggregation με εφαρμογή στην αναγνώριση ανθρώπινης δραστηριότητος. Οι αλγόριθμοι οι οποίοι εξετάζονται για σκοπούς aggregation είναι οι FedAvg, FedPer και FedMA. Τα αποτελέσματα των πειραμάτων δείχνουν μια μέτρια ακρίβεια όσον αφορά την ταξινόμηση και την ανίχνευση της ανθρώπινης δραστηριότητας. Στη συγκεκριμένη περίπτωση δεν έχουν ληφθεί υπόψη ανομοιογένειες και ανωμαλίες δεδομένων (data abnormalities) όπως επίσης δεν έχει αναλυθεί καμία τεχνολογία τύπου Tiny ML Inference. Μια άλλη εργασία στην οποία λαμβάνεται υπόψη η ετερογένεια των δεδομένων σε ένα σύστημα Federated Learning είναι η [109]. Δυστυχώς και σε αυτή την εργασία, η περίπτωση των ανωμαλιών στα δεδομένα δεν ερευνάται σαν πιθανότητα ούτε εξετάζονται περιορισμοί στους υπολογιστικούς πόρους, γεγονότα τα οποία μπορούν να εμφανιστούν σε ένα νέφος υπολογιστικού περιβάλλοντος στο άκρο του(edge cloud environment) για την περίπτωση προγνώσεων.

Στην εργασία[99] παρουσιάζεται ένα εξελιγμένο σύστημα Federated Learning, όπου ένας αλγόριθμος gradient descent χρησιμοποιείται τοπικά σε κάθε κόμβο σε συνδυασμό με έναν αλγόριθμο προσαρμοσμένου ελέγχου πόρων. Στην προαναφερθείσα εργασία ο αλγόριθμος ελέγχου βρίσκει κάθε φορά τη βέλτιστη συχνότητα μεταξύ των τοπικών και των καθολικών updates (global aggregation). Μόλις η βέλτιστη συχνότητα επιτευχθεί η καθολική συνάρτηση απωλειών ή αλλιώς global loss function τείνει σε μια ελάχιστη τιμή και την ίδια στιγμή ένα βέλτιστο σημείο στην χρήση των υπολογιστικών πόρων του συστήματος έχει κατορθωθεί. Παρά το γεγονός ότι τα τελικά αποτελέσματα δείχνουν μια καλή ακρίβεια πρόγνωσης πουθενά δεν παρατηρείται χρήση βαθέων και εξελιγμένων νευρωνικών δικτύων ούτε παρατηρείται μεθοδολογία επιλογής κόμβων κατά τη διάρκεια του global aggregation. Αυτές οι περιπτώσεις λαμβάνονται υπόψη στην ερευνητική εργασία στο περιοδικό[A5] όπου η ακρίβεια φαίνεται να αυξάνεται κατακόρυφα. Και στις 2 προαναφερθείσες ερευνητικές εργασίες, υπολογιστικοί περιορισμοί στο άκρο του υπολογιστικού νέφους για την εκτέλεσή του τμήματος Inference του αλγορίθμου δεν αναφέρονται καθόλου. Επιπλέον δεν υπάρχει αναφορά σε χρήση Tiny ML τεχνολογίας γεγονός που θα βοηθούσε πολύ την επίλυση των περιορισμών στη χρήση υπολογιστικών πόρων στο άκρο του υπολογιστικού νέφους (edge resources constraints problem).

Από την άλλη μεριά, στην εργασία [148], το ανοιχτού κώδικα framework ML (TFLM) Tensorflow Lite Micro παρουσιάζεται σαν ένα framework ML Inference για να τρέχει αλγορίθμους βαθιάς μηχανικής μάθησης σε συστήματα με περιορισμό υπολογιστικών πόρων και σε συσκευές όπως τα ενσωματωμένα συστήματα. Το TFLM αντιμετωπίζει και δίνει εξαιρετικές λύσεις όσον αφορά τους περιορισμούς στους πόρους και οποίοι συνήθως παρουσιάζονται σε συστήματα στο άκρο του υπολογιστικού νέφους και τους περιορισμούς

σχετικά με λειτουργία της πλατφόρμας σε πέραν του ενός υπολογιστικού παρόχους πράγμα αρκετά καινοτόμο. Παρ’ όλα αυτά η περίπτωση της υιοθέτησης ενός σεναρίου Federated Learning δεν εξετάζεται καθόλου στην προαναφερθείσα εργασία όταν χρειάζεται να χρησιμοποιηθεί η αρχιτεκτονική server client σε πέραν του ενός υπολογιστικούς παρόχους.

Σε αντίθεση με την παραπάνω αναφερθείσα έρευνα, η τρέχουσα παράγραφος της διδακτορικής διατριβής [A6] παρουσιάζει και καινοτομεί στο ότι έχει δημιουργήσει μια λύση σε ένα πρόβλημα δυναμικού υπολογισμού της συχνότητας μεταξύ των global aggregations χρησιμοποιώντας συγκεκριμένο χρονικό περιορισμό(time budget) ενώ ταυτόχρονα η ακρίβεια πρόγνωσης δείχνει σημαντικά βελτιωμένη σε σχέση με την υπόλοιπη έρευνα. Επιπρόσθετα ένα δέντρο απόφασης για τη βελτιστοποίηση του συστήματος προκειμένου να αποφασίζει κάθε φορά το εάν ένα απλά εκπαιδευμένο μοντέλο θα χρησιμοποιηθεί για το Inference ή ένα περαιτέρω βελτιωμένο και κβάντοποιημένο μοντέλο Tiny ML χρειάζεται να χρησιμοποιηθεί για την διαδικασία του Inference σαν έξτρα λειτουργικότητα. Η βελτιστοποίηση του μοντέλου Inference με τη χρήση τεχνολογίας Tiny ML εξασφαλίζει υψηλότερη ακρίβεια προγνώσεων από την κλασική περίπτωση χρήσης εκπαιδευμένου μοντέλου Federated Learning, ενώ ταυτόχρονα πετυχαίνει και σημαντική μείωση στο μέγεθος του μοντέλου, ένα γεγονός πολύ βασικό στην χρήση μοντέλων σε περιβάλλοντα υπολογιστικού νέφους στο άκρο τους (Edge Multi Cloud environments) [A6].

7.3 Ερευνητική Προσέγγιση και Αρχιτεκτονική Αλγορίθμου

7.3.1 Γενική Προσέγγιση

Στην παρούσα παράγραφο παρουσιάζουμε εν συντομία την γνωστή αρχιτεκτονική αλγορίθμου όπως παρουσιάστηκε στην προηγούμενη ενότητα 6.3.1 Federated Learning όπου χρησιμοποιήσαμε τον αλγόριθμο gradient descent για να ελαχιστοποιήσουμε την συνολική συνάρτηση απωλειών σε ένα περιβάλλον όπου συνυπάρχουν πολλοί παροχή υπολογιστικού νέφους όπως επίσης και πολλοί κόμβοι στο άκρο τους. Υπενθυμίζουμε την συνάρτηση γενικών απωλειών ως ακολούθως:

$$w^* \triangleq argmin F(w) \ (31)$$

Στην παραπάνω εξίσωση w^* είναι η παράμετρος διάνυσμα του συνολικού μοντέλου όταν φτάνει στο ολικό ελάχιστο. F(w) είναι οι καθολικοί συνάρτηση απωλειών που υπολογίζεται μετά την διαδικασία του global aggregation και λαμβάνει χώρα στον κεντρικό διακομιστή (server) σαν αποτέλεσμα συλλογής δεδομένων από όλους τους συμμετέχοντες κόμβους σύμφωνα με την ακόλουθη εξίσωση:

$$F(w) = \frac{\sum_{i=1}^{N} D_i F_i(w)}{D} \ (32)$$

Στην παραπάνω εξίσωση η παράμετρος N ορίζει τον αριθμό των συμμετεχόντων κόμβων (client or edge nodes), η παράμετρος D_i ορίζει το μέγεθος των δεδομένων για κάθε τοπικό κόμβο και η παράμετρος D_i δίνει το συνολικό μέγεθος ολόκληρου του dataset του συστήματος FL multi-cloud. Έτσι, το συνολικό μέγεθος των δεδομένων δίνεται από την ακόλουθη εξίσωση:

$$D \triangleq \sum_{i=1}^{N} D_i \quad (33)$$

Υπενθυμίζουμε ότι όπως αναφέρθηκε και στην προηγούμενη ενότητα 6.3.1 μετά από κάθε διαδικασία global aggregation οι παράμετροι των μοντέλων μηχανικής μάθησης (διανύσματα παραμέτρων) διαμορφώνονται με βάση την ακόλουθη εξίσωση:

$$w_i(t) = \widetilde{w_i}\,(\mathrm{t}-1) - \mathrm{n}\,\nabla F_i\,(\widetilde{w_i}\,(t-1) \ (34)$$

όπου t είναι η τρέχουσα ολοκλήρωση (iteration) και t-1 η προηγούμενη ολοκλήρωση (t-1), και n είναι η παράμετρος που δείχνει το ρυθμό εκμάθησης του αλγορίθμου (learning rate).

Η υπόλοιπη περιγραφή της συγκεκριμένης διαδικασίας σε λεπτομέρεια έχει παρουσιαστεί ενδελεχώς στην προηγούμενη ενότητα 6.3.1.

7.3.2 Τεχνική Εκτέλεσης Inference με χρήση τεχνολογίας TINY ML

Σε συνέχεια της προηγούμενης παραγράφου όπου δόθηκε ο τρόπος εκπαίδευσης ενός κατανεμημένου αλγόριθμου Federated Learning, εκκρεμεί ένα τελευταίο βήμα που αφορά το κομμάτι της δοκιμασίας της απόδοσης και εφαρμογής του αλγορίθμου σε πραγματικά δεδομένα. Αυτό το τμήμα αφορά το επονομαζόμενο Inference part του αλγορίθμου. Στη τρέχουσα παράγραφο εξετάζεται η εκτέλεση αυτού του τμήματος του αλγορίθμου σε κόμβους όπου υπάρχουν στο άκρο(Edge) ενός περιβάλλοντος ύπαρξης πέραν του ενός νεφο υπολογιστικών παροχών. Η συγκεκριμένη περιγραφείσα τοπολογία εισάγει στην συζήτηση τους περιορισμούς όσον αφορά τους υπολογιστικούς πόρους και συγκεκριμένα χρήση μνήμης ή επεξεργαστική ισχύς. Η περίπτωση της εκτέλεσης του Inference τμήματος του αλγορίθμου Federated Learning στη συγκεκριμένη περίπτωση, προϋποθέτει ότι θα εκτελείται σε πραγματικά δεδομένα τα οποία ευρίσκονται σε συσκευές(devices) ή διακομιστές(servers) στο άκρο του υπολογιστικού νέφος που όμως διαθέτουν μικρή ποσότητα δεδομένων και μικρή υπολογιστική ισχύ. Το επονομαζόμενο Edge computing αποτελεί την ανάγκη του σύγχρονου κόσμο λόγω των μεγάλων καινοτομιών που παρατηρούνται στο χώρο των IoT domain καθώς και λόγω της συμμόρφωσης και των νόμων προστασίας δεδομένων που υποχρεώνουν τις εταιρείες να κάνουν τις αντίστοιχες προγνώσεις και υπολογισμούς πάνω σε δεδομένα τα οποία βρίσκονται στο άκρο του υπολογιστικού νέφους αντί σε κεντρικούς διακομιστές (servers) ως μια νόμιμη τεχνική.

Πέραν των ανωτέρω αναφερθέντων, η μη ύπαρξη ενός συγκεκριμένου και καθολικού framework για την ανάπτυξη και βελτιστοποίηση μοντέλων που τρέχουν τη διαδικασία Inference σε κάποια “μικρή” συσκευή η διακομιστή στο άκρο του υπολογιστικού νέφους σημαίνει ότι αυτό το μοντέλο θα πρέπει να αναπτύσσεται από τον εκάστοτε προγραμματιστή. Δυστυχώς ο κάθε κατασκευαστής υλικού(hardware) έχει συγκεκριμένες και ξεχωριστές ανάγκες που πρέπει να ικανοποιηθούν. Χωρίς την ύπαρξη ενός γενικού framework Tiny ML όπου θα αξιολογεί την απόδοσή του υλικού(hardware) με έναν τρόπο ουδέτερο και ανεξάρτητο από τον έκαστο κατασκευαστή το προαναφερθέν πρόβλημα δεν θα μπορέσει να επιλυθεί.

