ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

"Πειραματική αποτίμηση επίδοσης κατανεμημένων συστημάτων μηχανικής μάθησης"

Ταξιάρχης Κούσκουρας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Επιβλέπων
Ιωάννης Κωνσταντίνου

Λαμία, 22/7 έτος 2022

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

INFORMATICS AND COMPUTATIONAL BIOMEDICINE

” Experimental evaluation of the performance of distributed machine learning systems”

Taxiarchis Kouskouras

Master thesis

Ioannis Konstantinou

Lamia

22/7/2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

“Πειραματική αποτίμηση επίδοσης κατανεμημένων συστημάτων μηχανικής μάθησης”

Ταξιάρχης Κούσκουρας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Ιωάννης Κωνσταντίνου

Λαμία, 22/7 έτος 2022

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο [«τίτλος εργασίας»] αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ

Ημερομηνία
22/7/2022

Υπογραφή

“Πειραματική αποτίμηση επίδοσης κατανεμημένων συστημάτων μηχανικής μάθησης”

Ταξιάρχης Κούσκουρας

Τριμελής Επιτροπή:

Ιωάννης Κωνσταντίνου (επιβλέπων)

Μαρία Κοζύρη

Γεώργιος Σταμούλης

ΠΕΡΙΛΗΨΗ

Η αύξηση των δεδομένων που παράγονται καθημερινά και η ανάγκη των οργανισμών να εκμεταλλευτούν τα δεδομένα αυτά έτσι ώστε να λύσουν προβλήματα όπως πχ να προτείνει σε πελάτες προϊόντα που είναι πιο πιθανόν να τους ενδιαφέρουν. Λύση σε τέτοια προβλήματα δίνει η μηχανική μάθηση με την εκπαίδευση μοντέλων για να λύνει προβλήματα. Λόγο του μεγάλου όγκου δεδομένων αλλά και την περιπλοκότητα των μοντέλων που χρειάζονται κάποια προβλήματα είναι αποτρεπτικό να εκπαιδευτούν τέτοια μοντέλα σε ένα μόνο υπολογιστή. Λόγο αυτού έχουν αναπτυχθεί διάφορες τεχνικές εκπαίδευσής των μοντέλων μηχανικής μάθησης με κατανεμημένο τρόπο. Σε αυτήν την διπλωματική θα δούμε την κατανεμημένη εκπαίδευση τριών μοντέλων μηχανικής μάθησης με δυο τεχνικές κατανεμημένης εκπαίδευσης του Tensorflow των Εξυπηρετητών Παραμέτρων και της στρατηγικής MultiWorkerMirroredStrategy και θα εξετάσουμε την ακρίβεια τους , τον χρόνο εκπαίδευση τους αλλά και την χρήση των πόρων της συστοιχίας με δυο διαφορετικά μεγέθη παρτίδας 128 και 256 και θα συγκρίνουμε τα αποτελέσματα μεταξύ τους έτσι ώστε να καταλήξουμε ποια θα ήταν η καλύτερη επιλογή για την συστοιχία μας

ABSTRACT

The increase in data that is produced daily and the need for organizations to take advantage of this data in order to solve problems such as recommending to customers products that are more likely to interest them. Machine learning provides a solution to such problems by training models to solve problems. Due to the large amount of data and the complexity of the models that need some problems, it is prohibitive to train such models on a single computer. Because of this, various techniques for training machine learning models in a distributed manner have been developed. In this thesis we will see the distributed training of three machine learning models with two distributed training techniques of Tensorflow of Parameter Servers and the MultiWorkerMirroredStrategy and we will examine their accuracy, their training time and the use of clusters resources with two different batch sizes 128 and 256 and we will compare the results between them so that we can come up with which would be the best choice for our cluster

Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Κίνητρο 1.1

Σήμερα η αύξηση στην παραγωγή δεδομένων έχει εκτοξευθεί περίπου στα 2.5 quintillion bytes. Αυτή η αύξηση προέρχεται από εταιρίες που αποθηκεύουν την συμπεριφορά των χρηστών , από τους ίδιους του χρήστες αλλά και από έξυπνες συσκευές. Μέσα από αυτά τα δεδομένα οι εταιρίες και διάφοροι οργανισμοί προσπαθούν να εξάγουν γνώση που μπορεί να χρησιμοποιηθεί σε υπηρεσίες όπως συστήματα συστάσεων κ.α. Η μηχανική μάθηση μπορεί να λύσει πολλά προβλήματα όπως αναφέραμε νωρίτερα με τα συστήματα συστάσεων όπου μοντέλα εκπαιδεύονται με δεδομένα επιλογής προϊόντων (μπορεί να είναι και άλλα πράγματα όπως μουσικά κομμάτια η υπηρεσίες) από άλλους πελάτες έτσι ώστε να μπορούν να προτείνουν προϊόντα που έχουν επιλεχθεί από παρόμοιους πελάτες. Για να εκπαιδευτούν τα μοντέλα μηχανικής μάθησης ερευνητές και ειδικοί του χώρου προσπαθούν να δημιουργήσουν καινούργιος αλγορίθμους εκπαίδευσης και γενικά τρόπους εκπαίδευσης έτσι ώστε να δημιουργήσουν μοντέλα με καλύτερη ακρίβεια αποτελεσμάτων. Ένα πρόβλημα που προσπαθούν να λύσουν πολύ ερευνητές είναι ότι λόγο της περιπλοκότητας κάποιόν μοντέλων η εκπαίδευση αλλά και ό όγκος των δεδομένων που χρειάζονται για την εκπαίδευση τους είναι αδύνατον να φιλοξενηθούν σε έναν υπολογιστή. Έτσι προσπαθούν να δημιουργήσουν διάφορες τακτικές κατανεμημένης εκπαίδευσης μοντέλων που στοχεύον είτε στην παραλληλοποιήση του μοντέλου είτε στην παραλληλοποιήση των δεδομένων

Συνεισφορά 1.2

Κατά την διάρκεια αυτής της διπλωματικής εκπαιδεύτηκαν κατανεμημένα τρία μοντέλα μηχανικής μάθησης πάνω στα δεδομένα του Mnist Dataset, του Fashion Mnist Dataset και του Kmnist Dataset Ειδικότερα:

- Δημιουργήθηκε μια συστοιχία υπολογιστών από 9 μηχανήματα στο υπολογιστικό νέφος του δικτυού Okeanos
- Σε κάθε μηχάνημα έγινε εγκατάσταση του υπολογιστικού πλαίσιού Tensorflow έκδοση 2.8.0
- Σε κάθε μηχάνημα έγινε εγκατάσταση του λογισμικού Apache Hadoop 3.2.1
- Στην συστοιχία υπολογιστών έγινε εγκατάσταση του κατανεμημένου συστήματος αρχείων HDFS

- Με την χρήστη του εργαλείου CLI της βιβλιοθήκης Tensorflow Datasets κατέβηκαν τα δεδομένα των Datasets Mnist, Fashion Mnist, Kmnist στην μορφή TFRecord

- Ανέβηκαν στα δεδομένα στο κατανεμημένο σύστημα αρχείων HDFS της συστοιχίας

- Εκπαιδεύτηκαν 3 μοντέλα ένα για κάθε Dataset με κατανεμημένο τρόπο ακολουθώντας την στρατηγική Εξυπηρετητών παραμέτρων χρησιμοποιώντας δυο μεγέθη παρτίδας 128 και 256

- Έγινε ανάλυση των πόρων που χρησιμοποιήθηκε για κάθε εκπαίδευση του μοντέλου

- Εκπαιδεύτηκαν 3 μοντέλα ένα για κάθε Dataset με κατανεμημένο τρόπο ακολουθώντας την στρατηγική MultiWorkerMirroredStrategy χρησιμοποιώντας δυο μεγέθη παρτίδας 128 και 256

- Έγινε ανάλυση των πόρων που χρησιμοποιήθηκε για κάθε εκπαίδευση του μοντέλου

- Συγκρίθηκαν οι διάφοροι τρόποι εκπαίδευσης

ΚΕΦΑΛΑΙΟ 2 Βιβλιογραφική επισκόπηση

Εισαγωγή 2.1

Σήμερα λόγο της κατακόρυφης αύξησης της παραγωγής δεδομένων από εταιρίες (αποθηκεύοντας δεδομένα από αγορές και από την συμπεριφορά των χρηστών), χρήστες του διαδικτύου (μέσα κοινωνικής δικτύωσης κλπ.) και γενικά από έξυπνες συσκευές που συνδέονται στο διαδίκτυο και παράγον δεδομένα η ανάγκη για εκπαίδευση μοντέλων μηχανικής μάθησης με κατανεμημένο τρόπο είναι πολύ μεγάλη ,διότι η αύξηση των δεδομένων που παράγεται είναι αποτρεπτική για την εκπαίδευση ενός μοντέλου μάθησης σε ένα μόνο μηχάνημα. Για αυτό τον λόγο ερευνητές από διάφορα πανεπιστήμια του κόσμου προσπαθούν να βρουν τρόπους έτσι ώστε να δημιουργήσουν στρατηγικές και αλγορίθμους που θα μειώσουν τον χρόνο που χρειάζεται ένα μοντέλων μηχανικής μάθησης να εκπαιδευτεί προσπαθώντας πάντα να κρατήσουν την ακρίβεια του μοντέλου όσο πιο ψηλά γίνεται.

Σχετικές εργασίες 2.2

Ερευνητές σε μια εργασία [1] που σχετίζεται με την εκπαίδευση κατανεμημένων μοντέλων μηχανικής μάθησης δημιουργώντας μια υλοποίηση ανοιχτού κώδικα της στρατηγικής Εξυπηρετητών Παραμέτρων (Parameter Server). Προσφέροντας τις εξής χαρακτηριστικά:

- **Αποδοτική επικοινωνία:** το ασύγχρονο μοντέλο επικοινωνίας δεν σταματάει τους υπολογισμούς (εκτός αν απαιτείται) . Έχει σχεδιαστεί έτσι ώστε να είναι κατάλληλο για εργασίες κατανεμημένης εκπαίδευσης μοντέλων μηχανικής μάθησης έτσι ώστε να μειώνεται όσο είναι δυνατόν η χρήση του δικτύου

- **Μοντέλα ευέλικτης συνέπειας:** Η χαλαρή συνέπεια «κρύβει» το κόστος συγχρονισμού και τις καθυστερήσεις. Έτσι επιτρέπεται η εξισορρόπηση στην αλγοριθμική σύγκλιση και την αποτελεσματικότητα του συστήματος. Αυτό βέβαια εξαρτάται από τα δεδομένα, στον αλγόριθμο και στο υλικό που χρησιμοποιείται.

- **Ελαστική επεκτασιμότητα:** Καινούργιοι κόμβοι μπορούν να προστεθούν χωρείς να χρειαστεί επανεκκίνηση το υπολογιστικό πλαίσιο

- **Ανοχή σε σφάλματα και ανθεκτικότητα:** Επιτρέπεται η επαναφορά μηχανών οι οποίες έχουν υποστεί μη-καταστροφικές αποτυχίες μέσα σε 1 δευτερόλεπτο χωρίς να σταματάνε τους υπολογισμούς.

- **Ευκολία στην χρήση:** Οι διαμοιρασμένες μεταβλητές που βρίσκονται στους εξυπηρετητές μπορούν να διαβαστούν εύκολα λόγου με τον τρόπο που έχουν αποθηκευτεί.

Για να δοκιμάσουν την αποτελεσματικότητα της υλοποίησης τους [1] χρησιμοποιήθηκε μια συστοιχία υπολογιστών με 16 φυσικούς πόρους , 192 GB μνήμη και δίκτυο με ταχύτητες 10 Gb για κάθε ένα μηχάνημα. 800 μηχανές είχαν τον ρόλο του εργάτη ενώ 200 είχαν τον ρόλο του Εξυπηρετητή παραμέτρων. Χρησιμοποιήθηκε ένας αλγόριθμος κατανεμημένης παλινδρόμησης (distributed regression algorithm) τελευταίας τεχνολογίας. Έγινε σύγκριση άλλων δυο συστημάτων όπως τα ονομάζουν Σύστημα Α και Σύστημα Β με την δικιά τους υλοποίησης του εξυπηρετητή παραμέτρων. Τα αποτελέσματα έδειξαν ότι το Σύστημα Β πέτυχε καλύτερο αποτέλεσμα σε μικρότερο χρόνο από το σύστημα Α και η υλοποίηση του εξυπηρετητή παραμέτρων πέτυχε καλύτερο αποτέλεσμα σε μικρότερο χρόνο από το Σύστημα Β. Ακόμη τα αποτελέσματα έδειξαν ότι η υλοποίηση του εξυπηρετητή παραμέτρων τους πέτυχε αποδοτικότερη χρήση των πόρων αφήνοντας λιγότερο από 2% αδρανή ενώ το Σύστημα Α και Σύστημα Β είχαν ποσοστά αδράνειας 32% και 52% αντίστοιχα.

Σε μια άλλη Έρευνα [2] οι ερευνητές αναγνωρίζοντας το πρόβλημα που μπορούν να δημιουργήσουν οι «μπαγιάτικες» (παλιές – μη ενημερωμένες) παράμετροι κατά την εκπαίδευση ενός μοντέλου μηχανικής μάθησης και πρότειναν μια λύση σε αυτό το πρόβλημα. Έτσι δημιούργησαν μια τεχνική που ονόμασαν SSPTable όπου βάση κάποιον κριτηρίων ο γρηγορότερος εργάτης θα πρέπει να περιμένει τον πιο αργό για να μην προχωρήσει πολύ παραπάνω από την διαδικασία εκπαίδευσης από αυτόν και έτσι να μην υπάρχει τεράστια διαφορά στις παραμέτρους τους. Η τεχνική δουλεύει βάση μιας σταθερά τη τιμή της οποίας μπορεί να αλλάξει ο χρήστης. Αυτή η μεταβλητή με όνομα staleness threshold δρα ως κατώφλι για το πόσο «μπαγιάτικη» μπορεί να είναι μια μεταβλητή και αν περάσει αυτό το κατώφλι ο εργάτης θα πρέπει να επικοινωνήσει με κάποιον εξυπηρετητή παραμέτρου έτσι ώστε να λάβει την ανανεωμένη παράμετρο. Στα πειράματα που πραγματοποιήσαν οι ερευνητές πάνω σε μια συστοιχία υπολογιστών με μηχανήματα τα οποία διέθεταν 8 πυρήνες (είτε με 2,3 GHz ή με 2,5 GHz) και μνήμη 23 GB σύγκριναν την τεχνική που υλοποιήσαν (SSP) , με μια τεχνική μάθησης όπου οι εργάτες σε κάθε βήμα περιμένουν ο ένας τον άλλον και συγχρονίζουν τις μεταβλητές τους (BSP) και την ασύγχρονη μάθηση όπου ο κάθε ένας εργάτης εκπαιδεύει μόνος του το μοντέλο μηχανικής μάθησης με τις δικές του μεταβλητές μέχρι να ζητήσει από κάποιον εξυπηρετητή παραμέτρων τις ανανεωμένες τιμές χωρίς να συγχρονίζεται με τους άλλους εργάτες. Κατά την διάρκεια των πειραμάτων χρησιμοποιήθηκε διαφορετικός αριθμός μηχανών (8 μηχανήματα, 32 μηχανήματα και 32 μηχανήματα με 10% minibatches).Σε όλα τα πειράματα εκτός το πείραμα με τα 8 μηχανήματα η μέθοδος που υλοποιήσανε (SSP) ήταν γρηγορότερη από τις άλλες δυο

Σε άλλη έρευνα [3] οι ερευνητές αρχικά εξετάζουν τις τεχνικές για την ενημέρωση των μεταβλητών στους εξυπηρετητές παραμέτρων. Οι τεχνικές είναι η σύγχρονη ανανέωση (BSP) την ασύγχρονη ανανέωση (ASP) και την τεχνική με την σταθερά που περιορίζει

πόσο «μπαγιάτικη» μπορεί να είναι μια μεταβλητή (SSP). Συμπεραίνουν ότι η SSP είναι μια ενδιάμεση λύση των άλλων δυο επιλογών και είναι η μια καλή επιλογή από άποψης χρόνου εκπαίδευσης και ακρίβειας του μοντέλου όμως ανακάλυψαν και προβλήματα που μπορούν να προκύψουν από αυτήν την σταθερά. Αρχικά συμπεραίνουν ότι είναι δύσκολο για κάποιο χρήστη να ξέρει εξαρχής ποια είναι η κατάλληλη τιμή για να επιλέξει για την σταθερά έτσι θα πρέπει να κάνει πολλά πειράματα από μια σειρά τιμών ώστε να πετύχει καλά αποτελέσματα, ακόμη όταν αλλάζει κάποια υπερ παράμετρος του νευρωνικού δικτύου θα πρέπει να αλλάξει και αυτή η τιμή. Επίσης μια σταθερά μπορεί να μην είναι κατάλληλη για όλη την διάρκεια της εκπαίδευσης . Για να λυθούν αυτά τα προβλήματα προτείνουν μια υλοποίηση όπου η τιμή του πόσο «μπαγιάτικη» μπορεί να είναι μια μεταβλητή αλλάζει δυναμικά (DSSP). Για να δοκιμάσουν την υλοποίηση τους οι ερευνητές χρησιμοποιήσαν μια συστοιχία υπολογιστών με 4 εξυπηρετητές IBM POWER8 όπου κάθε εξυπηρετητής διαθέτει 4 NVIDIA P100 Μονάδες Επεξεργασίας Γραφικών. Ακόμη κάθε εξυπηρετητής διαθέτει 512 GB μνήμη και 2 X 10 πυρήνες επεξεργαστή και η ταχύτητα δικτύου είναι 100 Gbps. Τα αποτελέσματα των πειραμάτων δείχνουν ότι ο DSSP και ο SSP πετυχαίνουν μεγαλύτερη ακρίβεια από τον ASP και ο ASP τελειώνει την εκπαίδευση πιο γρήγορα από ότι ο BSP. Τέλος όταν η σταθερά που ορίζει πόσο «μπαγιάτικη» μπορεί να είναι μια μεταβλητή του SSP είναι μεγάλη τελειώνει πιο γρήγορα την εκπαίδευση από τον DSSP με λιγότερη όμως ακρίβεια μοντέλου

Άλλοι ερευνητές [4] παρακολουθώντας την πορεία στην έρευνα της εκπαίδευσης των μοντέλων μηχανικής μάθησης αυξάνοντας την κλίμακα της «βαθιάς» μάθησης παίρνοντας υπόψιν στον αριθμό των παραμέτρων και στον αριθμό των παραδειγμάτων εκπαίδευσης αποφάσισαν να δημιουργήσουν ένα υπολογιστικό πλαίσιο. Στην έρευνα τους και την δημιουργία του υπολογιστικού πλαίσιού DistBelief οι ερευνητές είχαν υπόψιν το πρόβλημα της εκπαίδευσης ενός μοντέλου μηχανικής μάθησης με δισεκατομμύρια παραμέτρους και δεκάδες χιλιάδες πυρήνες Κεντρικών Μονάδων Επεξεργασίας. Έτσι το DistBelief μπορεί να αξιοποίηση συστοιχίες υπολογιστών με πολλές μηχανές και έτσι μπορεί να εκπαιδεύσει μεγάλο μοντέλα. Για την κατανεμημένη εκπαίδευση δημιουργήσαν 2 αλγορίθμους κατανεμημένης εκπαίδευσης. Τα πειράματα τα οποία δοκιμάστηκε το DistBelief ήταν ένα μοντέλο για αναγνώρισή αντικειμένων σε εικόνες και ένα μοντέλο για ακουστική επεξεργασία για αναγνώριση ομιλίας. Τα πειράματα έτρεξαν σε διαφορετικό αριθμό μηχανών και κατάφεραν 60% καλύτερη απόδοση από την καλύτερη απόδοση 21k στην κατηγορία ταξινόμησης ImageNet.

Σε άλλη Έρευνα [5] οι ερευνητές παρατήρησαν το πρόβλημα που παρουσιάζει η τεχνική των Εξυπηρετητών Παραμέτρων όταν χρειάζεται να επικοινωνήσουν οι εργάτες για να λάβουν το ανανεωμένο μοντέλο. Έτσι δημιούργησαν μια καινούργια τεχνική συγχρονισμού που την ονόμασα OSP στην οποία η καθυστέρηση αναμονής αφαιρείται

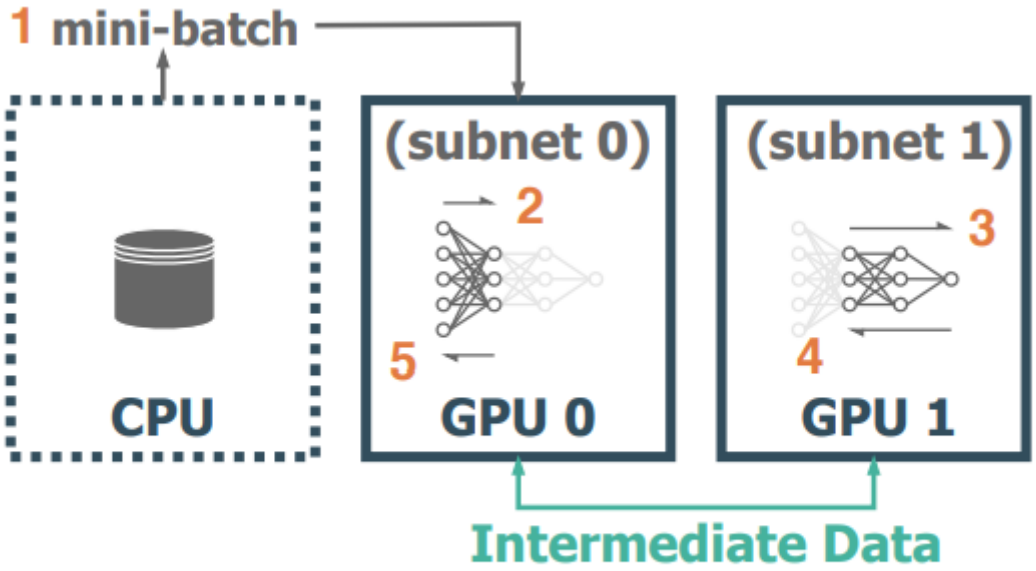
εκτελώντας εργασίες υπολογισμού και επικοινωνίας παράλληλα. Κατά την διάρκεια της έρευνας οι ερευνητές παραθέτουν μαθηματικές αποδείξεις για την σύγκλιση της τεχνικής που δημιούργησαν. Για να δοκιμάσουν την OSP την σύγκριναν με άλλες τεχνικές συγχρονισμού όπως ο BSP, ο KBatchAsync και ο SSP. Η συστοιχία που χρησιμοποιήθηκε για να δοκιμαστεί ο OSP ήταν μια συστοιχία υπολογιστών 8 μηχανές και μια άλλη συστοιχία με 16 μηχανές. Κάθε μηχανή είχε 4 πυρήνες (2,6 GHz) , 8 GB μνήμη και δίκτυο 10 Gbps. Οι τεχνικές δοκιμάστηκαν σε δυο τύπος προβλημάτων. Στον πρώτο (Convex Problems) ο OSP πέτυχε ταχύτητα σύγκλισης 6 φορές μεγαλύτερη από την τεχνική BSP και καλύτερη ποιότητα σύγκλισης σε κάθε επανάληψη. Για το δεύτερο πρόβλημα (Non-Convex Problems) η ταχύτητα σύγκλισης είναι μεγαλύτερη από τις άλλες δυο τεχνικές και παρατηρείται ότι ο BSP και ο KBatchAsync αργούν πιο πολύ όταν τρέχουν στην δεύτερη συστοιχία υπολογιστών (16 μηχανήματα) επειδή το κόστος επικοινωνίας αυξάνεται, η ποιότητα σύγκλισης επίσης είναι καλύτερη από τις άλλες δυο τεχνικές .

ΚΕΦΑΛΑΙΟ 3 Κατανεμημένοι τρόποι εκπαίδευσης

Η διαδικασία εκπαίδευσης ενός μοντέλου μηχανικής μάθησης λόγου του μεγάλου όγκου δεδομένων καθώς και της περιπλοκότητας τον μοντέλων που χρειάζονται για να λυθούν περίπλοκα προβλήματα μπορεί να πάρει από μέρες μέχρι βδομάδες μέχρι να ολοκληρωθεί. Για αυτόν τον λόγο ερευνητές ψάχνουν τρόπο να κατανέμουν την εκπαιδεύσει σε περισσότερα μηχανήματα. Οι δυο τρόποι κατανεμημένης εκπαίδευσης είναι η μέθοδος του παράλληλου μοντέλου (Model Parallelism) και η μέθοδος των παράλληλων δεδομένων (Data Parallelism)

Ενότητα 3.1 Η μέθοδος του παράλληλου μοντέλου (Model Parallelism)

Ένα πρόβλημα που δημιουργείται από την αυξανόμενη πολυπλοκότητα και του μεγάλου μεγέθους τους που απαιτούν τα μοντέλα μηχανικής μάθησης (πχ. Computer vision) σήμερα είναι ότι δεν μπορούν να φορτωθούν σε μια μόνο Μονάδα Επεξεργασίας Γραφικών (GPU) (ή σε ένα μόνο μηχάνημα που δεν διαθέτει Μονάδα Επεξεργασίας Γραφικών). Η τεχνική του παράλληλου μοντέλου χωρίζει το μοντέλο σε κομμάτια [6] και τα διανέμει σε διαφορετικά μηχανήματα – Μονάδες Επεξεργασίας Γραφικών (όταν δεν υπάρχει Μονάδα Επεξεργασίας Γραφικών στο σύστημα η Κεντρική Μονάδα Επεξεργασίας CPU αναλαμβάνει την εκπαίδευση) . Κάθε Κεντρική μονάδα επεξεργασίας αναλαμβάνει να υπολογίσει και να ανανεώσει τα βάρη του κομματιού του μοντέλου που της αναλογεί (συνήθως κάποιο στρώμα-επίπεδο layer του μοντέλου μηχανικής μάθησης) όπως βλέπουμε στην Εικόνα 1

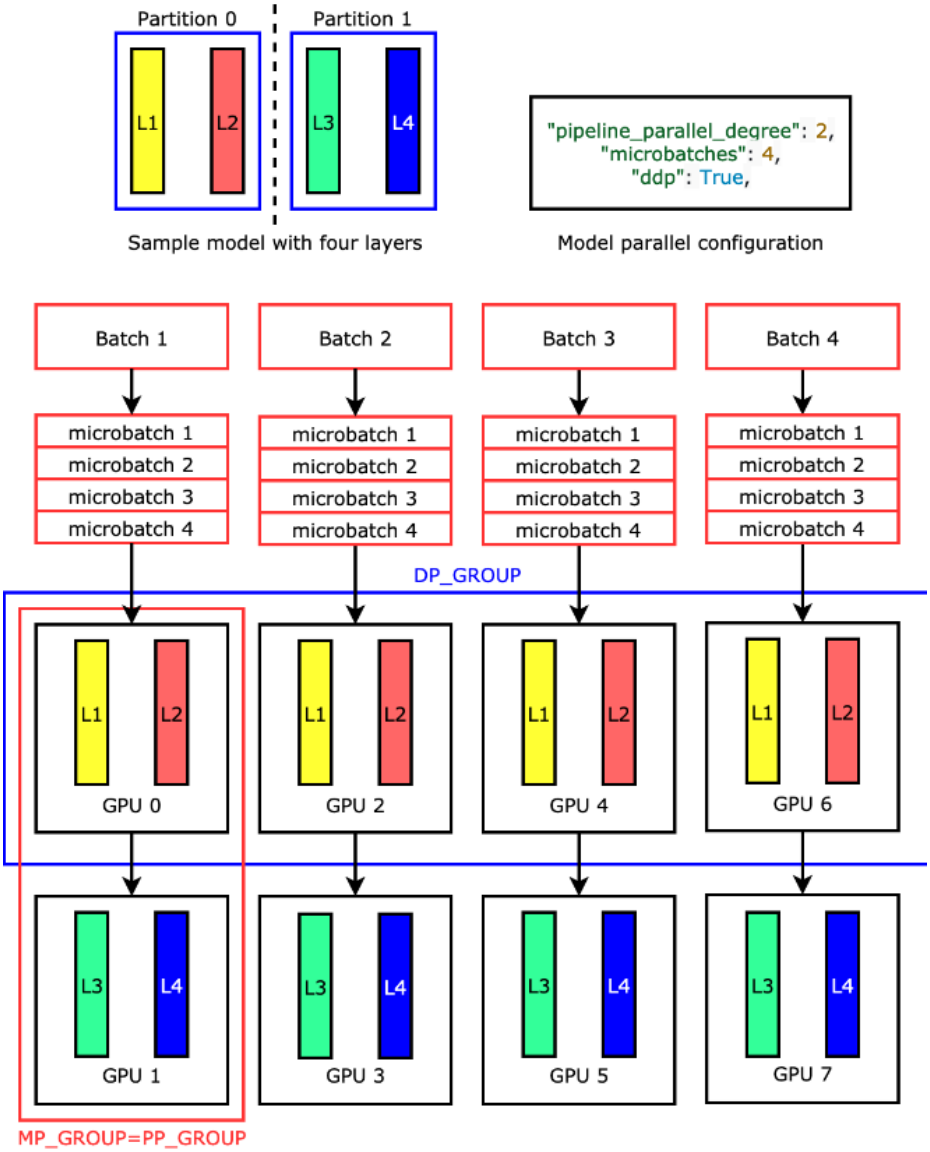


(b) Model Parallelism.

