ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

<<Big Data Εργασία 2 - Αναφορά Naive Bayes Classifier>>

Τσορμπάρη Παρασκευή (ics22050)

Φουλίδης Δημήτριος (iis22026)

Ομάδα cluster:

Βιτσιώτη Μαγδαληνή ics22031

Κατζός Γεώργιος ics22074

Κρίστι Ζέφι ics22084

Παρασκευή Τσορμπάρη ics22050

Πενλόγλου Χρυσόστομος Π. ics22116

Σπαράγγης Δημήτριος ics22066

Σταύρος Αλμπανόπουλος ics22056

Φουλίδης Δημήτριος iis22026

Θεσσαλονίκη, Ιανουάριος 2025

Πίνακας περιεχομένων

1. Ο κώδικας του προβλήματος

Ο κώδικας που αναπτύχθηκε πραγματοποιεί κατανεμημένη επεξεργασία ενός μεγάλου αρχείου δεδομένων με καταγραφές αξιολογήσεων προϊόντων από χρήστες του Amazon, χρησιμοποιώντας το Spark Framework, το PySpark API για τη διαχείριση του framework με Python και τη βιβλιοθήκη επεξεργασίας φυσικής γλώσσας Spark NLP. Σκοπός είναι η κατάλληλη επεξεργασία του αρχείου εισόδου και συγκεκριμένα του κειμένου της βαθμολογίας, ώστε να πραγματοποιηθεί ανάλυση συναισθήματος με έναν απλοϊκό κατά Bayes κατηγοριοποιητή σύμφωνα με την βαθμολογία που δίνει κάθε χρήστης και να μπορεί στη συνέχεια να προβλεφθεί η βαθμολογία των χρηστών (0-2:Αρνητική, 4-5:Θετική) βάσει του κειμένου της αξιολόγησης. Αρχικά, τα δεδομένα κριτικών φορτώνονται, φιλτράρονται και χαρακτηρίζονται θετικές και αρνητικές βαθμολογίες. Στη συνέχεια, εφαρμόζονται τεχνικές προεπεξεργασίας κειμένου, όπως η αφαίρεση κενών και διπλότυπων εγγραφών, lowercasing, η εξάλειψη λέξεων που δεν προσφέρουν κάτι στην ανάλυση συναισθήματος (stop words), καθώς και η λημματοποίηση και το stemming, για τη "κανονικοποίηση" του κειμένου σε μια συγκεκριμένη μορφή. Τα επεξεργασμένα δεδομένα μετατρέπονται σε αριθμητικά χαρακτηριστικά (πλήθος εμφανίσεων λέξεων) μέσω του Count Vectorizer, ενώ ο κατηγοριοποιητής εκπαιδεύεται και αξιολογείται με προκαθορισμένους διαχωρισμούς δεδομένων εκπαίδευσης και δοκιμής (80-20% και 40-60%). Τα αποτελέσματα αξιολογούνται με δείκτες ακρίβειας (precision) και ανάκλησης (recall), παρέχοντας μια συνολική εικόνα της αποτελεσματικότητας του συστήματος.

2. Βήματα που ακολουθήθηκαν

Για την υλοποίηση του Spark σε κατανεμημένο περιβάλλον χρησιμοποιούνται εικονικές μηχανές που τρέχουν Ubuntu 22.04.3 LTS στο IaaS περιβάλλον του Okeanos knossos με 16 GB μνήμη RAM, 8 πυρήνες και 30+30 GB αποθηκευτικό χώρο.