Λόγω όλων των προ-αναφερθέντων, οι τεχνολογίες TensorFlow Lite και TensorFlow Model Optimization Toolkit, παρέχουν εργαλεία προκειμένου να βελτιώνεται ο τρόπος εκτελέσεις της διαδικασίας Inference, σχετικά με τους περιορισμούς σε κατανάλωση μνήμης και υπολογιστική ισχύ. Γενικότερα η βελτιστοποίηση του μοντέλου για την εκτέλεσή του Inference είναι ένα πολύ δύσκολο case το οποίο ερευνάται και αναλύεται στην τρέχουσα παράγραφο.

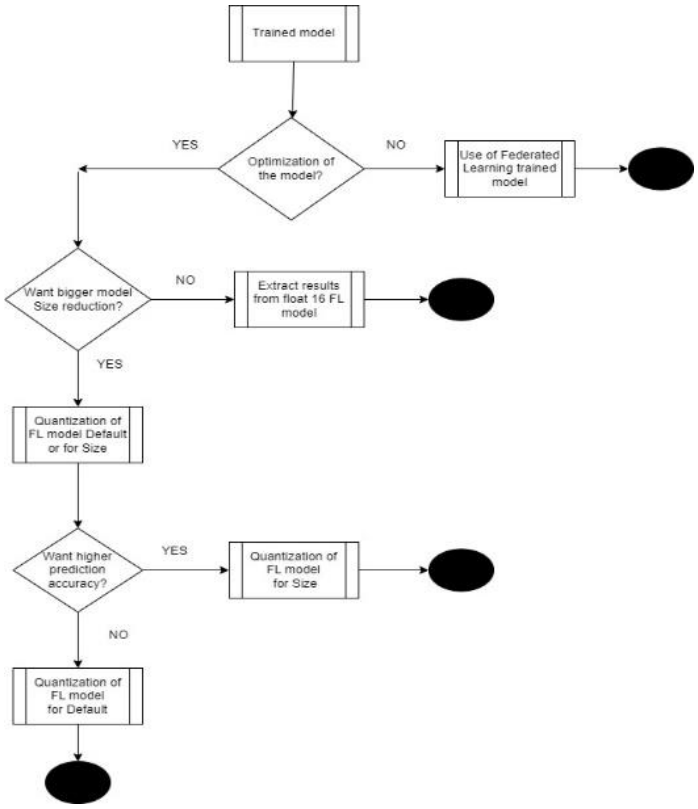
Οι παραπάνω απαιτήσεις φέρνουν στην επιφάνεια συνεπώς την υιοθέτηση του συγκεκριμένου Tiny ML framework του TensorFlow Lite. Κατόπιν συγκεκριμένης έρευνας, το framework το οποίο έχει ενταχθεί στο τμήμα Inference του υλοποιημένου μας FL Algorithm Εστιάζει στην ικανοποίηση των ακόλουθων κριτηρίων:

- Μικρό μέγεθος του εκπαιδευμένου μοντέλου και ακολούθως μικρός χρόνος και εύρος ζώνης προκειμένου να γίνει download στα edge devices των χρηστών.

- Μικρός χρόνος για την διαδικασία πρόβλεψης πάνω σε πραγματικά δεδομένα.
- Μικρό μέγεθος κατανάλωσης υπολογιστικών πόρων σχετικά με CPU or / and RAM / Disk Consumption.
- Αύξηση της ακρίβειας πρόγνωσης.

Αν και στις περισσότερες ερευνητικές περιπτώσεις τα παραπάνω κριτήρια φαίνονται αντιφατικά μεταξύ τους, στην περίπτωση του μοντέλου μας Federated Learning με τη χρήση τεχνολογίας Tiny ML για το τμήμα του TF Lite Inference model, η βελτιστοποίηση σε μέγεθος και σε χρονική καθυστέρηση δεν προκαλεί οποιαδήποτε μείωση στην ακρίβεια προγνώσεων. Αυτό είναι και πολύ καινοτόμο επίτευγμα για όλο το σύστημα Federated Learning που αναπτύχθηκε σε σύγκριση με άλλες ερευνητικές προηγούμενες δουλειές [A5]. Είναι πολύ σημαντικό να αναφέρουμε ότι αν και στις περισσότερες περιπτώσεις της χρήσης κβαντοποίησης στο μοντέλο TF lite υπάρχει μια πολύ μικρή μείωση γενικά στην ακρίβεια πρόγνωσης, στη δική μας εργασία πετύχαμε τουναντίον αύξηση προγνώσεων.

Κατά τη διάρκεια της βελτιστοποίησης του μοντέλου Inference, έχουν εφαρμοστεί διάφορες μέθοδοι κβαντοποίησης όπως μείωση μεγέθους μοντέλου, κβαντοποίηση τύπου float16 και κβαντοποίηση δυναμικής περιοχής. Έτσι μετά την εκπαίδευση του μοντέλου Federated Learning, η εκτέλεση του Inference στο άκρο του υπολογιστικού συστήματος λαμβάνει χώρα σύμφωνα με το ακόλουθο δένδρο αποφάσεων. Στο ακόλουθο σχήμα 37, το δέντρο βοηθάει στο να επιλέξουμε τον τρόπο βελτιστοποίησης και κβαντοποίησης που θα χρησιμοποιηθεί στο μοντέλο μας βασισμένοι απλά στο αναμενόμενο μέγεθος του μοντέλου και ακρίβεια του:



Σχήμα 37 – Δένδρο αποφάσεων για την βελτιστοποίηση του μοντέλου του νεφουπολογιστικού συστήματος Federated Learning πολλών παρόχων.

Σύμφωνα με το ανωτέρω σχήμα, μετά την εκπαίδευσή του FL algorithm, μια πράξη περαιτέρω δράσης λαμβάνει χώρα που αφορά την περαιτέρω βελτιστοποίηση του μοντέλου για το τμήμα του Inference. Για αυτό το λόγο ένα δένδρο λήψης αποφάσεων χρησιμοποιείται εάν ο σκοπός είναι η βελτιστοποίηση τότε μια εξέταση τις περιπτώσεις μείωσης του μεγέθους εφαρμόζεται. Εάν ο βασικός στόχος δεν είναι οι περαιτέρω μείωση του τρέχοντος μοντέλου τότε μια κβαντοποίηση τύπου float 16 model χρησιμοποιείται στο τμήμα Inference. Εάν μια επίτευξη μεγαλύτερης ακρίβειας πρόγνωσης είναι ο βασικός στόχος τότε η τεχνική του default quantization εφαρμόζεται.

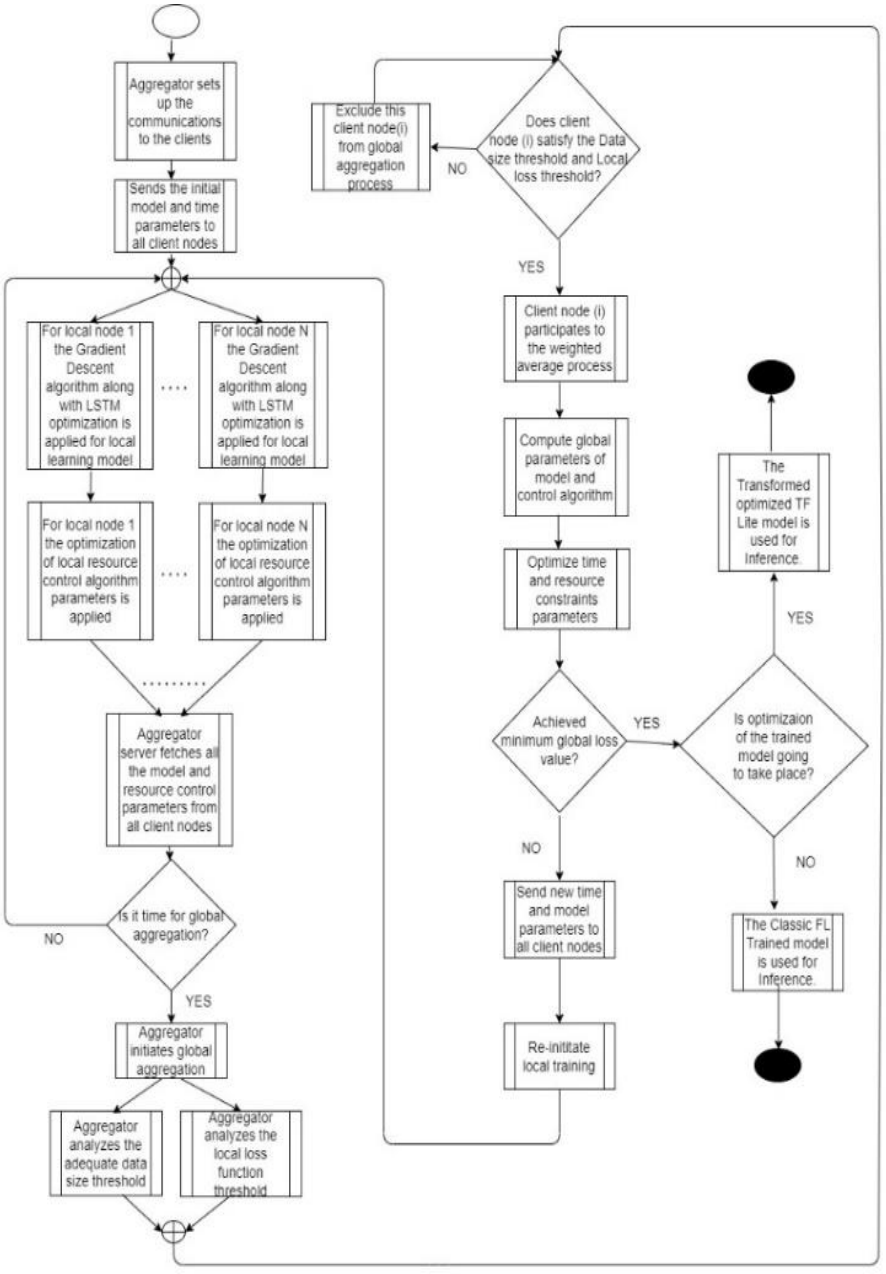
Είναι σημαντικό να αναφερθεί ότι για την περίπτωση της μεθόδου κβαντοποίησης τύπου δεδομένων float 16 data format, ένα αντιπροσωπευτικό dataset χρησιμοποιείται. Το TensorFlow Lite υποστηρίζει την μετατροπή των βαρών του μοντέλου σε τιμές τύπου 16-bit floating point κατά τη διάρκεια μετατροπής από το κλασικό TensorFlow στο TensorFlow Lite's flat buffer format. Αυτό επιδρά σε μια διπλάσια μείωση του μεγέθους του ολικού μοντέλου. Προκειμένου να κβαντοποιήσουμε τα δεδομένα των μεταβλητών όπως την είσοδο-έξοδο των ενδιάμεσων επιπέδων του μοντέλου, χρειάζεται να παρέχουμε ένα αντιπροσωπευτικό Dataset το οποίο γεννάται με τη βοήθεια μιας γεννήτριας δεδομένων με την χρήση συγκεκριμένης εισόδου δεδομένων.

Συμπερασματικά όταν το βελτιστοποιημένο μοντέλο του Tensorflow Lite καλείται να εκτελεστεί από το δέντρο υποστήριξης αποφάσεων στον αλγόριθμο μας Federated Learning, η διαδικασία deployment για το κομμάτι του Inference μοντέλου σε συσκευές/διακομιστές λίγων πόρων όσον αφορά το υλικό (hardware) ευρισκόμενες στο άκρο(edge) του υπολογιστικού νέφους έχει ως ακολούθως:

1. Φόρτωση του μοντέλου σε μορφή .tflite αρχείου.
2. Μετασχηματισμός των δεδομένων εισόδου raw.
3. Εκτέλεσή του inference χρησιμοποιώντας το TensorFlow Lite API που περιλαμβάνει ενέργειες όπως χτίσιμο του interpreter (building the interpreter) και καθορισμό tensor παραμέτρων (allocating tensors).
4. Μετατροπή της εξόδου του τελικού μοντέλου χρησιμοποιώντας τους tensors ώστε το μοντέλο αυτό να είναι χρήσιμο για να χρησιμοποιηθεί από κάθε είδους εφαρμογή.