Εικόνα 1 εικόνα από το paper: EFFICIENT AND ROBUST PARALLEL DNN TRAINING THROUGH MODEL PARALLELISM ON MULTI-GPU PLATFORM

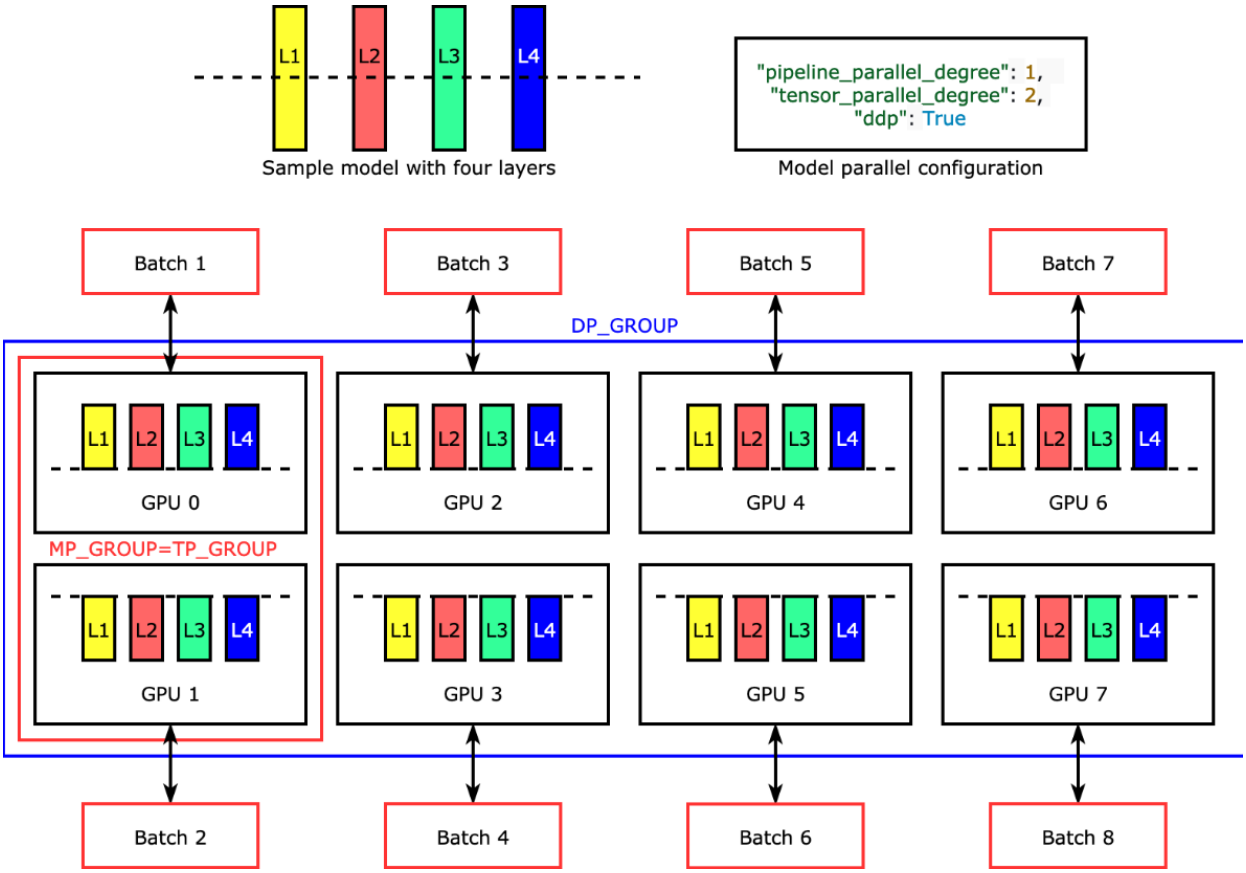
Ένα παράδειγμα ενός παράλληλου μοντέλου είναι η υπηρεσία της Amazon SageMaker [7] όπου παρέχει δυο τύπος παραλληλοποίησης μοντέλου. Ο πρώτος είναι παραλληλοποιήση αγωγού (Pipeline parallelism) και παραλληλοποιήση Tensor (Tensor parallelism)

Η παραλληλοποιήση αγωγού χωρίζει τα σετ των επιπέδων ή των εργασιών μοιράζοντας τας στα σετ των συσκευών , κάθε διεργασία παραμένει άθικτη. Για να γίνει αυτή η διεργασία ο προγραμματιστής θα πρέπει να ορίσει τον αριθμό του διαχωρισμού του μοντέλου. Μια απεικόνιση της παραλληλοποίησης αγωγού μπορούμε να δούμε στην Εικόνα 2



Εικόνα 2 Pipeline Parallelism εικόνα απο https://docs.aws.amazon.com/sagemaker/latest/dg/model-parallel-intro.html

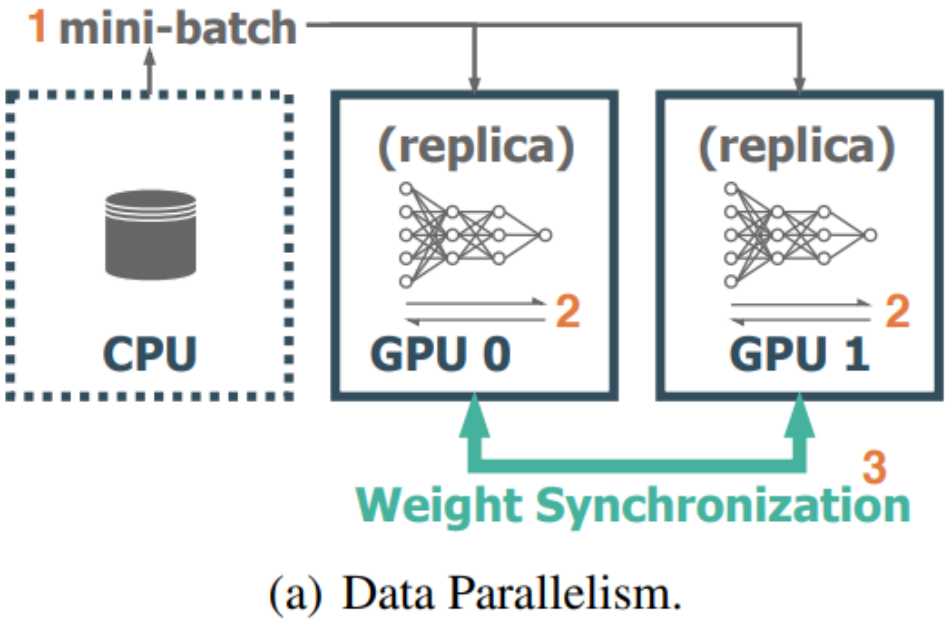
Ο άλλος τρόπος παραλληλοποίησης που προσφέρει το SageMaker η παραλληλοποιήση Tensor διαχωρίζει επίπεδα σε περισσότερα κομμάτια και τα μοιράζει στις συσκευές έτσι ώστε να εκτελεστούν παράλληλα. Ένα παράδειγμα αυτού του τρόπου παραλληλοποίησης μοντέλου μπορούμε να δούμε στην Εικόνα 3



Εικόνα 3 Tensor Parallelism εικόνα απο https://docs.aws.amazon.com/sagemaker/latest/dg/model-parallel-intro.html

Ενότητα 3.2 Η μέθοδος των παράλληλων δεδομένων (Data parallel)

Για την επιλογή της παραλληλοποίησης των δεδομένων [6] κάθε Μονάδα Επεξεργασίας Γραφικών (ή Κεντρική Μονάδα Επεξεργασίας όταν δεν υπάρχει Μονάδα Επεξεργασίας Γραφικών στα μηχανήματα) διαθέτει ένα πιστό αντίγραφο του μοντέλου. Τα δεδομένα όμως χωρίζονται και διαμοιράζονται σε κάθε μια μηχανή και έτσι κάθε μια εκπαιδεύει το τοπικό της μοντέλο με τα δικά της δεδομένα. Οι υπολογισμοί μετά μεταφέρονται στους εξυπηρετητές παραμέτρων (Parameter Servers) έτσι ώστε να ενωθούν. Αυτή η διαδικασία ονομάζεται συγχρονισμός βαρών του μοντέλου. Στην Εικόνα 4 μπορούμε να δούμε μια απεικόνιση της παραλληλοποίησης δεδομένων.



(a) Data Parallelism.

Εικόνα 4 Παραλληλοποιήση δεδομένων εικόνα από το paper: EFFICIENT AND ROBUST PARALLEL DNN TRAINING THROUGH MODEL PARALLELISM ON MULTI-GPU PLATFORM

Ένα παράδειγμα αυτής της τεχνική είναι η στρατηγική των εξυπηρετητών παραμέτρων Όπου υπάρχουν οι κόμβοι εργάτες και οι κόμβοι εξυπηρετητές παραμέτρων. Οι κόμβοι εργάτες όπως είπαμε και πιο πριν υπολογίζουν τα δικά τους αποτελέσματα των βαρών βάση των κομματιών δεδομένων που τους έχουν ανατεθεί πάνω στο τοπικό τους μοντέλο. Οι εξυπηρετητές παραμέτρων διαχειρίζονται τις μεταβλητές καθώς και την

ενημέρωση τους με τις τιμές των εργατών. Πάνω στις μεταβλητές είναι σύνηθες να εφαρμόζονται και τεχνικές τεμαχισμού (sharding) , ανάλογα με κάποια κριτήρια, και διαμοιράζονται σε πολλούς εξυπηρετητές παραμέτρων για να μπορούν να διαχειρίζονται μεγάλου μεγέθους μεταβλητές. Στην Εικόνα 5 μπορούμε να δούμε μια απεικόνιση της στρατηγικής Εξυπηρετητών παραμέτρων.

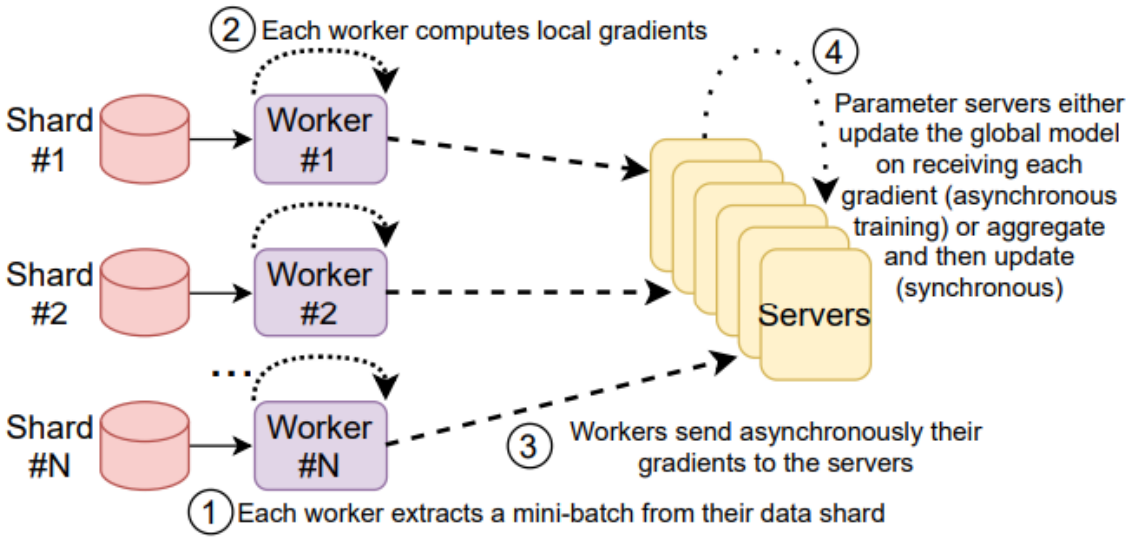


Fig. 2: Parameter Server Training

Εικόνα 5 Στρατηγική Εξυπηρετητών παραμέτρων εικόνα απο το paper A Performance Evaluation of Distributed Deep Learning Frameworks on CPU Clusters Using Image Classification Workloads

ΚΕΦΑΛΑΙΟ 4 Κατανεμημένα συστήματα επεξεργασίας και αποθήκευσης για εκτέλεση αλγορίθμων μηχανικής μάθησης:

Σε αυτό το κεφάλαιο θα γίνει ανάλυση των υπολογιστικών πλαισίων που χρησιμοποιήθηκαν. Στο υποκεφάλαιο 4.1 αναλύομε το υπολογιστικό πλαίσιο Tensorflow αρχίζοντας στην ενότητα 4.1.1 με μια σύντομη αναδρομή της ιστορίας του υπολογιστικού πλαισιού Tensorflow. Έπειτα θα μιλήσουμε για το Tensorflow API όπου θα γίνει ανάλυση των επιλογών που δίνει το Tensorflow για εκπαίδευση σε ποιες συσκευές και ποιες γλώσσες προγραμματισμού μπορεί να χρησιμοποιήσει ο προγραμματιστής που θέλει να το επιλέξει. Μετά στην ενότητα 4.1.3 κάνουμε μια ανάλυση για το πως δουλεύει το Tensorflow. Στις επόμενες δυο ενότητες δηλαδή για την ενότητα 4.1.4 και 4.1.5 μιλάμε για Tensorflow Dataset και τα TFRecords. Τέλος στην ενότητα 4.1.6 γίνεται ανάλυση των στρατηγικών κατανεμημένης εκπαίδευσης που προσφέρει το υπολογιστικό πλαίσιο. Στο υποκεφάλαιο 4.2 μιλάμε για το κατανεμημένο σύστημα αρχείων HDFS του υπολογιστικού πλαισιού Apache Hadoop περιγράφοντας την αρχιτεκτονική του και τον τρόπο λειτουργίας του.

Υποκεφάλαιο 4.1 Το υπολογιστικό πλαίσιο Tensorflow

Ενότητα 4.1.1 Η Ιστορία του Tensorflow

Το Tensorlofw είναι ένα υπολογιστικό πλαίσιο ανοιχτού κώδικα που χρησιμοποιείται για την δημιουργία και εκπαίδευση μοντέλων μηχανικής μάθησης. Το Tensorflow δημιουργήθηκε από την ομάδα Google Brain. Η Google Brain είναι μία ομάδα της Google η οποία ως κύριο στόχο της έχει την έρευνα της βαθιάς μάθησης και τεχνητής νοημοσύνης (deep learning artificial intelligence). Στην αρχή δημιουργήθηκε για εσωτερική χρήση και έπειτα το 2015 το υπολογιστικό πλαίσιο διατέθηκε στο κοινό με άδεια Apache licence 2.0. Το 2019 δημιουργήθηκε μια νέα έκδοση του Tensorflow 2.0 μια βελτίωση της προηγούμενης έκδοσης. Το Tensorflow μπορεί να χρησιμοποιηθεί σαν βιβλιοθήκη με πολλές γλώσσες προγραμματισμού όπως C++, java , javascript και python.

Ενότητα 4.1.2 Το Tensorflow API

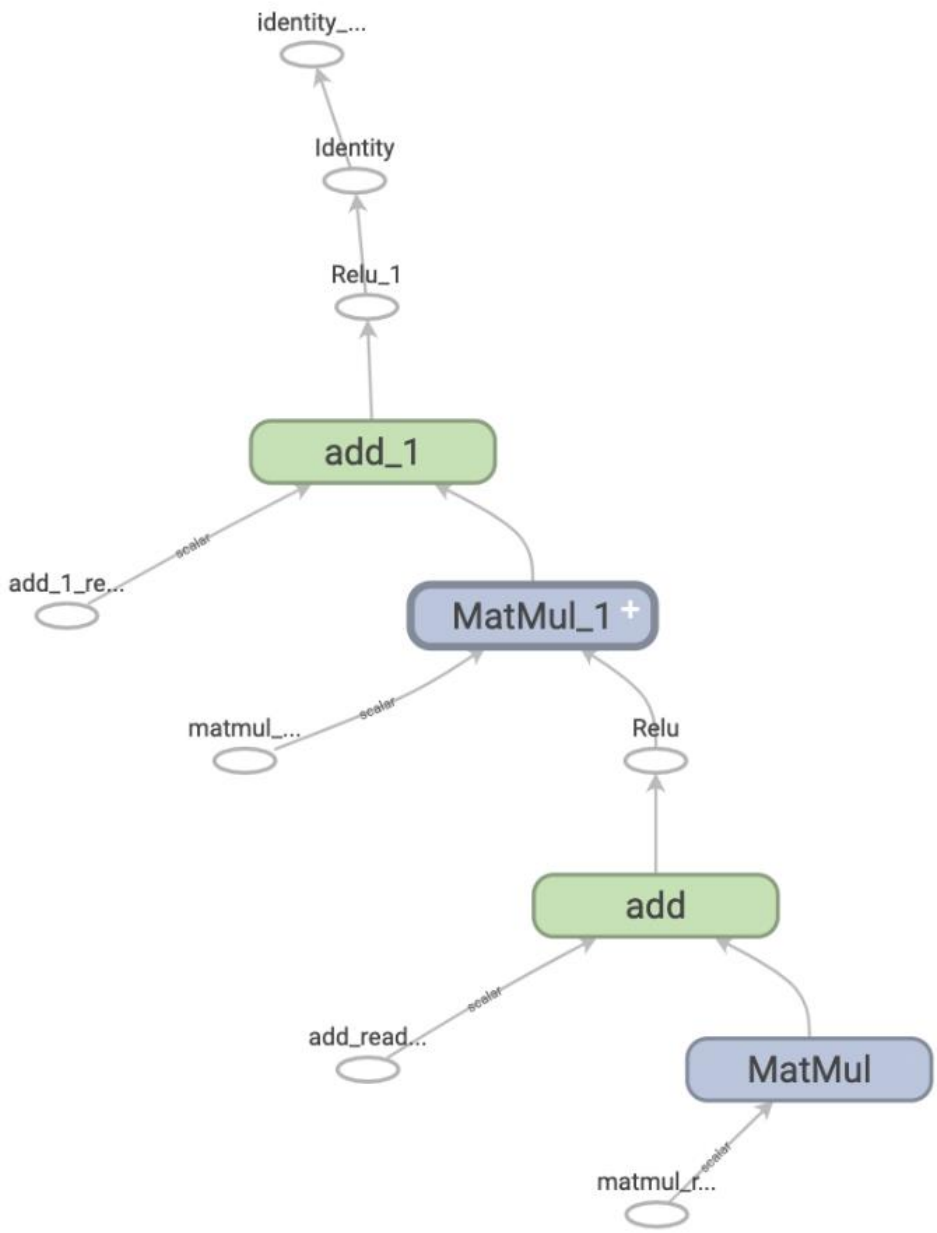
Η έκδοση του Tensorflow 2.0 ήρθε να δώσει λύσει σε προβλήματα που είχαν εκφράσει οι χρήστες του κάνοντας το υπολογιστικό πλαίσιο ευκολότερο στην χρήση (π.χ. με την εισαγωγή του Keras API ενός άλλου υπολογιστικού πλαισιού για εκπαίδευση μοντέλων μηχανικής μάθησης) και πιο αποδοτικό. Ακόμη αυτή η αναβάθμιση έκανε την κατανεμημένη εκπαίδευση πολύ πιο εύκολη. Πολλές εταιρίες χρησιμοποιούν το

Tensorflow για την εκπαίδευση μοντέλων μηχανικής μάθησης όπως η Google, η Twitter, η Coca Cola κ.α. [8]. Το υπολογιστικό πλαίσιο Tensorflow με την υποστήριξη πολλών γλωσσών προγραμματισμού επιτρέπει την χρήση του σε πολλές και διαφορετικών τύπων συσκευές. Το Tensorflow διαθέτει την δυνατότητα εκπαίδευσης ενός μοντέλου μηχανικής μάθησης με τον κλασσικό τρόπο δηλαδή με την συγγραφή κώδικα με γλώσσες προγραμματισμού όπως η C++ και η Python αλλά και την εκπαίδευση μοντέλων μηχανικής μάθησης σε προγράμματα περιήγησης (browsers) με την χρήση της Javascript [9] ακόμη, διαθέτει δυνατότητα εκπαίδευσης μοντέλων μηχανικής μάθησης σε κινητά, σε μικροελεγκτές αλλά και σε ακροδικτυακές συσκευές (edge devices) [10]. Το Tensorflow διαθέτει την δυνατότητα χρήσης της Μονάδα Επεξεργασίας Γραφικών (GPU) όταν υπάρχει στο σύστημα για την δημιουργία και την εκπαίδευση των μοντέλων μηχανικής μάθησης αλλά και της Κεντρικής μονάδας επεξεργασίας (CPU) όταν το σύστημα μας δεν διαθέτει Μονάδα επεξεργασίας γραφικών. Η χρήση της Μονάδας Επεξεργασίας Γραφικών προτιμάται έναντι της κεντρικής μονάδας επεξεργασίας για την εκπαίδευση μοντέλων μηχανικής μάθησης αφού ο τρόπος με τον οποίον είναι κατασκευασμένη η κάρτα γραφικών επιφέρει γρηγορότερη εκπαίδευση. Όταν το σύστημα διαθέτει Μονάδα Επεξεργασίας Γραφικών το Tensorflow χρησιμοποιεί την Κεντρική Μονάδα Επεξεργασίας για διάφορες εργασίες που σχετίζονται με τα δεδομένα όπως προεπεξεργασία, προανάκτηση δεδομένων (prefetch) [11] προτού τελειώσει η διαδικασία που εκτελεί η Μονάδα Επεξεργασίας Γραφικών ή διαδικασίες η οποίες δεν είναι υλοποιημένες από το υπολογιστικό πλαίσιο για Μονάδες επεξεργασίας γραφικών όπως η εργασία cast (tf.cast) [12].

Ενότητα 4.1.3 Πως δουλεύει το υπολογιστικό πλαίσιο Tensorflow

Αρχικά για να δούμε πως δουλεύει το υπολογιστικό πλαίσιο θα πρέπει να δώσουμε προσοχή στους δυο τρόπους λειτουργίας που διαθέτει το υπολογιστικό πλαίσιο για εκτέλεση. Οι δυο τρόποι εκτέλεσης είναι η «πρόθυμη» εκτέλεση (eager execution) και η εκτέλεση γράφου (graph execution). Κατά την «πρόθυμη» εκτέλεση υπολογιστικό πλαίσιο οι εργασίες εκτελούνται από την γλώσσα προγραμματισμού που έχουμε διαλέξει εργασία προς εργασία με την σειρά και τα αποτελέσματα αυτών των εργασιών επιστρέφονται στην γλώσσα προγραμματισμού που έχουμε επιλέξει [13]. Αυτός ο τρόπος εκτέλεσης είναι ιδιαίτερα χρήσιμος κατά την διάρκεια ελέγχου του κώδικα που αναπτύσσουμε για διαδικασίες αποσφαλμάτωσης. Ο δεύτερος τρόπος εκτέλεσης με την χρήση γράφων προσφέρει αποδοτικότερες εκτελέσεις αλλά και την φορητότητα των υπολογισμών (πχ: οι εργασίες από ένα πρόγραμμα γραμμένο με την γλώσσα προγραμματισμού python μπορούν να εκτελεστούν σε άλλο σύστημα που χρησιμοποιεί το Tensorflow API με άλλη γλώσσα προγραμματισμού πχ: C++ απλά χρησιμοποιώντας του Γράφους που παράχθηκαν με την εκτέλεση Γράφων). Οι Γράφοι του Tensorflow [13] είναι δομές δεδομένων όπου περιέχουν σετ από αντικείμενα Tensorflow εργασιών. Μια εργασία Tensorflow [14] είναι ένας κόμβος σε ένα Tensorflow Γράφο όπου παίρνει ως είσοδο ένα ή περισσότερους Tensorflow Tensors [15]μια απεικόνιση που χρησιμοποιεί το Tensorflow για πολυδιάστατους πίνακες στοιχείων. Τα αντικείμενα εργασιών Tensorflow αναπαριστούν μονάδες υπολογισμών πάνω σε δεδομένα

Tensorflow Tensors. Οι Γράφοι μπορούν να αποθηκευτούν και να φορτωθούν σε συστήματα για εκτέλεση.



Εικόνα 6 Απεικόνιση ενός γράφου εικόνα απο https://www.tensorflow.org/guide/intro_to_graphs

Στην Εικόνα 6 μπορούμε να δούμε ένα παράδειγμα ενός Tensorflow γράφου από ένα νευρωνικό δίκτυο δύο επιπέδων. Οι Γράφοι δίνουν επιπλέον δυνατότητα στον μεταγλωττιστή να επιβάλει βελτιώσεις με την χρήση μεταμορφώσεων (Transformations). Το υπολογιστικό πλαίσιο Tensorflow διαθέτει δικό του σύστημα βελτιστοποίησης Γράφον που ονομάζεται Grappler. Ο Grappler μπορεί να επιβάλει διάφορες μεταμορφώσεις στου Γράφους για να βελτιώσει την εκτέλεση τους κάποιες από αυτές είναι [16]:

- Αριθμητική βελτιστοποίηση: απλοποιεί αριθμητικές εργασίες εξαλείφοντας κοινές υπό εκφράσεις και απλοποιώντας αριθμητικές δηλώσεις

- Βελτιστοποίηση Μνήμης: Αναλύει τον Γράφο και ψάχνει να βρει την κορύφωση ως προς την χρήση της μνήμης και έπειτα εισάγει εργασίες αντιγραφής από την Κεντρικής Μονάδας Επεξεργασίας–Μονάδα Επεξεργασίας Γραφικών για να κάνει εναλλαγή από την μνήμη της Μονάδας Επεξεργασίας Γραφικών στην μνήμη της Κεντρικής Μονάδας Επεξεργασίας έτσι ώστε να μειώσει την μέγιστη χρήση της μνήμης

- Ξεχωρίζει υπό μέρη των υπολογισμών όπου είναι ανεξάρτητα και τα προγραμματίζει να εκτελεστούν σε διαφορετικά νήματα (threads) ή συσκευές (devices)

- Εκτελεί μικρές εργασίες στην Κεντρική Μονάδα Επεξεργασίας (αυτή η βελτιστοποίηση είναι κλειστή από προεπιλογή)

- Βελτιστοποιητής «κλαδέματος» (pruning): Ο βελτιστοποιητής «κλαδεύει» κόμβους οι οποίοι δεν επηρεάζουν το αποτέλεσμα του Γράφου. Συνήθως εκτελείται πρώτη αυτή η βελτιστοποίηση έτσι ώστε να μικρύνει το μέγεθος του Γράφου και να κάνει την διαδικασία επεξεργασίας-βελτιστοποίησης του Γράφου πιο γρήγορη

- Βελτιστοποιητής αναδίπλωσης σταθερών (constant folding): Συμπεραίνει στατικά την τιμή των Tensors αναδιπλώνοντας σταθερούς κόμβους στον Γράφο και υλοποιεί το αποτέλεσμα χρησιμοποιώντας σταθερές

Για να εκμεταλλευτεί ο προγραμματιστής της βελτιστοποίησης των Γράφων μπορεί για την γλώσσα προγραμματισμού python να χρησιμοποιήσει τον decorator @tf.function στις συναρτήσεις για να δημιουργήσει μια Tensorflow συνάρτηση (Function) . Η Tensorflow συνάρτηση παίρνει ως είσοδο μια κανονική συνάρτηση και παράγει τον κατάλληλο Γράφο για την συνάρτηση. Στην ιστοσελίδα του Tensorflow γίνεται σύγκριση του ίδιου κώδικα με τον Grappler ενεργοποιημένο και απενεργοποιημένο[16].

Στην πρώτη εκτέλεση όπου ο βελτιστοποιητής Grappler είναι απενεργοποιημένος η εκτέλεση του οποίου διήρκησε 0.0010882619999392773 δευτερόλεπτα ενώ με τον βελτιστοποιητή ενεργοποιημένο η εκτέλεση διήρκησε 0.0007714039998063527 δευτερόλεπτα που είναι περίπου 1,4 φορές γρηγορότερη εκτέλεση από την προηγούμενη.

Ενότητα 4.1.4 Η βιβλιοθήκη tf.data του Tensorflow

Για την διαχείριση των δεδομένων το υπολογιστικό πλαίσιο προσφέρει την βιβλιοθήκη tf.data .Με την βιβλιοθήκη το Tensorflow δίνει την δυνατότητα [17] στους προγραμματιστές να δημιουργήσουν με ευκολία περίπλοκες ροές εισαγωγής δεδομένων. Επίσης μπορεί να χειριστή μεγάλο όγκο δεδομένων διαφορετικού είδους καθώς και να εκτελέσει πάνω τους διάφορους σύνθετους μετασχηματισμούς. Για παράδειγμα η ροή εισόδου για ένα μοντέλο που διαχειρίζεται εικόνες μπορεί να χρησιμοποιεί δεδομένα που βρίσκονται σε κάποιο κατανεμημένο σύστημα αρχείων (πχ: Hadoop File system) , έπειτα να εφαρμόζει πάνω τους κάποια επεξεργασία και τέλος να χωρίζει τα δεδομένα σε παρτίδες (batching). Η βιβλιοθήκη προσφέρει με ευκολία στον προγραμματιστή μέσω του Dataset να διαβάσει δεδομένα , να τα επεξεργαστεί κλπ. Τα Dataset [11] αναπαριστούν μια ακολουθία δεδομένων και υποστηρίζει την συγγραφή αποδοτικών ροών εισόδου. Η χρήση των Dataset συνήθως ακολουθούν την εξής διαδικασία [11]:

- Δημιουργία ενός Dataset πηγή (source Dataset) από τα δεδομένα εισαγωγής (input Dataset)
- Προεπεξεργασία των δεδομένων μέσω μετασχηματισμών του Dataset
- Ανάγνωση (Iterate) του Dataset και επεξεργασία των στοιχείων

Για να διαχειριστή τον μεγάλο όγκου δεδομένων τα Dataset δεν φορτώνουν όλα τα στοιχεία στην μνήμη άλλα η ανάγνωση γίνεται διαδοχικά τύπου ζωντανής ροής (Streaming). Τα Dataset προσφέρουν στον προγραμματιστή την δυνατότητα να εφαρμόσει με ευκολία διάφορες εργασίες πάνω στα δεδομένα μέσω της apply(). Προσφέρει χρήσιμες δυνατότητες όπως το shuffle(n) όπου δημιουργώντας στην μνήμη μια περιοχή προσωρινής αποθήκευσης (buffer) μεγέθους n βάζει δεδομένα από το Dataset έτσι ώστε να τα αναμίξει. Ακόμη προσφέρει την δυνατότητα δημιουργίας παρτίδων από τα δεδομένα με την batch(n) όπου χωρίζει τα δεδομένα σε παρτίδες που η καθεμία περιέχει n αριθμό στοιχείων, δίνει επίσης με την repeat(n) την επανάληψη του dataset κατά n φορές (αυτό είναι πολύ χρήσιμο για την εκπαίδευση μοντέλων με κατανεμημένο τρόπο).

Ενότητα 4.1.5 Ο τύπος αρχείων TFRecord

Ο Τύπος αρχείων TFRecord είναι [18] μια απλή μορφή για την αποθήκευση μιας αλληλουχίας δυαδικών εγγραφών. Αυτό γίνεται με την χρήση των Protocol Buffers [19] που είναι μια βιβλιοθήκη για πολλές πλατφόρμες (cros-platfrom) και πολλές γλώσσες (cross-language) για αποτελεσματική σειριοποίηση δομημένων δεδομένων. Τα μηνύματα πρωτοκόλλων δηλώνονται σε αρχεία .proto και έχουν την μορφή «κλειδί» : τιμή. Η βιβλιοθήκη προσφέρει υποστηρίξει για πολλών τύπων τιμών.

Ενότητα 4.1.6 Οι στρατηγικές κατανεμημένης εκπαίδευσης του Tensorflow

Υπάρχουν δυο τρόποι εκπαιδεύσεις ενός μοντέλου μηχανικής μάθησης σε κατανεμημένα συστήματα ο ένας είναι παράλληλου μοντέλου (model parallel) και ο άλλος παράλληλων δεδομένων (data parallel). Η Τεχνική κατανεμημένης εκπαίδευσης παράλληλου μοντέλου χωρίζει ένα μοντέλο σε περισσότερα κομμάτια και τα μοιράζει στους κόμβους του κατανεμημένου συστήματος έτσι ώστε κάθε κόμβος να εκπαιδεύει ένα κομμάτι του μοντέλου. Η άλλη επιλογή των παράλληλων δεδομένων μοιράζει στους κόμβους κομμάτια από τα δεδομένα και όλοι οι κόμβοι του κατανεμημένου συστήματος εκπαιδεύουν μια αντιγραφή του ίδιου μοντέλου με τα δεδομένα που τους έχουν ανατεθεί. Αυτήν την στιγμή που γράφεται αυτή η εργασία το υπολογιστικό πλαίσιο Tensorflow υποστηρίζει μόνο την επιλογή παράλληλων δεδομένων και προσφέρει αρκετές στρατηγικές για κατανεμημένη εκπαίδευση [20]. Η επιλογή των παράλληλων δεδομένων περιλαμβάνει δυο τρόπους εκπαίδευσης της ασύγχρονης και της σύγχρονης εκπαίδευσης. Κατά την σύγχρονη εκπαίδευση όλοι οι κόμβοι εργάτες (workers) της συστοιχίας εκπαιδεύουν το μοντέλο πάνω σε διαφορετικά κομμάτια δεδομένων και στο τέλος του βήματος εκπαίδευσης συγκεντρώνουν τα αποτελέσματα τους. Στην ασύγχρονη εκπαίδευση όλοι οι κόμβοι εργάτες εκπαιδεύουν ανεξάρτητα ο κάθε ένας πάνω στα δεδομένα και ενημερώνουν τις τιμές ασύγχρονα χωρίς να περιμένουν ο ένας τον άλλο.