Αρχικά, πραγματοποιείται SSH σύνδεση με την εικονική μηχανή που η ομάδα μας δημιούργησε μέσω PuTTY και αφού γίνει είσοδος και αλλαγή password ενημερώνονται τα πακέτα, πραγματοποιούνται οι απαραίτητες ρυθμίσεις και εγκαθίστανται τα απαραίτητα εργαλεία. Πιο συγκεκριμένα, εκτελέστηκαν εντολές **sudo apt update** και **sudo apt upgrade** για ενημέρωση των ήδη εγκατεστημένων πακέτων. Έπειτα, γίνεται εγκατάσταση του python package manager και των numpy και spark-nlp βιβλιοθηκών (**sudo apt install python3-pip**, **python3 -m pip install numpy**, **python3 -m pip install spark-nlp**). Δεν παραλείπεται και η εγκατάσταση της Java, αφού χρησιμοποιείται από το Hadoop και το Spark, με την εντολή **sudo apt install default-jdk**. Κατόπιν, γίνεται η παραμετροποίηση του συστήματος προσθέτοντας κλειδί passwordless SSH για την απρόσκοπτη επικοινωνία μεταξύ εμας ως master και των slaves που θα καθοριστούν στη συνέχεια (Δημιουργία SSH key με **ssh-keygen -t rsa**, προσθήκη του στο authorized_keys αρχείο και αλλαγή δικαιωμάτων του αρχείου ώστε κανείς άλλος να μη μπορεί να το επεξεργαστεί βάσει κανόνων SSH).

Μετά, γίνεται λήψη και αποσυμπίεση του Hadoop για κατανεμημένο deployment μέσω της εντολής **wget** και με την εντολή **sudo vim /etc/hosts** συμπληρώνεται η IP της δικής μας μηχανής (master) και των μηχανών που θα χρησιμοποιήσουμε κάθε φορά ως slaves, μαζί

με αναγνωριστικά ονόματα δίπλα από κάθε IP. Έτσι, μπορούμε να αναφερόμαστε στους slaves με τα ονόματα που έχουμε επιλέξει και όχι με την IP τους. Στη συνέχεια ορίζεται μεταβλητή περιβάλλοντος JAVA_HOME για να γνωρίζει το Hadoop που εγκαταστάθηκε η Java με τις εντολές **echo export JAVA_HOME=/usr/lib/jvm/default-java >> .bashrc** και **. .bashrc**. Έπειτα, εγκαθίστανται οι κανόνες firewall που υποδείχθηκαν στο εργαστήριο και σετάρεται η πρόσβαση SSH αντιγράφοντας το SSH κλειδί μας με την εντολή **ssh-copy-id -i $HOME/.ssh/id_rsa.pub ubuntu@όνομα_slave** από τον master σε κάθε slave. Προχωρούμε σε αρχικοποίηση του συστήματος αρχείων HDFS του Hadoop με την εντολή **~/hadoop/bin/hdfs namenode -format** και δημιουργείται ο φάκελος myhdfs στο προεπιλεγμένο path (/home/ubuntu/myhdfs/) που ορίζει το hdfs-site.xml στο hadoop/etc/hadoop.

Συνεχίζοντας τη ρύθμιση, γίνεται λήψη και αποσυμπίεση του Spark με **wget** στο home directory. Ο φάκελος που δημιουργείται μετονομάζεται σε "spark" για ευκολία στην αναφορά και χρήση του και στη συνέχεια προστίθενται οι απαραίτητες ρυθμίσεις περιβάλλοντος στο αρχείο .bashrc, καθορίζοντας τη μεταβλητή SPARK_HOME (**echo export SPARK_HOME=/home/ubuntu/spark >> .bashrc**), η οποία δείχνει στη διαδρομή εγκατάστασης του Spark, και η μεταβλητή PYSPARK_PYTHON (**echo export PYSPARK_PYTHON=python3 >> .bashrc**), η οποία ορίζει τη χρήση της Python 3 για το PySpark. Για να εφαρμόσουν οι αλλαγές αυτές, εκτελείται η εντολή **..bashrc**. Τέλος, ρυθμίστηκαν οι μεταβλητές περιβάλλοντος για την υποστήριξη της εκτέλεσης του Spark σε ένα κατανεμημένο περιβάλλον, με τη διαμόρφωση του αρχείου spark-env.sh (εντολή **echo export JAVA_HOME=/usr/lib/jvm/default-java >> ~/spark/conf/spark-env.sh** που υποδεικνύει στο Spark την τοποθεσία εγκατάστασης της Java και **echo export SPARK_MASTER_HOST=master >> ~/spark/conf/spark-env.sh** που καθορίζει το όνομα του master node στο cluster). Επιπλέον, έγινε επεξεργασία και του αρχείου workers στο spark/conf προσθέτοντας τα αναγνωριστικά ονόματα που θέσαμε προηγουμένως για τους hosts του cluster μας, ώστε να οριστούν οι worker κόμβοι που θα απαρτίζουν το cluster.

