

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΑΡΑΧΕ HADOOP ΜΕ ΑΡΑΧΕ
SPARK ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ



ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΔΡΟΜΟΣ ΔΗΜΗΤΡΙΑΔΗΣ

ΑΡΤΑ, ΙΟΥΛΙΟΣ, 2021



ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ: ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ: ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΑΡΑΧΕ ΗΑDΟΟΡ ΜΕ ΑΡΑΧΕ SPARK
ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ**

Πρόδρομος Δημητριάδης

Επιβλέπων:

Χρήστος Γκόγκος,

Αναπληρωτής Καθηγητής

Άρτα, Ιούλιος, 2021

COMPARISON OF APACHE HADOOP WITH APACHE SPARK FOR BIG DATA PROCESSING

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 2021

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
Χρήστος Γκόγκος,
Αναπληρωτής Καθηγητής
2. Μέλος επιτροπής
Χρυσόστομος Στύλιος,
Καθηγητής
3. Μέλος επιτροπής
Αλέξανδρος Τζάλλας,
Επίκουρος Καθηγητής

Ο Προϊστάμενος του Τμήματος Ευριπίδης Γλαβάς,

Καθηγητής

Υπογραφή

© Δημητριάδης, Πρόδρομος, 2021.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Δημητριάδης, Πρόδρομος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση των προπτυχιακών σπουδών μου, θα ήθελα πρώτα απ' όλα να ευχαριστήσω την οικογένειά μου για όλη τη βοήθεια και τη στήριξη που μου παρείχαν αυτά τα χρόνια. Έπειτα, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, τον κ. Γκόγκο Χρήστο (Αναπληρωτής Καθηγητής), για τη βοήθεια και την καθοδήγηση που μου πρόσφερε, ώστε να υλοποιήσω την πτυχιακή μου εργασία. Τέλος, θα ήθελα να ευχαριστήσω και τον συμφοιτητή μου Νάστο Βασίλη για τη σημαντική βοήθεια που μου έδωσε κατά τη διάρκεια της πτυχιακής μου εργασίας.

ΠΕΡΙΛΗΨΗ

Κατά τη διάρκεια της πτυχιακή εργασίας αναλύθηκαν τα εργαλεία Apache Hadoop και Apache Spark και δημιουργήθηκαν οι εγκαταστάσεις τους για single-node και multi-node μορφή. Για τη multi-node μορφή δημιουργήθηκε μία συστάδα υπολογιστών, όπου αποτελείται από 5 υπολογιστές, σε ένα από τα εργαστήρια του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Ιωαννίνων στην Άρτα. Με τη χρήση των εργαλείων Apache Hadoop και Apache Spark, καθώς και τις βιβλιοθήκες τους για μηχανική εκμάθηση Apache Mahout και MLlib, αλλά και το Hadoop Streaming, δημιουργήθηκαν τρεις εφαρμογές. Συγκεκριμένα, μία εφαρμογή σύστασης ταινιών με τη χρήση Mahout με δεδομένα χρήστες, ταινίες, αξιολογήσεις και χρονική σήμανση, και ένα παράδειγμα με τη χρήση Hadoop Streaming για ανάλυση βαθμολογίας ταινιών με δεδομένα χρήστες, ταινίες, αξιολογήσεις και χρονική σήμανση. Τέλος, δημιουργήθηκε ακόμα μία εφαρμογή σύστασης ταινιών με δεδομένα ταινίες και κριτικές διάφορων χρηστών με τη χρήση MLlib, Collaborative Filtering και ALS (Alternating Least Squares).

Λέξεις-κλειδιά: συστάδα υπολογιστών, μηχανική εκμάθηση, ανάλυση βαθμολογίας, σύσταση ταινιών

ABSTRACT

During of my thesis, the Apache Hadoop and Apache Spark tools were analyzed and their installations for single-node and multi-node format were created. For the multi-node format, a cluster consisting of 5 computers was created, in one of the laboratories of the Department of Informatics and Telecommunications of the University of Ioannina in Arta. Using the Apache Hadoop and Apache Spark tools, as well as their Apache Mahout and MLlib machine learning libraries, as well as Hadoop Streaming, three applications were designed. Specifically, a movie recommendation system app using Mahout with user data, movies, ratings, and timestamps, and an example using Hadoop Streaming for movie ratings breakdown with user data, movies, ratings, and timestamps. Last, another movie recommendation system app for movies and reviews of various users was designed using MLlib, Collaborative Filtering and ALS (Alternating Least Squares).