Οι στρατηγικές που υποστηρίζει το Tensorflow είναι [20]:

- MirroredStrategy: σύγχρονη κατανεμημένη εκπαίδευση σε πολλές Μονάδες Επεξεργασίας Γραφικών σε ένα μηχάνημα. Δημιουργεί ένα αντίγραφο για κάθε μια από τις Μονάδες Επεξεργασίας Γραφικών. Κάθε μια μεταβλητή είναι ίδια για κάθε ένα αντίγραφο καθώς εφαρμόζεται μια τεχνική που ονομάζεται «καθρεπτισμός» και έτσι όλες οι μεταβλητές από όλα τα αντίγραφα μαζί δημιουργούν μια καθρεπτισμένη μεταβλητή (Mirrored Variable). Αυτές οι μεταβλητές παραμένουν ενημερωμένες μεταξύ τους σε κάθε γύρω εφαρμόζοντας πανομοιότυπες ενημερώσεις. Για να επικοινωνήσουν μεταξύ τους τα αντίγραφα και να ενημερώσουν τις τιμές του αποδοτικοί αλγόριθμοι επικοινωνίας χρησιμοποιούνται από το υπολογιστικό πλαίσιο έτσι ώστε να μειώσει όσο γίνεται

τον χρόνο που παίρνει η επικοινωνία. Η Προεπιλογή είναι ο αλγόριθμος της NVIDIA Collective Communication (NCCL) αλλά δίνεται η επιλογή και άλλων αλγορίθμων, ακόμα επιτρέπει στον προγραμματιστή να γράψει τον δικό του αλγόριθμο επικοινωνίας.

- TPUStrategy: Το Tensorflow επιτρέπει την κατανεμημένη εκπαίδευση πάνω σε ειδικά κατασκευασμένο υλικό ASIC που είναι φτιαγμένο για να επιταχύνει την διαδικασία της μηχανικής μάθησης που ονομάζονται TPUs. Η στρατηγική TPUStrategy εφαρμόζει την ίδια λογική με την στρατηγική MirroredStrategy εφαρμόζοντας σύγχρονη εκπαίδευση μεταξύ πολλών πυρήνων TPU

- MultiWorkerMirroredStrategy: Βασισμένη πάνω στην στρατηγική MirroredStrategy η στρατηγική MulitWorkerMirroredStrategy εφαρμόζει σύγχρονη εκπαίδευση σε κόμβους μιας συστοιχίας όπου ο κάθε ένας πιθανά διαθέτει πολλές Μονάδες Επεξεργασίας Γραφικών. Όμοια με την στρατηγική MirroredStrategy δημιουργεί αντίγραφα και «καθρεπτισμένες» μεταβλητές για κάθε εργάτη.

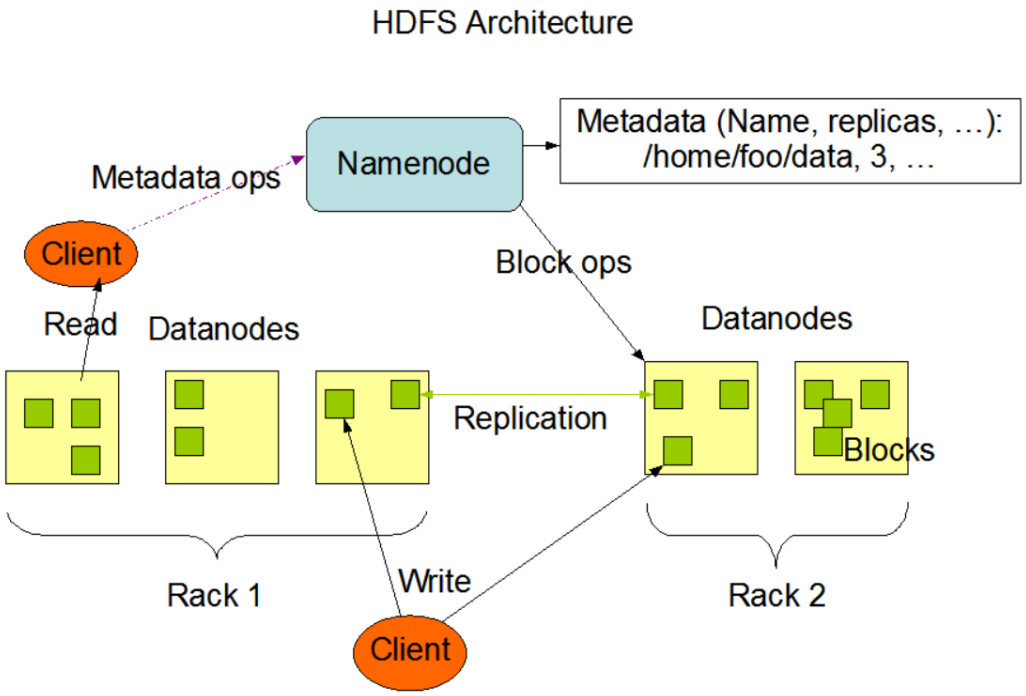
- ParameterServerStrategy: Μια πολύ κοινή στρατηγική για την εκπαίδευση μοντέλων μηχανικής μάθησης χρησιμοποιώντας την επιλογή παράλληλων δεδομένων είναι η χρήση της στρατηγικής εξυπηρετητών παραμέτρων (Parameter Server). Κατά την στρατηγική εξυπηρετητών παραμέτρων μέσα σε μια συστοιχία υπολογιστών υπάρχουν κόμβοι οι οποίοι έχουν τον ρόλο των εργατών (Workers) και κάποιοι κόμβοι που έχουν τον ρόλο των εξυπηρετητών παραμέτρων (Parameter Servers). Οι μεταβλητές δημιουργούνται στους κόμβους που έχουν τον ρόλο των εξυπηρετητών παραμέτρων όπου διαβάζονται από τους εργάτες και ανανεώνονται ασύγχρονα σε κάθε βήμα της εκπαίδευσης. Η υλοποίηση της στρατηγικής εξυπηρετητών παραμέτρων ParameterServerStrategy του Tensorflow [21] προσθέτει ακόμα έναν ρόλο στην στρατηγική των εξυπηρετητών παραμέτρων τον ρόλο του αρχηγού-οργανωτή (Chief). Ο Αρχηγός δημιουργεί πόρους, προγραμματίζει διαδικασίες εκπαίδευσης, διαχειρίζεται τις αποτυχίες των εργασιών (task failures) και αποθηκεύει την κατάσταση του μοντέλου σε μια συγκεκριμένη στιγμή (checkpoints).

- CentralStorageStrategy: Η στρατηγική CentralStoragetStrategy του Tensorflow εφαρμόζει σύγχρονη εκπαίδευση. Οι μεταβλητές δεν είναι «καθρεπτισμένες» όπως είδαμε στις παραπάνω στρατηγικές αλλά τοποθετούνται στην Κεντρική Μονάδα Επεξεργασίας και οι εργασίες αντιγράφονται σε κάθε μια από τις τοπικές Μονάδες Επεξεργασίας Γραφικών. Αν υπάρχει μόνο μια όλες οι μεταβλητές και οι εργασίες θα αντιγραφούν σε αυτήν.

Το κατανεμημένο σύστημα HDFS 4.2

Ενότητα 4.2.1 Το κατανεμημένο σύστημα αρχείων HDFS

Το κατανεμημένο σύστημα αρχείων HDFS διατίθεται από το υπολογιστικό πλαίσιο Apache Hadoop. Το Apache Hadoop είναι ένα υπολογιστικό πλαίσιο εκτέλεσης εργασιών κατανεμημένα σε μια συστοιχία υπολογιστών. Όσο αναφορά το κατανεμημένο σύστημα αρχείων HDFS αποτελεί μια συχνή λύση για την φιλοξενία δεδομένων σε συστήματα εκτέλεσης εργασιών στο νέφος . Το κατανεμημένο σύστημα HDFS αποτελείται από δυο τύπους κόμβων ο ένας είναι ο NameNode και ο δεύτερος τύπος είναι οι DataNodes καθώς ακολουθεί μια αρχιτεκτονική Αρχηγού/Εργάτη (Master/Worker architecture) [22]. Ο NameNode είναι ο αρχηγός του συστήματος και οι DataNodes είναι οι εργάτες. Ο NameNode γνωρίζει που βρίσκεται κάθε αρχείο , διαχειρίζεται το γενικό σύστημα αρχείων και ελέγχει σε ποιο μηχάνημα θα αποθηκευτεί κάθε αρχείο . Οι DataNodes διαθέτουν κομμάτια των αρχείων και είναι υπεύθυνοί να εκτελούν εντολές διαχείρισης του τοπικού τους συστήματος αρχείων πάντα με εντολή του NameNode. Ένα αρχείο όταν εισάγεται στο HDFS αρχικά «τεμαχίζεται» σε ένα ή περισσότερα κομμάτια και έπειτα αποφασίζεται από τον NameNode σε ποιόν DataNode θα αποθηκευτεί. Για κάθε κομμάτι του αρχείου δημιουργούνται αντίγραφά και μοιράζονται σε περισσότερους από έναν DataNode έτσι ώστε μια πιθανή αποτυχία ενός DataNode να μην σημαίνει ολική αποτυχία του συστήματος αρχείων. Στην Εικόνα 7 μπορούμε να δούμε μια εικονική αναπαράσταση του συστήματος αρχείων HDFS



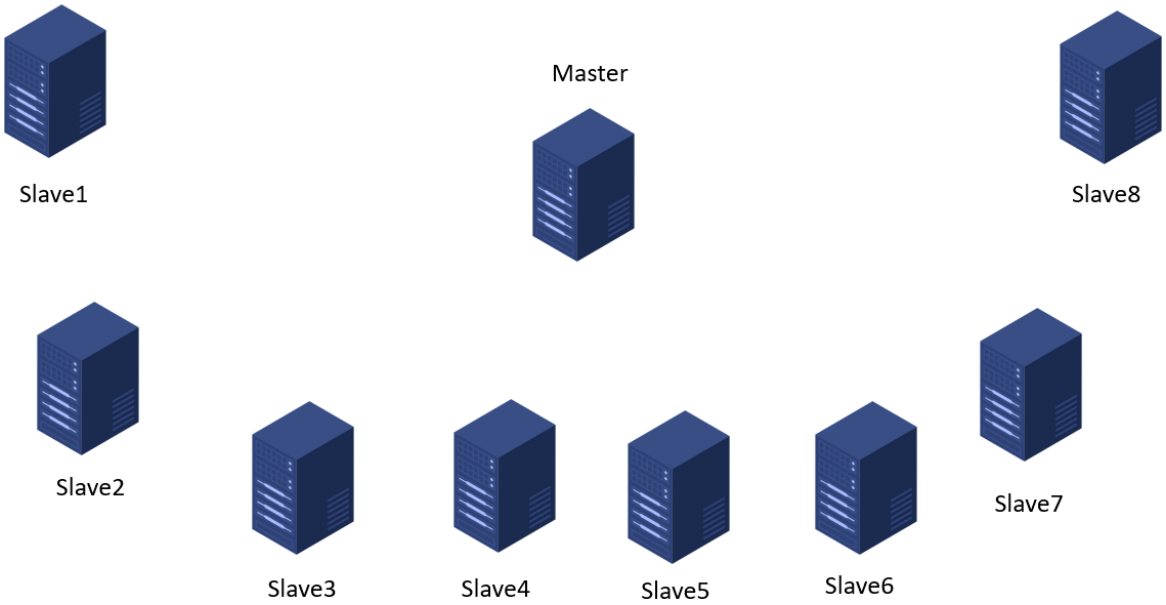
Εικόνα 7 Η αρχιτεκτονική του HDFS εικόνα απο https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

ΚΕΦΑΛΑΙΟ 5 Πειραματική αποτίμηση:

Σε αυτό το κεφάλαιο αρχικά γίνεται μια περιγραφή της συστοιχίας των υπολογιστών μας στο κεφάλαιο 5.1.1. Έπειτα στο επόμενο κεφάλαιο 5.1.2 μιλάμε για τα δεδομένα που χρησιμοποιήθηκαν σ την εκπαίδευση των μοντέλων και παρουσιάζουμε κάποια παραδείγματα από αυτά. Στην τελευταία ενότητα 5.1.3 μιλάμε για την πειραματική αποτίμηση των μοντέλων καθώς και την χρήση των πόρων για κάθε στρατηγική και για κάθε ένα μέγεθος παρτίδας

Ενότητα 5.1.1 Η συστοιχία υπολογιστών

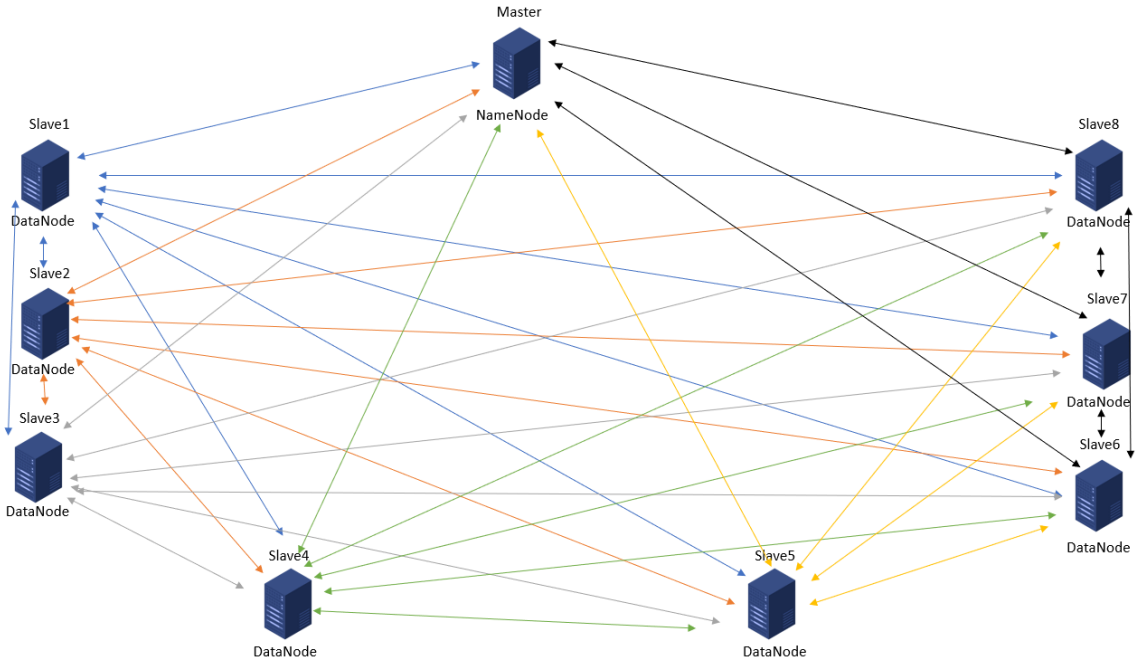
Η κατανεμημένη συστοιχία που χρησιμοποιήσαμε αποτελείται από 9 μηχανήματα που φιλοξενούνται στις υποδομές της υπηρεσίας Okeanos. Το υπολογιστικό νέφος Okeanos δημιουργήθηκε με σκοπό να προσφέρει υπολογιστικές υπηρεσίες και υπηρεσίες αποθήκευσης για τους Έλληνες ερευνητές. Είναι στηριγμένο πάνω σε διάφορα έργα ανοιχτού κώδικα (Linux, Google Zaneti, Python/Django Κλπ.) με τα οποία μπορεί και προσφέρει μια γρήγορη , εύκολη και ασφαλή πρόσβαση σε εικονικούς πόρους. Κάθε ερευνητής αποκτά πρόσβαση σε πόρους όπως CPU, Ram , εικονικούς δίσκους και εικονικές μηχανές που μπορεί να διαχειριστεί μέσω μιας διαδικτυακής διεπαφής χρήστη [23]. Κάθε μηχάνημα διαθέτουν Κεντρικές Μονάδες Επεξεργασίας με 4 πυρήνες (Οι μηχανές μας δεν διαθέτουν Μονάδες Επεξεργασίας Γραφικών). Τα μηχανήματα επίσης διαθέτουν 4 GB μνήμη και δυο σκληρούς δίσκους 30 GB ο κάθε ένας. Το λειτουργικό σύστημα που χρησιμοποιούν οι μηχανές είναι Ubuntu Linux 16.04 LTS. Το ένα μηχάνημα έχει ονομαστεί Master γιατί κατά την εκτέλεση εργασιών με τα υπολογιστικά πλαίσια του αναθέτουμε τον ρόλο του αρχηγού και του δίνονται περισσότερες αρμοδιότητες από τα άλλα μηχανήματα. Τα άλλα μηχανήματα έχουν όνομα Slave 1-8 και κατά την εκτέλεση εργασιών έχουν ρόλο εργάτη (κατά την εκτέλεση εργασιών με την στρατηγική των Εξυπηρετητών Παραμέτρων κάποια μηχανήματα επιλέγονται για τον ρόλο του Εξυπηρετητή Παραμέτρων και κάποια του εργάτη). Στην Εικόνα 8 μπορούμε να δούμε μια αναπαράστασή της συστοιχίας μας.



Εικόνα 8 Αναπαράσταση της συστοιχίας υπολογιστών μας

Ενότητα 5.1.2 Τα δεδομένα-Datasets

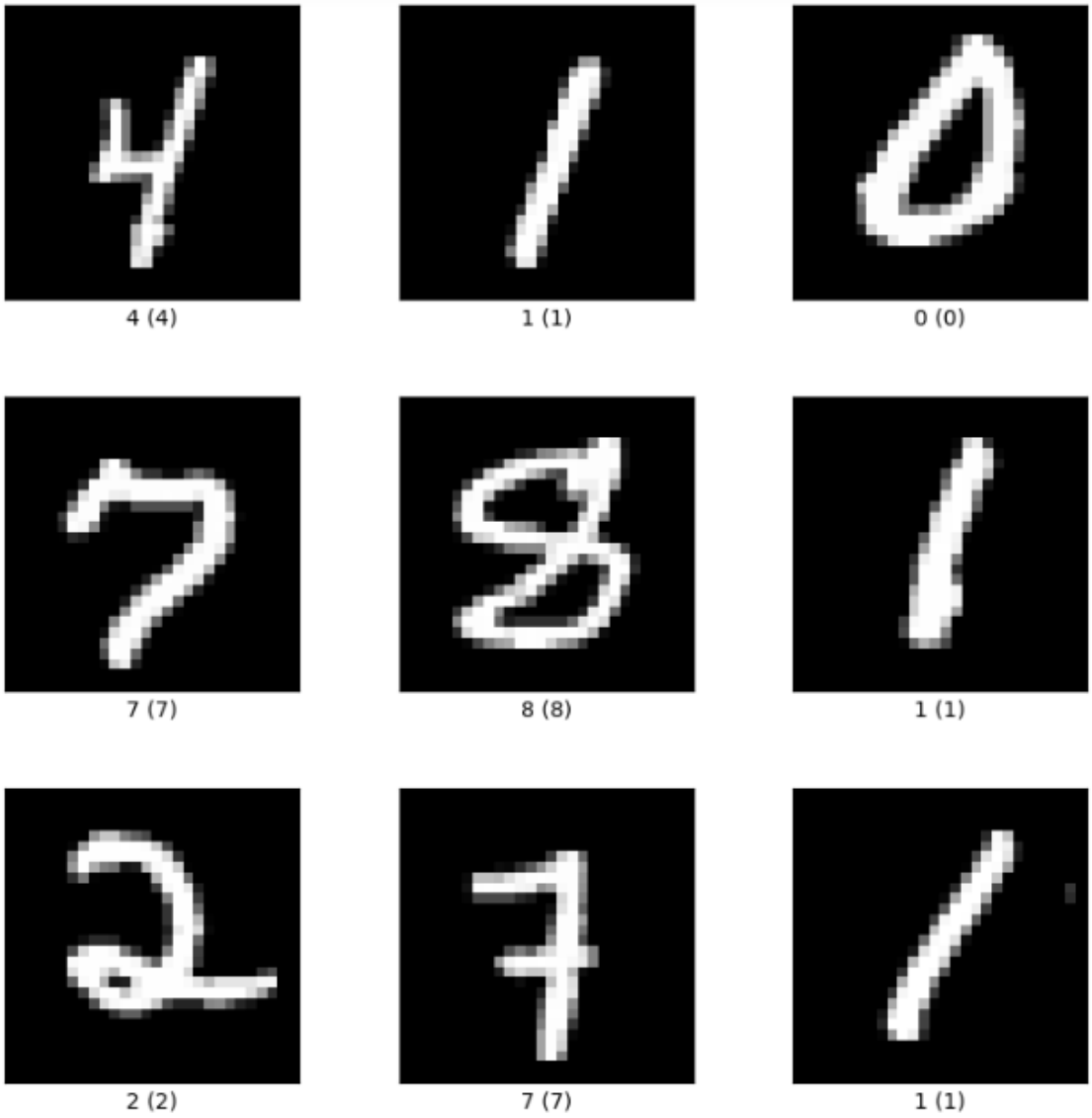
Για την δοκιμή της κατανεμημένης εκπαίδευσης των μοντέλων μηχανικής μάθησης στην συστοιχία υπολογιστών μας εκτελέστηκαν 3 μοντέλα μηχανικής μάθησης. Για το πρώτο πείραμα δημιουργήθηκε ένα μοντέλο που εκπαιδεύτηκε με τα δεδομένα από το Mnist Dataset. Για να κατεβάσουμε τα δεδομένα του Mnist Dataset στην μορφή TFRecord χρησιμοποιήθηκε το εργαλείο του Tensorflow Tensorflow-Datasets [24].Το Tensorflow-Datasets είναι μια βιβλιοθήκη που μας επιτρέπει να κατεβάσουμε τα δεδομένα την ώρα της εκτέλεσης του κώδικα όταν εκτελούμε την εκπαίδευση των μοντέλων μηχανικής μάθησης σε ένα μηχάνημα. Επειδή όμως εμείς θέλουμε να εκπαιδεύσουμε τα μοντέλα με κατανεμημένο τρόπο δεν μας αρκεί αυτή η προσέγγιση έτσι εκμεταλλευόμαστε την δυνατότητα της βιβλιοθήκης Tensorflow-Datasets να εκτελεστεί ως εργαλείο στην γραμμή εντολών (CLI) [25]. Με την χρήση αυτού του εργαλείου κατεβάζουμε τα δεδομένα στην μορφή TFRecord και έπειτα τα ανεβάζουμε τα δεδομένα στο κατανεμημένο σύστημα αρχείων HDFS έτσι ώστε όλα τα μηχανήματα στην συστοιχία να έχουν πρόσβαση στα δεδομένα. Μια γραφική αναπαράσταση του κατανεμημένου συστήματος αρχείων της συστοιχίας μας μπορούμε να δούμε στην Εικόνα 9



Εικόνα 9 Αναπαράσταση του κατανεμημένου συστήματος αρχείου HDFS της συστοιχίας υπολογιστών μας

Όπως βλέπουμε και στην Εικόνα 9 το μηχάνημα με το όνομα Master έχει υποχρεώσεις αρχηγού και έτσι είναι το μόνο μηχάνημα με τον ρόλο NameNode όπου διαχειρίζεται το συνολικό σύστημα αρχείων. Τα άλλα μηχανήματα με όνομα Slave δηλαδή από Slave 1-8 έχουν υποχρεώσεις εργάτη και έτσι έχουν όλοι τον ρόλο του DataNode όπου εκτελούν εργασίες που τους αναθέτει ο NameNode και διαχειρίζονται το τοπικό τους σύστημα αρχείων (τα βέλη που απεικονίζουν την δυνατότητα επικοινωνίας μεταξύ των μηχανών έχουν διαφορετικό χρώμα για να είναι πιο εύκολα να διακρίνουμε τις σχέσεις μεταξύ των μηχανών).

Το Dataset Mnist [26] διαθέτει 70000 εγγραφές χωρισμένες σε 60000 εγγραφές για εκπαίδευση και 10000 εγγραφές για επαλήθευση. Οι εγγραφές του Dataset Mnist διαθέτει εικόνες με χειρόγραφους αριθμούς και επιγραφές (labels) με τον αριθμό που αναπαριστούν. Το Mnist dataset περιέχει δέκα κατηγορίες μια για κάθε αριθμό από τον αριθμό 0 έως τον αριθμό 9 .Παράδειγμα των εγγραφών μπορούμε να δούμε στην Εικόνα 10

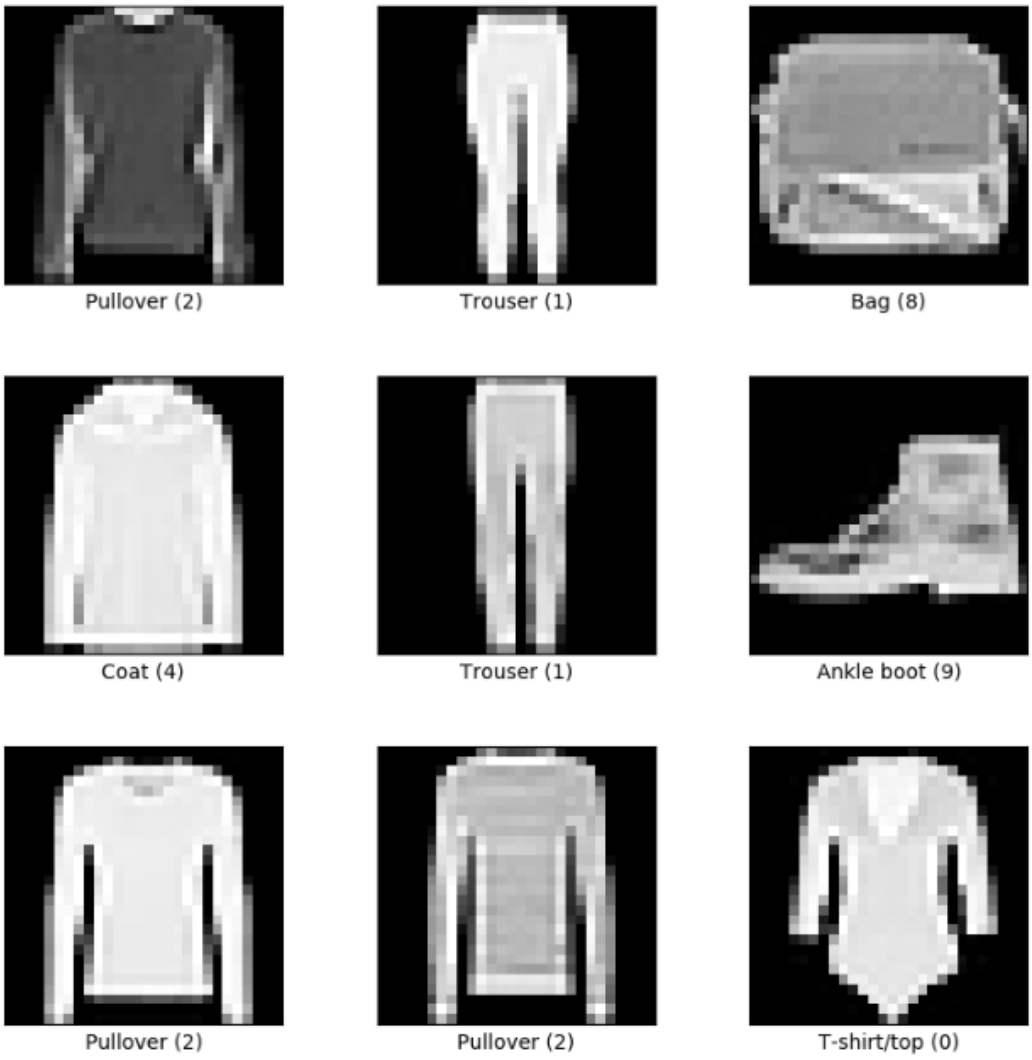


Εικόνα 10 Παράδειγμα των δεδομένων του Dataset Mnist εικόνα απο https://www.tensorflow.org/datasets/catalog/mnist

Επιπλέον από τις εγγραφές εκπαίδευσης δημιουργούμε ακόμη ένα σύνολο εγγραφών 6000 για να ελέγχουμε την ποιότητα του μοντέλου σε κάθε εποχή. Τελικά στην εκπαίδευση έχουμε 54000 εγγραφές για εκπαίδευση , 6000 για έλεγχο στο τέλος για κάθε εποχής και 10000 εγγραφές όπου στο τέλος της εκπαίδευσης ελέγχουμε την ακρίβεια του μοντέλου σε εγγραφές που δεν έχει «δει» το μοντέλο κατά την διάρκεια της εκπαίδευσης.