Επειδή κατα την ολοκλήρωση εγκατάστασης όλων των απαιτούμενων πακέτων και λόγω του μεγάλου μεγέθους του αρχείου εισόδου ο διαθέσιμος αποθηκευτικός χώρος δεν επαρκεί, πραγματοποιείται επέκταση σου. Αφού δημιουργείται ο δεύτερος δίσκος (30GB) στο περιβάλλον του Okeanos knossos, δημιουργείται νέο partition ext4 με default partition type, start και end sectors τιμές και γίνεται mount σε φάκελο /storage. Δίνονται read, write και execution δικαιώματα για αυτόν και ρυθμίζεται το αρχείο fstab που είναι υπεύθυνο για τις προσαρτήσεις των δίσκων κατά την έναρξη του συστήματος. Στο πλαίσιο αυτό, τροποποιήθηκαν οι ρυθμίσεις του Spark, ώστε να αναγνωρίζει και να αξιοποιεί και τον δεύτερο δίσκο. Η αλλαγή αφορά την επεξεργασία του αρχείου spark-defaults.conf στο /spark/conf με την εντολή **nano spark-defaults.conf** την εισαγωγή της εντολής **spark.local.dir /storage,** / ώστε τα αρχεία προσωρινής αποθήκευσης που παράγονται κατά την διάρκεια της εκτέλεσης του Spark να αποθηκεύονται στο φάκελο /storage και στο root από προεπιλογή, ακόμα και οταν δεν καθορίζω τη συγκεκριμένη παράμετρο κατά την εκτέλεση με spark-submit. Τέλος, γίνεται προσθήκη των όλων αναγνωριστικών που επιλέχθηκαν για τις μηχανές που θα συμμετέχουν στο cluster στο σχετικό configuration file του Hadoop (hadoop/etc/hadoop/workers) και του Spark (spark/conf/workers) ώστε με το spark submit να ξεκινάνε και στους slaves (με την εντολή **nano**).

Σχετικά με τη διαδικασία εκτέλεσης και έχοντας εγκαταστήσει το Hadoop και το Spark σε όλες τις εικονικές μηχανές που καθορίστηκαν για μέλη του cluster, ο master ξεκινά το Hadoop και το YARN (τον resource manager για το cluster) με τις εντολές **~/hadoop/sbin/start-dfs.sh** και **~/hadoop/sbin/start-yarn.sh**. Εκτελεί την εντολή **~/hadoop/bin/hadoop fs -mkdir /user** για τη δημιουργία ενός φακέλου στο HDFS και την εντολή **~/hadoop/bin/hdfs dfs -put Tools_and_Home_Improvement.jsonl user/ubuntu/Tools_and_Home_Improvement.jsonl** για τη προσθήκη του αρχείου εισόδου στο HDFS από το τοπικό directory στο οποίο το έχουμε αποθηκεύσει. Ακόμα, εκτελεί την **~/spark/sbin/start-all.sh** για να ξεκινήσει το cluster. Για να ξεκινήσει η κατανεμημένη εκτέλεση μένει πλέον να χρησιμοποιηθεί η σχετική εντολή **spark-submit**.

3. Χρόνοι εκτέλεσης

Χρόνοι εκτέλεσης με όλους τους hosts στο cluster, με hosts/2 και 1 host (n=6 ενεργά hosts, δείτε προβλήματα που αντιμετωπίσαμε παρακάτω), όπως εμφανίστηκαν μετά την ολοκλήρωση των εκτελέσεων στο Spark Web UI. Ο κώδικας εκτελέστηκε 3 φορές για κάθε host configuration και 3 φορές για κάθε train/test split (6+6+6 φορές σύνολο).