Keywords: cluster, machine learning, ratings breakdown, movie recommendation system

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

<u>ΕΙΣΑΓΩΓΗ.....</u>	<u>XV</u>
<u>1 BIG DATA.....</u>	<u>1</u>
1.1 ΟΡΙΣΜΟΣ	1
1.2 ΤΑ 3 V	2
1.3 ΤΕΧΝΟΛΟΓΙΕΣ	3
<u>2 ΕΡΓΑΛΕΙΑ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΔΕΔΟΜΕΝΩΝ</u>	<u>6</u>
2.1 ΑΡΑΧΕ ΗΑDOOP	6
2.1.1 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΤΡΟΠΟΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ	7
2.1.2 ΒΑΣΙΚΑ ΥΠΟΣΥΣΤΗΜΑΤΑ ΟΙΚΟΣΥΣΤΗΜΑΤΟΣ ΗΑDOOP	7
2.1.3 ΥΠΟΛΟΙΠΑ ΥΠΟΣΥΣΤΗΜΑΤΑ ΟΙΚΟΣΥΣΤΗΜΑΤΟΣ ΗΑDOOP	12
2.2 ΑΡΑΧΕ SPARK	15
2.2.1 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΤΡΟΠΟΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ	15
2.2.2 ΤΟ STACK ΤΟΥ SPARK	17
<u>3 ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΟΥ</u>	<u>20</u>
3.1 ΕΓΚΑΤΑΣΤΑΣΗ ΑΡΑΧΕ ΗΑDOOP	20
3.1.1 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ SINGLE NODE	20
3.1.2 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ MULTI NODE	29
3.1.3 ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ	36
3.2 ΕΓΚΑΤΑΣΤΑΣΗ ΑΡΑΧΕ SPARK	40
3.2.1 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ SINGLE NODE	40
3.2.2 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ MULTI NODE	43
3.2.3 ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ	47
<u>4 ΣΥΓΚΡΙΣΗ ΗΑDOOP ΜΕ SPARK</u>	<u>53</u>
<u>5 ΕΦΑΡΜΟΓΕΣ</u>	<u>58</u>
5.1 ΕΦΑΡΜΟΓΕΣ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ ΗΑDOOP	58
5.1.1 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ ΜΑΗΟΥΤ	58
5.1.2 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ ΗΑDOOP STREAMING	63
5.2 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ SPARK	66
<u>6 ΣΥΜΠΕΡΑΣΜΑΤΑ</u>	<u>69</u>
<u>ΒΙΒΛΙΟΓΡΑΦΙΑ</u>	<u>71</u>
<u>ΠΑΡΑΡΤΗΜΑ</u>	<u>73</u>

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1 1 Big Data	1
Εικόνα 1.3 1 Operational Big Data Technologies (Πηγή: https://www.edureka.co/blog/top-big-data-technologies/)	4
Εικόνα 1.3 2 Analytical Big Data Technologies (Πηγή: https://www.edureka.co/blog/top-big-data-technologies/)	5
Εικόνα 1.3 3 Big Data Technologies (Πηγή: https://www.edureka.co/blog/top-big-data-technologies/).....	5
Εικόνα 2.1 1 Apache Hadoop	6
Εικόνα 2.1 2 Η έμπνευση του logo του Hadoop	6
Εικόνα 2.1.2 1 HDFS Architecture (Πηγή: https://phoenixnap.com/kb/apache-hadoop-architecture-explained)	8
Εικόνα 2.1.2 2 YARN Architecture (Πηγή: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html).....	10
Εικόνα 2.1.2 3 MapReduce WordCount Process (Πηγή: https://www.edureka.co/blog/mapreduce-tutorial/).....	11
Εικόνα 2.2 1 Apache Spark	15
Εικόνα 2.2.1 1 Συστήματα όπου τρέχει το Spark (Πηγή: https://spark.apache.org/)	16
Εικόνα 2.2.2 1 Το Stack του Spark (Πηγή: https://spark.apache.org/).....	17
Εικόνα 3.1.1 1 Δημιουργία χρήστη Hadoop.....	21
Εικόνα 3.1.1 2 Δημιουργία ζεύγος κλειδιών ssh και ορισμός της τοποθεσίας αποθήκευσης	21
Εικόνα 3.1.1 3 ssh localhost	22
Εικόνα 3.1.1 4 Hadoop Related Options	23
Εικόνα 3.1.1 5 Παραμετροποίηση αρχείου <code>hadoop-env.sh</code>	24
Εικόνα 3.1.1 6 Παραμετροποίηση <code>core-site.xml</code>	25
Εικόνα 3.1.1 7 Παραμετροποίηση <code>hdfs-site.xml</code>	26
Εικόνα 3.1.1 8 Παραμετροποίηση <code>mapred-site.xml</code>	27
Εικόνα 3.1.1 9 Παραμετροποίηση <code>yarn-site.xml</code>	28
Εικόνα 3.1.1 10 Εκκίνηση namenode, datanode και secondary namenodes	28
Εικόνα 3.1.1 11 Εκκίνηση resourcemanager και nodemanages.....	29
Εικόνα 3.1.1 12 Επαλήθευση πως όλα τρέχουν σωστά.....	29
Εικόνα 3.1.2 1 Παραμετροποίηση αρχείου <code>hadoop-env.sh</code>	30
Εικόνα 3.1.2 2 Ανάθεση ονομάτων master/slaves στις IP	31
Εικόνα 3.1.2 3 Παραμετροποίηση <code>bashrc</code> αρχείου	32
Εικόνα 3.1.2 4 Παραμετροποίηση <code>core-site.xml</code>	32
Εικόνα 3.1.2 5 Παραμετροποίηση <code>hdfs-site.xml</code>	33
Εικόνα 3.1.2 6 Παραμετροποίηση <code>yarn-site.xml</code>	34
Εικόνα 3.1.2 7 Παραμετροποίηση <code>mapred-site.xml</code>	34