7.3.3 Συνολική ροή δεδομένων του συνολικού αλγορίθμου με χρήση Tiny ML

Όπως περιεγράφηκε στις προηγούμενες παραγράφους 7.3.1 και 7.3.2 τόσο για το κομμάτι εκπαίδευσης του συνολικού αλγορίθμου όσο και για το κομμάτι Inference, η συνολική ροή όπου περιγράφει όλα τα βήματα του αλγορίθμου μας για το FL Multi cloud and edge σύστημα, φαίνεται ως ακολούθως:



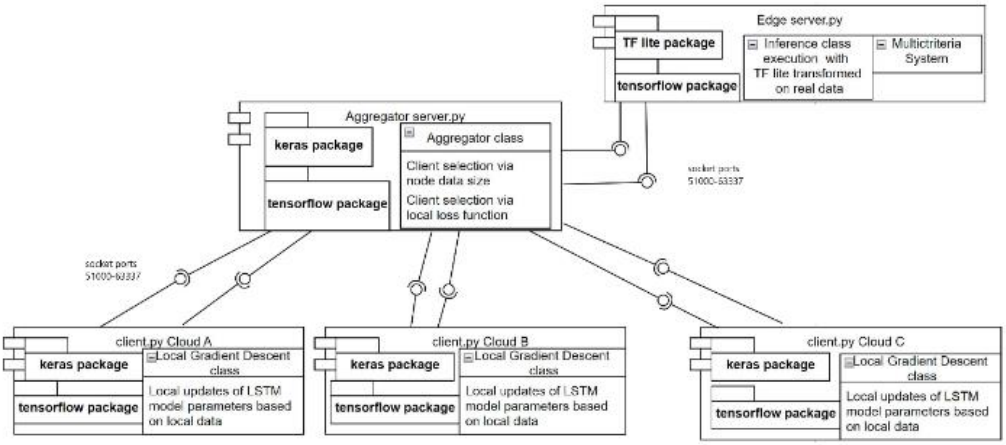
Σχήμα 38 – Γενικό Διάγραμμα της ροής του αλγορίθμου Federated Learning στο Άκρο (Edge) του υπολογιστικού Νέφους πολλών παρόχων.

7.4 Αρχιτεκτονική Υλοποίησης του Λογισμικού

Στην παρούσα παράγραφο δίνουμε μια περιγραφή του αρχιτεκτονικού σχήματος και της τεχνολογίας υλοποίησης του προτεινόμενου συστήματος αλγορίθμου. Πιο συγκεκριμένα μια από-κεντρικοποιημένη αρχιτεκτονική Πελάτη-Εξυπηρετητή (Client-Server) πολλών νεφο-υπολογιστικών παροχών αποτελούμενη και από κόμβους ευρισκόμενους στο άκρο με χρήση αλγορίθμου Federated Learning αναλύεται. Το συγκεκριμένο σχήμα κατανέμεται σε πολλούς νέφο-υπολογιστικούς παρόχους πολλών κόμβων. Ο κεντρικός διακομιστής (aggregation server) μπορεί να ευρίσκεται σε οποιοδήποτε νέφος υπολογιστικό περιβάλλον και οποιοδήποτε κόμβο από τους συμμετέχοντες παρόχους. Κάθε κόμβος(client) εκτελεί τοπικα επεξεργασία δεδομένων καθώς επίσης και απομακρυσμένες ενέργειες. Το τελικό βήμα του Inference μπορεί να λάβει χώρα σε μια μικρή συσκευή ή διακομιστή ευρισκόμενη στο άκρο ακολουθώντας μια αρχιτεκτονική τύπου Tiny ML το οποίο σημαίνει ότι μπορεί να γίνεται offload & push του εκπαιδευμένου μοντέλου σε έναν μικρό διακομιστή στο άκρο. Αυτό το σύστημα το οποίο βρίσκεται κοντά στον τελικό χρήστη ονομάζεται " Inference Edge node".

Όπως έχει ήδη αναφερθεί για το για το προσαρμόσιμο σύστημα Federated Learning υλοποιημένο σε γλώσσα Python και χρησιμοποιώντας τα packages Tensorflow & Keras, η διαδικασία εκμάθησης ξεκινάει με τοπικα βήματα εκμάθησης σε κάθε κόμβο όπου η βελτιστοποίηση των παραμέτρων επιτυγχάνεται χρησιμοποιώντας τον αλγόριθμο long short term memory (LSTM). Χρησιμοποιώντας τον βελτιστοποιητή gradient descent με εφαρμογή και χρυσή του ειδικού αλγορίθμου LSTM, επιτρέπει σε εξαρτήσεις κοντινής μνήμης να γίνεται εκμάθηση σε σειριακά δεδομένα και ακολούθως οι τοπικοί παράμετροι του μοντέλου να ανανεώνονται αντίστοιχα. Η τοπική συνάρτηση απωλειών του κάθε κόμβου ελαχιστοποιείται βασιζόμενη στο περιεχόμενο δεδομένων που βρίσκεται σε κάθε κόμβο. Μόλις οι ανανεώσεις των τοπικών παραμέτρων των κόμβων ολοκληρωθούν, ο κεντρικός διακομιστής ή αλλιώς global aggregation server περιμένει να εκτελέσει την διαδικασία του global aggregation κάποιο σημείο απομακρυσμένα στο νέφος. Οι κόμβοι οι οποίοι συμμετέχουν σε αυτή τη διαδικασία aggregation είναι καταλλήλως επιλεγμένοι. Μετά τα βήματα aggregation και τις ανανεώσεις των παραμέτρων, μόλις ένα ολικό ελάχιστο στην ολική συνάρτηση απωλειών επιτευχθεί, η διαδικασία του Inference βασισμένου σε πραγματικά δεδομένα τα οποία ευρίσκονται στον διακομιστή στο άκρο(edge server) ακολουθείτε χρησιμοποιώντας τα πλεονεκτήματα της βελτιστοποίησης του Tensorflow Lite package. Σε εκείνη τη χρονική στιγμή, η αρχιτεκτονική μας μπορεί να επιλέξει μια συσκευή στο άκρο με ελαττωμένου ς υπολογιστικούς πόρους π.χ. RAM & CPU, ώστε να λάβουμε τελικά αποτελέσματα προγνώσεων ακολουθώντας το δένδρο υποστήριξης αποφάσεων όπως φάνηκε στο σχήμα 37 προηγουμένως.

Η βασική λογική της εργασίας μας και της υλοποίησης μπορεί να δειχθεί στο ακόλουθο διάγραμμα UML (Σχήμα 39) όπου παρουσιάζει τα βασικά χρησιμοποιούμενα packages keras & Tensorflow καθώς επίσης και διεπαφές τεχνολογίας Python ή αλλιώς «interfaces modules of Python»:



Σχήμα 39 – Γενική Αρχιτεκτονική Λογισμικού του αλγορίθμου Federated Learning στο Άκρο (Edge) του υπολογιστικού Νέφους πολλών παρόχων.

7.5 Αξιολόγηση

7.5.1 Πειραματικό Περιβάλλον-Setup

Σε αυτή την παράγραφο, παρουσιάζουμε μια αξιολόγηση της προτεινόμενης αρχιτεκτονικής του νεφο υπολογιστικού συστήματος Federated Learning πολλών παροχών με έκταση στο άκρο χρησιμοποιώντας τεχνολογία Tiny ML. Στη συγκεκριμένη πειραματική αρχιτεκτονική χρησιμοποιούμε διάφορους τύπους δεδομένων σχετικών με κατανάλωση επεξεργαστικής ισχύος (datasets over CPU consumption) χρησιμοποιώντας 2 κόμβους και ένα κόμβο με περιορισμένους πόρους ευρισκόμενος στο άκρο του υπολογιστικού νέφους. Στη συνέχεια αυξάνουμε τον αριθμό των κόμβων σε 5 και επαναλαμβάνουμε το όλο πείραμα.

Στο πείραμα που εκτελούμε στην παρούσα παράγραφο, χρησιμοποιούμε συλλογές δεδομένων (datasets) που έχουν να κάνουν με κατανάλωση υπολογιστικών πόρων που χρησιμοποιεί μια εφαρμογή εξομοίωσης υπολογισμών για τη δυναμική υγρών. Η συγκεκριμένη εφαρμογή δείχνει τα αποτελέσματα του μοντέλου δεδομένων όσον αφορά την κινητικότητα των υγρών και αναφέρεται σε ένα πιλοτικό παράδειγμα του έργου από την εταιρεία ICON company (https://www.iconcfd.com/ (accessed on 1 September 2023). Κατά την διάρκεια αξιολόγησης των πειραματικών αποτελεσμάτων, χρησιμοποιήσαμε 2 κατηγορίες εξομοιώσεων: low-fi (low-fidelity) & hi-fi (high-fidelity). Ο πρώτος τύπος εξομοιώσεων ο οποίος ονομάζεται low-fi, χρησιμοποιεί μεσαίο αριθμό pixels στις εικόνες το οποίο σημαίνει χαμηλή κατανάλωση επεξεργαστικής ισχύος για την αναγκαία επεξεργασία δεδομένων, καθότι η συγκεκριμένη περίπτωση εξομοιώσεων χρησιμοποιείται για εκπαιδευτικούς σκοπούς(Σχήμα 40). Ο έτερος τύπος εξομοιώσεων, επονομαζόμενος και ως hi-fi, χρησιμοποιεί αρκετούς υπολογιστικούς πόρους, και συγκεκριμένα επεξεργαστική ισχύ, καθότι αναπαριστά μεγάλο αριθμό από pixels εικόνων για εξομοιώσεις σε πραγματικά παραγωγικά περιβάλλοντα(Σχήμα 41).

Για τις ανάγκες της εξομοίωσης/πειράματος απαιτείται χρήση αυτοματοποιημένου συστήματος για τη διευθέτηση και τη λειτουργία συστημάτων εξοπλισμού και λογισμικού. Στη συγκεκριμένη περίπτωση χρειάστηκε να εξεταστεί το θέμα της ενορχήστρωσης (orchestration), στην οποία ένα αυτοματοποιημένο σύστημα διευθετεί, ελέγχει και διαχειρίζεται όλες τις πτυχές μιας υπηρεσίας. Σε αντίθεση με ένα γενικό εργαλείο αυτοματοποίησης, που εστιάζει σε μια πτυχή λειτουργίας ενός κέντρου δεδομένων, ένα σύστημα ενορχήστρωσης συντονίζει όλα τα υποσυστήματα που απαιτούνται για τη λειτουργία μιας υπηρεσίας συμπεριλαμβανομένης διάθεσης περιεκτών(Containers/Docker Instances) και τις διευθετήσεις τόσο της επικοινωνίας και της αποθήκευσης μέσω δικτύου. Στο συγκεκριμένο πείραμα χρησιμοποιήθηκε μια τεχνολογία που αναπτύχθηκε στην google και αργότερα έγινε διαθέσιμη ως ανοιχτός πηγαίος κώδικας. Αυτή η τεχνολογία είναι ιδιαίτερα δημοφιλής και λέγεται Kubernetes. Στην αρχιτεκτονική του Kubernetes Orchestrator (Αρχιτεκτονική μέσω Ενορχηστρωτή), χρησιμοποιείται ένας κύριος κόμβος ο οποίος εκτελεί τα συστατικά στοιχεία του επιπέδου ελέγχου για τη συστάδα από τα docker instances (περιέκτες). Στο Kubernetes χρησιμοποιείται ο όρος επίπεδο ελέγχου(Control Plane-master-κύριος) για τα στοιχεία λογισμικού που χρησιμοποιεί ένας ιδιοκτήτης για να δημιουργεί και διαθέτει περιέκτες. Όταν το λογισμικό του επιπέδου ελέγχου εκτελείται σε έναν κόμβο αυτός ο κόμβος είναι γνωστός ως κύριος κόμβος. Οι υπόλοιποι κόμβοι λέγονται κόμβοι-εργάτες(worker nodes) και εκτελούν εντολές του κύριου κόμβου. Τα συστατικά

στοιχεία του επιπέδου ελέγχου επικοινωνούν μεταξύ τους και μερικά από αυτά επικοινωνούν με εξωτερικά τελικά σημεία. Όλη η εσωτερική επικοινωνία εκτελείται μέσω ενός διακομιστή API. Οι διάφοροι κόμβοι-εργάτες που έχουμε χρησιμοποιήσει διαθέτουν το λογισμικό της συγκεκριμένης εφαρμογής εξομοίωσης (ICON company) και διαχειρίζονται από τον Orchestrator Kubernetes.