Για το επόμενο μοντέλο μηχανικής μάθησης χρησιμοποιήθηκε το Dataset Fashion Mnist [27] όπου παρέχονται 70000 εγγραφές. Από αυτές τις εγγραφές οι 60000 παρέχονται για εκπαίδευση ενώ οι άλλες 10000 για επαλήθευση. Οι εγγραφές του Dataset Fashion Mnist περιέχουν φωτογραφίες από διαφορετικού τύπου ρούχων και επιγραφές με την

κατηγορία στην οποία ανήκει η εγγραφή. Το Dataset Fashion Mnist διαθέτει δέκα κατηγορίες: T-shirt/Top , παντελόνι, πουλόβερ, φόρεμα, παλτό, σανδάλια, μπλούζα, αθλητικά παπούτσια, τσάντα, μπότες. Μια απεικόνιση των δεδομένων μπορούμε να δούμε στην Εικόνα 11

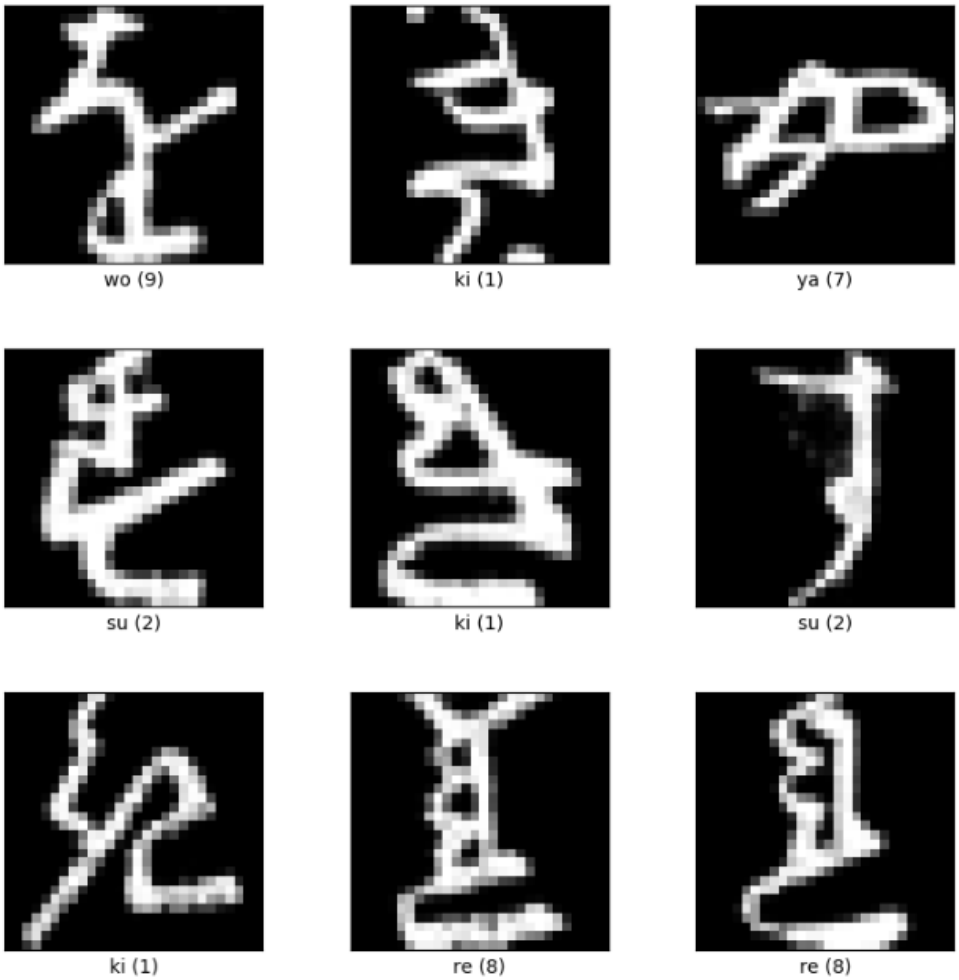


Εικόνα 11 Παράδειγμα δεδομένων του Dataset Fashion Mnist εικόνα απο https://www.tensorflow.org/datasets/catalog/fashion_mnist

Επιπλέον από τις εγγραφές εκπαίδευσης δημιουργούμε ακόμη ένα σύνολο εγγραφών 6000 για να ελέγχουμε την ποιότητα του μοντέλου σε κάθε εποχή. Τελικά στην εκπαίδευση έχουμε 54000 εγγραφές για εκπαίδευση , 6000 για έλεγχο στο τέλος για κάθε εποχής και 10000 εγγραφές όπου στο τέλος της εκπαίδευσης ελέγχουμε την ακρίβεια του μοντέλου σε εγγραφές που δεν έχει «δει» το μοντέλο κατά την διάρκεια της εκπαίδευσης.

Στο τελευταίο μοντέλο μηχανικής μάθησης για την εκπαίδευση του χρησιμοποιούμε το Dataset Kuzushiji-MNIST [28] όπου είναι ένα πιο καινούργιο dataset που μοιάζει με το κλασσικό Dataset Mnist. Το Kuzushiji-MNIST διαθέτει 70000 εγγραφές. Από αυτές τις

εγγραφές οι 60000 παρέχονται για εκπαίδευση ενώ οι άλλες 10000 για επαλήθευση. Μέσα στις εγγραφές του Dataset Kuzushiji-MNIST εμπεριέχονται χειρόγραφες εικόνες από χαρακτήρες και επιγραφές με τον αριθμό της κατηγορίας της οποίας ανήκει η εικόνα . Οι εγγραφές του Dataset Kuzushiji-MNIST περιέχουν 10 κατηγορίες διαφορετικών χαρακτήρων. Μια απεικόνιση των δεδομένων μπορούμε να δούμε στην Εικόνα 12



Εικόνα 12 παράδειγμα δεδομένων απο το Dataset Kmnist εικόνα απο https://www.tensorflow.org/datasets/catalog/kmnist)

Επιπλέον από τις εγγραφές εκπαίδευσης δημιουργούμε ακόμη ένα σύνολο εγγραφών 6000 για να ελέγχουμε την ποιότητα του μοντέλου σε κάθε εποχή. Τελικά στην εκπαίδευση έχουμε 54000 εγγραφές για εκπαίδευση , 6000 για έλεγχο στο τέλος για κάθε εποχής και 10000 εγγραφές όπου στο τέλος της εκπαίδευσης ελέγχουμε την ακρίβεια του μοντέλου σε εγγραφές που δεν έχει «δει» το μοντέλο κατά την διάρκεια της εκπαίδευσης.

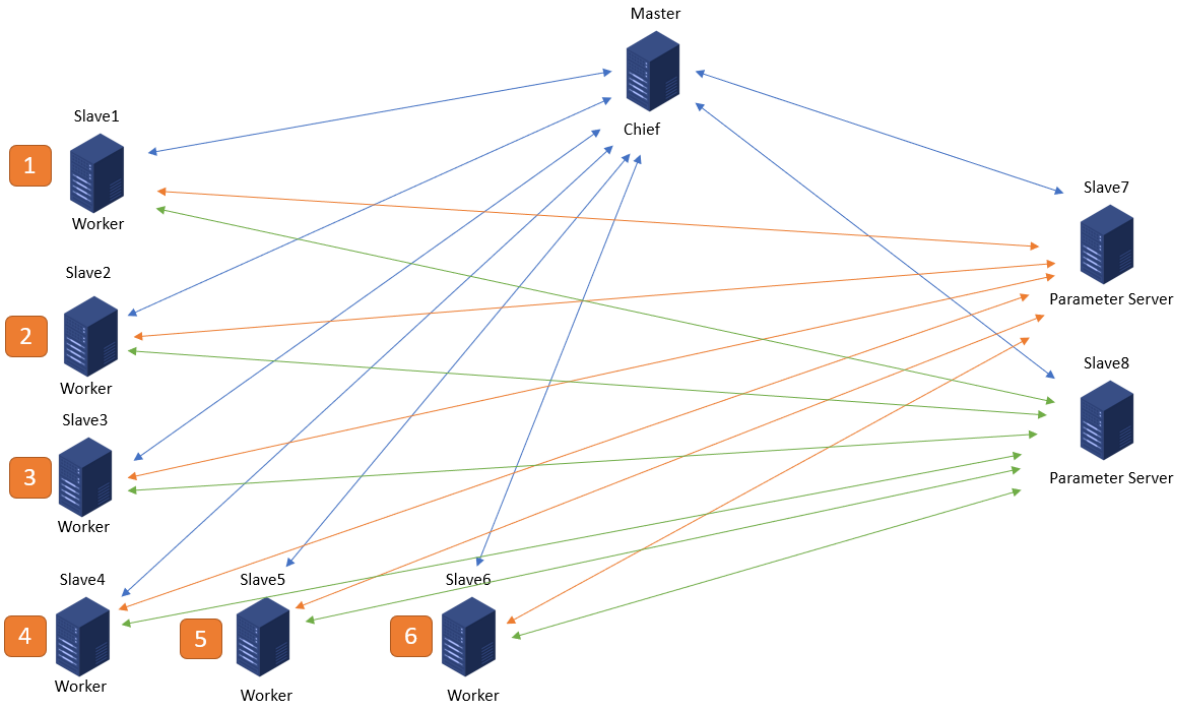
Ενότητα 5.1.3 Στρατηγικές εκτέλεσης κατανεμημένης μηχανικής μάθησης

Για να εξετάσουμε την κατανεμημένη εκπαίδευση των μοντέλων διαλέξαμε δυο διαφορετικές στρατηγικές. Με την χρήση του υπολογιστικού πλαίσιού Tensorflow οι στρατηγικές που επιλέχθηκαν είναι η στρατηγική των εξυπηρετητών παραμέτρων (Parameter Server Strategy) και MultiWorkerMirroredStrategy.

Για την στρατηγική των εξυπηρετητών παραμέτρων από το υπολογιστικό πλαίσιο Tensorflow διατίθενται 3 ρόλοι: ο ρόλος του αρχηγού-οργανωτή (chief) , ο ρόλος του εργάτη (worker) και ο ρόλος του εξυπηρετητή παραμέτρων (ps – parameter server). Η συστοιχία υπολογιστών στην οποία εκπαιδεύσαμε τα μοντέλα περιλαμβάνει 9 μηχανές και οι ρόλοι είναι ως εξής:

- Αρχηγός-Οργανωτής (chief) : Το μηχάνημα με όνομα Master
- Εργάτης (Worker): Τα μηχανήματα με ονόματα Slave1, Slave2, Slave3, Slave4, Slave5, Slave6
- Εξυπηρετητής Παραμέτρων (Parameter Server): Τα μηχανήματα με ονόματα Slave7, Slave8

Η δουλειά του Master (με τον ρόλο του Αρχηγού-οργανωτή) είναι να δημιουργεί πόρους, να αποθηκεύει την κατάσταση του μοντέλου, να χειρίζεται αποτυχίες εργασιών και να αναθέτει εργασίες. Τα μηχανήματα που έχουν τον ρόλο εργάτης και εξυπηρετητής Παραμέτρων περιμένουν μέχρι να πάρουν κάποια εντολή από τον Αρχηγό-Οργανωτή. Τα μηχανήματα που έχουν τον ρόλο του εργάτη δηλαδή ο Slave1, Slave2, Slave3, Slave4, Slave5 και Slave6 λαμβάνουν εργασίες από τον Master και διαβάζουν τις μεταβλητές από τα μηχανήματα που έχουν τον ρόλο του Εξυπηρετητή Παραμέτρων χωρίς να έχουν άμεση επικοινωνία με τα άλλα μηχανήματα με τον ρόλο εργάτη στην συστοιχία υπολογιστών. Τα μηχανήματα με τον ρόλο του Εξυπηρετητή Παραμέτρων δηλαδή ο Slave7 και Slave8 έχουν τον ρόλο να ενημερώνουν τις μεταβλητές και να μοιράζουν τις μεταβλητές στα μηχανήματα εργάτες που τις ζητάνε. Για τα μηχανήματα που έχουν τον ρόλο του Εξυπηρετητή Παραμέτρων το υπολογιστικό πλαίσιο Tensorflow χρειάζεται να ορίσουμε τον τρόπο με τον οποίο θα χωρίζονται οι μεταβλητές μεταξύ των Εξυπηρετητών Παραμέτρων εμείς επιλέξαμε μια κλασσική λύση με ελάχιστο όριο [29] (minimum size partitioner), όπου δηλώνουμε τον κατώτατο αριθμό bytes για κάθε κομμάτι (shard) της μεταβλητής και τον μέγιστο αριθμό κομματιών. Μια αναπαράσταση της συστοιχίας υπολογιστών μας για την στρατηγικής Εξυπηρετητών Παραμέτρων μπορούμε να δούμε στην Εικόνα 13



Εικόνα 13 Αναπαράσταση της στρατηγικής Εξυπηρετητών Παραμέτρων στην συστοιχία των υπολογιστών μας

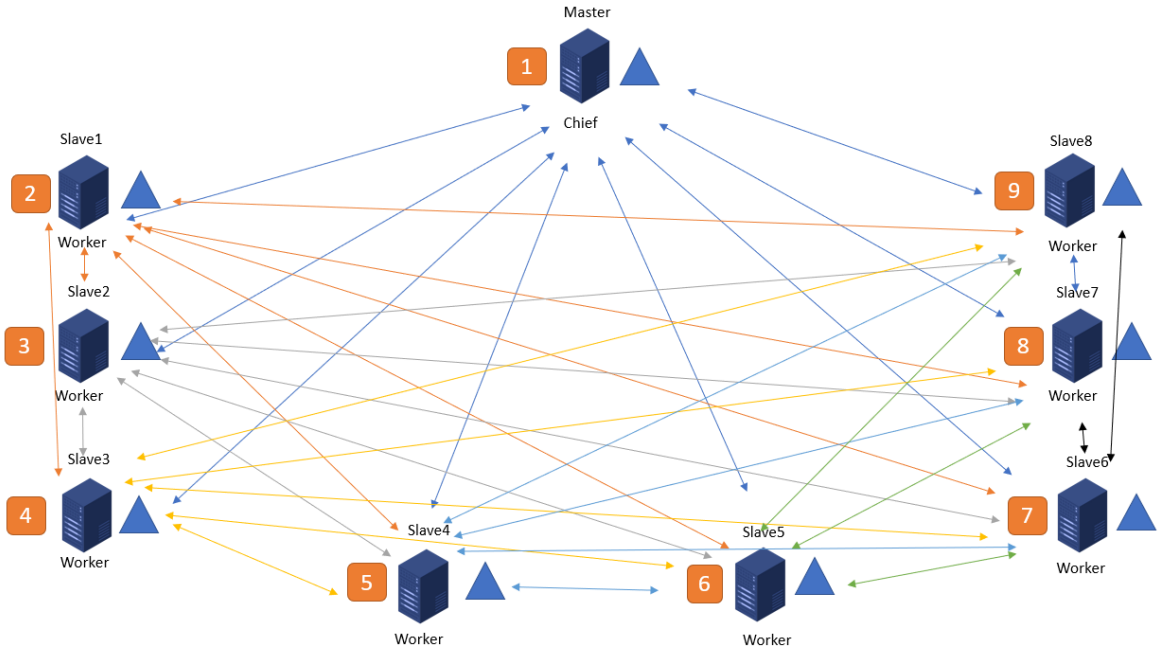
Όπως βλέπουμε στην Εικόνα 13 το μηχάνημα Master επικοινωνεί με όλα τα άλλα μηχανήματα ανεξαρτήτως ρόλου. Τα μηχανήματα εργάτες επικοινωνούν με τον οργανωτή-αρχηγό και με τα μηχανήματα με τον ρόλο Εξυπηρετητής Παραμέτρων. Ακόμη στην εικόνα μπορούμε να διακρίνουμε δίπλα στα μηχανήματα με τον ρόλο του εργάτη ότι υπάρχουν κάποια πορτοκαλί τετράγωνα, αυτά αναπαριστούν τα κομμάτια (shards) των δεδομένων που αντιστοιχούν σε κάθε μηχάνημα. Τα μηχανήματα διαβάζουν τα δεδομένα μετά από εντολή του Αρχηγού από το κατανεμημένο σύστημα HDFS της συστοιχίας των υπολογιστών μας. Οι προκαθορισμένες ρυθμίσεις της στρατηγικής Εξυπηρετητών Παραμέτρων δεν κατακερματίζει (sharding) τα δεδομένα και έτσι όλα τα μηχανήματα διαβάζουν όλα τα δεδομένα και εκπαιδεύουν τα τοπικά τους μοντέλα πάνω στα ίδια δεδομένα. Εμείς επιλέγουμε να κατακερματίσουμε τα δεδομένα με τις υλοποιημένες μεθόδους του Tensorflow. Το Tensorflow διαθέτει 2 επιλογές για κατακερματισμό των δεδομένων την επιλογή DATA και την επιλογή FILE

[30]. Η επιλογή FILE προϋποθέτει ότι τα δεδομένα είναι χωρισμένα σε αρχεία πλήθους τουλάχιστον ίσο με το πλήθος των μηχανημάτων που έχουν τον ρόλο του εργάτη έτσι ώστε κάθε μηχάνημα να αναλάβει ένα αρχείο (ή περισσότερα από ένα αρχείο). Η επιλογή DATA για κατακερματισμό των δεδομένων από ένα αρχείο και τα μοιράζει στους εργάτες αρχικά όλοι οι εργάτες διαθέτουν όλο το αρχείο αλλά χρησιμοποιούν μόνο τα δεδομένα που τους αναλογούν και απορρίπτουν τα άλλα δεδομένα. Εμείς λόγο ότι τα δεδομένα εκπαίδευσης βρίσκονται μόνο σε ένα αρχείο επιλέγουμε την επιλογή DATA.

Για την στρατηγική MultiWorkerMirroredStrategy το υπολογιστικό πλαίσιο Tensorflow διαθέτει 2 ρόλους για τα μηχανήματα της συστοιχίας, αυτός του Αρχηγού-εργάτη και αυτός του εργάτη. Ειδικότερα για την συστοιχία υπολογιστών μας οι ρόλοι των μηχανημάτων είναι οι ακόλουθοι:

- Αρχηγός - Εργάτης (Chief – Worker): Το μηχάνημα με όνομα Master
- Εργάτης (Worker): Τα μηχανήματα με όνομα Slave1, Slave2, Slave3, Slave4, Slave5, Slave6, Slave7, Slave8

Οι αρμοδιότητες του μηχανήματος με τον ρόλο του Αρχηγού – Εργάτη είναι ίδιες με αυτές του ρόλου Εργάτη αλλά με κάποιες επιπλέον αρμοδιότητες. Δηλαδή ο Αρχηγός-Εργάτης σε αντίθεση με την στρατηγική των Εξυπηρετητών Παραμέτρων συμμετέχει και αυτός στην εκπαίδευση του μοντέλου ενώ παράλληλα έχει και άλλες δουλειές όπως το να αποθηκεύσει την κατάσταση του μοντέλου , να κρατήσει κάποια αρχεία περίληψης κλπ. Αν για οποιοδήποτε λόγο δεν δηλώσουμε κάποιο μηχάνημα με τον ρόλο του Αρχηγού-Εργάτη το πρώτο μηχάνημα στην σειρά αναλαμβάνει αυτόν τον ρόλο .Τα μηχανήματα με το ρόλο του Εργάτη έχουν την υποχρεώσει να συμμετάσχουν στην διαδικασία της εκπαίδευσης του μοντέλου. Μια αναπαράσταση της συστοιχίας των υπολογιστών μας με την χρήση της στρατηγικής MultiWorkerMirroredStrategy μπορούμε να δούμε στην Εικόνα |X|



Εικόνα 14 Αναπαράσταση της στρατηγικής MultiWorkerMirroredStrategy στην συστοιχία υπολογιστών μας

Στην Εικόνα 14 παρατηρούμε όπως και στην αντίστοιχη αναπαράσταση της στρατηγικής των Εξυπηρετητών παραμέτρων τα πορτοκαλί αριθμημένα τετράγωνα δίπλα από κάθε μηχάνημα. Αυτά τα τετράγωνα αναπαριστούν τα κομμάτια των δεδομένων εκπαίδευσης που αναλογούν σε κάθε μηχάνημα. Η στρατηγική MultiWorkerMirroredStrategy χρησιμοποιεί από τις προκαθορισμένες ρυθμίσεις κατακερματισμού επιλέγοντας αυτόματα την στρατηγική που θα χρησιμοποιήσει. Στην περίπτωση μας η στρατηγική που θα επιλεχτεί είναι η DATA αφού ο αριθμός του πλήθος των αρχείων που περιέχονται τα δεδομένα εκπαίδευσης είναι 1. Ακόμη μπορούμε να παρατηρήσουμε το μπλε τρίγωνο δίπλα από κάθε μηχάνημα το οποίο αναπαριστά τις «καθρεπτιζόμενες» μεταβλητές (mirrored variables) . Αυτές οι μεταβλητές πρέπει να είναι ίδιες για όλα τα μηχανήματα και έτσι στο τέλος κάθε βήματος εκπαίδευσης τα μηχανήματα ανταλλάσσουν μεταξύ τους τους υπολογισμούς τους. Όλα τα μηχανήματα είναι συνδεδεμένα μεταξύ τους έτσι ώστε να μπορούν να επικοινωνούν.

Ενότητα 5.1.4 Τα αποτελέσματα των πειραμάτων

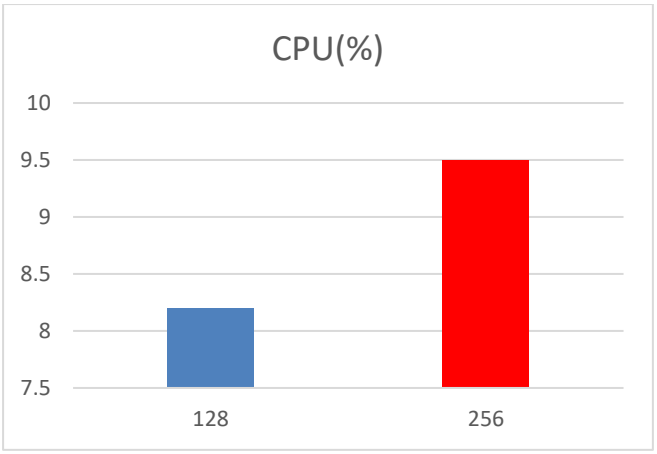
Ενότητα 5.1.4.α Πρώτο μοντέλο (Mnist)

Το μοντέλο που χρησιμοποιήθηκε για την πρώτη εκπαίδευση με το Dataset Mnist όπως είπαμε και στην ενότητα 5.1.2 είναι ένα διαδοχικό βαθύ νευρωνικό δίκτυο με στρώμα εισαγωγής των δεδομένων (Flatten layer) σχήματος (28,28) όσο δηλαδή το μέγεθος μιας εικόνας. Έπειτα ακολουθεί ένα «Πυκνό» στρώμα (Dense layer) με 300 νευρώνες και συνάρτηση ενεργοποίησης (activation function) την relu. Μετά ακολουθεί άλλο ένα «Πυκνό» στρώμα με 100 νευρώνες και συνάρτηση ενεργοποίησης την relu. Τέλος το μοντέλο περιλαμβάνει ένα «Πυκνό» στρώμα με 10 νευρώνες όσες δηλαδή και οι κατηγορίες των εικόνων και συνάρτηση ενεργοποίησης την softmax. Τον κώδικα για το μοντέλο μπορούμε να δούμε στην Εικόνα 15

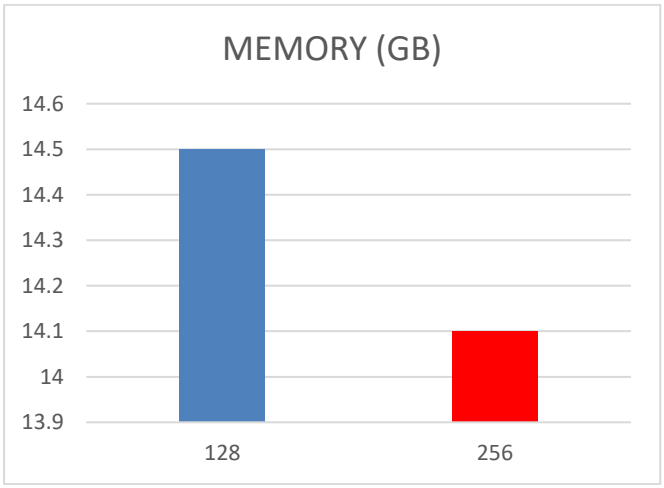
```
model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(300, activation='relu'),
        tf.keras.layers.Dense(100, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
])
```

Εικόνα 15 ο Κώδικας του πρώτου μοντέλου

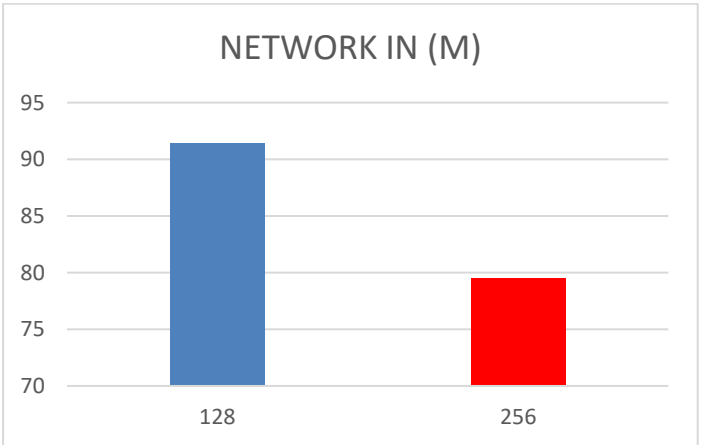
Για την στρατηγική των Εξυπηρετητών Παραμέτρων γίναν δυο εκτελέσεις με διαφορετικό αριθμό παρτίδας (batch size). Για το πρώτο πείραμα ο αριθμός παρτίδας ήταν 128 ενώ για την δεύτερη ήταν 256. Στις εικόνες 16, 17, και 18 μπορούμε να δούμε την μέση χρήση πόρων για αριθμό παρτίδας 128 και 256. Η μέση χρήση της Κεντρικής Μονάδας Επεξεργασίας για 128 ήταν 8,2% ενώ για 256 ήταν 9.5% , η μέση χρήση της μνήμης για αριθμό παρτίδας 128 ήταν 14.5 GB ενώ για 256 ήταν 14.1 GB, Τέλος για την χρήση του δικτύου η μέση τιμή 91.4 M για τον αριθμό παρτίδας 128 και 79.5 M για τον αριθμό παρτίδας 256. Η μέση χρήση των πόρων ήταν μεγαλύτερη για την εκπαίδευση με τον αριθμό παρτίδας 128 εκτός βέβαια στην μέση χρήση της Κεντρική Μονάδας Επεξεργασίας.



Εικόνα 16 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



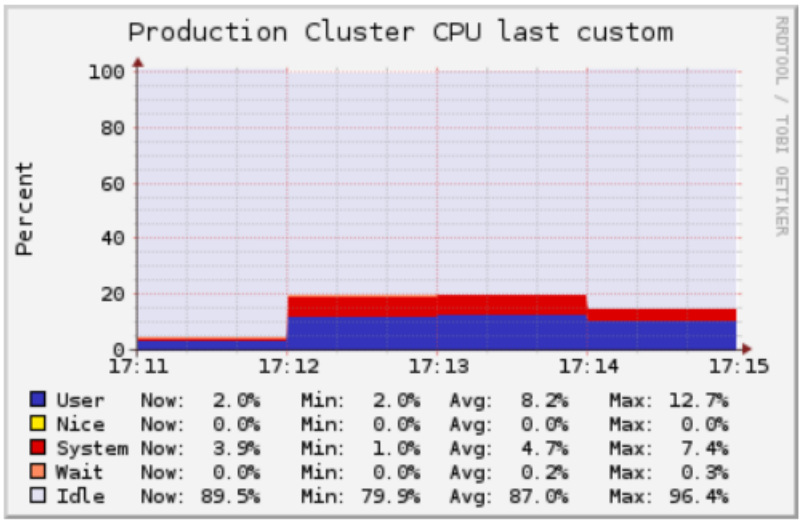
Εικόνα 17 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



Εικόνα 18 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

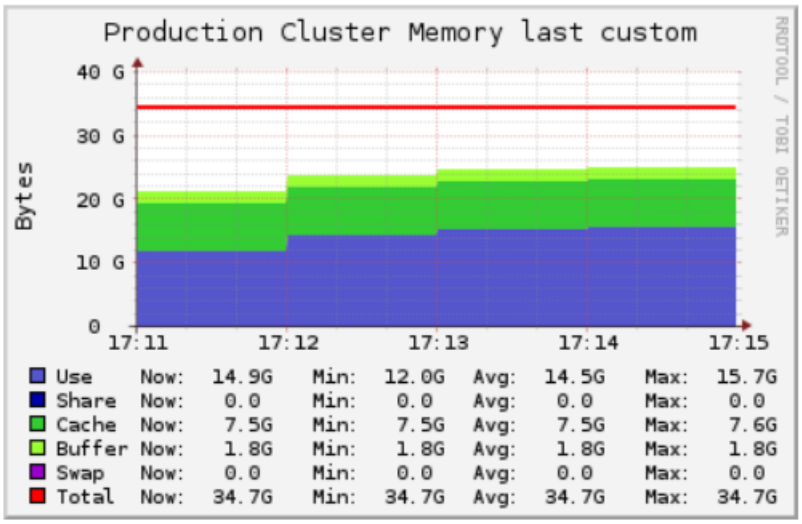
Πιο αναλυτικά για την χρήση της Κεντρικής Μονάδας Επεξεργασίας στην συστοιχία για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική των Εξυπηρετητών Παραμέτρων και για αριθμό παρτίδας 128 μπορούμε να δούμε την Εικόνα 19 να παρατηρήσουμε ότι η μέγιστη χρήση της Κεντρική Μονάδας Επεξεργασίας ήταν 12,7% και η ελάχιστη χρήση ήταν 2%

.