Εικόνα 3.1.2 8 Ρύθμιση Yarn μέσα στον 1ο slave	35
Εικόνα 3.1.2 9 Ρύθμιση yarn-site.xml στον 1ο slave	35
Εικόνα 3.1.3 10 Hadoop NameNode UI Overview	36
Εικόνα 3.1.3 11 NameNode Storage & DFS Storage Types	37
Εικόνα 3.1.3 12 DataNode usage histogram	37
Εικόνα 3.1.3 13 Browse Directory	38
Εικόνα 3.1.3 14 Περιβάλλον παρακολούθησης DataNode	38
Εικόνα 3.1.3 15 Περιβάλλον παρακολούθησης εργασιών cluster	39
 Εικόνα 3.2.1 1 Επιβεβαίωση έκδοσης Java, Scala & Git	40
Εικόνα 3.2.1 2 Spark Configuration	41
Εικόνα 3.2.1 3 Εκκίνηση master server	41
Εικόνα 3.2.1 4 Εκκίνηση slave server	42
Εικόνα 3.2.1 5 spark-shell	42
Εικόνα 3.2.1 6 pyspark	43
 Εικόνα 3.2.2 1 Παραμετροποίηση αρχείου spark-env.sh	44
Εικόνα 3.2.2 2 Ανάθεση ονομάτων master/slaves στις IP	45
Εικόνα 3.2.2 3 Ορισμός των workers	46
Εικόνα 3.2.2 4 Διαμόρφωση host αρχείου	47
 Εικόνα 3.2.3 1 Περιβάλλον χρήστη και διαχείρισης Spark (start-master.sh)	48
Εικόνα 3.2.3 2 Περιβάλλον χρήστη και διαχείρισης Spark (start-slave.sh)	48
Εικόνα 3.2.3 3 Εκκίνηση worker χρησιμοποιώντας 1 πυρήνα	49
Εικόνα 3.2.3 4 Εκκίνηση worker χρησιμοποιώντας 512 MB	50
Εικόνα 3.2.3 5 Περιβάλλον χρήστη και διαχείρισης spark-shell (χωρίς να έχει εκτελεστεί κάποια εργασία)	51
Εικόνα 3.2.3 6 Περιβάλλον χρήστη και διαχείρισης spark-shell (έχοντας εκτελεστεί μία εργασία)	51
Εικόνα 3.2.3 7 Περιβάλλον χρήστη και διαχείρισης pyspark (χωρίς να έχει εκτελεστεί κάποια εργασία)	52
Εικόνα 3.2.3 8 Περιβάλλον χρήστη και διαχείρισης pyspark (έχοντας εκτελεστεί μία εργασία)	52
 Εικόνα 4 1 Hadoop VS Spark	53
 Εικόνα 5.1.1 1 Αρχιτεκτονική Μηχανής Συστάσεων (Πηγή: https://www.tutorialspoint.com/mahout/mahout_recommendation.htm)	59
Εικόνα 5.1.1 2 Hadoop Jobs (1)	61
Εικόνα 5.1.1 3 Hadoop Jobs (2)	61
Εικόνα 5.1.1 4 output.txt	62
Εικόνα 5.1.1 5 Αποτέλεσμα συστάσεων για τον χρήστη με id=4	63
 Εικόνα 5.1.2 1 Εκτέλεση RatingsBreakdown.py	64
Εικόνα 5.1.2 2 Βαθμολογίες & οι φορές όπου πραγματοποιήθηκαν οι αξιολογήσεις	64
Εικόνα 5.1.2 3 Hadoop All Applications UI (έχοντας εκτελεστεί το παράδειγμα) [1η]	64
Εικόνα 5.1.2 4 Hadoop All Applications UI (έχοντας εκτελεστεί το παράδειγμα) [2η]	65
 Εικόνα 5.2 1 Εκτέλεση προγράμματος	67
Εικόνα 5.2 2 Spark UI (έχοντας εκτελεστεί η εφαρμογή)	67