Μέσω της εφαρμογής Command Line Interface(CLI) που διαθέτει ο κύριος ενορχηστρωτής κόμβος (Kubernetes) μπορούμε και δίνουμε εντολές προκειμένου να σχηματιστούν στην αρχή 2 docker instances(περιέκτες) με χαρακτηριστικά 4 cores VCPU, 8 GB RAM and 30 GB Hard Disk και έναν docker instance(περιέκτης) που παίζει το ρόλο του διακομιστή στο άκρο(Edge Server). Αυτός ο Edge Server docker διαθέτει περιορισμένους υπολογιστικούς πόρους 2 cores VCPU, 4 GB RAM and 10 GB Hard Disk. Στη δεύτερη φάση των πειραμάτων, όπου η ανάγκη για επεξεργασία δεδομένων παραγωγικής εικόνας είναι σαφώς μεγαλύτερη, δίνουμε εντολές μέσω CLI Kubernetes ώστε να δημιουργηθούν 5 docker instances(περιεκτών) με χαρακτηριστικά 4 cores VCPU, 8 GB RAM and 30 GB Hard Disk. Ο Edge Server docker διαθέτει τις ίδιες με πριν διαστάσεις.

Στις περιπτώσεις πειραμάτων που εξετάζουμε, οι τύποι δεδομένων που χρησιμοποιούνται αφορούν κυρίως κατανάλωση επεξεργαστικής ισχύος και τα όποια δεδομένα διαβάζονται από έναν βελτιστοποιητή Stochastic Gradient Descent (SGD). Ο συνολικός αλγόριθμος FL Algorithm χρησιμοποιεί την ευρέως γνωστή μέθοδο δειγματοληψίας με mini-batches χρησιμοποιώντας κομμάτια κάθε φορά δεδομένων ή αλλιώς data batches για την ανάγνωση των δεδομένων από τους διάφορους κόμβους του συνολικού συστήματος. Η συγκεκριμένη μέθοδος θεωρεί την περίπτωση όπου όλοι οι κόμβοι έχουν ίδιο αριθμό δεδομένων σε κάθε ολοκλήρωση(iteration) εάν ο ίδιος τύπος δεδομένων χρησιμοποιείται παντού. Ο βελτιστοποιητής SGD χρησιμοποιείται σε συνδυασμό με το νευρωνικό δίκτυο βαθιάς μηχανικής μάθησης LSTM που έχει ως ρόλο την ελαχιστοποίηση της τοπικής συνάρτησης απωλειών[120]. Στην πραγματικότητα τα περισσότερα δεδομένα είναι κατανεμημένα με τυχαίο τρόπο σε όλους τους κόμβους (clients) με διαφορετικό μέγεθος και περιεχόμενο θεωρώντας ότι ένας συνδυασμός IID και non-IID χρησιμοποιείται. Με άλλα λόγια θεωρούμε ότι χρησιμοποιούνται πιθανοτικά πανομοιότυπη κατανομή δεδομένων αλλά και μη πανομοιότυπη κατανομή δεδομένων στους διάφορους κόμβους(clients). Ο συνολικός αριθμός των δεδομένων που χρησιμοποιούνται στο πείραμά μας είναι περίπου 2000 σημεία (CPU time series points) που αντιπροσωπεύουν μετρήσεις κατανάλωσης επεξεργαστικής ισχύος (CPU measurements). Χωρισμός δεδομένων έχει λάβει χώρα στο συγκεκριμένο πείραμα, όπου το 75% περίπου χρησιμοποιείται για σκοπούς εκπαίδευσης του μοντέλου του αλγορίθμου και τα υπόλοιπα δεδομένα χρησιμοποιούνται για λόγους επικύρωσης στο βήμα του Inference.

Οι βασικοί παράμετροι ελέγχου του συστήματος πολλών οι νεφο υπολογιστικών παροχών Federated Learning έχουν ως ακολούθως:

- γ=10
- τmax = 10
- control parameter ϕ=0.01 for LSTM algorithm [120]
- gradient descent step size n=0.1

Οι χρησιμοποιούμενοι LSTM αλγόριθμοι στους επιμέρους κόμβους χρησιμοποιούν την ακόλουθη αρχιτεκτονική και hyper-parameters:

- 1 hidden layer με 50 κόμβους-nodes
- 1 drop-out layer (at the output) με rate=2
- Ένα dense layer με 60-time steps ανά σειρά εισόδου δεδομένων-input sequence.

7.5.2 Αποτελέσματα Πειραμάτων και Συμπεράσματα

Για κάθε κόμβο-client μετρικές που αφορούν την κατανάλωση επεξεργαστικής ισχύος χρησιμοποιούνται προκειμένου να υπολογίσουμε την ακρίβεια πρόγνωσης με βάση κάποιους ορισμούς όπως:

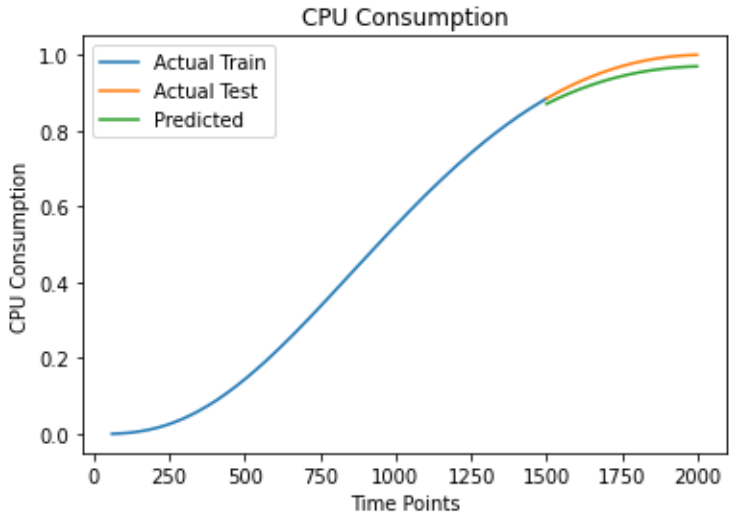
- Mean Absolute Error (MAE)
- Mean Squared Error or Loss (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Percentage Error (MAPE)
- Symmetric Mean Absolute Percentage Error (SMAPE)

Οι ορισμοί των συγκεκριμένων μεγεθών έχουν περιγραφεί ήδη στο κεφάλαιο 6 μέσω μαθηματικών εκφράσεων. Τα συγκεκριμένα μεγέθη δίνουν μια εκτίμηση της ακρίβειας που επιτυγχάνεται για κάθε τύπο δεδομένων και για κάθε αρχιτεκτονικό τρόπο κατανομής των δεδομένων μέσα στο σύστημα Federated Learning cluster.

Είναι σημαντικό να επισημανθεί ότι προκειμένου να συγκρίνουμε τις περιπτώσεις όπου εφαρμόζουμε κανονικό - Full Federated Learning Trained model εν σχέση με την περίπτωση που εφαρμόζουμε μειωμένο μοντέλο-TF Lite model, ακολουθούμε τα βήματα του δέντρου αποφάσεων στο σχήμα 37 και θεωρούμε τις ακόλουθες παραμέτρους επιλογής στο αρχείο configuration file του FL system:

- Παραμετροποίηση ή όχι του τμήματος Inference?
- Ελάττωση του μεγέθους του Inference μοντέλου ή όχι?
- Επιπλέον αύξηση ή όχι του τελικού αποτελέσματος ακρίβειας πρόγνωσης μετά την εκτέλεση του τμήματος Inference του αλγορίθμου μας?

Ακολούθως, σύμφωνα με τις επιλογές βελτιστοποίησης καταλήγουμε σε πολύ θετικά αποτελέσματα όσον αφορά την ακρίβεια όπως αναφέρθηκε ανωτέρω. Αρχικά χρησιμοποιήσαμε το ακόλουθο προφίλ δεδομένων της κατανάλωσης επεξεργαστικής ισχύος(περίπτωση εξομοίωσης low fidelity) και έχουμε την ακόλουθη αξιολόγηση προγνώσεων:



Σχήμα 40 – Dataset για Αυξανόμενη Κατανάλωση Επεξεργαστικής Ισχύος (CPU increasing)

Χρησιμοποιώντας ως μέτρο σύγκρισης τα αποτελέσματα των μετρικών μεταξύ της περιπτώσης χρήσης ή όχι βελτιστοποιημένου μοντέλου Inference στο άκρο του υπολογιστικού νέφους(edge server - TF Lite model) καθώς και τις 3 επιλογές της βελτιστοποιήσεως ή κβαντοποιήσεως του μοντέλου Inference φαίνονται στον ακόλουθο πίνακα 17 με χρήση 2 κόμβων ως clients στο Federated Learning σύστημα:

Πίνακας 17: Αποτελέσματα για Αυξανόμενη Κατανάλωση Επεξεργαστικής Ισχύος (CPU increasing) με 2 clients

Metrics	FL model without Tiny ML Inference	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	0.25497	**0.00197**	0.0020	**0.0019**
MAE	0.02058	0.021788	0.01780	**0.01383**
MSE	0.00045	0.00050	0.00033	**0.00020**
RMSE	0.02059	0.02179	0.01780	0.01383
MAPE	2.12632	2.24949	1.83899	1.42949
Size of Model file (KB)	60	22	31	22

Η σύγκριση των παραπάνω τιμών με την περίπτωση ενός μοντέλου Federated Learning χωρίς χρήση τεχνολογίας Tiny ML Inference σε επί τοις 100 (%) τιμές φαίνεται στον παρακάτω πίνακα:

Πίνακας 18: Αποτελέσματα σύγκρισης σχετικά με πρόγνωση για Κατανάλωση Επεξεργαστικής Ισχύος (CPU increasing) με 2 clients με χρήση ή όχι Tiny ML.

Metrics	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	- 99%	- 99%	- 99%
MAE	- 6%	-13%	-32%
MSE	+ 11%	- 26%	-55%
RMSE	+ 6%	- 13%	-32%
MAPE	+ 6%	- 13%	-32%
Size of Model file (KB)	-63%	-48%	-63%

Χρησιμοποιώντας ως μέτρο σύγκρισης τα αποτελέσματα των μετρικών μεταξύ της περιπτώσης χρήσης ή όχι βελτιστοποιημένου μοντέλου Inference στο άκρο του υπολογιστικού νέφους(edge server - TF Lite model) καθώς και τις 3 επιλογές της βελτιστοποιήσεως ή κβαντοποιήσεως του μοντέλου Inference φαίνονται στον ακόλουθο πίνακα 19 με χρήση 5 κόμβων ως clients στο Federated Learning σύστημα:

Πίνακας 19: Αποτελέσματα για Αυξανόμενη Κατανάλωση Επεξεργαστικής Ισχύος (CPU increasing) με 5 clients

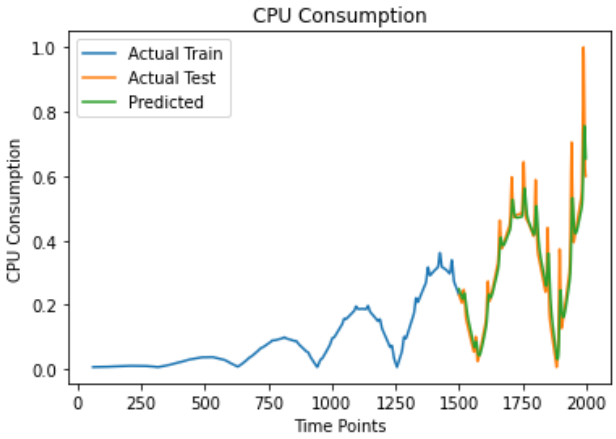
Metrics	FL model without Tiny ML Inference	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	0.24255	0.00300	0.00297	0.00199
MAE	0.03255	0.02304	0.02173	0.02169
MSE	0.00109	0.00055	0.00049	0.00049
RMSE	0.03255	0.02304	0.02173	0.02169
MAPE	3.37104	2.38365	2.24683	2.24167
Size of Model file (KB)	60	22	31	22

Η σύγκριση των παραπάνω τιμών με την περίπτωση ενός μοντέλου Federated Learning χωρίς χρήση τεχνολογίας Tiny ML Inference σε επί τοις 100 (%) τιμές φαίνεται στον παρακάτω πίνακα:

Πίνακας 20: Αποτελέσματα σύγκρισης σχετικά με πρόγνωση για Κατανάλωση Επεξεργαστικής Ισχύος (CPU increasing) με 5 clients με χρήση ή όχι Tiny ML.