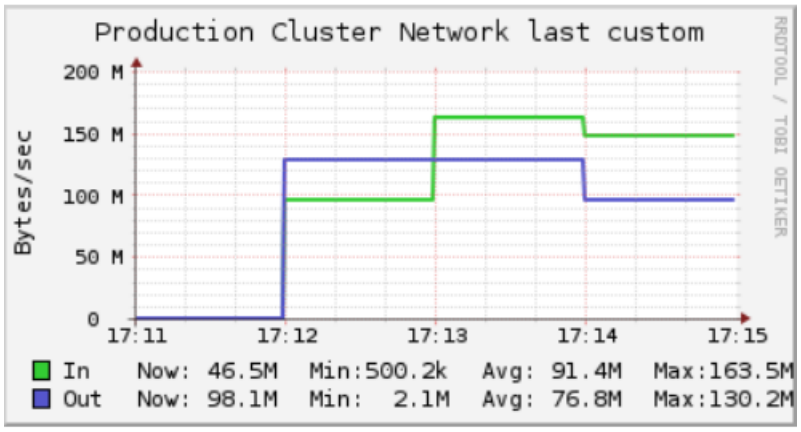
Εικόνα 19Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε την Εικόνα 20 όπου η μέγιστη χρήση της μνήμης ήταν 15,7 G ενώ η ελάχιστη ήταν 12 G



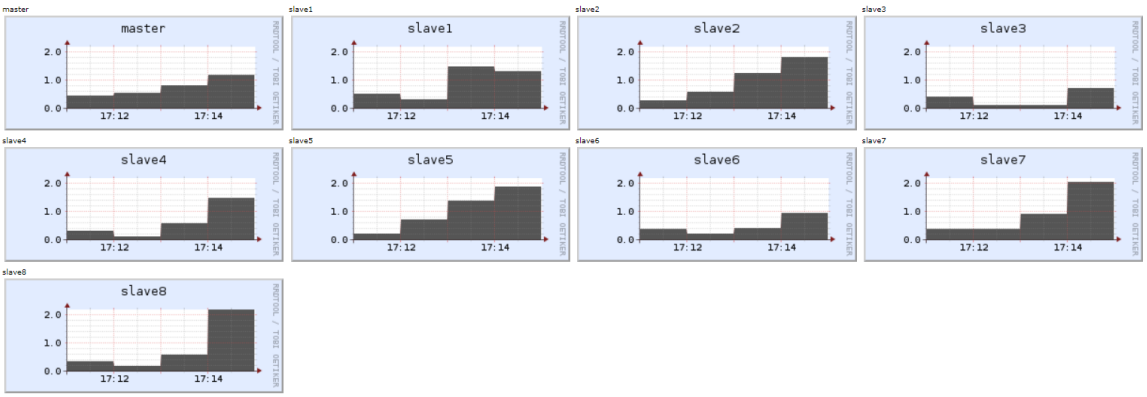
Εικόνα 20 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την εικόνα 21 όπου η ελάχιστη χρήση του δικτυού ήταν 500,2 k ενώ η μέγιστη 163,5 M



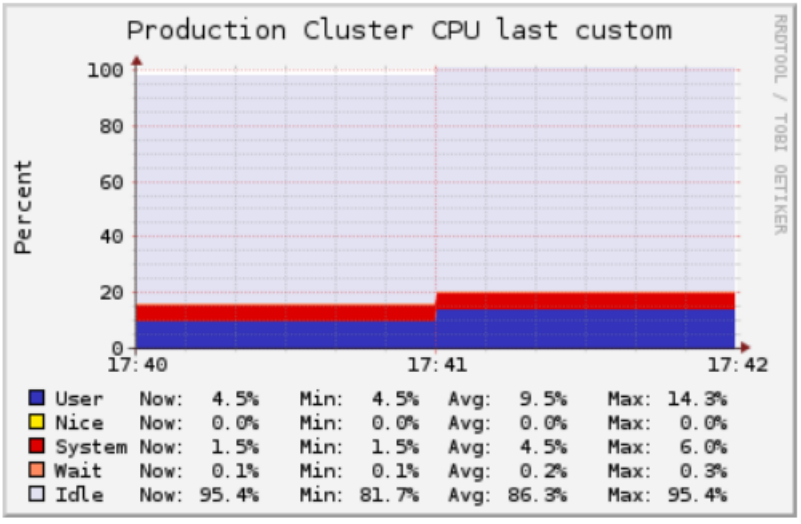
Εικόνα 21 Απεικόνιση χρήσης του δικτυού

Τέλος στην εικόνα 22 μπορούμε να δούμε το φόρτο κάθε μηχανήματος κατά την εκπαίδευση



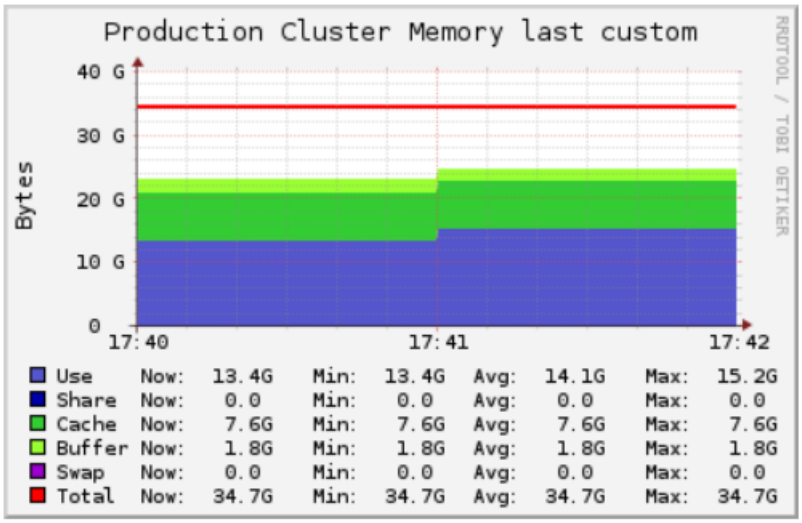
Εικόνα 22 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Όσο αναφορά για την εκπαίδευση του μοντέλου με αριθμό παρτίδας 256 μπορούμε να δούμε την αναλυτικότερη χρήση της Κεντρικής μονάδας Επεξεργασίας στην Εικόνα 23 Όπου η μέγιστη χρήση είναι 14,3% και η ελάχιστη 4,5%



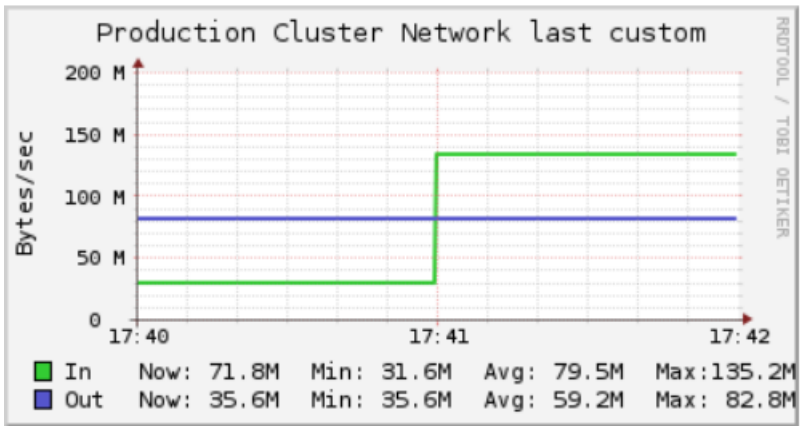
Εικόνα 23 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Όσο αναφορά την χρήση της μνήμης μπορούμε να δούμε στην Εικόνα 24 η μέγιστη χρήση της μνήμης ήταν 15,2 G ενώ η ελάχιστη 13,4 G



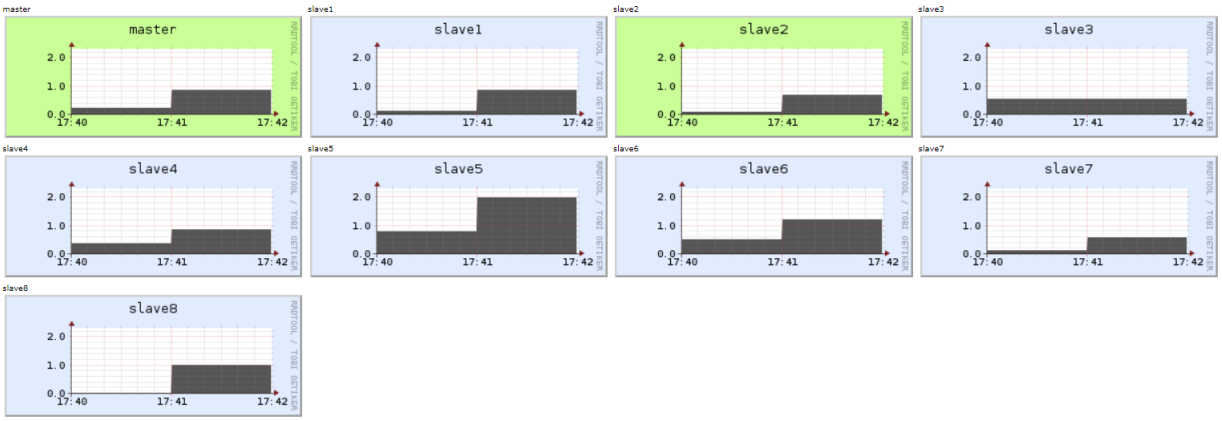
Εικόνα 24 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την Εικόνα 25 όπου η μέγιστη χρήση ήταν 135,2 M και η ελάχιστη 31,6 M



Εικόνα 25 Απεικόνιση χρήσης του δικτυού

Τέλος στην εικόνα 26 μπορούμε να δούμε το φόρτο κάθε μηχανήματος κατά την εκπαίδευση



Εικόνα 26 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Στον Πίνακα 1 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Mnist με αριθμό παρτίδας 128 και στρατηγική Εξυπηρετητών Παραμέτρων

Epoch	Accuracy(%)
1	0.8721
2	0.9739
3	0.979
4	0.9843
5	0.9855
6	0.9918
7	0.9914
8	0.9944
9	0.9932
10	0.9954
11	0.9958
12	0.9963
13	0.9961
14	0.9973
15	0.9977
16	0.9958
17	0.9973
18	0.9986
19	0.9968
20	0.999

21	0.9983
22	0.9973
23	0.9974
24	0.9978
25	0.9982

Πίνακας 1 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Και ο χρόνος εκπαίδευσης ήταν περίπου 173.895 δευτερόλεπτα και η ακρίβεια του μοντέλου με δεδομένα που δεν έχει δει κατά την εκπαίδευση ήταν 0.9804

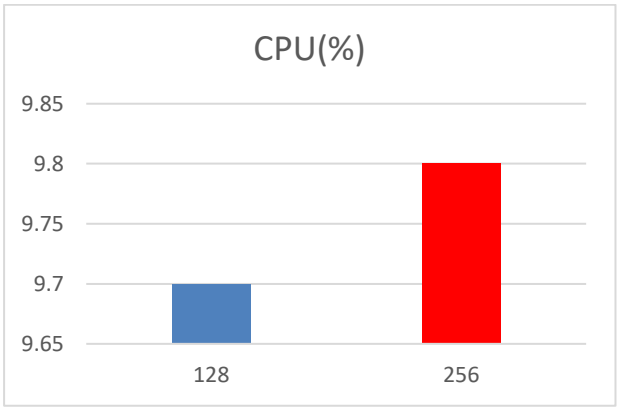
Ενώ για τον αριθμό παρτίδας 256 μπορούμε να δούμε τον Πίνακα 2

Epoch	Accuracy(%)
1	0.8079
2	0.952
3	0.9688
4	0.9766
5	0.9816
6	0.9872
7	0.9863
8	0.9927
9	0.9931
10	0.9946
11	0.9953
12	0.9974
13	1.0015
14	0.998
15	0.9965
16	0.9959
17	0.9979
18	0.9989
19	0.9909
20	0.9993
21	0.9993
22	0.9983
23	0.9963
24	0.9989
25	0.9996

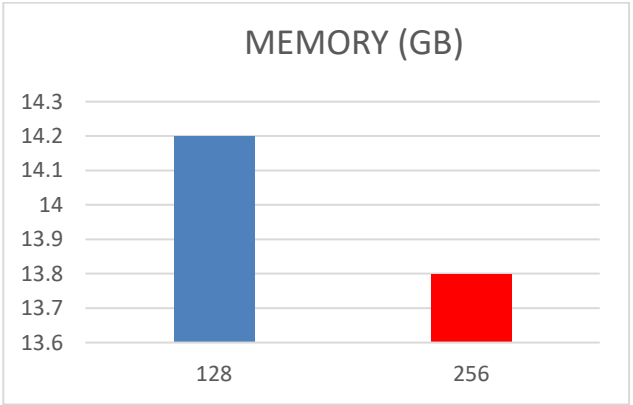
Πίνακας 2 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Ο χρόνος εκπαίδευσης είναι περίπου 107.155 δευτερόλεπτα και η επαλήθευση του μοντέλου με δεδομένα που δεν έχει δει κατά την εκπαίδευση ήταν 0.9838

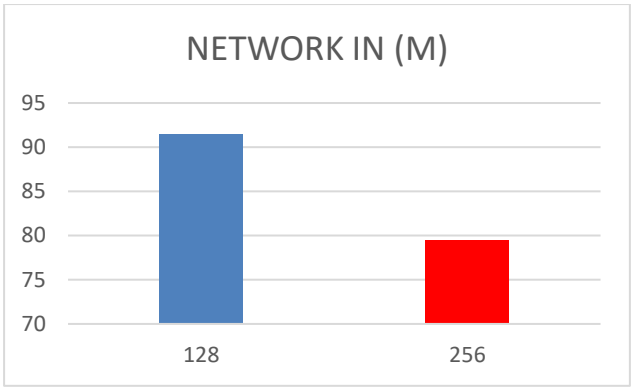
Για την στρατηγική των MultiWorkerMirroredStrategy γίναν δυο εκτελέσεις με διαφορετικό αριθμό παρτίδας (batch size). Για το πρώτο πείραμα ο αριθμός παρτίδας ήταν 128 ενώ για την δεύτερη ήταν 256. Στις εικόνες 27,28 και 29 μπορούμε να δούμε την μέση χρήση πόρων για αριθμό παρτίδας 128 και 256. Η μέση χρήση της Κεντρικής Μονάδας Επεξεργασίας για 128 ήταν 9.7% ενώ για 256 ήταν 9.8% , η μέση χρήση της μνήμης για αριθμό παρτίδας 128 ήταν 14.2 GB ενώ για 256 ήταν 13,8 GB , Τέλος για την χρήση του δικτύου η μέση τιμή 91.4 M για τον αριθμό παρτίδας 128 και 79.4 M για τον αριθμό παρτίδας 256. Η μέση χρήση των πόρων για την εκπαίδευση του μοντέλου με αριθμό παρτίδας 128 ήταν μεγαλύτερη εκτός από την χρήση της κεντρικής μονάδας επεξεργασίας που ήταν μικρότερη από την εκπαίδευση του μοντέλου με αριθμό παρτίδας 256



Εικόνα 27 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



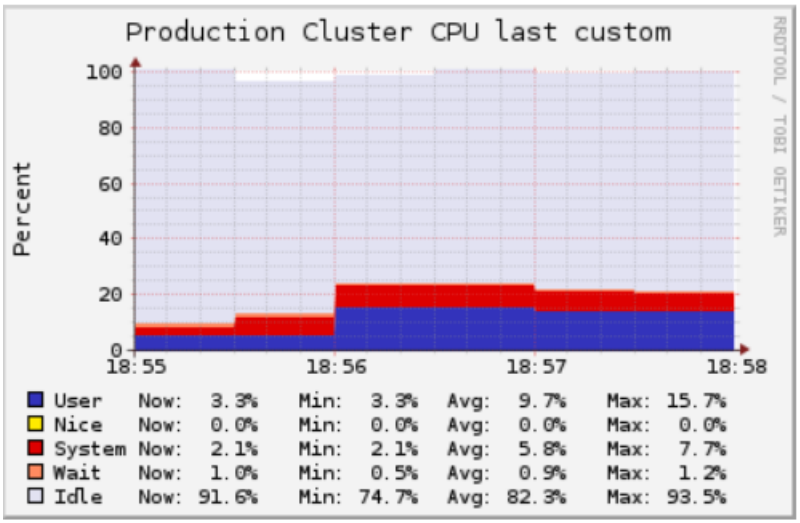
Εικόνα 28 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



Εικόνα 29 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του πρώτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

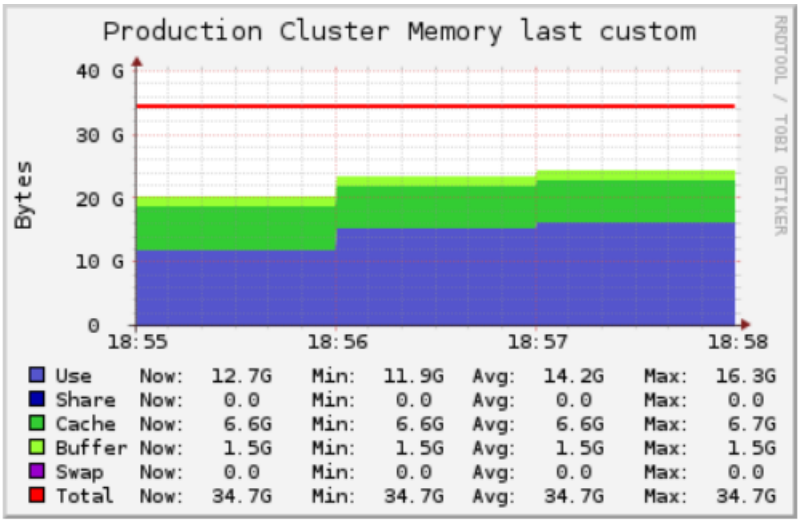
Ειδικότερα για την εκπαίδευση με τον αριθμό παρτίδας 128 μπορούμε να δούμε αναλυτικά την χρήση των πόρων

Για την χρήση της Κεντρικής Μονάδας Επεξεργασίας μπορούμε να δούμε στην Εικόνα 30 η μέγιστη τιμή της ήταν 15.7% ενώ η ελάχιστη ήταν 3.3%



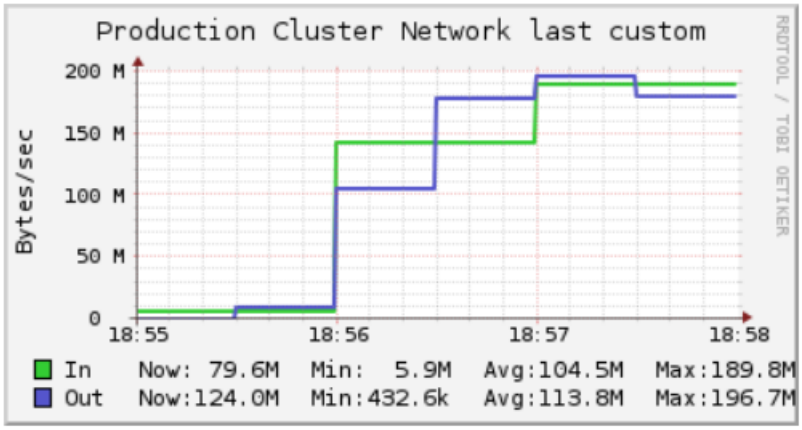
Εικόνα 30 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε την Εικόνα 31 η μέγιστη χρήση της μνήμης 16,3 GB ενώ η ελάχιστη χρήση ήταν 11,9 GB



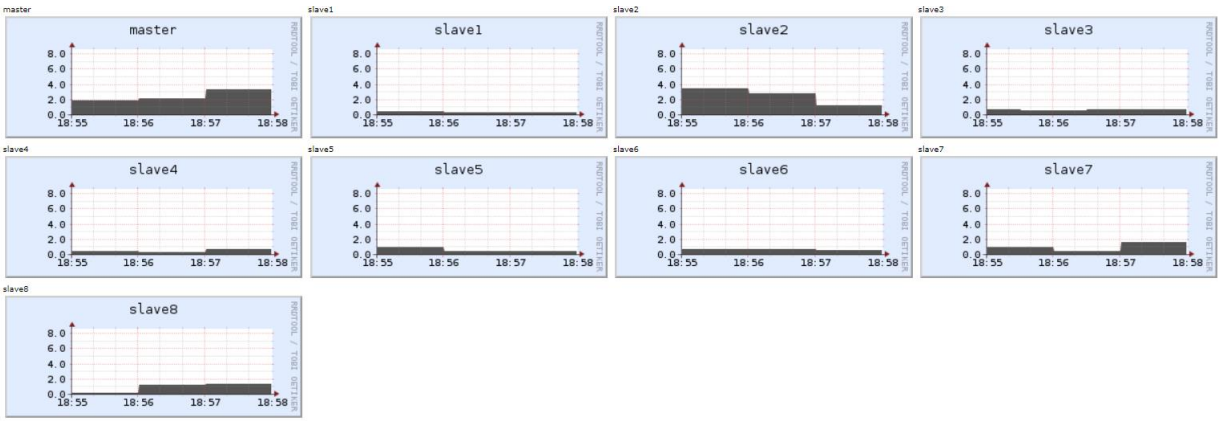
Εικόνα 31 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την Εικόνα 32 η ελάχιστη τιμή ήταν 5,9 M ενώ η μέγιστη 189,8 M



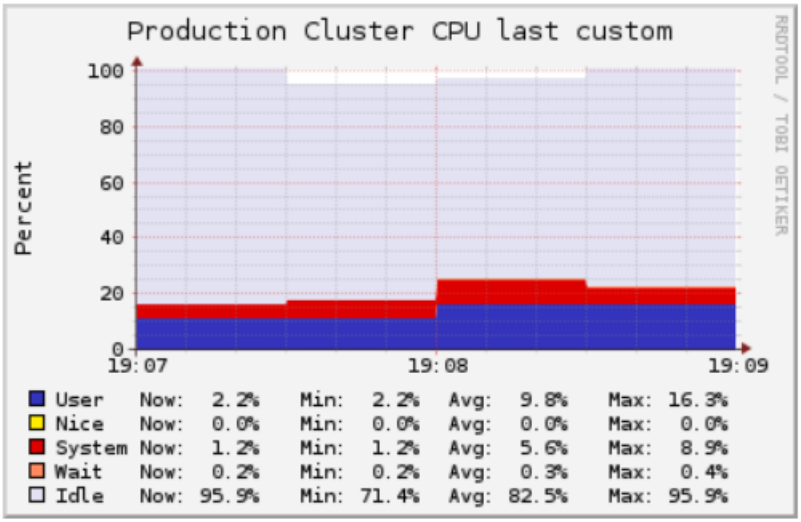
Εικόνα 32 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 33 μπορούμε να δούμε το φόρτο κάθε μηχανήματος κατά την εκπαίδευση



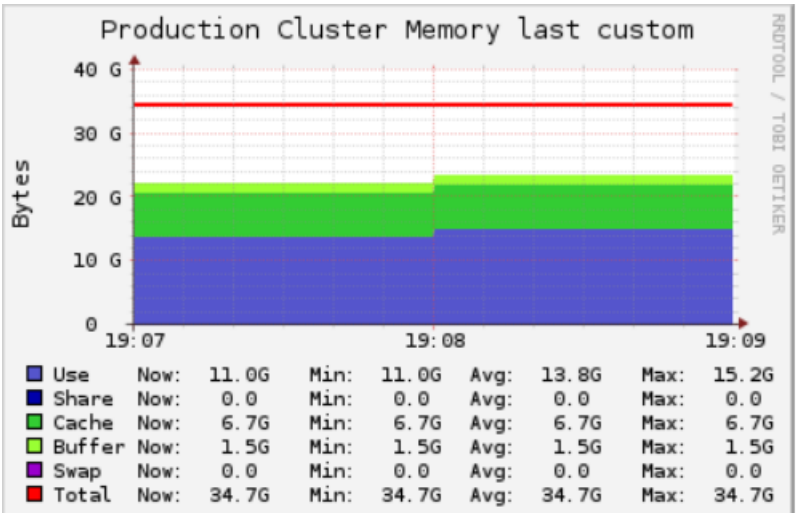
Εικόνα 33 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Όσο αναφορά για την εκπαίδευση του μοντέλου με τον αριθμό παρτίδας 256 για την χρήση της Κεντρικής Μονάδας Επεξεργασίας μπορούμε να δούμε την Εικόνα 34 η μέγιστη τιμής χρήσης ήταν 16,3% και η ελάχιστη τιμή ήταν 2,2%



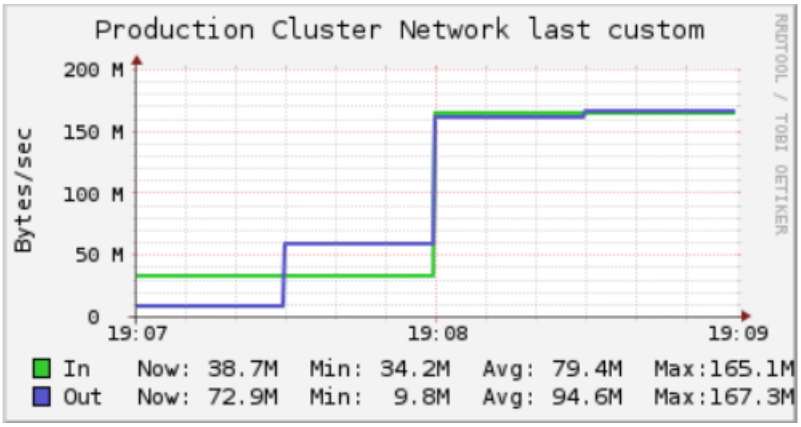
Εικόνα 34 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης στην εικόνα 35 μπορούμε να δούμε ότι η μέγιστη χρήση της μνήμης ήταν 15,2 GB και η ελάχιστη ήταν 11 GB



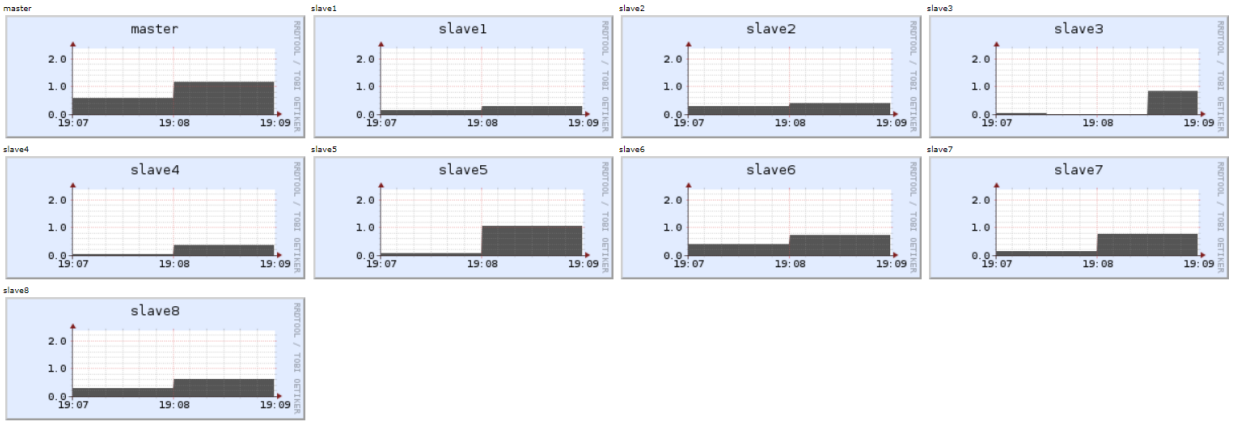
Εικόνα 35 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε στην Εικόνα 36 η μέγιστη τιμή ήταν 165,1 M και η ελάχιστη ήταν 34,2 M



Εικόνα 36 Απεικόνιση χρήσης του δικτυού

Τέλος μπορούμε να δούμε την κατανομή του φόρτου στα μηχανήματα της συστοιχίας στην Εικόνα 37



Εικόνα 37 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

στον Πίνακα 3 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Mnist με αριθμό παρτίδας 128 και στρατηγική MultiWorkerMirroredStrategy

Epoch	Accuracy(%)
1	0.8224
2	0.9383
3	0.9545
4	0.963
5	0.9706
6	0.976
7	0.9763
8	0.9812
9	0.9854
10	0.9847
11	0.9897
12	0.9908
13	0.9921
14	0.9924
15	0.9945
16	0.994
17	0.9959
18	0.9977
19	0.9956

20	0.9982
21	0.9982
22	0.9982
23	0.9986
24	0.9983
25	0.9989

Πίνακας 3 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Η διάρκεια εκπαίδευσης ήταν περίπου 144.703 δευτερόλεπτα και η επαλήθευση του μοντέλου με δεδομένα που δεν έχει δει κατά την εκπαίδευση ήταν 0.9804

Για την εκπαίδευση με αριθμό παρτίδας 256 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο στον πίνακα 4

Epoch	Accuracy(%)
1	0.7425
2	0.9118
3	0.9363
4	0.9445
5	0.9533
6	0.9602
7	0.9627
8	0.9691
9	0.9731
10	0.9748
11	0.9793
12	0.9817
13	0.9828
14	0.9845
15	0.9853
16	0.9856
17	0.9891
18	0.9896
19	0.9898
20	0.9918
21	0.992
22	0.9935
23	0.9946
24	0.9944
25	0.9951

Πίνακας 4 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Ο Χρόνος εκπαίδευσης ήταν περίπου 85.640 δευτερόλεπτα και η επαλήθευση του μοντέλου με δεδομένα που δεν έχει δει κατά την εκπαίδευση ήταν 0.9780

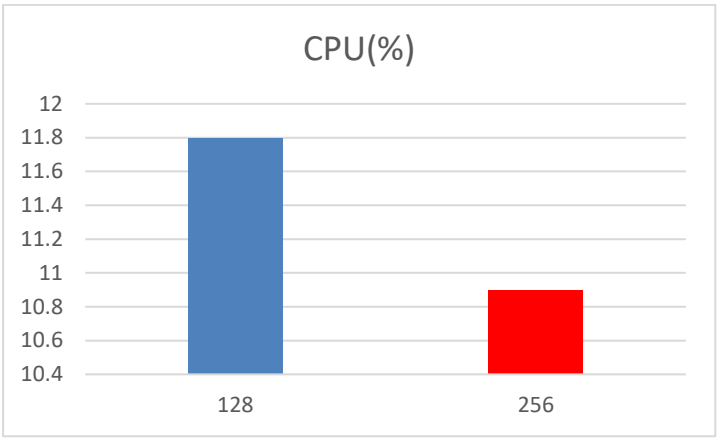
Ενότητα 5.1.4.β Δεύτερο μοντέλο (Fashion Mnist)

Το μοντέλο που χρησιμοποιήθηκε για την πρώτη εκπαίδευση με το Dataset Fashion Mnist όπως είπαμε και στην ενότητα 5.1.2 είναι ένα διαδοχικό βαθύ νευρωνικό δίκτυο με στρώμα εισαγωγής των δεδομένων (Flatten layer) σχήματος (28,28) όσο δηλαδή το μέγεθος μιας εικόνας. Έπειτα ακολουθεί ένα «Πυκνό» στρώμα (Dense layer) με 128 νευρώνες και συνάρτηση ενεργοποίησης (activation function) την relu. Τέλος το μοντέλο περιλαμβάνει ένα «Πυκνό» στρώμα με 10 νευρώνες όσες δηλαδή και οι κατηγορίες των εικόνων .Τον κώδικα για το μοντέλο μπορούμε να δούμε στην Εικόνα 34

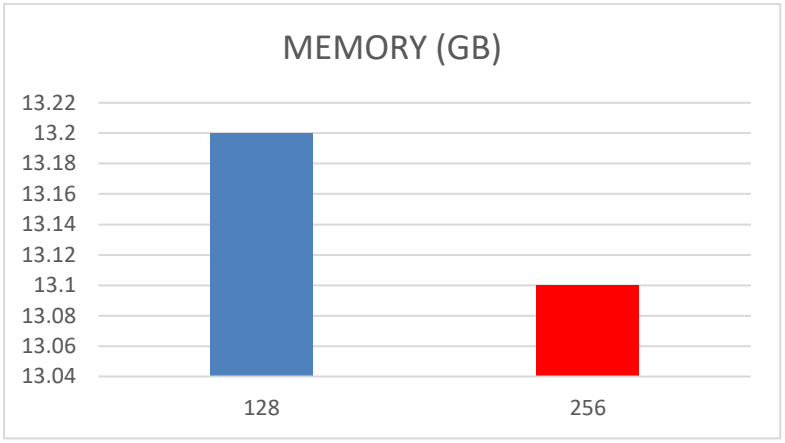
```
model = tf.keras.Sequential([
tf.keras.layers.Flatten(input_shape=(28, 28)),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(10)
])
```

Εικόνα 38 Ο κώδικας του δεύτερου μοντέλου

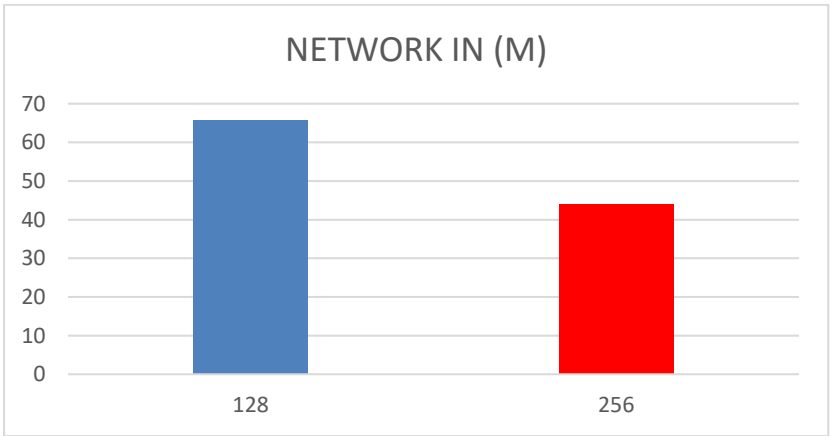
Για την στρατηγική Εξυπηρετητών παραμέτρων και για μέγεθος παρτίδας 128 και 256 μπορούμε να στις εικόνες 39,40 και 41| . Η μέση χρήση της Κεντρικής μονάδας επεξεργασίας ήταν 11,8% για μέγεθος παρτίδας 128 και 10,9% για μέγεθος παρτίδας 256. Η μέση χρήση της μνήμης ήταν 13,2 GB για μέγεθος παρτίδας 128 και 13,1 GB για μέγεθος παρτίδας 256. Τέλος η μέση χρήση του δικτυού ήταν 65,8 M για μέγεθος παρτίδας 128 και 43,9M για μέγεθος παρτίδας 256. Όπου η εκπαίδευση με μέγεθος παρτίδας 128 είχε μέση χρήση των πόρων μεγαλύτερη από αυτήν με μέγεθος εκπαίδευσης 256



Εικόνα 39 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

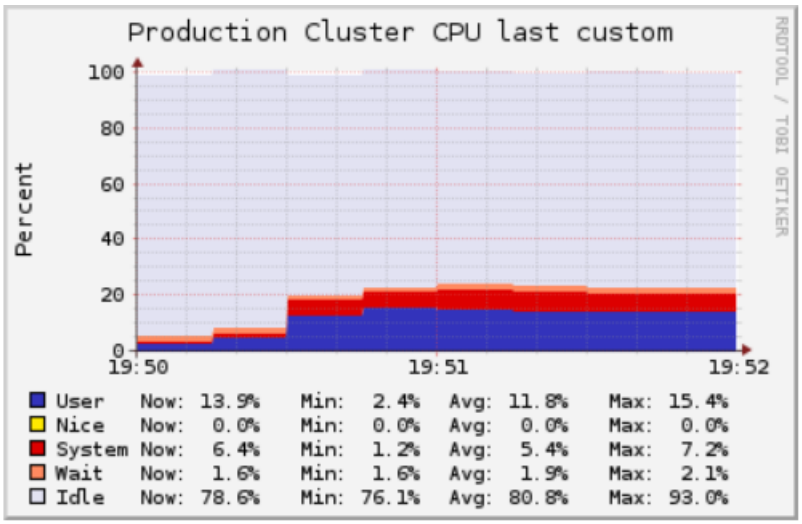


Εικόνα 40 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



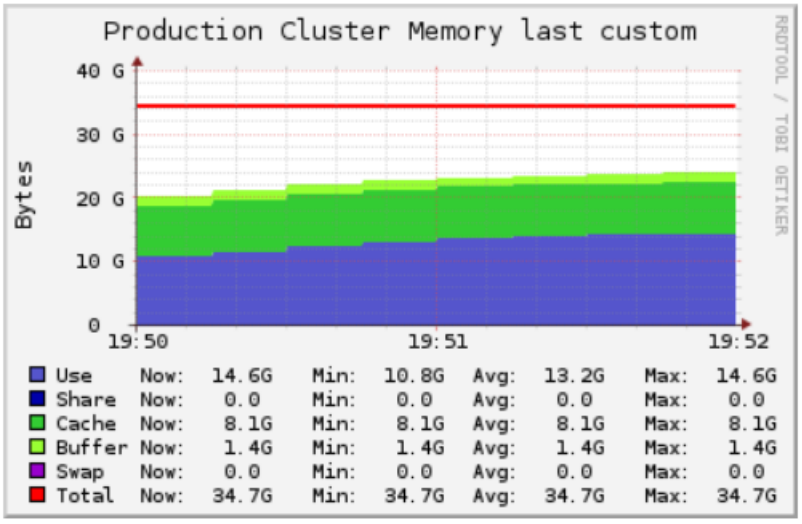
Εικόνα 41 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

Ειδικότερα για την χρήση πόρων για την εκπαίδευση με μέγεθος παρτίδας 128 μπορούμε να δούμε την Εικόνα 42. Η μέγιστη χρήση της Κεντρικής μονάδας Επεξεργασίας ήταν 15,4% ενώ η ελάχιστη χρήση ήταν 2.4%



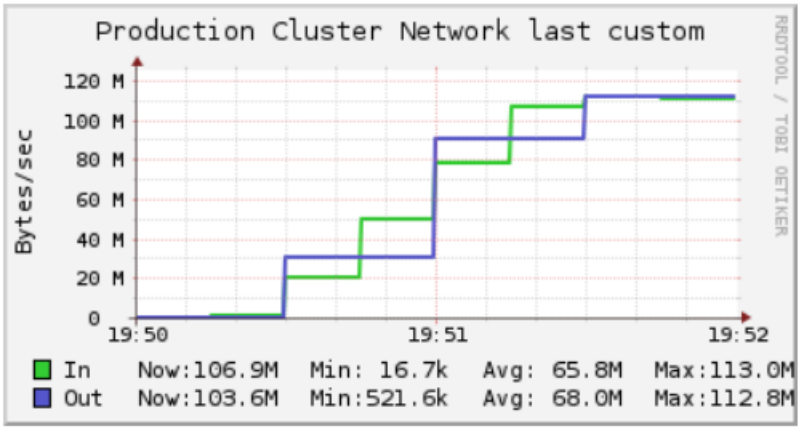
Εικόνα 42 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε την Εικόνα 43 η μέγιστη χρήση της μνήμης ήταν 14,6 G και η ελάχιστη τιμή ήταν 10,8 G.