Εικόνα 5.2 3 Ταινίες που έχει δει ο χρήστης με ID=567 (i).....	68
Εικόνα 5.2 4 Ταινίες που έχει δει ο χρήστης με ID=567 (ii).....	68
Εικόνα 5.2 5 Ταινίες που προτείνει το σύστημα στον χρήστη με ID=567.....	68

ΓΛΩΣΣΑΡΙΟ

- **Framework:** Πλαίσιο όπου μπορεί να αλλάξει τη γενική λειτουργικότητα επιλεκτικά με πρόσθετο κώδικα γραμμένο από τον χρήστη, παρέχοντας έτσι λογισμικό για συγκεκριμένη εφαρμογή.
- **Scale up:** Ένα μηχανήμα υψηλών προδιαγραφών και υψηλής αξιοπιστίας με εκθετική αύξηση κόστους.
- **Virtual Machine:** Εικονική μηχανή όπου επιτρέπεται η πρόσβαση σε οποιοδήποτε λογισμικό.
- **Daemons:** Πρόγραμμα υπολογιστή που λειτουργεί ως διαδικασία παρασκηνίου, αντί να βρίσκεται υπό τον άμεσο έλεγχο ενός διαδραστικού χρήστη.
- **BigTable:** Κατανεμημένο σύστημα αποθήκευσης όπου έχει σχεδιαστεί για την αντιμετώπιση μεγάλων συνόλων δεδομένων.
- **API REST:** Διεπαφή προγραμματισμού εφαρμογών που προσαρμόζεται με τους περιορισμούς της αρχιτεκτονικής REST και επιτρέπει την αλληλεπίδραση με τις υπηρεσίες RESTful.
- **Avro:** Σύστημα σειριοποίησης δεδομένων.
- **Thrift:** Πλαίσιο επεκτάσιμης ανάπτυξης γλωσσικών υπηρεσιών. Συνδυάζει μία στοίβα λογισμικού με μία μηχανή δημιουργίας κώδικα για τη δημιουργία υπηρεσιών που λειτουργούν μεταξύ διάφορων γλωσσών (Python, Java, C++, Ruby κλπ).
- **Data Frames:** Δομή δεδομένων τύπου πίνακα διαθέσιμη σε γλώσσες προγραμματισμού, όπως python.
- **Ping (Packet Internet Groper):** Μέθοδος για τον εντοπισμό της διαθεσιμότητας και της απόδοσης ενός απομακρυσμένου πόρου του δικτύου.
- **DAG (Directed Acyclic Graph):** Κατευθυνόμενο Άκυκλο Γράφημα (γράφημα με βελάκια), όπου μετατρέπει ένα λογικό σχέδιο εκτέλεσης σε ένα σχέδιο φυσικής εκτέλεσης.
- **Script:** Σενάριο όπου επιτρέπει τον έλεγχο μίας ή περισσότερων εφαρμογών.
- **Kerberos:** Πρωτόκολλο ελέγχου ταυτότητας δικτύου υπολογιστών που λειτουργεί βάσει εισιτηρίων για να επιτρέπει στους κόμβους να επικοινωνούν μέσω ενός μη

ασφαλούς δικτύου για να αποδεικνύουν την ταυτότητά του μεταξύ τους με ασφαλή τρόπο.