Metrics	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	- 99%	- 99%	- 99%
MAE	- 29%	- 33%	- 33%
MSE	- 49%	- 55%	- 55%
RMSE	- 29%	- 33%	- 33%
MAPE	- 29%	- 33%	- 33%
Size of Model file (KB)	-63%	-48%	-63%

Στη συνέχεια χρησιμοποιήσαμε το ακόλουθο προφίλ δεδομένων της κατανάλωσης επεξεργαστικής ισχύος (περίπτωση εξομοίωσης high fidelity) με αυξανόμενο φορτίο με Spikes και έχουμε την ακόλουθη αξιολόγηση προγνώσεων:



Σχήμα 41 – Dataset για Κατανάλωση Επεξεργαστικής Ισχύος με Spikes (CPU consumption Spikes)

Χρησιμοποιώντας ως μέτρο σύγκρισης τα αποτελέσματα των μετρικών μεταξύ της περιπτώσης χρήσης ή όχι βελτιστοποιημένου μοντέλου Inference στο άκρο του υπολογιστικού νέφους(edge server - TF Lite model) καθώς και τις 3 επιλογές της βελτιστοποιήσεως ή κβαντοποιήσεως του μοντέλου Inference φαίνονται στον ακόλουθο πίνακα 21 με χρήση 2 κόμβων ως clients στο Federated Learning σύστημα:

Πίνακας 21: Αποτελέσματα πρόγνωσης για Κατανάλωση Επεξεργαστικής Ισχύος με Spikes (CPU Spikes) με 2 clients

Metrics	FL model without Tiny ML Inference	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	0.20302	0.00201	0.00199	0.00200
MAE	0.04534	0.04239	0.04366	0.04219
MSE	0.00474	0.00433	0.00442	0.00434
RMSE	0.04534	0.04239	0.04366	0.04368
MAPE	23.89639	20.98909	22.45505	22.06845
Size of Model file (KB)	60	22	31	22

Η σύγκριση των παραπάνω τιμών με την περίπτωση ενός μοντέλου Federated Learning χωρίς χρήση τεχνολογίας Tiny ML Inference σε επί τοις 100 (%) τιμές φαίνεται στον παρακάτω πίνακα:

Πίνακας 22: Αποτελέσματα σύγκρισης σχετικά με πρόγνωση για Κατανάλωση Επεξεργαστικής Ισχύος με Spikes (CPU Spikes) με 2 clients με χρήση ή όχι Tiny ML.

Metrics	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	- 99%	-99%	-99%
MAE	-6%	-3%	-7%
MSE	-7%	-6%	-8%
RMSE	-6%	-3%	-3,5%
MAPE	-12%	-6%	-7%
Size of Model file (KB)	-63%	-48%	-63%

Χρησιμοποιώντας ως μέτρο σύγκρισης τα αποτελέσματα των μετρικών μεταξύ της περίπτωσης χρήσης ή όχι βελτιστοποιημένου μοντέλου Inference στο άκρο του υπολογιστικού νέφους(edge server - TF Lite model) καθώς και τις 3 επιλογές της βελτιστοποιήσεως ή κβαντοποιήσεως του μοντέλου Inference φαίνονται στον ακόλουθο πίνακα 23 με χρήση 5 κόμβων ως clients στο Federated Learning σύστημα:

Πίνακας 23: Αποτελέσματα πρόγνωσης για Κατανάλωση Επεξεργαστικής Ισχύος με Spikes (CPU Spikes) με 5 clients

Metrics	FL model without Tiny ML Inference	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	0.22020	0.00200	0.00199	0.00200
MAE	0.04446	0.03794	0.04845	0.04374
MSE	0.00467	0.00341	0.00553	0.00449
RMSE	0.04446	0.03794	0.04845	0.04374
MAPE	22.65313	18.93766	25.12222	22.53382
Size of Model file (KB)	60	22	31	22

Η σύγκριση των παραπάνω τιμών με την περίπτωση ενός μοντέλου Federated Learning χωρίς χρήση τεχνολογίας Tiny ML Inference σε επί τοις 100 (%) τιμές φαίνεται στον παρακάτω πίνακα:

Πίνακας 24: Αποτελέσματα σύγκρισης σχετικά με πρόγνωση για Κατανάλωση Επεξεργαστικής Ισχύος με Spikes(CPU Spikes) με 5 clients με χρήση ή όχι Tiny ML.

Metrics	FL model with Tiny ML Inference (DEFAULT)	FL model with Tiny ML Inference (FLOAT16)	FL model with Tiny ML Inference (SIZE)
Execution Time Prediction	-99%	-99%	-99%
MAE	-14%	-8%	-1,5%
MSE	-26%	-18%	-3,8%
RMSE	-14%	-9%	-1,6%
MAPE	-16%	-11%	-0,5%
Size of Model file (KB)	-63%	-48%	-63%

Λαμβάνοντας υπόψη όλα τα παραπάνω πειραματικά αποτελέσματα, μπορούμε να καταλάβουμε ότι χρησιμοποιώντας τεχνικές μετασχηματισμό και βελτιστοποίησης σε ένα εκπαιδευμένο μοντέλο για σκοπούς Inference, μπορούμε να επιτύχουμε τόσο κατάλληλη χρήση πόρων όσον αφορά το χώρο στο δίσκο (μικρότερο μέγεθος μοντέλου) και συνάμα καλύτερη ακρίβεια πρόγνωσης σε σύγκριση με προηγούμενες εργασίες όπως [A5]. Γενικότερα μιλώντας, κάποιος θα περίμενε ότι τα αποτελέσματα ακριβείας θα γίνονται χειρότερα καθώς το μέγεθος του μοντέλου ελαττώνεται και κβαντοποιείται [151]. Παρόλαυτά, για το συγκεκριμένο σύστημα Federated Learning που αναπτύξαμε τα αποτελέσματα είναι καλύτερα.

Συμπερασματικά, για όλους τους τύπους των δεδομένων (dataset types), φαίνεται ότι σχετικά με το μέγεθος MAE οι τιμές της τρέχουσας εργασίας παρουσιάζουν μια βελτίωση

από 6% έως 33% στις περισσότερες των περιπτώσεων συγκρινόμενες με αυτές της αντίστοιχης εργασίας [A5]. Φυσικά, η περίπτωση χρήσης του μοντέλου Tensorflow lite όπου εφαρμόζεται κβαντοποίηση είτε χρησιμοποιώντας την default τιμή ή ακολουθώντας την μεθοδολογία μεγέθους, δίνει καλύτερα αποτελέσματα ακρίβειας προγνώσεων όπως έχει ήδη φανεί στο σχήμα 37 του δένδρου λήψης αποφάσεων. Αυτό είναι αναμενόμενα εξαιτίας της μεθοδολογίας βελτιστοποίησης που έχει χρησιμοποιηθεί. Όπως είναι γνωστό, η μετρική MAE παρουσιάζει τα αποτελέσματα της διαφοράς μεταξύ συνεχών τιμών στο χρόνο σχετικά με την πραγματική μετρημένη κατανάλωση της επεξεργαστικής ισχύος και της τιμής που έχει προκύψει από πρόγνωση. Γενικά η μετρική MAE παραβλέπει τον θόρυβο στα δεδομένα και τους outliers δίνοντας καλύτερα αποτελέσματα όταν αυτοί δεν υπάρχουν[129].

Όσον αφορά τις μετρικές MSE (loss definition) και RMSE, υπάρχει μια βελτίωση συγκρινόμενες με αυτές της εργασίας [A5] περίπου έως 55% αναλόγως φυσικά την περίπτωση και τον τύπο των δεδομένων. Για την περίπτωση της κατανάλωσης επεξεργαστικής ισχύος δεδομένων αυξανομένης κατανάλωσης (CPU Increasing dataset consumption) αλλά και της περιπτώσεως επεξεργαστικής ισχύος με Spikes (CPU Spikes dataset consumption) φαίνεται ότι έχουμε καλύτερα αποτελέσματα όσον αφορά την ακρίβεια προγνώσεων. Οι συγκεκριμένοι τύποι δεδομένων και ο αλγόριθμος επιλογής κόμβων που έχουν χρησιμοποιηθεί αποδίδουν πάρα πολύ καλά όσον αφορά τις τιμές των τοπικών συναρτήσεων απωλειών καθώς οποιαδήποτε παρουσία θορύβου αυτή παραβλέπεται. Αυτά τα data spikes’ στις τιμές απωλειών αποκλείονται κατά τη διάρκεια της συνολικής διαδικασίας Federated Learning.

Γενικά μιλώντας, τα αποτελέσματα στην τρέχουσα παράγραφο της παρούσας διδακτορικής διατριβής δείχνουν μια σοβαρή αύξηση στην ακρίβεια προβλέψεων σε σύγκριση με προηγούμενες παρουσιασμένες εργασίες [A5], [99], [147], [105], [91], [111]. Επιπλέον, είναι πολύ σημαντικό να αναφερθεί ότι βελτιστοποιώντας και κβαντοποιώντας το βασικό Federated Learning μοντέλο χρησιμοποιώντας διάφορες τεχνικές Tiny ML, το τελικό συνολικό μέγεθος του μοντέλου γιατί την διαδικασία του Inference ελαττώνεται περίπου 65%. Με αυτό τον τρόπο ο περιορισμός των πόρων ως κριτήριο για τα συστήματα ευρισκόμενα στο άκρο του υπολογιστικού νέφους (Edge systems) ικανοποιείται πάρα πολύ καλά. Ο χρόνος για τη σύγκλιση του συνολικού αλγορίθμου Federated Learning ελαττώνεται πάρα πολύ σε συνάρτηση με αυτόν της εργασίας[99] καθότι μόνο οι επιλεγμένοι κόμβοι συνεισφέρουν στο ολικό iteration γιατί εκπαίδευση του συνολικού Federated Learning αλγόριθμου. Και τα 2 προαναφερθέντα αποτελέσματα είναι αρκετά καινοτόμα και προσφέρουν μεγάλα πλεονεκτήματα σε περιβάλλοντα που βρίσκονται στο άκρο υπολογιστικού νέφους πέραν του ενός παρόχου.

7.6 Συμπεράσματα

Οι μέθοδοι Federated learning έχουν αρχικά προταθεί ώστε να εκπαιδεύουν από κοινού ένα μοντέλο μηχανικής μάθησης χρησιμοποιώντας διάφορους κόμβους και την ίδια στιγμή να διαβεβαιώνουν για την ασφάλεια των δεδομένων του κάθε κόμβου. Με την ανάπτυξη των τεχνολογιών στο άκρο του υπολογιστικού νέφους σε περιβάλλοντα πολλών παρόχων, το μοντέλο μας Federated Learning το οποίο χρησιμοποιεί στη λειτουργία του το μοντέλο Federated Learning, μπορεί να πετύχει πολλές βελτιστοποιήσεις σε μέγεθος και σε καθυστέρηση χωρίς να προκαλεί την οποιαδήποτε μείωση στην ακρίβεια προβλέψεων. Αντιθέτως, μπορούμε να βρούμε πολλά κέρδη και βελτιώσεις όσον αφορά την ακρίβεια προγνώσεων στην σχετική διαδικασία του Inference, χρησιμοποιώντας διάφορες μεθόδους κβαντοποίησης. Η μεθοδολογίασε αυτή την περίπτωση είναι αρκετά καινούργια και πολύ καινοτόμα σε σχέση με όλο το Federated Learning σύστημα που έχει αναπτυχθεί και σε σύγκριση με προηγούμενες εργασίες [A5].

Στην τρέχουσα εργασία, πριν ένα καθολικό βήμα aggregation εκτελεστή, εξαιρούμε οποιοδήποτε κόμβο του υπολογιστικού νέφους που πιθανά θα μειώσει τη συνολική ακρίβεια προγνώσεων και την απόδοση του συστήματος. Το άλλο κομμάτι της τρέχουσας εργασίας στην τρέχουσα παράγραφο αφορά έναν γνωστό αλγόριθμο ελέγχου ο οποίος χρησιμοποιείται για να πετύχει βέλτιστη συχνότητα μεταξύ των τοπικών ανανεώσιμων των παραμέτρων των μοντέλων και του καθολικού aggregation με στόχο την επίτευξη ενός ολικού ελαχίστου. Σαν αποτέλεσμα όλων των προαναφερθέντων, βελτιώσεις έως 65% στην ακρίβεια προγνώσεων και στο μέγεθος του μοντέλου κατά τη διαδικασία του Inference παρατηρείται σε σχέση με προηγούμενες εργασίες [A5], [99]. Όλα τα προαναφερθέντα κατορθώματα είναι αρκετά καινοτόμα και πολύ βοηθητικά στις διαδικασίες που τρέχουν σε περιβάλλοντα υπολογιστικών νεφών κυρίως ευρισκόμενα στο άκρο τους.