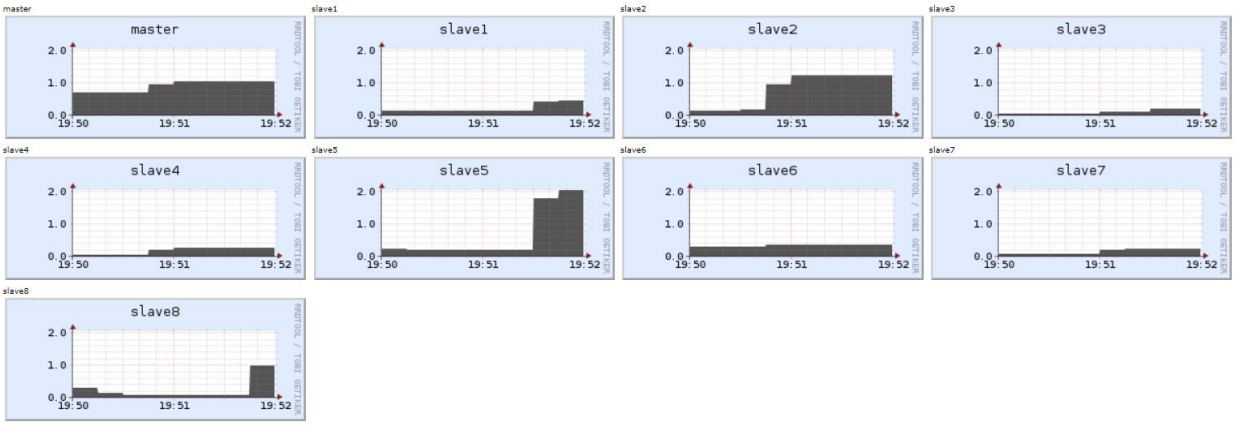
Εικόνα 43 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την Εικόνα 44 η μέγιστη χρήση του δικτυού ήταν 113M και η ελάχιστη χρήση ήταν 16.7k



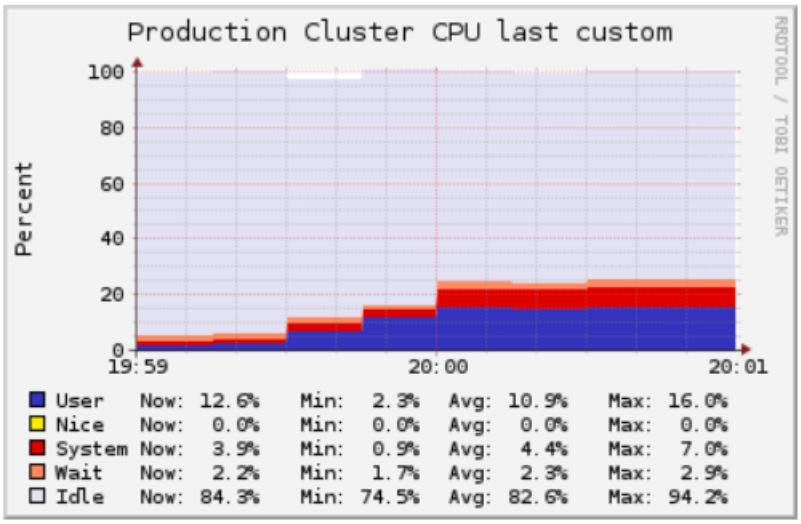
Εικόνα 44 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 45 μπορούμε να δούμε το φόρτο εργασίας κάθε μηχανήματος κατά την διάρκεια της εκπαίδευσης.



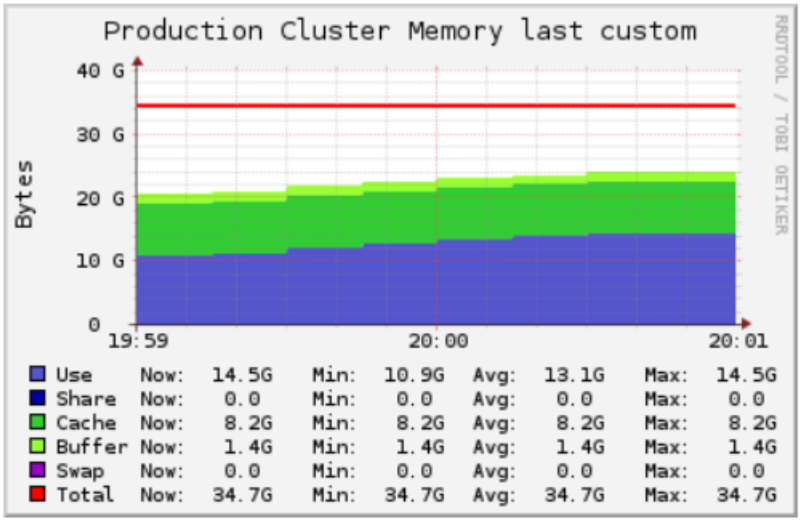
Εικόνα 45 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Όσο αναφορά την εκπαίδευση με μέγεθος παρτίδας 256 μπορούμε να δούμε την Εικόνα 46 η μέγιστη χρήση της Κεντρική Μονάδας Επεξεργασίας ήταν 16% ενώ η ελάχιστη τιμή ήταν 2.3%



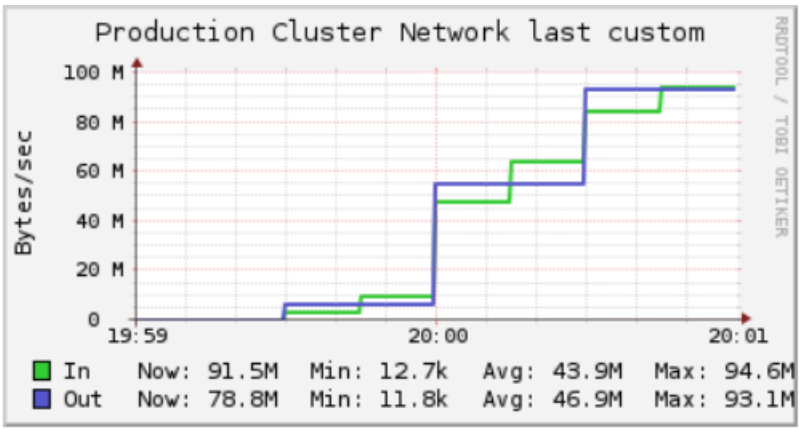
Εικόνα 46 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Όσο αναφορά την χρήση της μνήμης μπορούμε να δούμε την Εικόνα 47 όπου η μέγιστη χρήση ήταν 14,5 G ενώ η ελάχιστη ήταν 10,9G



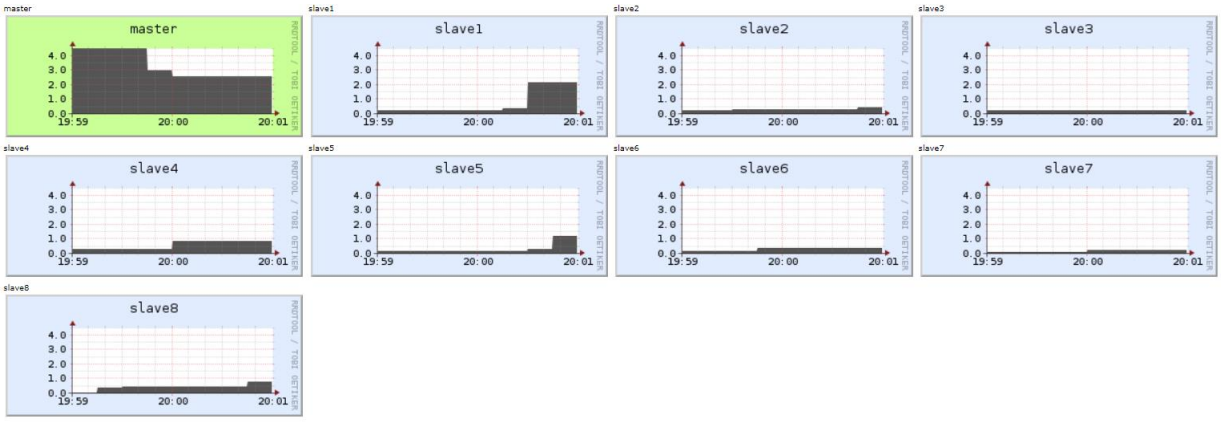
Εικόνα 47 Απεικόνιση χρήσης της μνήμης

Όσο αναφορά το δίκτυο μπορούμε να δούμε την Εικόνα 48 όπου η μέγιστη χρήση του δικτυού ήταν 94,6M ενώ η ελάχιστη ήταν 12,7 k



Εικόνα 48 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 49 μπορούμε να δούμε το φόρτο εργασίας των μηχανήματων κατά την εκπαίδευση



Εικόνα 49 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

στον Πίνακα 5 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Fashion Mnist με αριθμό παρτίδας 128 και στρατηγική Εξυπηρετητών Παραμέτρων

Epoch	Accuracy(%)
1	0.7301
2	0.8499
3	0.8616
4	0.8658
5	0.8764
6	0.8882
7	0.8934
8	0.8945
9	0.8966
10	0.8993
11	0.9087
12	0.9147
13	0.9136
14	0.9126
15	0.9128
16	0.916
17	0.9244
18	0.9251
19	0.9241
20	0.9248
21	0.9251
22	0.9294
23	0.9348
24	0.9341
25	0.9353

Πίνακας 5 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Ο χρόνος εκπαίδευσης ήταν περίπου 121.245 δευτερόλεπτα και η ακρίβεια του μοντέλου με δεδομένα που δεν έχει δει κατά την εκπαίδευση ήταν 0.8862

Ενώ για την εκπαίδευση με αριθμό παρτίδας 256 μπορούμε να δούμε τον Πίνακα 6

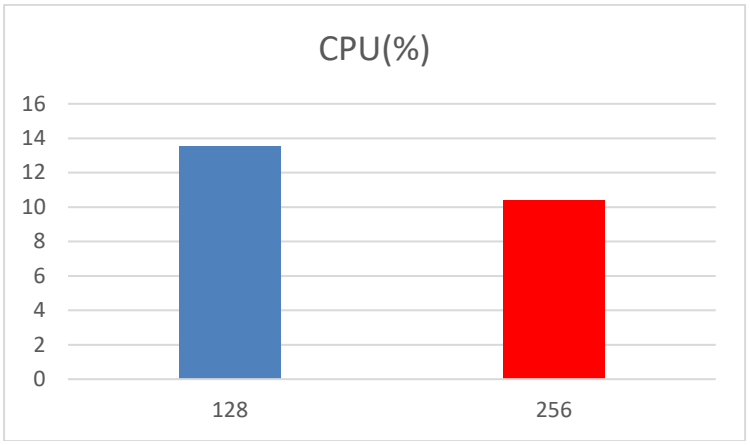
Epoch	Accuracy(%)
1	0.677
2	0.8314
3	0.85
4	0.857
5	0.8686
6	0.8731

7	0.8818
8	0.883
9	0.8824
10	0.8881
11	0.8927
12	0.8993
13	0.8939
14	0.8996
15	0.8982
16	0.9025
17	0.905
18	0.9093
19	0.9075
20	0.909
21	0.9098
22	0.9149
23	0.9172
24	0.9198
25	0.9205

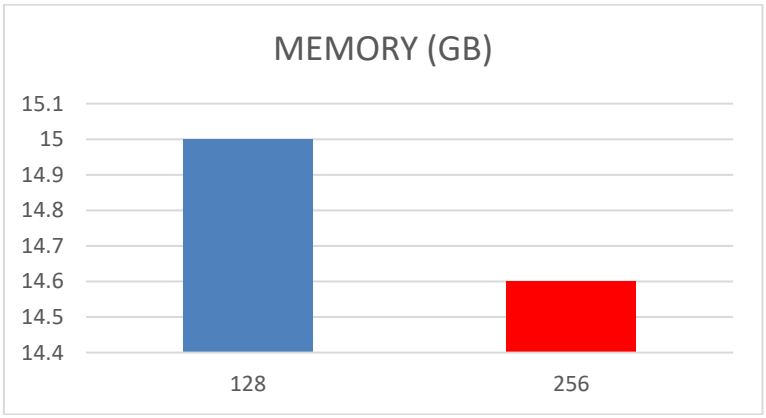
Πίνακας 6 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Ο χρόνος εκπαίδευσης διήρκησε περίπου 89.151 δευτερόλεπτα ενώ η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε δει κατά την διάρκεια της εκπαίδευσης ήταν 0.8920

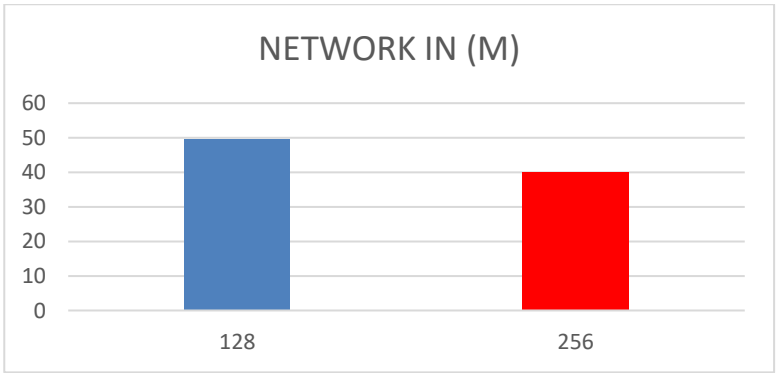
Για την εκπαίδευση του μοντέλου με την στρατηγική MultiWorkerMirroredStrategy μπορούμε να δούμε στις εικόνες 50,51 και 52. Η μέση χρήστη της Κεντρικής Μονάδας Επεξεργασίας για μέγεθος παρτίδας 128 ενώ ήταν 13,5% ενώ ήταν 10,4% για την εκπαίδευση με μέγεθος 256. Για την μέση χρήση της μνήμης ήταν 15 GB για μέγεθος παρτίδας 128 και 14,6 GB για μέγεθος παρτίδας 256. Η μέση χρήση του δικτυού ήταν 49,5 M για μέγεθος παρτίδας 128 και 40,1M για μέγεθος παρτίδας 256. Η μέση χρήση των πόρων ήταν μεγαλύτερη για την εκπαίδευση με μέγεθος παρτίδας 128.



Εικόνα 50 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

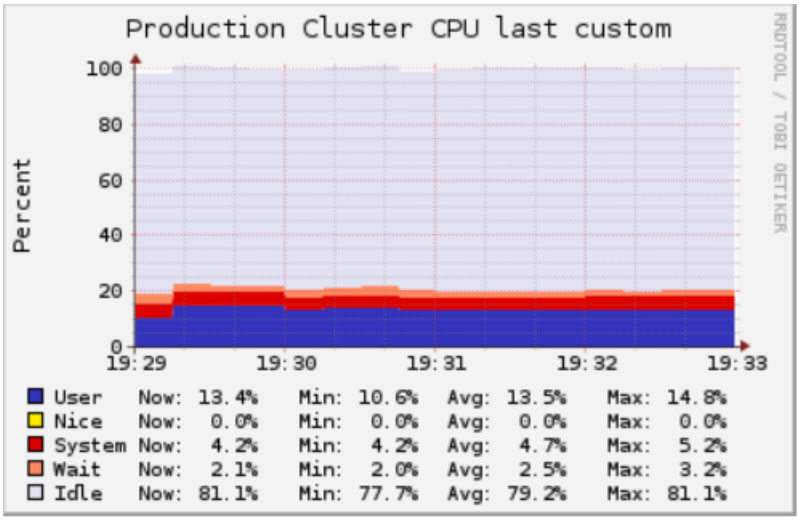


Εικόνα 51 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



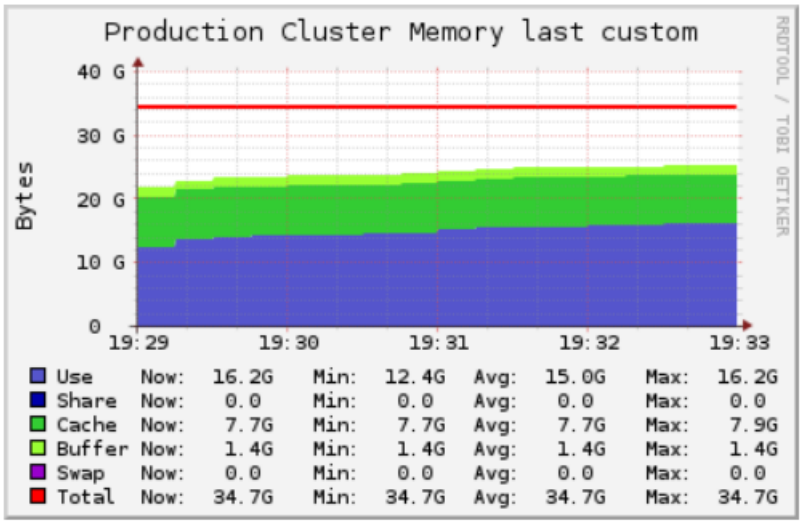
Εικόνα 52 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του δεύτερου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

Ειδικότερα για την χρήση της κεντρικής μονάδας επεξεργασίας με μέγεθος παρτίδας 128 μπορούμε να δούμε την Εικόνα 53 όπου η μέγιστη χρήση ήταν 14,8% και η ελάχιστη 10.6%



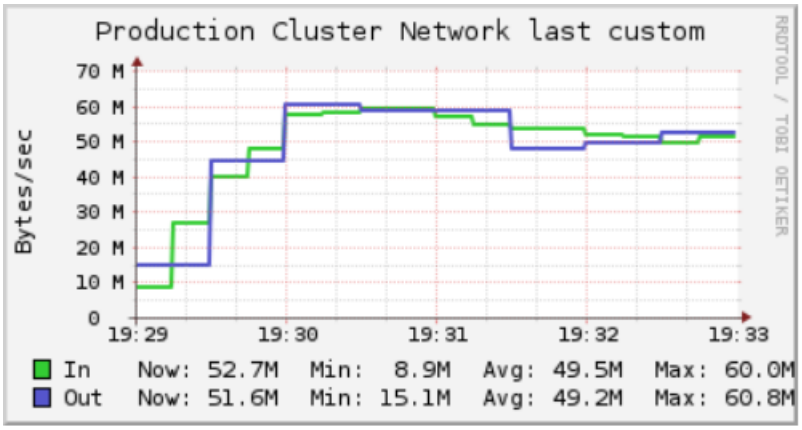
Εικόνα 53 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε στην Εικόνα 54 ότι η μέγιστη χρήση της μνήμης ήταν 16,2 GB και η ελάχιστη ήταν 12,4 GB



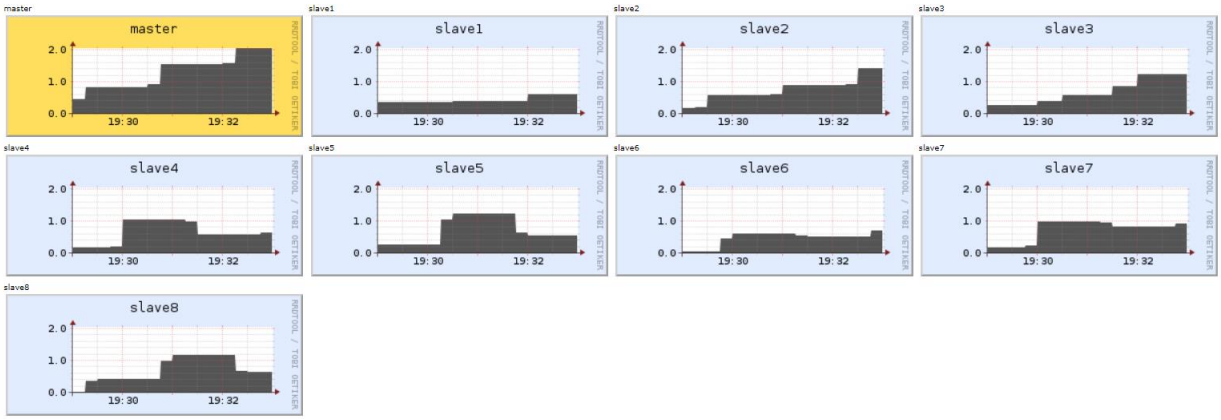
Εικόνα 54 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτύου μπορούμε να δούμε την Εικόνα 55 η μέγιστη χρήση του δικτυού ήταν 60 M και η ελάχιστη ήταν 8,9 M



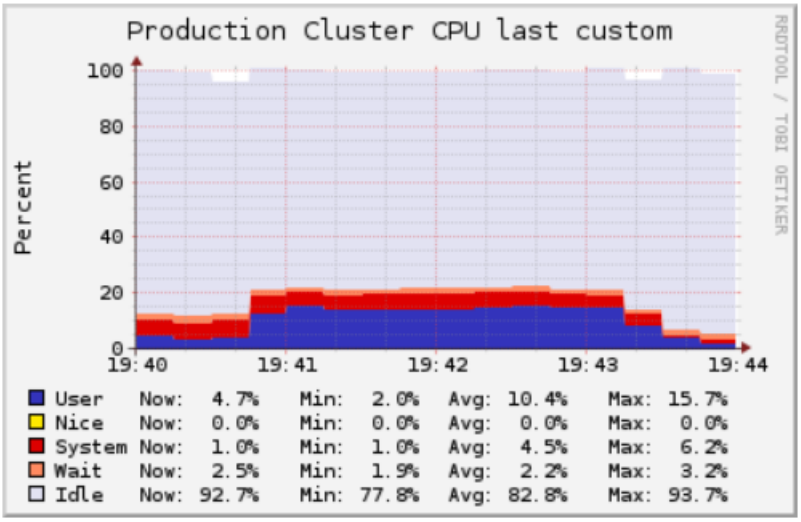
Εικόνα 55 Απεικόνιση χρήσης του δικτυού

Τέλος στην εικόνα 56 μπορούμε να δούμε το φόρτο εργασίας κάθε μηχανήματος κατά την διάρκεια εκπαίδευσης του μοντέλου



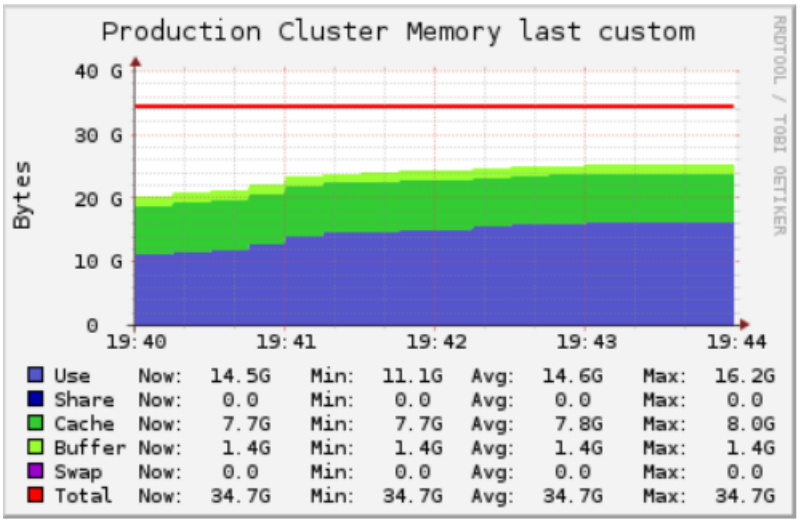
Εικόνα 56 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Για μέγεθος παρτίδας 256 μπορούμε να δούμε στην Εικόνα 57 την μέγιστη χρήση της κεντρικής μονάδας επεξεργασίας που ήταν 15,7% και την ελάχιστη που ήταν 2%



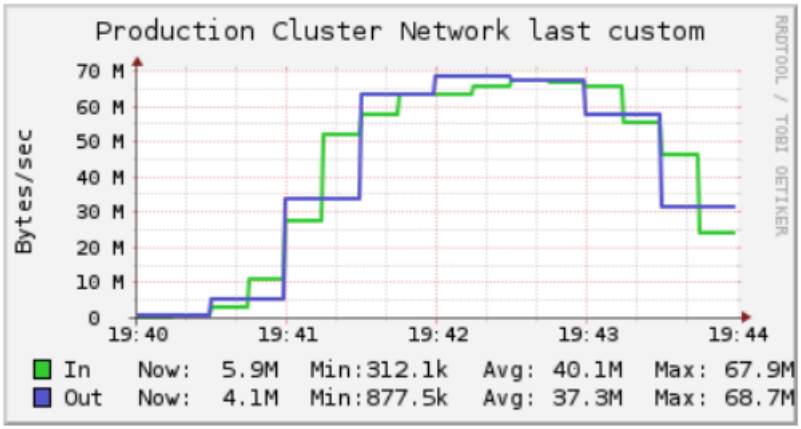
Εικόνα 57 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε την Εικόνα 58 όπου η μέγιστη χρήση της μνήμης ήταν 16,2 GB ενώ η ελάχιστη ήταν 11,1 GB



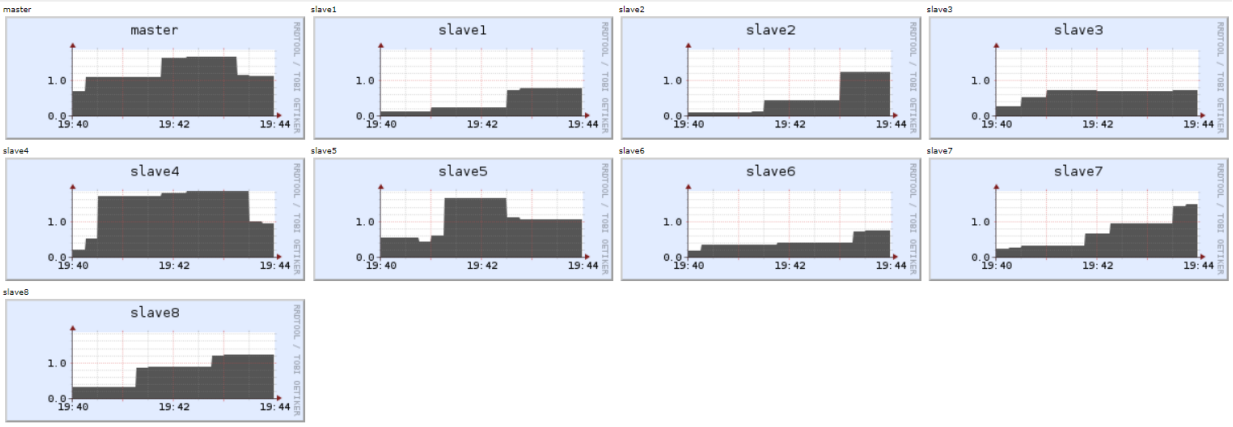
Εικόνα 58 Απεικόνιση χρήσης της μνήμης

Όσο αναφορά την χρήση του δικτυού μπορούμε να δούμε στην Εικόνα 59 την μέγιστη χρήση του δικτυού ήταν 67,9 M και η ελάχιστη ήταν 312,1 k