- **LDAP (Lightweight: Directory Access Control):** Πρωτόκολλο ανοιχτού κώδικα προτύπου για την πρόσβαση σε υπηρεσίες καταλόγου, όπου τρέχει πάνω από το επίπεδο μεταφοράς ενός δικτύου.

ΕΙΣΑΓΩΓΗ

Σκοπός της παρούσας εργασίας είναι η κατανόηση των Big Data και η εξοικείωση με τα λογισμικά Apache Hadoop και Apache Spark. Όσοι θέλουν να ασχοληθούν με τον κλάδο των «Μεγάλων Δεδομένων» η γνώση της χρήσης των παραπάνω εργαλείων θα τους βοηθήσει σε αρκετά μεγάλο βαθμό. Η εργασία αποτελείται από 5 κεφάλαια. Τα 4 πρώτα αφορούν το θεωρητικό κομμάτι, ενώ στο 5^ο παρουσιάζονται δύο παραδείγματα και μία εφαρμογή συστάσεων με τη χρήση των παραπάνω εργαλείων.

Στο 1^ο κεφάλαιο θα αναφερθούμε στα Big Data ή αλλιώς «Μεγάλα Δεδομένα» και θα περιέχει τον ορισμό τους, τον συμβολισμό και την έννοια των 3 V καθώς και διάφορες τεχνολογίες πάνω στον κλάδο αυτόν. Έπειτα στο 2^ο κεφάλαιο περιγράφονται δύο βασικά εργαλεία μοντελοποίησης δεδομένων, το Apache Hadoop και Apache Spark, καθώς και κάποια τεχνικά χαρακτηριστικά τους μαζί με τα υποσυστήματα του οικοσυστήματος του Hadoop, το stack του Spark και τους τρόπους με τους οποίους προγραμματίζονται. Το 3^ο κεφάλαιο σχετίζεται με την εγκατάσταση και παραμετροποίηση των εργαλείων αυτών και το περιβάλλον χρήστη και διαχείρισης. Συγκεκριμένα θα δείξουμε τον τρόπο εγκατάστασης των δύο εργαλείων σε single-node και σε multi-node μορφή. Στο 4^ο κεφάλαιο γίνονται συγκρίσεις ανάμεσα στο Hadoop με το Spark και τέλος στο 5^ο κεφάλαιο θα δούμε μία εφαρμογή συστάσεων ταινιών με τη χρήση της βιβλιοθήκης Apache Mahout για Hadoop, ένα παράδειγμα ανάλυσης βαθμολογίας με τη χρήση Hadoop Streaming για Hadoop και μία εφαρμογή συστάσεων ταινιών με τη χρήση Collaborative Filtering και της βιβλιοθήκης MLlib για το PySpark.

1 BIG DATA



Εικόνα 1 1 Big Data

1.1 ΟΡΙΣΜΟΣ

Μέσω μεγάλων εταιρειών και των μέσων κοινωνικής δικτύωσης, όπως είναι το Facebook, το Twitter και το Instagram και όχι μόνο, καθημερινά παράγονται εκατομμύρια byte δεδομένων. Τα Big Data συνήθως περιλαμβάνουν σύνολα δομημένων, ημιδομημένων και αδόμητων δεδομένων με τεράστια μεγέθη τα οποία είναι πολύ μεγάλα για επεξεργασία, αποθήκευση, διαχείριση και ανάλυση από ένα μόνο υπολογιστικό σύστημα, με αποτέλεσμα τα παραδοσιακά εργαλεία να μην είναι σε θέση να χειριστούν τέτοιες ποσότητες. Γενικά μία συστάδα Μεγάλων Δεδομένων είναι ένα σύστημα πολλών υπολογιστών με αρκετά μεγάλο χώρο στο δίσκο, αλλά και συνήθως με πάρα πολύ RAM. Στα συστήματα αυτά συνήθως υπάρχουν εργαλεία όπως το Apache Hadoop, Hive, Pig, Sqoop, Spark, Impala και Kafka. Το κάθε ένα έχει τον δικό του σκοπό μέσα σε ένα Big Data flow. Στην παρούσα εργασία θα αναλυθούν τα εργαλεία Hadoop και Spark.

1.2 ΤΑ 3 V

Στα Μεγάλα Δεδομένα υπάρχουν τα γνωστά 3 V, τα οποία συμβολίζουν τον Όγκο, την Ταχύτητα και την Ποικιλομορφία.