Σε μελλοντική εργασία υπάρχουν πολλές σκέψεις οι οποίες θα μπορούσαν να υλοποιηθούν όσον αφορά τους τρόπους κρυπτογράφησης των παραμέτρων των τοπικών μοντέλων και ανταλλαγής μεταξύ των κόμβων και του κεντρικού διακομιστή του συστήματος.

8 Επίλογος

Στο συγκεκριμένο κεφάλαιο παρουσιάζουμε τα γενικά συμπεράσματα της μελέτης μας. Επίσης, περιγράφουμε τις δυνητικές μελλοντικές επεκτάσεις της συγκεκριμένης ερευνητικής προσπάθειας.

8.1 Συμπεράσματα

Στην παρούσα διδακτορική διατριβή πραγματοποιήθηκε επισκόπηση της βιβλιογραφίας, που σχετίζεται με τρόπους παρακολούθησης κατανεμημένων εφαρμογών σε υπολογιστικά νέφη με έμφαση σε μεθόδους όταν υπάρχουν πολλοί πάροχοι. Επίσης αναζητήθηκαν μεθοδολογίες για βέλτιστη προσαρμογή πόρων κατανεμημένων εφαρμογών πάλι με έμφαση σε περιβάλλοντα πολλών παρόχων υπολογιστικού νέφους. Τέλος, αναζητήθηκαν εργασίες σχετικές με εφαρμογή Federated Learning μοντέλων με τεχνική επιλογής Clients και βέλτιστη χρήση πόρων κυρίως χρόνου, με τελικό στόχο την διασφάλιση κατανάλωσης λιγότερων πόρων και αύξηση της ακρίβειας ζήτησης πόρων των διαφόρων εφαρμογών στο multi cloud περιβάλλον. Επιπλέον, αναζητήθηκαν και εργασίες σχετικές με εκτέλεση των παραπάνω Federated Learning αλγορίθμων στο Άκρο του Νέφους (Edge) σε περιβάλλοντα περιορισμένων υπολογιστικών πόρων.

Με οδηγό τα κενά που εντοπίστηκαν κατά τη βιβλιογραφική μελέτη, η διατριβή προτείνει έναν ευέλικτο τρόπο παρακολούθησης κατανεμημένων εφαρμογών σε υπολογιστικά νέφη παρόχων και εφαρμόζοντας καινοτόμα τεχνική ιεραρχικής και σύνθετης επεξεργασίας συμβάντων. Επιπλέον, προτείνεται και μια τεχνική με την βοήθεια αλγορίθμου για βέλτιστη αναπροσαρμογή πόρων και με βασική παράμετρο τον χρόνο για αποφυγή καθυστερήσεων στις παρεχόμενες υπηρεσίες στους τελικούς χρήστες. Τέλος, η τεχνική του Federated Learning μοντέλο με τεχνική επιλογής Clients όπως αναφέρθηκε εξασφαλίζει την proactive δέσμευση και αποδέσμευση πόρων για αδιάλειπτη λειτουργία κατανεμημένων στο cloud εφαρμογών. Σε αυτή την τεχνική προστέθηκε και η περίπτωση εκτέλεσης του Inference κομματιού της διαδικασίας Federated Learning στο Άκρο του υπολογιστικού Νέφους πολλών παρόχων.

Η προσέγγισή μας αξιολογήθηκε σε πραγματικές συνθήκες μέσω εγκατάστασης των προτεινόμενων μεθόδων σε περιβάλλοντα πολλών παρόχων τόσο ιδιωτικού όσο και δημόσιου υπολογιστικού νέφους και αντιστοίχων μετρήσεων που έλαβαν χώρα. Τα αποτελέσματα είναι πολύ θετικά για την εξαγωγή αποτελεσμάτων σε πραγματικό χρόνο και βοηθητικά σε λήψη αποφάσεων που έχουν να κάνουν με αναπροσαρμογή των αναγκαίων πόρων του νέφους και παροχή κατ’ επέκταση υψηλής ποιότητας υπηρεσιών. Σημαντικό ρόλο σε αυτά κατέχει ο αριθμός των μετρήσεων που γίνονται καθώς αυτό επηρεάζει την διαδικασία μάθησης των αλγορίθμων που χρησιμοποιούνται. Επίσης σχετικά με το προτεινόμενο σύστημα παρακολούθησης, η αξιολόγηση έδειξε ότι γίνεται χρήση επαρκών επιπέδων κατανάλωσης μνήμης και επεξεργαστικής ισχύος στις υπό παρακολούθηση εικονικές μηχανές που φιλοξενούν τις εφαρμογές, και με πολύ ικανοποιητικό επίπεδο επεξεργασίας σύνθετων συμβάντων σε σύγκριση με άλλες τεχνολογίες πχ. Siddhi CEP Engine. Τέλος και η πρόγνωση πόρων με το σύστημα Federated

Learning μοντέλο με τεχνική επιλογής Clients έδειξε πολύ καλά στοιχεία ακρίβειας προγνώσεων και καλύτερης χρήσης πόρων στο edge, καλύτερα από αυτά που παρουσιάζονται σε άλλες εργασίες στην διεθνή βιβλιογραφία. Το σενάριο χρήσης πολλών παρόχων χρησιμοποιήθηκε καθώς και η Τεχνολογία Tiny ML σε συνάρτηση με Docker εικονικές μηχανές (VMs) στο άκρο (Edge) του υπολογιστικού νέφους.

8.2 Μελλοντική Έρευνα

Η βασική συνεισφορά της Διατριβής θα μπορούσε να επεκταθεί μελλοντικά με βάση τις λύσεις που θα δοθούν στις ακόλουθες ερωτήσεις:

- Πόσους παρόχους νεφουπολογιστικών υπηρεσιών θα μπορούσαμε να χρησιμοποιήσουμε όσον αφορά τις μετρήσεις των χρόνων εκκίνησης των εικονικών μηχανών που θα μπορούσαν να διαμορφώσουν πληρέστερα τον penalty calculator για βέλτιστη αναπροσαρμογή πόρων (reconfiguration);
- Είναι δυνατόν στην διαδικασία αναπροσαρμογής πόρων(reconfiguration) να ληφθούν υπόψιν και χρόνοι μετάβασης (migration) εικονικών μηχανών από τον ένα Cloud πάροχο στον άλλο?
- Πώς θα μπορούσαμε να προστατέψουμε το Federated Learning σύστημα από πιθανούς κακόβουλους (malicious) συμμετέχοντες κόμβους(clients) που θα μπορούσαν να στείλουν λανθασμένες ή «μολυσμένες» παραμέτρους τοπικών μοντέλων που στο τέλος θα δημιουργούσαν πρόβλημα στην συνολική διαδικασία του global aggregation του συνολικού αλγορίθμου?
- Θα μπορούσαμε να χρησιμοποιήσουμε trusted score metrics σαν κριτήρια συμμετοχής κόμβων στο Federated Learning σύστημα αλλά και στο Edge node ώστε να αναβαθμίσουμε την ασφάλεια της διαδικασίας και κατ' επέκταση την ακρίβεια προγνώσεων?

ΔΗΜΟΣΙΕΥΣΕΙΣ ΤΕΛΙΚΗΣ ΚΡΙΣΗΣ ΔΙΑΤΡΙΒΗΣ

Στα πλαίσια της παρούσας εργασίας Τελικής Κρίσης του Διδακτορικού έχουν πραγματοποιηθεί οι παρακάτω δημοσιεύσεις:

A1. Stefanidis, V.-A., Verginadis, Y., Baur, D., Przezdziek, T., Mentzas, G. (2020), Reconfiguration Penalty Calculation for Cross-Cloud Application Adaptations, In D. Ferguson, M. Helfert, C. Pahl (eds), Proceedings of the 10th International Conference on Cloud Computing and Services Science (pp. 355-362), CLOSER 2020, Prague, Czech Republic, 7-9 May.

A2. Vassilis Stefanidis, Yiannis Verginadis, Ioannis Patiniotakis and Gregoris Mentzas. “Distributed Complex Event Processing in Multi-Clouds”, Service-Oriented and Cloud Computing: 7th IFIP WG 2.14 European Conference, ESOCC 2018, Como, Italy, September 12-14, 2018, pp 105-119.

A3. Daniel Baur, Frank Griesinger, Yiannis Verginadis, Vasilis Stefanidis, Ioannis Patiniotakis: “A Model Driven Engineering Approach for Flexible and Distributed Monitoring of Cross-Cloud Applications”. 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), Dec. 17-20 2018, Zurich, Switzerland, pp.31-40.

A4. Geir Horn, Pawel Skrzypek, Marcin Prusinski, Katarzyna Materka, Vassilis Stefanidis, Yiannis Verginadis: “MELODIC: Selection and Integration of Open Source to Build an Autonomic Cross-Cloud Deployment Platform”. Software Technology: Methods and Tools: 51st International Conference 2019, pp.364-377.

A5. Stefanidis, V.-A.; Verginadis, Y.; Mentzas, G. MulticloudFL: Adaptive Federated Learning for Improving Forecasting Accuracy in Multi-Cloud Environments. Journal MDPI - Information 2023, vol. 14, issue 12, 662, pp.2-28. Doi: https://doi.org/10.3390/info14120662

A6. Stefanidis, V.-A.; Verginadis, Y.; Mentzas, G. Federated Learning in Multi Clouds and Resources Constraint Devices at the Edge. IISA2024 Conference Committee, IEEE Proceedings, 17-18 July 2024, Chania, Greece.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Qi, Z.: Cloud Computing: state-of-the-art and research challenges. J. Internet Serv. Appl. 1(1), 7-18 (2010)

2. Ambrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A.,Lee, G., Patterson, D.A., Rabkin, A., Stoca, I., Zaharia, M.: Above the clouds: a berkeley view of cloud computing. Technical reports UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009)

3. Ghanam, Y., Ferreira, J., Maurer, F.: Emerging issues and challenges in cloud computing – a hybrid approach. J. Softw. Eng. Appl. 5(11A), 923 (2012)

4. Sun, Y., Zhang, J., Xiong, Y., Zhu, G.: Data Security and privacy in cloud computing. Int. J. Distrib. Sens. Netw. 10(7), 190903 (2014)

5. Lama P, Zhou X (2012) Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In: Proceedings of the 9th International Conference on Autonomic Computing, ICAC ’12. ACM, New York, NY, USA. pp 63–72

6. Dawoud W, Takouna I, Meinel C (2011) Elastic virtual machine for fine-grained cloud resource provisioning. Glob Trends Comput Commun Syst 269, pp:11–25

7. Addis B, Ardagna D, Panicucci B, Squillante MS, Zhang L (2013) A hierarchical approach for the resource management of very large cloud platforms. IEEE Trans Dependable Secure Comput 10, pp:253–272

8. Bu X, Rao J, Xu C-Z (2011) Model-free learning approach for coordinated configuration of virtual machines and appliances. In: 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. IEEE, Washington, DC, USA. pp 12–21

9. Jung G, Hiltunen MA, Joshi KR, Schlichting RD, Pu C (2010) Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In: International Conference on Distributed Computing Systems. IEEE, Washington, DC, USA. pp 62–73

10. Han R, Guo L, Ghanem MM, Guo Y (2012) Lightweight resource scaling for cloud applications. In: International Symposium on Cluster, Cloud and Grid Computing. IEEE, Washington, DC, USA. pp 644–651.

11. Beloglazov A, Abawajyb J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28:755–768

12. Shen Z, Subbiah S, Gu X, Wilkes J (2011) Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In: Proceedings of the 2Nd ACM Symposium on Cloud Computing, SOCC ’11. ACM, New York, NY, USA. pp 5–1514

13. Padala P, Hou K-Y, Shin KG, Zhu X, Uysal M, Wang Z, Singhal S, Merchant A (2009) Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European Conference on Computer Systems, EuroSys ’09. ACM, New York, NY, USA. pp 13–26

14. Lim HC, Babu S, Chase JS (2010) Automated control for elastic storage. In: Proceedings of the 7th International Conference on Autonomic sComputing, ICAC ’10. ACM, New York, NY, USA. pp 1–10

15. Q. Zhang, L. Cheng and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” J. Internet Services Appl., vol. 1, no. 1, pp. 7-18, May 2010

16. H. Takabi, J.B.D.Joshi and G.-J. Ahn, “Security and privacy challenges in cloud computing environments,” IEEE Security Privacy, vol.8, no. 6, pp. 24-31, Nov./Dec. 2010

17. K. Ren, C. Wang, and Q. Wang, “Security challenges for the public cloud,” IEEE Internet Comput., vol. 16, no. 1, pp. 69-73, Jan./Feb. 2012.