Εικόνα 59 Απεικόνιση χρήσης του δικτυού

Τέλος μπορούμε να δούμε στην Εικόνα 60 το φόρτο εργασίας σε κάθε μηχάνημα κατά την διάρκεια της εκπαίδευσης



Εικόνα 60 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

στον Πίνακα 7 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Fashion Mnist με αριθμό παρτίδας 128 και στρατηγική MultiWorkerMirroredStrategy

Epoch	Accuracy(%)
1	0.7069
2	0.8139
3	0.8347
4	0.8457
5	0.8447
6	0.8466
7	0.8573
8	0.8587
9	0.8635
10	0.8709
11	0.8662
12	0.8696
13	0.8775
14	0.8698
15	0.8716
16	0.8811
17	0.8754
18	0.8821
19	0.8872
20	0.881
21	0.883
22	0.8867
23	0.8853
24	0.8838
25	0.8926

Πίνακας 7 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 259.300 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8577

Όσο αναφορά την εκπαίδευση με μέγεθος παρτίδας 256 μπορούμε να δούμε στον Πίνακα 8 την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Fashion Mnist και στρατηγική MultiWorkerMirroredStrategy

Epoch	Accuracy(%)
1	0.6107
2	0.785
3	0.8058
4	0.8249
5	0.8259
6	0.8281
7	0.8332
8	0.8386
9	0.8466
10	0.8551
11	0.8543
12	0.8491
13	0.859
14	0.8555
15	0.8523
16	0.8602
17	0.8685
18	0.8735
19	0.8628
20	0.8691
21	0.869
22	0.867
23	0.8682
24	0.8669
25	0.8723

Πίνακας 8 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 202.254 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8592

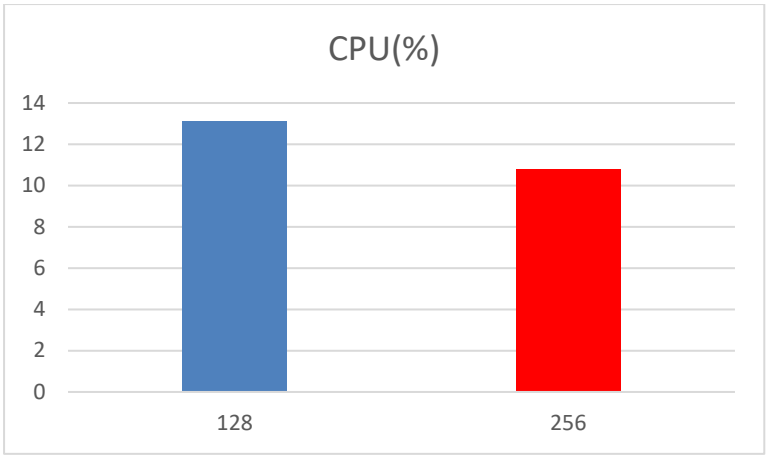
Ενότητα 5.1.4.γ Τρίτο μοντέλο (Kmnist)

Το μοντέλο που χρησιμοποιήθηκε για την πρώτη εκπαίδευση με το Dataset Kmnist όπως είπαμε και στην ενότητα 5.1.2 είναι ένα διαδοχικό βαθύ νευρωνικό δίκτυο με στρώμα εισαγωγής των δεδομένων (Flatten layer) σχήματος (28,28) όσο δηλαδή το μέγεθος μιας εικόνας. Έπειτα ακολουθεί ένα «Πυκνό» στρώμα (Dense layer) με 128 νευρώνες και συνάρτηση ενεργοποίησης (activation function) την relu. Τέλος το μοντέλο περιλαμβάνει ένα «Πυκνό» στρώμα με 10 νευρώνες όσες δηλαδή και οι κατηγορίες των εικόνων .Τον κώδικα για το μοντέλο μπορούμε να δούμε στην Εικόνα 53

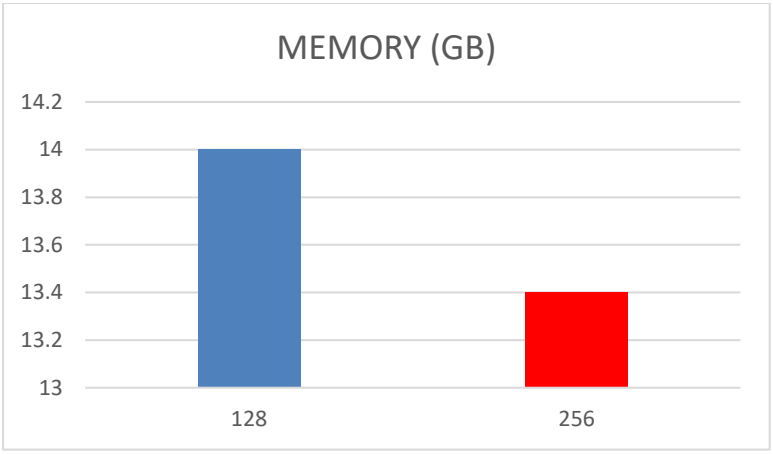
```
model = tf.keras.Sequential([
tf.keras.layers.Flatten(input_shape=(28, 28)),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(10)
])
```

Εικόνα 61 Ο κώδικας του τρίτου μοντέλου

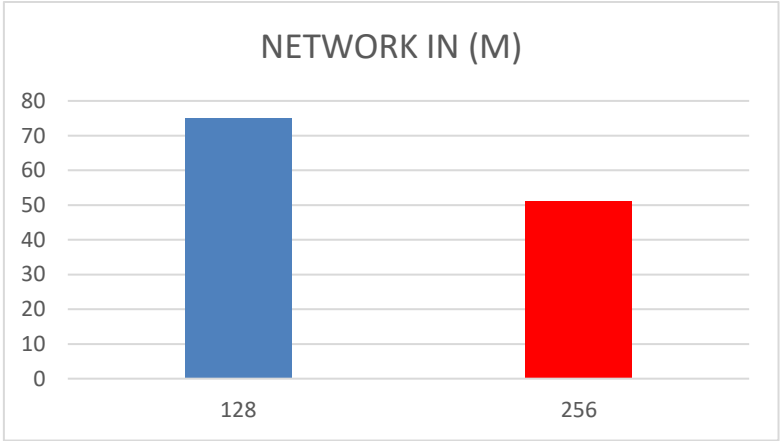
Για την στρατηγική Εξυπηρετητών παραμέτρων και για μέγεθος παρτίδας 128 και 256 μπορούμε να δούμε στις εικόνες 62, 63 και 64. Η μέση χρήση της Κεντρικής μονάδας επεξεργασίας ήταν 13,1% για μέγεθος παρτίδας 128 και 10,8% για μέγεθος παρτίδας 256. Η μέση χρήση της μνήμης ήταν 14 GB για μέγεθος παρτίδας 128 και 13,4 GB για μέγεθος παρτίδας 256. Τέλος η μέση χρήση του δικτυού ήταν 75 M για μέγεθος παρτίδας 128 και 51,1 για μέγεθος παρτίδας 256. Για την εκπαίδευση με μέγεθος παρτίδας 128 η μέση χρήση των πόρων ήταν μεγαλύτερη από αυτήν με μέγεθος παρτίδας 256



Εικόνα 62 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

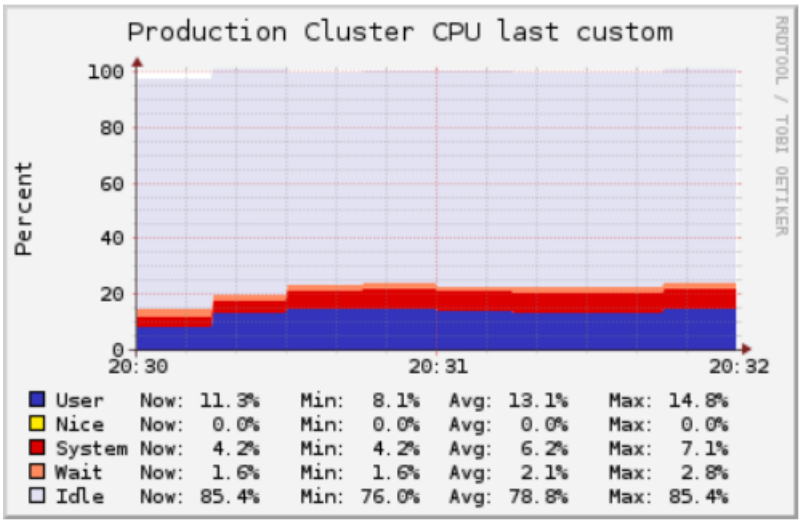


Εικόνα 63 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



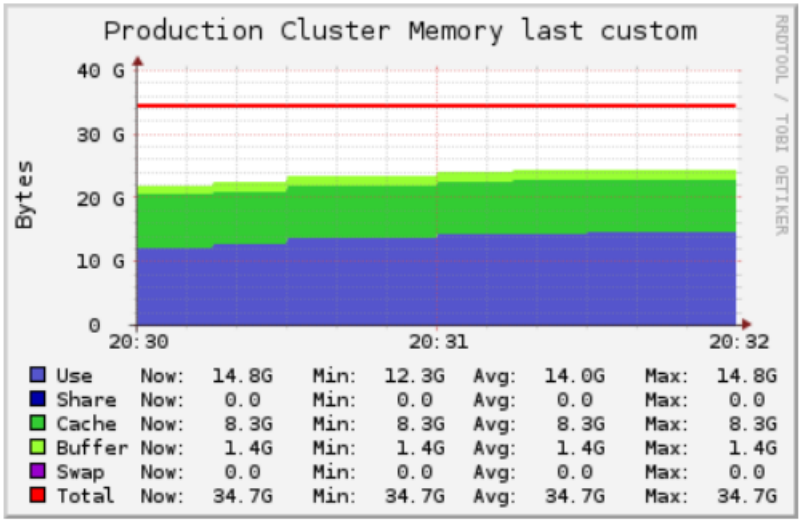
Εικόνα 64 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική Εξυπηρετητών παραμέτρων για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

Ειδικότερα για την χρήση της κεντρικής μονάδας επεξεργασίας με μέγεθος παρτίδας 128 μπορούμε να δούμε την Εικόνα 65 όπου η μέγιστη χρήση ήταν 14,8% και η ελάχιστη 8,1%



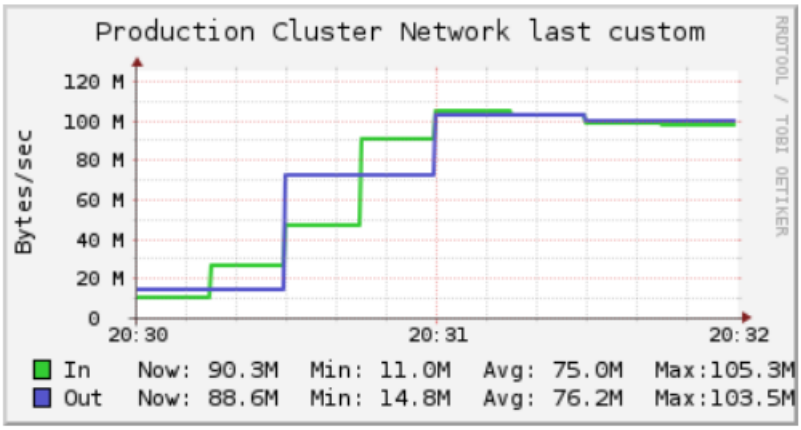
Εικόνα 65 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε στην Εικόνα 66 ότι η μέγιστη χρήση ήταν 14,8 G και η ελάχιστη ήταν 12,3 G



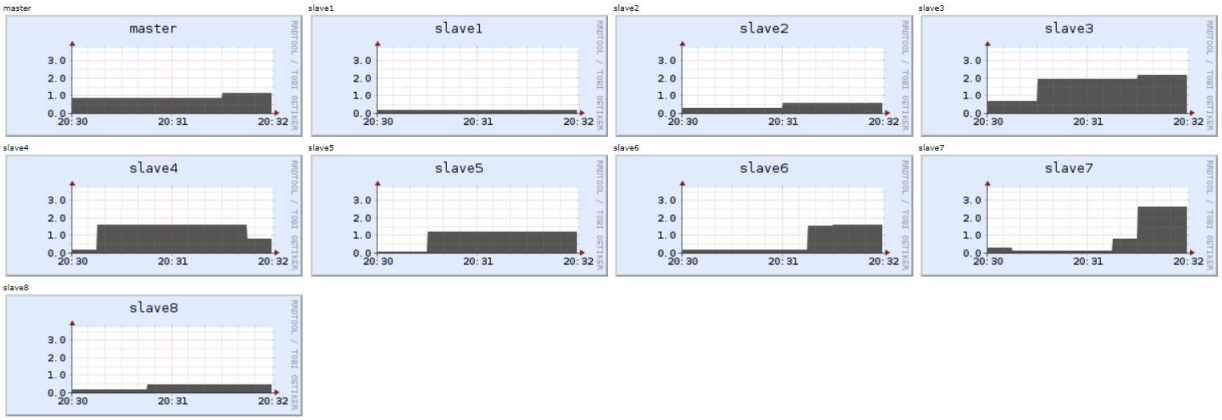
Εικόνα 66 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την Εικόνα 67 ότι η μέγιστη χρήση ήταν 105,3 M και η ελάχιστη 11 M



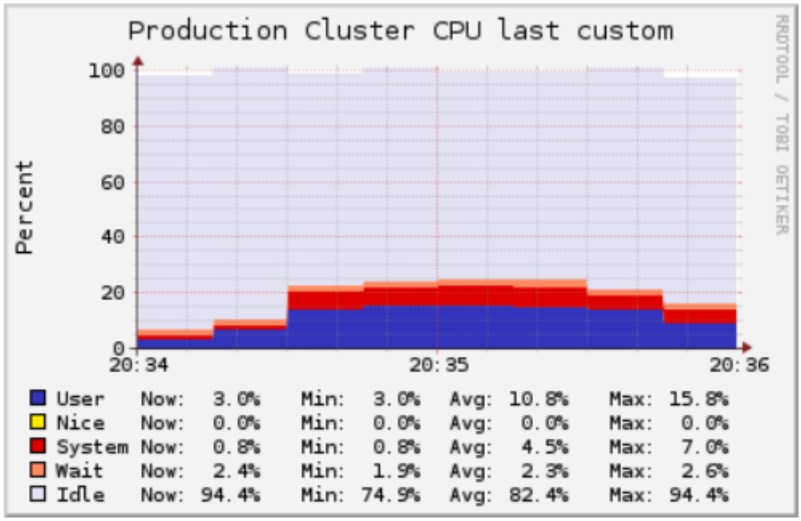
Εικόνα 67 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 68 μπορούμε να δούμε την κατανομή του φόρτου εργασίας κατά την εκπαίδευση στα μηχανήματα



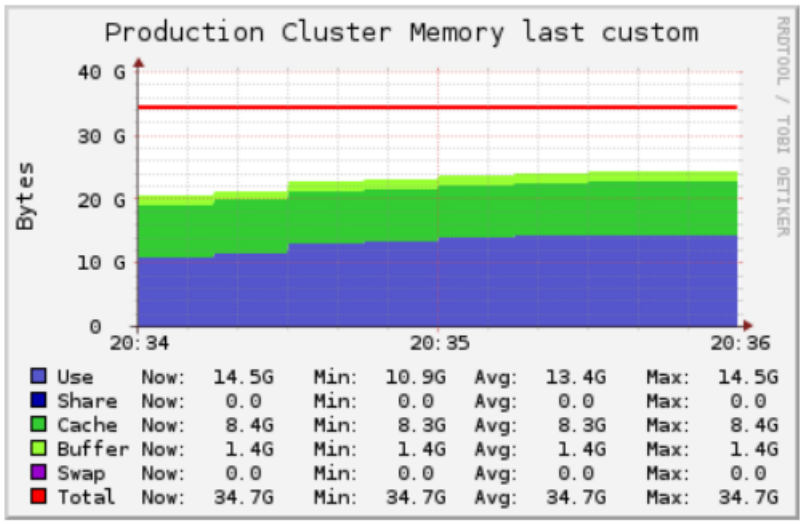
Εικόνα 68 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Όσο αναφορά την εκπαίδευση για μέγεθος παρτίδας 256 μπορούμε να δούμε στην Εικόνα 69 ότι η μέγιστη χρήση της κεντρικής μονάδας επεξεργασίας ήταν 15,8% και η ελάχιστη ήταν 3%



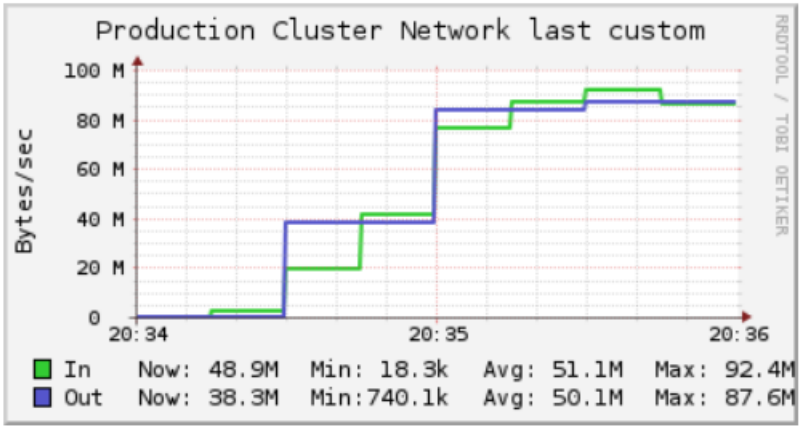
Εικόνα 69 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης μπορούμε να δούμε στην Εικόνα 70 ότι η μέγιστη χρήση ήταν 14,5 G και η ελάχιστη 10,9 G



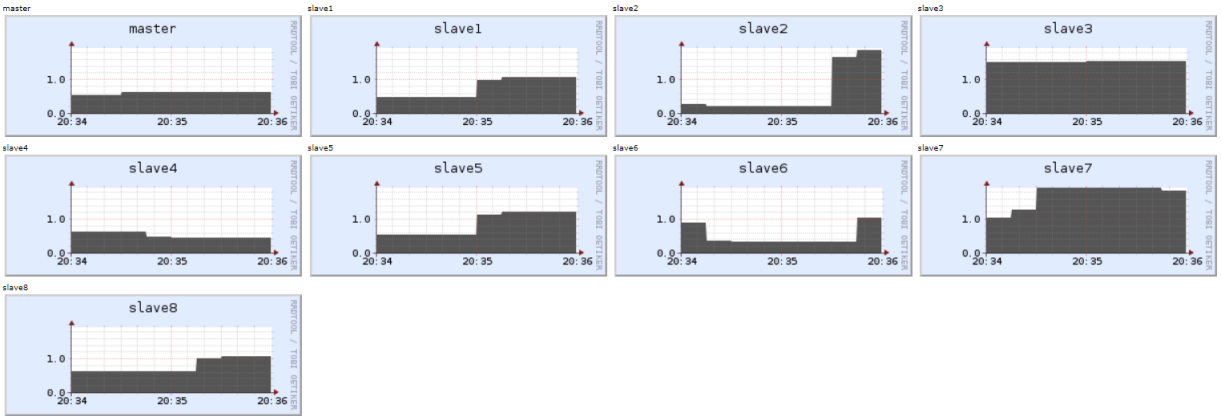
Εικόνα 70 Απεικόνιση χρήσης της μνήμης

Όσο αναφορά την χρήση του δικτυού μπορούμε να δούμε στην Εικόνα 71 ότι η μέγιστη χρήση ήταν 92,4 M και η ελάχιστη ήταν 18,3 k



Εικόνα 71 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 72 μπορούμε να δούμε την κατανομή του φόρτου εργασίας κατά την εκπαίδευση στα μηχανήματα



Εικόνα 72 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

στον Πίνακα 9 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Kmnist με αριθμό παρτίδας 128 και στρατηγική Εξυπηρετητών παραμέτρων

Epoch	Accuracy(%)
1	0.832
2	0.9375
3	0.9578
4	0.9665
5	0.9703
6	0.9769
7	0.9824
8	0.986
9	0.9885
10	0.9883
11	0.9905
12	0.9928
13	0.9935
14	0.9943
15	0.9949
16	0.9956
17	0.9956
18	0.9965
19	0.9972
20	0.9964
21	0.9971
22	0.9975
23	0.9967
24	0.9981
25	0.9984

Πίνακας 9 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 125.570 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8758

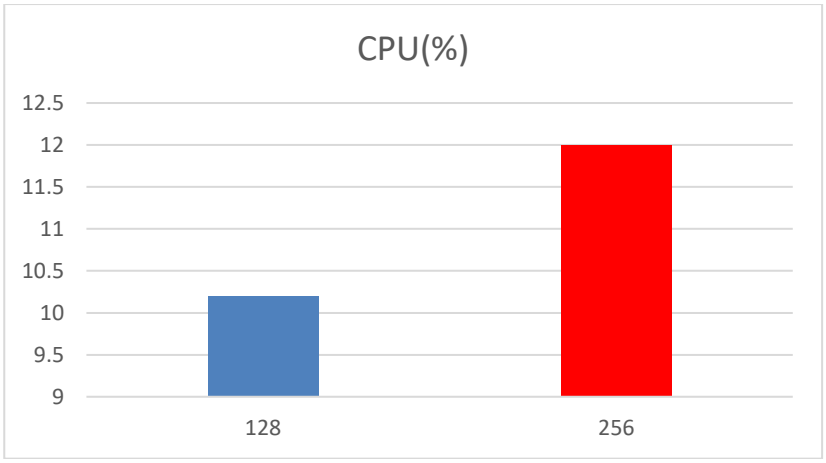
στον Πίνακα 10 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Kmnist με αριθμό παρτίδας 256 και στρατηγική Εξυπηρετητών παραμέτρων

Epoch	Accuracy(%)
1	0.7698
2	0.9034
3	0.9337
4	0.9441
5	0.9546
6	0.9613
7	0.9665
8	0.9709
9	0.979
10	0.9812
11	0.9822
12	0.9852
13	0.9879
14	0.9897
15	0.9902
16	0.992
17	0.9933
18	0.9939
19	0.9947
20	0.9952
21	0.9963
22	0.9968
23	0.9968
24	0.9979
25	0.9984

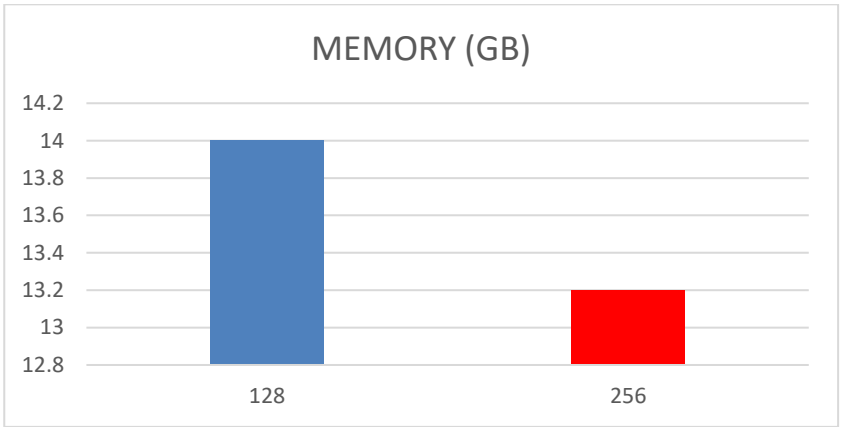
Πίνακας 10 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 82.683 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8855

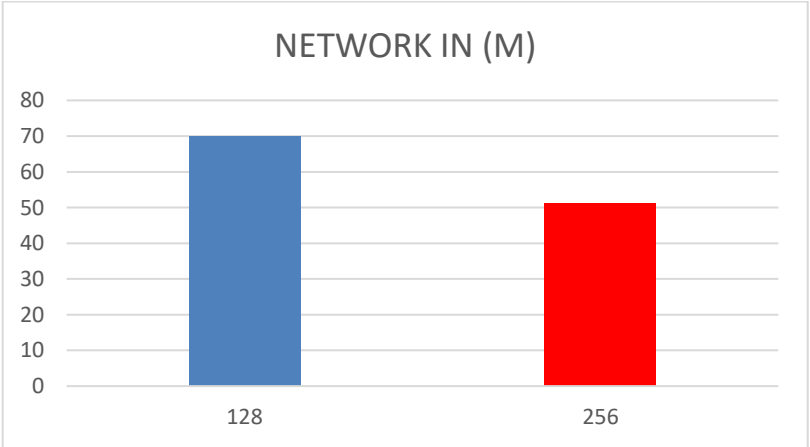
Για την στρατηγική MultiWorkerMirroredStrategy και για μέγεθος παρτίδας 128 και 256 μπορούμε να δούμε στις εικόνες 73, 74 και 75 Η μέση χρήση της Κεντρικής μονάδας επεξεργασίας ήταν 10,2% για μέγεθος παρτίδας 128 και 12% για μέγεθος παρτίδας 256. Η μέση χρήση της μνήμης ήταν 14 GB για μέγεθος παρτίδας 128 και 13,2 GB για μέγεθος παρτίδας 256. Τέλος η μέση χρήση του δικτυού ήταν 69,9 M για μέγεθος παρτίδας 128 και 51,3 M για μέγεθος παρτίδας 256. Για την εκπαίδευση με μέγεθος παρτίδας 128 η μέση χρήση των πόρων ήταν μεγαλύτερη από αυτήν με μέγεθος παρτίδας 256 εκτός από την μέση χρήση της κεντρικής μονάδας επεξεργασίας



Εικόνα 73 Γραφική απεικόνιση της μέσης χρήση κεντρικής μονάδας επεξεργασίας για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

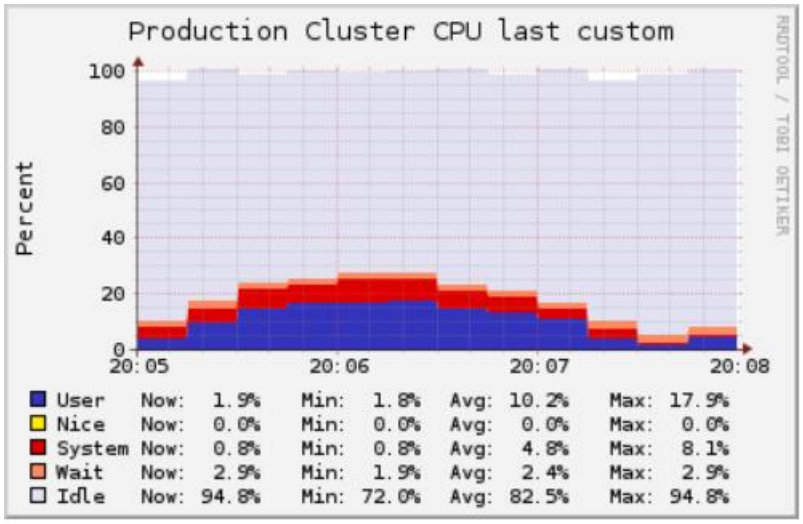


Εικόνα 74 Γραφική απεικόνιση της μέσης χρήση Μνήμης για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256



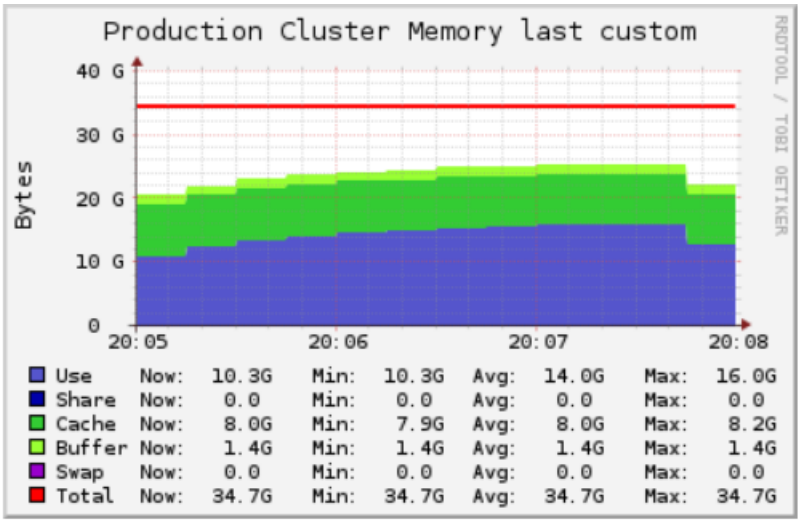
Εικόνα 75 Γραφική απεικόνιση της μέσης χρήση Δικτυού για την εκπαίδευση του τρίτου μοντέλου με την στρατηγική MultiWorkerMirroredStrategy για δυο παραλλαγές με διαφορετικό μέγεθος παρτίδας 128 και 256

Ειδικότερα για την χρήση της κεντρικής μονάδας επεξεργασίας με μέγεθος παρτίδας 128 μπορούμε να δούμε την Εικόνα 76 όπου η μέγιστη χρήση ήταν 17,9% και η ελάχιστη 1,8%



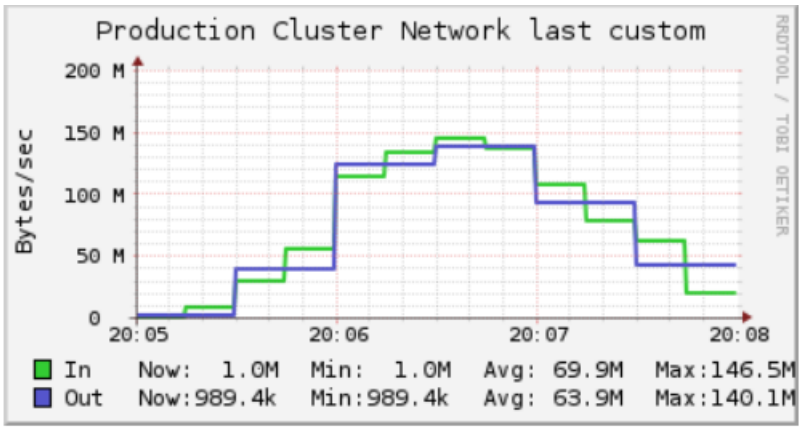
Εικόνα 76 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την χρήση της μνήμης στην εικόνα 77 βλέπουμε ότι η μέγιστη χρήση ήταν 16 G και η ελάχιστη ήταν 10,3 G



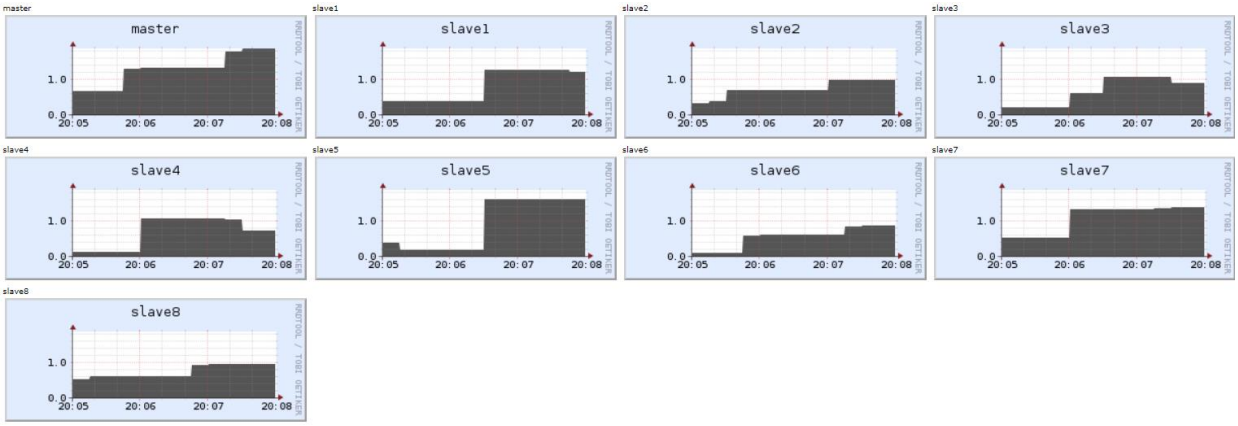
Εικόνα 77 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού βλέπουμε στην εικόνα 78 ότι η μέγιστη χρήση του δικτυού ήταν 146,5 M και η ελάχιστη ήταν 1 M