- **Volume (Όγκος):** Ο Όγκος σχετίζεται περισσότερο με τα Μεγάλα Δεδομένα, διότι ο όγκος τους είναι αρκετά μεγάλος. Στα μέσα κοινωνικής δικτύωσης για παράδειγμα, ο όγκος αναφέρεται στην ποσότητα δεδομένων που δημιουργούνται μέσω ιστότοπων, πυλών και διαδικτυακών εφαρμογών. Το Facebook έχει δισεκατομμύρια χρήστες όπου ο καθένας από αυτούς αποθηκεύει πολλές φωτογραφίες. Το YouTube, το Twitter, το Instagram κι αυτά έχουν επίσης δισεκατομμύρια χρήστες που καθημερινά αναρτούν βίντεο, tweets και εικόνες [1]. Στα Big Data λοιπόν, ο όγκος αφορά την ανάγκη αποθήκευσης και επεξεργασίας Terabytes ακόμα και Petabytes δεδομένων. Καθώς περνάνε τα χρόνια θα υπάρχουν όλο και περισσότερες τεράστιες συλλογές. Ένα άλλο παράδειγμα είναι συστήματα σύστασης ταινιών. Το Netflix από τον Μάιο του 2019 έχει 13.612 τίτλους. Από το 2016 έχει ολοκληρώσει τη μετεγκατάστασή τους στις Amazon Web Services (AWS) με δεκάδες Petabytes δεδομένων να έχουν μεταφερθεί και υπολογίζεται ότι αποθηκεύει περίπου 105 TB δεδομένων μόνο για ένα βίντεο [2].
- **Velocity (Ταχύτητα):** Η ταχύτητα αφορά τον υψηλό ρυθμό με τον οποίο παράγονται νέα δεδομένα. Το Facebook για παράδειγμα καθημερινά έχει να διαχειριστεί μία πληθώρα δημοσιεύσεων, όπου πρέπει να τις επεξεργαστεί, να τις αρχειοθετήσει και να είναι σε θέση να τις ανακτήσει. Ένα άλλο παράδειγμα είναι οι αισθητήρες. Στην καθημερινότητά μας χρησιμοποιούμε αρκετούς αισθητήρες πια, είτε φορώντας μία έξυπνη συσκευή, όπως smartwatch και smartband, είτε έχοντας μέσα στο σπίτι μας διάφορες έξυπνες συσκευές, όπως είναι στα συστήματα φωτισμού, θέρμανσης, κλιματισμού και αρκετά ακόμα άλλα. Όλοι αυτοί οι αισθητήρες των συσκευών παράγουν συνέχεια αρκετά νέα δεδομένα και πρέπει να σε ενημερώνουν ανά τακτά χρονικά διαστήματα [1].
- **Variety (Ποικιλομορφία):** Η ποικιλομορφία αφορά τη συγκέντρωση δεδομένων από διάφορες πηγές και σε διάφορες μορφές. Ορισμένες μορφές δεδομένων είναι

φωτογραφίες, δεδομένα αισθητήρων, tweets, κρυπτογραφημένα πακέτα και πολλά άλλα ακόμη. Καθένα από αυτά είναι διαφορετικά το ένα από το άλλο και μεγάλο μέρος τους δεν είναι δομημένο, με αποτέλεσμα να μην χωράνε εύκολα σε υπολογιστικά φύλλα ή σε ορισμένες βάσεις δεδομένων [1]. Ωστόσο, υπάρχουν και κάποια δεδομένα τα οποία είναι σε δομημένη μορφή, όπως για παράδειγμα το Netflix, το οποίο συλλέγει δεδομένα όπως η ώρα της ημέρας, η διάρκεια της παρακολούθησης, η δημοτικότητα και διάφορες πληροφορίες που σχετίζονται με την αναζήτηση των χρηστών. Όλη αυτή η ποικιλομορφία δεδομένων αποτελεί τον φορέα ποικιλίας των Big Data [2].