18. T. Dillon, C. Wu, and E. Chang, ``Cloud computing: Issues and challenges, ''in Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl. (AINA), Apr. 2010, pp. 27-33.

19. C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, ``A framework for native multi-tenancy application development and management,'' in Proc. 9th IEEE Int. Conf. E-Commerce Technol., Jul. 2007, pp. 551-558.

20. A. Gupta and D. Milojicic, ``Evaluation of HPC applications on cloud,'' in Proc. 6th Open Cirrus Summit (OCS), Oct. 2011, pp. 22-26.

21. P. Bientinesi, R. Iakymchuk, and J. Napper, ``HPC on competitive cloud resources,'' in Handbook of Cloud Computing. Boston, MA, USA: Springer, 2010, pp. 493-516.

22. A. Iosup, N. Yigitbasi, and D. Epema, ``On the performance variability of production cloud services,'' in Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid), May 2011, pp. 104-113.

23. A. Shieh, S. Kandula, A. G. Greenberg, and C. Kim, ``Seawall: Performance isolation for cloud datacenter networks,'' in Proc. HotCloud, 2010, p. 1.

24. V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, C. Kim, and A. Greenberg, ``EyeQ: Practical network performance isolation at the edge,'' in Proc. 10th USENIX Conf. Netw. Syst. Design Implement., 2013, pp. 297-312.

25. B. P. Rimal and M. Maier, ``Workflow scheduling in multi-tenant cloud computing environments,'' IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 1, pp. 290-304, Jan. 2017.

26. J. Kirschnick, J. M. A. Calero, L. Wilcock, and N. Edwards, ``Toward an architecture for the automated provisioning of cloud services,'' IEEE Commun. Mag., vol. 48, no. 12, pp. 124-131, Dec. 2010.

27. The Anatomy of APM. Accessed: Oct. 25, 2017. [Online]. Available: http://www.apmdigest.com/

28. A. Taherkordi, F. Zahid, Y. Verginadis, G. Horn, “Future Cloud Systems Design: Challenges and Research Directions”, Journal of IEEE Access, pp. 74120-74150, Sept. 2018

29. L. Schubert, J. Domaschka, and P. Guisset. PaaSage-making cloud usage easy, CloudScape. Accessed: Apr. 2018. [Online]. Available: https://paasage.ercim.eu/

30. C. Chapman, M. Musolesi, W. Emmerich, and C. Mascolo, ``Predictive resource scheduling in computational grids,'' in Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS), Mar. 2007, pp. 1-10.

31. C. K. Chui and G. Chen, Kalman Filtering. Cham, Switzerland: Springer, 2017.

32. S. Islam, J. Keung, K. Lee, and A. Liu, ``Empirical prediction models for adaptive resource provisioning in the cloud,'' Future Generat. Comput. Syst., vol. 28, no. 1, pp. 155-162, 2012.

33. Cugola, G., Margara, Al.: Processing Flows of Information: From Data stream to complex event processing. Journal of ACM Computing Surveys (CSUR) 44 (Issue 3 – article 15), pp. 15:1-15:62, 2012.

34. Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I.: Approaching the Internet of Things through Integrating SOA and Complex Event Processing. In Z. Sun, & J. Yearwood (Eds.), Hand-book of Research on Demand-Driven Web Services: Theory, Technologies, and Applica-tions (pp. 304-323). Hershey, PA: IGI Global. doi:10.4018/978-1-4666-5884-4.ch014, 2014.

35. Leitner,P., Inzinger, C., Hummer, W., Satzger, B., Dustdar, S.: Application-Level Performance Monitoring of Cloud Services Based on Complex Event Processing Paradigm. In: 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2012.

36. Hirzel, M.: Partition and Compose: Parallel Complex Event Processing. In: DEBS '12-Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 191-200,ACM, Berlin, Germany, 2012.

37. Ku,T., Long-Zhu, Y., Yuan-Hu, K.: A Novel Distributed Complex Event Processing for RFID Application. In: 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, South Korea, 2008.

38. Hirzel, M.: Partition and Compose: Parallel Complex Event Processing. In: DEBS '12-Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 191-200,ACM, Berlin, Germany, 2012.

39. Ku,T., Long-Zhu, Y., Yuan-Hu, K.: A Novel Distributed Complex Event Processing for RFID Application. In: 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, South Korea, 2008.

40. Paraiso,F., Hermosillo, G., Rouvoy, R., Seinturier, L.: A Middleware Platform to Federate Complex Event Processing. In: 2012 IEEE 16th International Enterprise Distributed Object Computing Conference (EDOC), pp. 113-122, Beijing, China, 2012.

41. Flouris, I., Manikaki, V., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Mock, M., Bothe, S., Skarbovsky, I., Fournier, F., Štajcer, M., Križan, T., Yom-Tov, J., Volarević, M.: FERARI: A Prototype for Complex Event Processing over Streaming Multi-cloud Platforms. In: DEBS '16 Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems. pp. 348-349. Irvine, CA, USA, 2016.

42. Seinturier,L., Merle,P., Rouvoy, R., Romero, D., Schiavoni, V., Stefani, J-B.: A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures. Jour-nal of Software: Practice and Experience (SPE) 42(5), pp. 559-583, 2012.

43. Schultz-Moller, N., Migliavacca, M., Pietzuch, P.: Distributed Complex Event Processing with Query Rewriting. In: Proceedings of the Third ACM International Conference on Dis-tributed Event-Based Systems, DEBS 2009, Nashville, Tennessee, USA, 2009.

44. Mdhaffar, A., Halima, R., Jmaiel, M., Freisleben, B.: A Dynamic Complex Event Processing Architecture for Cloud Monitoring and Analysis. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2013.

45. Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I.: Approaching the Internet of Things through Integrating SOA and Complex Event Processing. In Z. Sun, & J. Yearwood (Eds.), Hand-book of Research on Demand-Driven Web Services: Theory, Technologies, and Applica-tions (pp. 304-323). Hershey, PA: IGI Global. doi:10.4018/978-1-4666-5884-4.ch014, 2014.

46. Leitner,P., Inzinger, C., Hummer, W., Satzger, B., Dustdar, S.: Application-Level Performance Monitoring of Cloud Services Based on Complex Event Processing Paradigm. In: 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 2012.

47. Leitner, P.,Hummer, W., Satzger, B., Inzinger, C., Dustdar, S.: CloudScale- a Novel Middleware for Building Transparently Scaling Cloud Applications. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 434-440. ACM, Trento, Italy, 2012.

48. Zeginis, C., Kritikos, K., Plexoudakis,D.: Event Pattern Discovery for Cross-layer Adaptation of Multi-Cloud Applications. International Journal of Systems and Service-Oriented Engineering, pp. 78-103, 2015.

49. Garcia de Prado, A., Ortiz, G., Boubeta-Puig, J.: CARED-SOA: a Context-Aware Event-Driven Service Oriented Architecture. IEEE Access Journal, pp. 4646 – 4663, 2017.

50. Garcia de Prado, A., Ortiz, G., Boubeta-Puig, J.: COLLECT: Collaborative Context-aware Service Oriented Architecture for Intelligent Decision-making in the Internet of Things. Journal of Expert Systems with Applications, pp. 231-248, 2017.

51. Etzion, O., & Niblett, P. (2010). Event processing in action (20). Manning Publications Company.

52. Mule Soft Homepage, www.mulesoft.com

53. Rademakers, T., Dirksentt, J.: Open-source ESBs in action.1st edn. Greenwich, CT: Man-ning, 2009.

54. Apache Active MQ Homepage, http://activemq.apache.org/

55. Esper CEP engine Homepage, http://www.espertech.com/esper/

56. Quartz Esper module: https://docs.mulesoft.com/mule-user-guide/v/3.6/quartz-connector

57. Siddhi WSO2 Homepage, https://docs.wso2.com/display/CEP400/SiddhiQL+Guide+3.0#SiddhiQLGuide3.0-Joins

58. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica et al., "Above the clouds: A berkley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, University of California, Berkeley, Tech. Rep., 2009

59. A. Verma, G. Kumar, R. Koller, A. Sen et al., "CosMig: Modeling the Impact of Reconfiguration in a Cloud", 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, 25-27 July 2011, Signapore.

60. EC2, Amazon elastic compute cloud. http://aws.amazon.com/

61. Google app engine. Available online at: http://code.google.com/appengine/

62. A. Taherkordia, F. Zahida, Y. Verginadis, and G. Horn. Future Cloud Systems Design: Challenges and Research Directions. IEEE Access 6: 74120-74150, 2018.

63. M. Mao, M. Humphrey et al., "A Performance Study on the VM Startup Time in the Cloud", 2012 IEEE Fifth International Conference on Cloud Computing, 2012, Honolulu, HI, USA, DOI: 10.1109/CLOUD.2012.103.

64. F. Salfner, P. Troger, A. Polze et al., "Downtime Analysis of Virtual Machine Live Migration", DEPEND 2011: The Fourth International Conference on Dependability, August 21-27, 2011 – French Riviera.

65. Z. Izzah, M. Yusoh, M. Tang et al., "A penalty-based grouping genetic algorithm for multiple composite SaaS components clustering in Cloud", 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, Seoul, South Korea, DOI: 10.1109/CLOUD.2012.103.

66. K. Saniee et al., "A Simple Expression for Multivariate Language Interpolation", January 2008, DOI: 10.1137/08S010025.

67. G. Uyanik, N. Guler et al., "A study on multiple linear regression analysis" , Procedia - Social and Behavioral Sciences 106, pp 234 – 240 (2013).

68. P. Kokoszka, G.Young, "KPSS test for functional time series", Colorado State University, Colorado, USA, Tech. Rep., 2015

69. Rutherford, A.(2001). Introducing ANOVA and ANCOVA: a GLM approach. London: Sage Publications, London, UK.

70. G. Horn, P. Skrzypek, M. Prusinski, K. Materka, V. Stefanidis, Y. Verginadis. MELODIC: Selection and Integration of Open Source to Build an Autonomic Cross-Cloud Deployment Platform. International Conference on TOOLS 50+1: Technology of Object-Oriented Languages and System, 14-19 October,2019, Innopolis, Russia.

71. G. Horn and P. Skrzypek, "MELODIC: Utility Based Cross Cloud Deployment Optimisation," 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), Krakow, 2018, pp. 360-3.

72. http://camel-dsl.org/

73. Hutcheson, G.D.(2011). Ordinary Least-Squares Regression. In L. Moutinho and G.D. Hutcheson, The SAGE Dictionary of Quantitative Management Research. Pages 224-228.

74. https://memcached.org/

75. https://www.influxdata.com/

76. D. Baur and J. Domaschka, "Experiences from building a cross-cloud orchestration tool", Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms. ACM, 2016.

77. https://jclouds.apache.org/

78. https://en.wikipedia.org/wiki/Fast_Fourier_transform

79. https://en.wikipedia.org/wiki/Percentile

80. https://www.tutorialspoint.com/batch_script/batch_script_quick_guide.htm

81. Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., & Chan, K. (2018). When edge meets learning: Adaptive control for resource-constrained distributed machine learning. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. https://doi.org/10.1109/INFOCOM.2018.8486403

82. Alahakoon, D., Nawaratne, R., Xu, Y., De Silva, D., Sivarajah, U., & Gupta, B. (2020). Self-Building Artificial Intelligence and Machine Learning to Empower Big Data Analytics in Smart Cities. Journal of Information Systems Frontiers. https://doi.org/10.1007/s10796-020-10056-x

83. Inés Sittón-Candanedo, Ricardo S Alonso, Juan M Corchado, Sara Rodríguez-González, Roberto Casado-Vara (2019), "A review of edge computing reference architectures and a new global edge proposal".