	Hosts	**Hosts/2**	**1 Host**
1.	40 min	1.1 h	2.8 h
2.	40 min	1.1 h	2.9 h
3.	40 min	1 h	2.8 h
4.	41 min	1 h	3.0 h
5.	42 min	1 h	2.8 h
6.	40 min	1 h	3.1 h

4. Προβλήματα που προέκυψαν και πώς αντιμετωπίστηκαν

1. **HDFS Connection Refused**: Κατά την προσπάθεια αποστολής του αρχείου εισόδου στο HDFS με την εντολή **hdfs dfs -put**, εμφανίστηκε το σφάλμα "connection refused". Για την επίλυση του προβλήματος, χρειάστηκε να σταματήσουν οι υπηρεσίες HDFS και YARN, καθώς και να εκτελεστεί η εντολή **~/hadoop/bin/hdfs namenode -format** για την εκ νέου μορφοποίηση του NameNode και την αρχικοποίηση του συστήματος αρχείων HDFS. Μετά την ολοκλήρωση αυτής της διαδικασίας, το πρόβλημα επιλύθηκε.

2. **Αδυναμία εκκίνησης Spark Workers**: Εκτελώντας την εντολή **jps**, παρατηρήθηκε ότι οι διεργασίες worker του Spark δεν ξεκινούσαν, εμποδίζοντας την κατανεμημένη εκτέλεση του προγράμματος. Μετά από διάφορες προσπάθειες με τα configurations, η λύση ήταν η διαγραφή της εικονικής μηχανής και η ρύθμιση του συστήματος από την αρχή. Μετά την επαναφορά, το πρόβλημα επιλύθηκε και το Spark λειτουργούσε κανονικά.

3. **Disk Full error σε ψευδο-κατανεμημένη εκτέλεση**: Κατά την εκτέλεση αρχικών δοκιμών σε ψευδο-κατανεμημένη λειτουργία, παρουσιάστηκε σφάλμα πως ο δίσκος ήταν γεμάτος. Κατά τη διερεύνηση του προβληματος, διαπιστώθηκε η σημασία του spark configuration spark.local.dir για τον καθορισμό του νέου δίσκου ως σημείο αποθήκευσης των προσωρινών αρχείων εκτέλεσης που απαιτούσαν χώρο, αλλά και η αναγκαιότητα καθαρισμού του δίσκου από τα κατάλοιπα προσωρινών αρχείων μετά από κάθε εκτέλεση. Χρησιμοποιώντας τις εντολές **df -h** για την εμφάνιση του διαθέσιμου χώρου στους δίσκους και την εντολή **du -h | sort -rh | head -n 10** για την αναζήτηση των 10 μεγαλύτερων αρχείων/φακέλων στο directory, εντοπίστηκε ο φάκελος spark/work με τα προσωρινά αρχεία εκτέλεσης, καθώς και φάκελοι τύπου blockmgr-*** και spark-*** στο /storage. Η διαγραφή αυτών των αρχείων μετά από κάθε εκτέλεση έλυσε το πρόβλημα του αποθηκευτικού χώρου.

4. **Access Denied κατά την είσοδο**: Το μεγαλύτερο πρόβλημα που είχαμε σχετικά με τις εκτελέσεις. Στο VM του ενός μέλους της ομάδας κατά τη διαδικασία ενός login, παρουσιάστηκε το σφάλμα "access denied", παρόλο που δίνονταν τα σωστά διαπιστευτήρια (username και password). Επιχειρήθηκε προσπάθεια εκ νέου διαγραφής της εικονικής μηχανής και ρύθμισης της, αλλά αδυνατούμε να ξεκινήσουμε νεο VM στο περιβάλλον του Okeanos knossos (Error κατά το Building). Παρά τις προσπάθειες δημιουργίας μηχανής με λιγότερους πόρους και νέες IP, το πρόβλημα παράμεινε. Σχετικά με το VM του δεύτερου μέλους της ομάδας, το building αποτύχαινε από τη πρώτη κιόλας απόπειρα δημιουργίας της μηχανής. Παρόμοιο πρόβλημα αντιμετώπισε και μέλος της ομάδας cluster. Τελικά, μετά από συνεννόηση με την ομάδα του cluster μας και άδεια από τον διδάσκοντα, αποφασίστηκε η εκτέλεση του κώδικα μας από master ενός από τα VM της ομάδας cluster, με όσους hosts είχαν παραμείνει ζωντανοί.