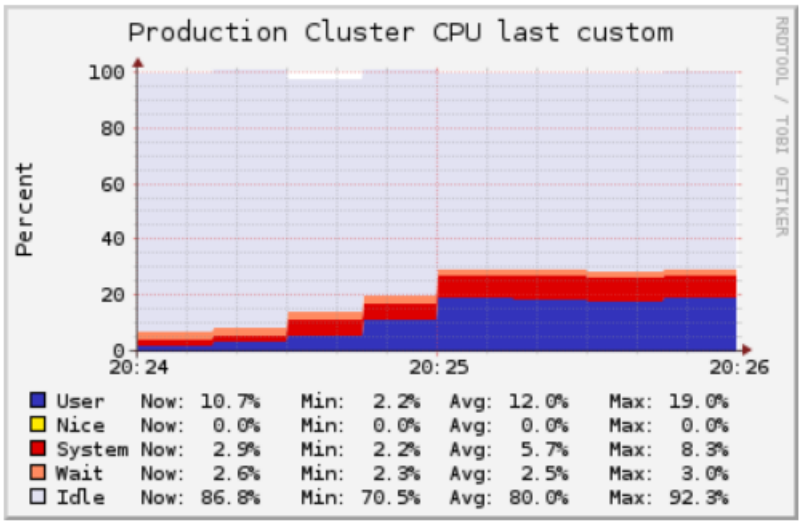
Εικόνα 78 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 79 μπορούμε να δούμε την κατανομή του φόρτου εργασίας κατά την εκπαίδευση στα μηχανήματα



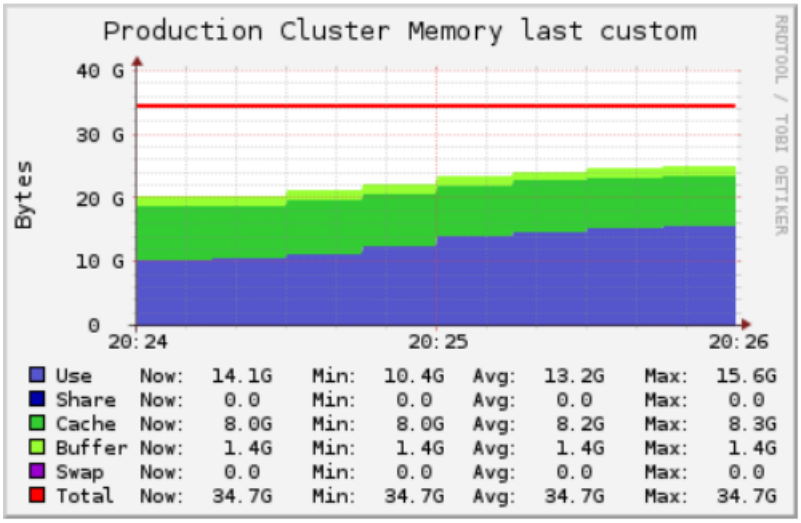
Εικόνα 79 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Όσο αναφορά την εκπαίδευση για μέγεθος παρτίδας 256 μπορούμε να δούμε στην εικόνα 80 ότι η μέγιστη χρήση της κεντρικής μονάδας επεξεργασίας ήταν 19% και η ελάχιστη ήταν 2,2 %



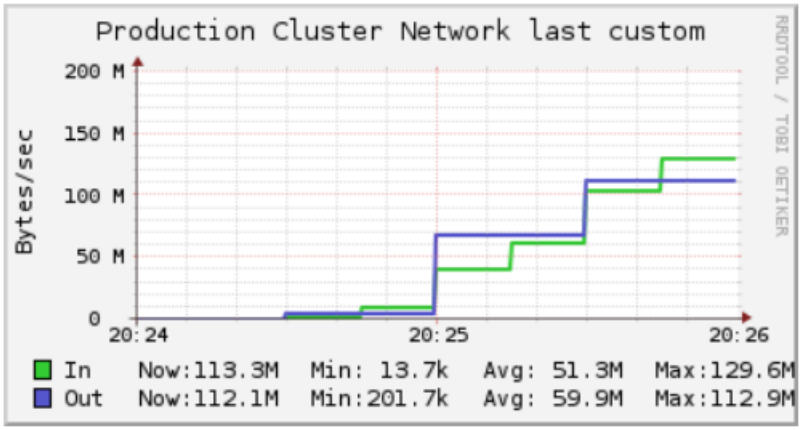
Εικόνα 80 Απεικόνιση χρήσης της κεντρικής μονάδας επεξεργασίας

Για την μνήμη μπορούμε να δούμε στην Εικόνα 81 ότι η μέγιστη χρήση ήταν 15,6 G και η ελάχιστη ήταν 10,4 G



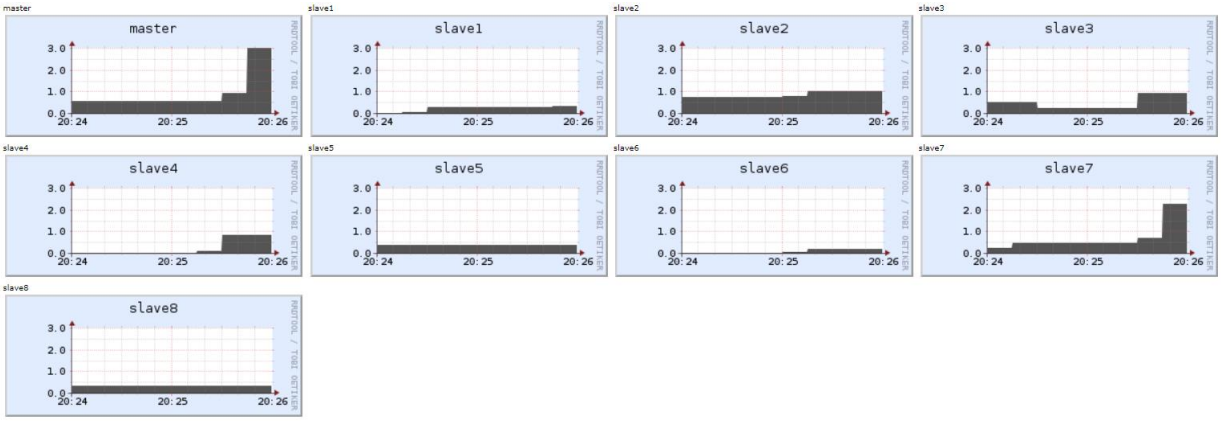
Εικόνα 81 Απεικόνιση χρήσης της μνήμης

Για την χρήση του δικτυού μπορούμε να δούμε την εικόνα 82 όπου η μέγιστη χρήση του δικτυού ήταν 129,6 M και η ελάχιστη ήταν 13,7 k



Εικόνα 82 Απεικόνιση χρήσης του δικτυού

Τέλος στην Εικόνα 83 μπορούμε να δούμε την κατανομή του φόρτου εργασίας κατά την εκπαίδευση στα μηχανήματα



Εικόνα 83 Απεικόνιση διαμοιρασμού του φόρτου εργασίας στα μηχανήματα

Στον Πίνακα 11 μπορούμε να δούμε τις εποχές και την ακρίβεια του μοντέλου στο τέλος της εποχής για το μοντέλο που εκπαιδεύτηκε πάνω στα δεδομένα του Dataset Kmnist με αριθμό παρτίδας 128 και στρατηγική MultiWorkerMirroredStrategy

Epoch	Accuracy(%)
1	0.6781
2	0.8359
3	0.8647
4	0.8818
5	0.8901
6	0.9017
7	0.9079
8	0.9163
9	0.9199
10	0.9257
11	0.9331
12	0.9347
13	0.9428
14	0.9411
15	0.9431
16	0.9468
17	0.9507
18	0.956

19	0.9537
20	0.9595
21	0.9583
22	0.9617
23	0.9598
24	0.963
25	0.9622

Πίνακας 11 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 128

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 145.943 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8687

Για την εκπαίδευση με μέγεθος παρτίδας 256 μπορούμε να δούμε τον Πίνακα 12

Epoch	Accuracy(%)
1	0.5919
2	0.7953
3	0.8311
4	0.8483
5	0.8556
6	0.8745
7	0.8844
8	0.8914
9	0.8923
10	0.9016
11	0.9057
12	0.9087
13	0.9165
14	0.9165
15	0.9175
16	0.9281
17	0.9378
18	0.9266
19	0.9342
20	0.9354
21	0.9345
22	0.9409
23	0.9413
24	0.9439
25	0.9494

Πίνακας 12 Η ακρίβεια του μοντέλου στο τέλος κάθε εποχής για μέγεθος παρτίδας 256

Όπου η διάρκεια εκπαίδευσης διήρκησε περίπου 75.110 δευτερόλεπτα και η ακρίβεια του μοντέλου σε δεδομένα που δεν είχε «δει» κατά την εκπαίδευση ήταν 0.8538

ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα

Στον Πίνακα 13 μπορούμε να δούμε μια σύγκριση της μέσης χρήσης των πόρων και για τα τρία μοντέλα για την στρατηγική των εξυπηρετητών παραμέτρων και για τα δυο μεγέθη παρτίδας

	1ο μοντέλο		2ο μοντέλο		3ο μοντέλο	
ΠΟΡΟΙ	128	256	128	256	128	256
CPU(%)	8.2	9.5	11.8	10.9	13.1	10.8
MEMORY (GB)	14.5	14.1	13.2	13.1	14	13.4
NETWORK IN (M)	91.4	79.5	65.8	43.9	75	51.1

Πίνακας 13 μέσης χρήσης των πόρων και για τα τρία μοντέλα για την στρατηγική των εξυπηρετητών παραμέτρων και για τα δυο μεγέθη παρτίδας

Για το πρώτο μοντέλο μπορούμε να δούμε ότι αν εξαιρέσουμε την μέση χρήση της κεντρικής μονάδας επεξεργασίας η εκπαίδευση με μέγεθος παρτίδας 128 έχει μεγαλύτερη μέση χρήση των πόρων. Για το δεύτερο μοντέλο επίσης η μέση χρήση των πόρων ήταν μεγαλύτερη για την εκπαίδευση με μέγεθος παρτίδας 128. Τέλος για το τρίτο μοντέλο η μέση χρήση των πόρων για την εκπαίδευση με μέγεθος παρτίδας 128 ήταν μεγαλύτερη. Η μεγαλύτερη μέση χρήση της Κεντρικής μονάδας επεξεργασίας ήταν για την εκπαίδευση του τρίτου μοντέλου και με μέγεθος παρτίδας 128. Η μεγαλύτερη μέση χρήση της μνήμης ήταν για το τρίτο μοντέλο και για μέγεθος παρτίδας 128. Τέλος η μεγαλύτερη μέση χρήση του δικτυού ήταν για την εκπαίδευση του πρώτου μοντέλου και μέγεθος παρτίδας 128

Μοντέλο	Ακρίβεια(%)	Χρόνος(sec)
1ο Μοντέλο (Mnist)	0.9804	173.895
2ο Μοντέλο (Fashion Mnist)	0.8862	121.245
3ο Μοντέλο (Kmnist)	0.8758	125.57

Πίνακας 14 Οι χρόνοι εκπαίδευσης και η ακρίβεια του κάθε μοντέλου με την στρατηγική Εξυπηρετητών Παραμέτρων και μέγεθος παρτίδας 128

Στον Πίνακα 14 μπορούμε να δούμε πόσο χρόνο διήρκησε η εκπαίδευση του κάθε μοντέλου αλλά και την ακρίβεια του χρησιμοποιώντας την στρατηγική Εξυπηρετητών παραμέτρων με μέγεθος παρτίδας 128

Μοντέλο	Ακρίβεια(%)	Χρόνος(sec)
1ο Μοντέλο (Mnist)	0.9838	107.155
2ο Μοντέλο (Fashion Mnist)	0.892	89.151
3ο Μοντέλο (Kmnist)	0.8855	82.683

Πίνακας 15 Οι χρόνοι εκπαίδευσης και η ακρίβεια του κάθε μοντέλου με την στρατηγική Εξυπηρετητών Παραμέτρων και μέγεθος παρτίδας 256

Στον Πίνακα 15 μπορούμε να δούμε πόσο χρόνο διήρκησε η εκπαίδευση του κάθε μοντέλου αλλά και την ακρίβεια του χρησιμοποιώντας την στρατηγική Εξυπηρετητών παραμέτρων με μέγεθος παρτίδας 256

Για το πρώτο μοντέλο επιτυγχάνεται καλύτερη ακρίβεια 0.9838 εκπαιδεύοντάς το μοντέλο με μέγεθος παρτίδας 256 ενώ πετυχαίνει και καλύτερο χρόνο 107,155 από την εκπαίδευση με μέγεθος παρτίδας 128.

Για το δεύτερο μοντέλο επιτυγχάνεται καλύτερη ακρίβεια 0.892 εκπαιδεύοντάς το μοντέλο με μέγεθος παρτίδας 256 ενώ πετυχαίνει και καλύτερο χρόνο 89.151 από την εκπαίδευση με μέγεθος παρτίδας 128.

Για το τρίτο μοντέλο επιτυγχάνεται καλύτερη ακρίβεια 0.8855 εκπαιδεύοντάς το μοντέλο με μέγεθος παρτίδας 256 ενώ πετυχαίνει και καλύτερο χρόνο 82.683 από την εκπαίδευση με μέγεθος παρτίδας 128.

Στον Πίνακα 16 μπορούμε να δούμε μια σύγκριση της μέσης χρήσης των πόρων και για τα τρία μοντέλα για την στρατηγική των MultiWorkerMirroredStrategy και για τα δυο μεγέθη παρτίδας

	1ο μοντέλο		2ο μοντέλο		3ο μοντέλο	
ΠΟΡΟΙ	128	256	128	256	128	256
CPU(%)	9.7	9.8	13.5	10.4	10.2	12
MEMORY (GB)	14.2	13.8	15	14.6	14	13.2
NETWORK IN (M)	91.4	79.4	49.5	40.1	69.9	51.3

Πίνακας 16 μέσης χρήσης των πόρων και για τα τρία μοντέλα για την στρατηγική MultiWorkerMirroredStrategy και για τα δυο μεγέθη παρτίδας

Για το πρώτο μοντέλο η μέση χρήση των πόρων εκτός της κεντρικής μονάδας επεξεργασίας ήταν μεγαλύτερη για την εκπαίδευση με μέγεθος παρτίδας 128 από αυτήν με μέγεθος παρτίδας 256. Για το δεύτερο μοντέλο η μέση χρήση των πόρων ήταν μεγαλύτερη για την εκπαίδευση με μέγεθος 128. Τέλος για το τρίτο μοντέλο η μέση χρήση των πόρων ήταν μεγαλύτερη. Η μεγαλύτερη μέση χρήση της Κεντρικής μονάδας Επεξεργασίας ήταν για την εκπαίδευση του δεύτερου μοντέλου με μέγεθος παρτίδας 128. Η μεγαλύτερη μέση χρήση της μνήμης ήταν για την εκπαίδευση του δεύτερου μοντέλου με μέγεθος παρτίδας 128. Η μεγαλύτερη μέση χρήση του δικτύου ήταν για την εκπαίδευση του πρώτου μοντέλου με μέγεθος παρτίδας 128

Μοντέλο	Ακρίβεια(%)	Χρόνος(sec)
1ο Μοντέλο (Mnist)	0.9804	144.703
2ο Μοντέλο (Fashion Mnist)	0.8577	259.3
3ο Μοντέλο (Kmnist)	0.8687	145.943

Πίνακας 17 Οι χρόνοι εκπαίδευσης και η ακρίβεια του κάθε μοντέλου με την στρατηγική MultiWorkerMirroredStrategy και μέγεθος παρτίδας 128

Στον Πίνακα 17 μπορούμε να δούμε πόσο χρόνο διήρκησε η εκπαίδευση του κάθε μοντέλου αλλά και την ακρίβεια του χρησιμοποιώντας την στρατηγική MultiWorkerMirroredStrategy με μέγεθος παρτίδας 128

Μοντέλο	Ακρίβεια(%)	Χρόνος(sec)
1ο Μοντέλο (Mnist)	0.978	85.64
2ο Μοντέλο (Fashion Mnist)	0.8592	202.254
3ο Μοντέλο (Kmnist)	0.8538	75.11

Πίνακας 18 Οι χρόνοι εκπαίδευσης και η ακρίβεια του κάθε μοντέλου με την στρατηγική MultiWorkerMirroredStrategy και μέγεθος παρτίδας 256

Στον Πίνακα 18 μπορούμε να δούμε πόσο χρόνο διήρκησε η εκπαίδευση του κάθε μοντέλου αλλά και την ακρίβεια του χρησιμοποιώντας την στρατηγική MultiWorkerMirroredStrategy με μέγεθος παρτίδας 256

Για το πρώτο μοντέλο η εκπαίδευση με μέγεθος παρτίδας 128 πετυχαίνει καλύτερη ακρίβεια 0.9804 αλλά με χειρότερο χρόνο 144.703 από την εκπαίδευση με μέγεθος παρτίδας 256 που είχε 85.64

Για το δεύτερο μοντέλο η εκπαίδευση με μέγεθος παρτίδας 256 πετυχαίνει καλύτερη ακρίβεια 0.8592 και πιο γρήγορη εκπαίδευση 202.254

Για το τρίτο μοντέλο η εκπαίδευση με μέγεθος παρτίδας 128 πέτυχε καλύτερη ακρίβεια 0.8687 αλλά με χειρότερο χρόνο εκπαίδευσης 145.943 όπου η εκπαίδευσε με μέγεθος παρτίδας 256 είχε 75.11

	Ακρίβεια(%)	Χρόνος(sec)
128_PS	0.9804	173.895
256_PS	0.9838	107.155
128_MW	0.9804	144.703
256_MW	0.978	85.64

Πίνακας 19 Σύγκριση ακρίβειας των δυο στρατηγικών για το πρώτο μοντέλο

Στον Πίνακα 19 μπορούμε να δούμε μια σύγκριση της εκπαίδευσης του πρώτου μοντέλου (PS = Εξυπηρετητές Παραμέτρων , MW = MultiWorkerStrategy). Για την εκπαίδευση με μέγεθος παρτίδας 128 οι δυο στρατηγικές πετυχαίνουν την ίδια ακρίβεια αλλά η στρατηγική MultiWorkerStrategy εκπαιδεύει το μοντέλο πιο γρήγορα . Για την εκπαίδευση με μέγεθος παρτίδας 256 η στρατηγική των Εξυπηρετητών Παραμέτρων πετυχαίνει λίγο καλύτερη ακρίβεια (0,9838 έναντι 0,9804) αλλά χρειάζεται 48,515 δευτερόλεπτα παραπάνω.

	Ακρίβεια(%)	Χρόνος(sec)
128_PS	0.8862	121.245
256_PS	0.892	89.151
128_MW	0.8577	259.3
256_MW	0.8592	202.254

Πίνακας 20 Σύγκριση ακρίβειας των δυο στρατηγικών για το δεύτερο μοντέλο

Στον Πίνακα 20 μπορούμε να δούμε μια σύγκριση της εκπαίδευσης του δεύτερου μοντέλου. Για την εκπαίδευση με μέγεθος παρτίδας 128 η στρατηγική των εξυπηρετητών παραμέτρων πετυχαίνει καλύτερη ακρίβεια και χρειάζεται λιγότερο χρόνο για να εκπαιδεύσει το μοντέλο. Για την εκπαίδευση με μέγεθος παρτίδας 256 πάλι η εκπαίδευση με την στρατηγική Εξυπηρετητών παραμέτρων πετυχαίνει καλύτερη ακρίβεια και χρειάζεται λιγότερο χρόνο εκπαίδευσης.

	Ακρίβεια(%)	Χρόνος(sec)
128_PS	0.8758	125.57
256_PS	0.8855	82.683
128_MW	0.8687	145.943
256_MW	0.8538	75.11

Πίνακας 21 Σύγκριση ακρίβειας των δυο στρατηγικών για το τρίτο μοντέλο

Στον Πίνακα 21 μπορούμε να δούμε μια σύγκριση της εκπαίδευσης του τρίτου μοντέλου. Για την εκπαίδευση με μέγεθος παρτίδας 128 η εκπαίδευση με την στρατηγική των εξυπηρετητών παραμέτρων πέτυχε καλύτερη ακρίβεια με καλύτερο χρόνο από την στρατηγική MultiWorkrerMirorredStrategy. Για την εκπαίδευση με μέγεθος παρτίδας 256 η εκπαίδευση με την στρατηγική των εξυπηρετητών παραμέτρων πέτυχε καλύτερη ακρίβεια με χειρότερο χρόνο από την στρατηγική MultiWorkrerMirorredStrategy.

Εν κατακλείδι Για το πρώτο μοντέλο οι στρατηγικές πετυχαίνουν την ίδια ακρίβεια 0.9804 με την στρατηγική MultiWorkerMirroredStrategy να πετυχαίνει καλύτερο χρόνο 144.703 έναντι 173.895 της στρατηγικής Εξυπηρετητών Παραμέτρων δηλαδή διαφορά 29.187 δευτερολέπτων . Για την εκπαίδευση με μέγεθος παρτίδας 256 η στρατηγική Εξυπηρετητών Παραμέτρων πετυχαίνει καλύτερη ακρίβεια 0.9838 έναντι 0.978 από την στρατηγική MultiWorkerMirroredStrategy δηλαδή μια μικρή διαφορά 0.0058 ακρίβειας αλλά σε περισσότερο χρόνο 107.155 από ότι η στρατηγική MultiWorkerMirroredStrategy 85.64 δηλαδή διαφορά 21.515 δευτερόλεπτα. Για το δεύτερο μοντέλο για την εκπαίδευση με μέγεθος παρτίδας 128 η στρατηγική Εξυπηρετητών Παραμέτρων πετυχαίνει καλύτερη ακρίβεια 0.8862 έναντι 0.8577 της στρατηγικής MultiWorkerMirroredStrategy δηλαδή διαφορά 0.0285 ακρίβειας σε καλύτερο χρόνο 121.245 έναντι 259.3 της MultiWorkerMirroredStrategy διαφορά 138.055 δευτερολέπτων. Για την εκπαίδευση με μέγεθος παρτίδας 256 η στρατηγική Εξυπηρετητών Παραμέτρων πετυχαίνει καλύτερη ακρίβεια 0.892 έναντι 0.8592 της στρατηγικής MultiWorkerMirroredStrategy δηλαδή διαφορά 0.0328 ακρίβειας σε καλύτερο χρόνο 89.151 έναντι 202.254 της MultiWorkerMirroredStrategy δηλαδή διαφορά 113.103 δευτερολέπτων . Για το τελευταίο μοντέλο και μέγεθος παρτίδας 128 η στρατηγική των εξυπηρετητών παραμέτρων πετυχαίνει καλύτερη ακρίβεια 0.8758 έναντι 0.8687 δηλαδή διαφορά 0.0071 ακρίβειας σε καλύτερο χρόνο 125.57 έναντι της MultiWorkerMirroredStrategy 145.943 δηλαδή διαφορά 20.373 δευτερολέπτων. Τέλος για την εκπαίδευση με μέγεθος παρτίδας 256 η στρατηγική εξυπηρετητών πετυχαίνει καλύτερη ακρίβεια 0.8855 έναντι 0.8538 δηλαδή διαφορά 0.0317 ακρίβειας με χειρότερο χρόνο όμως 82.683 από την στρατηγική MultiWorkerMirroredStrategy 75.11 διαφορά 7.573 δευτερολέπτων . Η στρατηγική Εξυπηρετητών παραμέτρων φαίνεται να πετυχαίνει καλύτερη ακρίβεια για το δεύτερο και τρίτο μοντέλο και σε φαίνεται να εκπαιδεύει το μοντέλο γρηγορότερα σε αυτές τις περιπτώσεις (αν εξαιρέσουμε το μέγεθος παρτίδας 256 του τρίτου μοντέλου όπου τελειώνει την εκπαιδεύσει αργότερα). Η στρατηγική MultiWorkerMirroredStrategy στο πρώτο μοντέλο πετυχαίνει ίδια ακρίβεια με την εκπαίδευση του μοντέλου με την στρατηγική των Εξυπηρετητών Παραμέτρων αλλά γρηγορότερα ενώ για την εκπαίδευσή με μέγεθος παρτίδας 256

πετυχαίνει ελαφρώς χειρότερη ακρίβεια με καλύτερο χρόνο τέτοιο ώστε να θεωρείται αμελητέα αυτή η διαφορά ακρίβειας . Μια πιθανή εξήγηση όπου τα μοντέλα που εκπαιδεύτηκαν με την στρατηγική MultiWorkerMirroredStrategy πετυχαίνουν κατά μέσο όρο χειρότερη ακρίβεια είναι ίσως ο αριθμός των δεδομένων για εκπαίδευση των μοντέλων αφού η στρατηγική Εξυπηρετητών Παραμέτρων έχει 6 εργάτες οπότε τα δεδομένα θα κατακερματιστούν σε 6 διαφορετικά κομμάτια μεγαλύτερου μεγέθους , από ότι στην στρατηγική MultiWorkerMirroredStrategy όπου τα ίδια δεδομένα θα κατακερματιστούν σε 9 διαφορετικά κομμάτια μικρότερου μεγέθους όσοι δηλαδή και οι εργάτες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] M. Li, “Scaling Distributed Machine Learning with the Parameter Server,” in *Proceedings of the 2014 International Conference on Big Data Science and Computing - BigDataScience ’14*, Beijing, China, 2014, pp. 1–1. doi: 10.1145/2640087.2644155.

[2] Q. Ho *et al.*, “More effective distributed ML via a Stale Synchronous Parallel parameter server,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, Red Hook, NY, USA, Sep. 2013, pp. 1223–1231.

[3] X. Zhao, A. An, J. Liu, and B. X. Chen, “Dynamic Stale Synchronous Parallel Distributed Training for Deep Learning.” arXiv, Aug. 16, 2019. doi: 10.48550/arXiv.1908.11848.

[4] J. Dean *et al.*, “Large Scale Distributed Deep Networks,” in *Advances in Neural Information Processing Systems*, 2012, vol. 25. Accessed: Jun. 30, 2022. [Online]. Available: https://papers.nips.cc/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html

[5] “OSP | Proceedings of the 48th International Conference on Parallel Processing.” https://dl.acm.org/doi/10.1145/3337821.3337828 (accessed Jul. 01, 2022).

[6] C.-C. Chen, C.-L. Yang, and H.-Y. Cheng, “Efficient and Robust Parallel DNN Training through Model Parallelism on Multi-GPU Platform.” arXiv, Oct. 28, 2019. doi: 10.48550/arXiv.1809.02839.

[7] “Introduction to Model Parallelism - Amazon SageMaker.” https://docs.aws.amazon.com/sagemaker/latest/dg/model-parallel-intro.html (accessed Jun. 28, 2022).

[8] “TensorFlow,” *TensorFlow*. https://www.tensorflow.org/ (accessed Jul. 09, 2022).

[9] “TensorFlow.js | Machine Learning for JavaScript Developers,” *TensorFlow*. https://www.tensorflow.org/js (accessed Jul. 09, 2022).

[10] “TensorFlow Lite | ML for Mobile and Edge Devices.” https://www.tensorflow.org/lite (accessed Jul. 09, 2022).

[11] “tf.data.Dataset | TensorFlow Core v2.9.1,” *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/data/Dataset (accessed Jul. 09, 2022).

[12] “Use a GPU | TensorFlow Core,” *TensorFlow*. https://www.tensorflow.org/guide/gpu (accessed Jul. 09, 2022).

[13] “Introduction to graphs and tf.function | TensorFlow Core,” *TensorFlow*. https://www.tensorflow.org/guide/intro_to_graphs (accessed Jul. 09, 2022).

[14] “tf.Operation | TensorFlow Core v2.9.1,” *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/Operation (accessed Jul. 09, 2022).

[15] “tf.Tensor | TensorFlow Core v2.9.1,” *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/Tensor (accessed Jul. 09, 2022).

[16] “TensorFlow graph optimization with Grappler | TensorFlow Core,” *TensorFlow*. https://www.tensorflow.org/guide/graph_optimization (accessed Jul. 09, 2022).

[17] “tf.data: Build TensorFlow input pipelines | TensorFlow Core,” *TensorFlow*. https://www.tensorflow.org/guide/data (accessed Jul. 09, 2022).

[18] "TFRecord and tf.train.Example | TensorFlow Core," *TensorFlow*. https://www.tensorflow.org/tutorials/load_data/tfrecord (accessed Jul. 09, 2022).
[19] "Protocol Buffers," *Google Developers*. https://developers.google.com/protocol-buffers (accessed Jul. 09, 2022).
[20] "Distributed training with TensorFlow | TensorFlow Core," *TensorFlow*. https://www.tensorflow.org/guide/distributed_training (accessed Jul. 09, 2022).
[21] "Parameter server training with ParameterServerStrategy | TensorFlow Core," *TensorFlow*. https://www.tensorflow.org/tutorials/distribute/parameter_server_training (accessed Jul. 09, 2022).
[22] "Apache Hadoop 3.3.1 – HDFS Architecture." https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html (accessed Mar. 02, 2022).
[23] "Home | ~okeanos IAAS." https://okeanos.grnet.gr/home/ (accessed Mar. 02, 2022).
[24] "TensorFlow Datasets," *TensorFlow*. https://www.tensorflow.org/datasets (accessed Jul. 03, 2022).
[25] "TFDS CLI | TensorFlow Datasets," *TensorFlow*. https://www.tensorflow.org/datasets/cli (accessed Jul. 03, 2022).
[26] "mnist | TensorFlow Datasets," *TensorFlow*. https://www.tensorflow.org/datasets/catalog/mnist (accessed Jul. 03, 2022).
[27] "fashion_mnist | TensorFlow Datasets." https://www.tensorflow.org/datasets/catalog/fashion_mnist (accessed Jul. 03, 2022).
[28] "kmnist | TensorFlow Datasets." https://www.tensorflow.org/datasets/catalog/kmnist (accessed Jul. 03, 2022).
[29] "tf.distribute.experimental.partitioners.MinSizePartitioner | TensorFlow Core v2.9.1." https://www.tensorflow.org/api_docs/python/tf/distribute/experimental/partitioners/MinSizePartitioner (accessed Jul. 05, 2022).
[30] "tf.data.experimental.AutoShardPolicy | TensorFlow Core v2.9.1," *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/data/experimental/AutoShardPolicy (accessed Jul. 05, 2022).