84. https://en.wikipedia.org/wiki/Edge_computing

85. Jacub Konecny, H Brendan Mc Mahan, Daniel Ramage and Peter Richtarik. Federated Optimization: Distributed machine learning for on-device intelligence.arXiv preprint arXiv:1610.02527,2016a

86. https://en.wikipedia.org/wiki/Federated_learning

87. Lim, W.Y.B, Luong, N.C., Hoang., D.T., Jiao., Y., Liang., Y-C., Yang., Q., Niyato., D., & Miao., C. (2020). Federated Learning in Mobile Edge Networks: A Comprehensive Survey. IEEE Journal of Communications Surveys & Tutorials, Volume 22, Issue 3, pp. 2031-2063. https://doi.org/10.1109/COMST.2020.2986024

88. https://webrate.org/site/bdtechtalks.com/

89. Chen, T., Giannakis, G.B., Suny, T., & Yin, W., (2018). LAG: Lazily Aggregated Gradient for Communication Efficient Distributed Learning. NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages: 5055–5065. https://arxiv.org/abs/1805.09965

90. Chen, Y., Slawski, M., Ning, Y., & Rangwala, H. (2020). Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. IEEE International Conference on Big Data (Big Data). DOI: 10.1109/BigData50022.2020.9378161

91. Lu, X., Liao, Y., Lio, P., & Hui, P. (2020). Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing. IEEE Access Journal, Volume: 8, pp. 48970 – 48981. https://doi.org/10.1109/ACCESS.2020.2978082

92. Imakura, A., Inaba, H., Okada, Y., & Sakurai, T. (2021). Interpretable collaborative data analysis on distributed data. Journal Expert Systems with Applications, Elsevier, vol. 177, pp. 1-16. https://doi.org/10.1016/j.eswa.2021.114891

93. Fantacci, R. & Picano, B.(2020). A Federated Learning Framework for Mobile Edge Computing Nodes. Journal of CAAI Transactions on Intelligence Technology, Volume5, Issue 1, pp. 15-21. https://doi.org/10.1049/trit.2019.0049.

94. Lim, W.Y.B, Luong, N.C., Hoang., D.T., Jiao., Y., Liang., Y-C., Yang., Q., Niyato., D., & Miao., C. (2020). Federated Learning in Mobile Edge Networks: A Comprehensive Survey. IEEE Journal of Communications Surveys & Tutorials, Volume 22, Issue 3, pp. 2031-2063. https://doi.org/10.1109/COMST.2020.2986024

95. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. Journal of ACM Computing Surveys, Vol. 41, issue 3, pp. 1-58. https://doi.org/10.1145/1541880.1541882.

96. Srivastava, Y., MuraliShiv, V. & Dubey, R. (2019). A Performance Evaluation of Loss Functions for Deep Face Recognition. 7th National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics, NCVPRIPG 2019. https://arxiv.org/abs/1901.05903

97. Martin-Doñas, J.M., Gomez, A.M., Gonzalez, J.A., & Peinado, A.M. (2018). A Deep Learning Loss Function Based on the Perceptual Evaluation of the Speech Quality. Journal of IEEE Signal Processing Letters, Volume: 25, Issue: 11, pp. 1680 – 1684. https://doi.org/10.1109/LSP.2018.2871419.

98. Stochastic Gradient Descent. Retrieved 1st February, 2021 from https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

99. Wang, S., Tuor, T., Salonidis, T., Leung, K., Makaya, C., He., T., & Chan., K. (2019). Adaptive Federated Learning in Resource Constrained Edge Computing Systems. IEEE Journal on Selected Areas in Communications, Volume: 37, Issue: 6, pp. 1205 – 1221. https://doi.org/10.1109/JSAC.2019.2904348.

100. Sakr, F.; Bellotti, F.; Berta, R.; De Gloria, A. Machine Learning on Mainstream Microcontrollers. Sensors 2020, 20, 2638

101. Ravaglia, L.; Rusci, M.; Nadalini, D.; Capotondi, A.; Conti, F.; Benini, L.; Benini, L. A TinyML Platform for On-Device Continual Learning with Quantized Latent Replays. IEEE J. Emerg. Sel. Top. Circuits Syst. 2021, 11, 789–802.

102. Konecny, J., McMahan, H.B., Yu, F.X., Richtarik, P., Suresh, A.R., Bacon, D. (2016b) Federated learning: Strategies for improving communication efficiency. Machine Learning. Retrieved 1st March 2021 from: https://arxiv.org/pdf/1610.05492.

103. Chen, Y., Slawski, M., Ning, Y., & Rangwala, H. (2020). Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. IEEE International Conference on Big Data (Big Data). DOI: 10.1109/BigData50022.2020.9378161

104. Xie, C., Koyejo, O. & Gupta, I. (2020). Asynchronous Federated Optimization. OPT2020: 12th Annual Workshop on Optimization for Machine Learning. https://arxiv.org/abs/1903.03934

105. Liu, Y., Yu, J.J.Q., Kang, J., Niyato, D., & Zhang, S. (2020). Privacy-preserving traffic flow prediction: A federated learning approach. Journal IEEE Internet of Things, Volume: 7, Issue: 8, pp. 7751 – 7763. https://doi.org/10.1109/JIOT.2020.2991401

106. Lu, X., Liao, Y., Lio, P., & Hui, P. (2020). Privacy-Preserving Asynchronous Federated Learning Mechanism for Edge Network Computing. IEEE Access Journal, Volume: 8, pp. 48970 – 48981. https://doi.org/10.1109/ACCESS.2020.2978082

107. Qin, Y., Matsutani, H. & Kondo, M. (2020). A Selective Model Aggregation Approach in Federated Learning for Online Anomaly Detection. 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00119

108. Ek, S., Portet, F., Lalanda, P., & Vega, G. (2020). Evaluation of Federated Learning Aggregation Algorithms Application to Human Activity Recognition. UbiComp-ISWC '20: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, pages 638–643. https://doi.org/10.1145/3410530.3414321

109. Arivazhagan, M., Aggarwal, V., Singh, A., & Choudhary, S. (2020). Federated Learning with Personalization Layers. AISTATS 2020: The 23rd International Conference on Artificial Intelligence and Statistics. https://arxiv.org/abs/1912.00818

110. Ye, Y., Li, S., Liu, F., Tang. Y., & Hu, W. (2020). EdgeFed: Optimized Federated Learning Based on Edge Computing. Journal of IEEE Access, Volume 8, pp. 209191-209198. https://doi.org/10.1109/ACCESS.2020.3038287

111. Nishio, T. & Yonetani, R. (2019). Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. ICC 2019 - 2019 IEEE International Conference on Communications (ICC). http://dx.doi.org/10.1109/ICC.2019.8761315

112. Huang, L., Yin, Y., Fu, Z., Zhang, S., Deng, H., & Liu, D. (2020). LoAdaBoost: loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data. Journal Plos ONE, Article e 0230706. https://doi.org/10.1371/journal.pone.0230706.

113. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated Optimization in Heterogeneous Networks. Proceedings of the 3rd MLSys Conference. https://arxiv.org/abs/1812.06127

114. https://www.tensorflow.org/lite/microcontrollers

115. Koizumi, Y., Saito, S., Uematsu, H., Harada, N., and Imoto, K. ToyADMOS: A dataset of miniaturemachine operating sounds for anomalous sound detection. In Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 308–312, November 2019.

116. Chowdhery, A., Warden, P., Shlens, J., Howard, A., and Rhodes, R. Visual wake words dataset. arXiv preprint arXiv:1906.05721, 2019.

117. Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T.,Tr¨oster, G., Mill´an, J. d. R., and Roggen, D. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. Pattern Recognition Letters, 34(15):2033–2042, 2013.

118. Robert David et. All, "Tensorflow Lite Micro: Embedded Machine Learning on Tiny ML Systems, Proceedings of the 4th MLSys Conference, San Jose, CA,USA,2021.

119. Daniel Beutel et all, Flower: A friendly federated learning framework, March 2022.

120. LSTM Algorithm Definition and Analysis. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Long_short-term_memory.

121. Definition of Outliers. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Outlier

122. Stochastic Gradient Descent. Retrieved 1st February, 2021 from https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

123. Definition and decription of Docker environments. Retrieved 1st September, 2020, from https://www.docker.com/

124. Definition of Mean Absolute Error. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Mean_absolute_error

125. Definition of Mean Squared Error. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Mean_squared_error

126. Definition of Root Mean Square Deviation or Root Mean Square Error. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Root-mean-square_deviation

127. Definition of Mean Absolute Percentage Error. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Mean_absolute_percentage_error

128. Definition of Symmetric Mean Absolute percentage Error. Retrieved 1st March, 2021 from https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error

129. Armstrong, J. S., & Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. International journal of forecasting, vol. 8(1), pp.69-80. https://econpapers.repec.org/RePEc:eee:intfor:v:8:y:1992:i:1:p:69-80

130. Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. International Journal of Forecasting, vol 9(4), pp. 527-529. https://econpapers.repec.org/RePEc:eee:intfor:v:9:y:1993:i:4:p:527-529

131. Goodwin, P., & Lawton, R. (1999). On the asymmetry of the symmetric MAPE. International journal of forecasting, vol. 15(4), pp. 405-408. https://econpapers.repec.org/RePEc:eee:intfor:v:15:y:1999:i:4:p:405-408

132. Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International journal of forecasting, vol. 22(4), pp. 679-688. http://dx.doi.org/10.1016/j.ijforecast.2006.03.001

133. Liu, Y., Yu, J.J.Q., Kang, J., Niyato, D., & Zhang, S. (2020). Privacy-preserving traffic flow prediction: A federated learning approach. Journal IEEE Internet of Things, Volume: 7, Issue: 8, pp. 7751 – 7763. https://doi.org/10.1109/JIOT.2020.2991401

134. https://www.tensorflow.org/lite/performance/model_optimization

135. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. arXiv 2014, arXiv:1412.6980.

136. Definition of AdaGrad Optimizer. Available online: https://optimization.cbe.cornell.edu/index.php?title=AdaGrad (accessed on 20 April 2020).

137. Definition of RMSProp Optimizer. Available online: https://optimization.cbe.cornell.edu/index.php?title=RMSProp (accessed on 20 April 2020).

138. Chen, Y.; Slawski, M.; Ning, Y.; Rangwala, H. Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. arXiv 2020, arXiv:1911.02134.

139. Chen, T.; Giannakis, G.B.; Suny, T.; Yin, W. LAG: Lazily Aggregated Gradient for Communication Efficient Distributed Learning. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2–8 December 2018.

140. Bengio, Y. Dropout: A SimpleWay to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. 2014, 15, 1929–1958.

141. Definition and Description of Tensorflow Environments. Available online: https://www.tensorflow.org/ (accessed on 20 April 2020).

142. Definition of Keras Library of Tensorflow. Available online: https://www.tensorflow.org/api_docs/python/tf/keras (accessed on 20 April 2020).

143. Definition of HDF5 File Format. Available online: https://www.hdfgroup.org/solutions/hdf5/ (accessed on 20 April 2020).

144. Definition and Description of Docker Environments. Available online: https://www.docker.com/ (accessed on 20 April 2020).

145. Definition and Description of Docker Environments. Available online: https://www.docker.com/ (accessed on 20 April 2020).

146. Chen, T.; Giannakis, G.B.; Suny, T.; Yin, W. LAG: Lazily Aggregated Gradient for Communication Efficient Distributed Learning. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2–8 December 2018.

147. Li, S.; Cheng, Y.; Liu, Y.;Wang,W.; Chen, T. Abnormal Client Behavior Detection in Federated Learning. In Proceedings of the 2nd International Workshop on Federated Learning for Data Privacy and Confidentiality (NeurIPS 2019), Vancouver, BC, Canada, 13 December 2019; pp. 1–7.

148. R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I.Nappier, M. Natraj, T. Wang, P. Warden, R. Rhodes, "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems", Proceedings of Machine Learning and Systems 3 (MLSys 2021), pp.1-12, 2021.

149. V. Falbo, T.Apicella, D. Aurioso, L.Danese, F. Bellotti, R. Berta, A. De Gloria,"Analyzing Machine Learning on Mainstream Microcontrollers", In Proceedings of the International Conference on Applications in Electronics Pervading Industry Environment and Society, ApplePies 2019, Pisa, Italy, 12–13 September 2019.

150. H.B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A. Arcas, "Communication-efficient learning of deep networks from decentralized data", arXiv preprint arXiv: 1602.05629, 2016

151. Tensorflow optimization, https://www.tensorflow.org/lite/performance/model_optimization , accessed January 2022.