Βέβαια, εκτός από τα γνωστά 3 V υπάρχουν ακόμα δύο, το Veracity και το Value. Το Veracity συμβολίζει την φιλαλήθεια και συγκεκριμένα αφορά τη διασφάλιση ποιότητας, αξιοπιστίας και ακρίβειας των δεδομένων. Τα δεδομένα συλλέγονται από πολλές πηγές, οπότε πρέπει ελέγχονται ώστε να μην υπάρχουν στο σύστημα μη έγκυρα δεδομένα που του προσθέτουν «θόρυβο». Το Value συμβολίζει την οικονομική αξία των δεδομένων και στο πόσο χρήσιμα είναι στη λήψη αποφάσεων. Για παράδειγμα, μία εταιρεία συλλέγοντας διάφορα δεδομένα από τους πελάτες της, μπορεί να γνωρίζει τα ενδιαφέροντάς τους ώστε να δημιουργεί τις σωστές προσφορές για τον καθένα, ακόμα και να αποφύγει κάποια απάτη [3].

1.3 ΤΕΧΝΟΛΟΓΙΕΣ

Οι Big Data Τεχνολογίες αφορούν την ανάλυση, την επεξεργασία και την εξαγωγή πληροφοριών από ένα εξαιρετικά περίπλοκο και μεγάλο σύνολο δεδομένων. Είναι απαραίτητες για την ανάλυση τεράστιων δεδομένων σε πραγματικό χρόνο και για προβλέψεις μείωσης κινδύνων. Όλες αυτές οι εργασίες δεν θα μπορούσαν να πραγματοποιηθούν με τα παραδοσιακά εργαλεία επεξεργασίας δεδομένων. Οι Big Data Τεχνολογίες χωρίζονται σε 2 κατηγορίες, στις Τεχνολογίες Μεγάλων Δεδομένων που χρησιμοποιούνται στην πράξη και στην Αναλυτική Μεγάλων Δεδομένων [4]

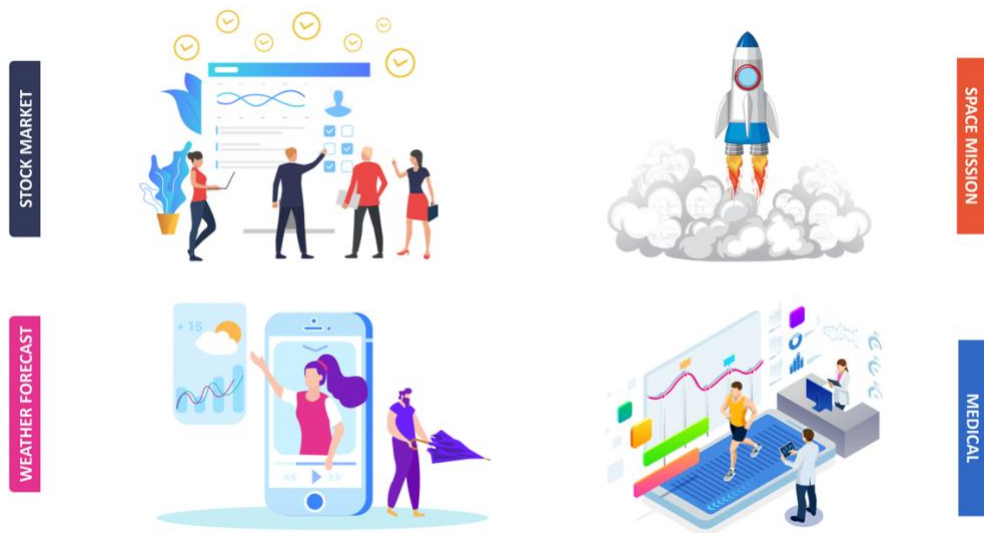
- **Τεχνολογίες Μεγάλων Δεδομένων που χρησιμοποιούνται στην πράξη:** Οι Τεχνολογίες Μεγάλων Δεδομένων που χρησιμοποιούνται στην πράξη, αφορούν τα

καθημερινά δεδομένα που δημιουργούμε. Στην παρακάτω εικόνα υπάρχουν κάποια συγκεκριμένα παραδείγματα, όπως είναι οι ηλεκτρονικές κρατήσεις εισιτηρίων, οι ηλεκτρονικές αγορές, τα δεδομένα από ιστότοπους μέσω κοινωνικής δικτύωσης και τα στοιχεία του υπαλλήλου οποιασδήποτε πολυεθνικής εταιρείας.



Εικόνα 1.3 1 Operational Big Data Technologies (Πηγή: <https://www.edureka.co/blog/top-big-data-technologies/>)

- **Αναλυτική Μεγάλων Δεδομένων:** Η Αναλυτική Μεγάλων Δεδομένων αφορά τις κρίσιμες επιχειρηματικές αποφάσεις που λαμβάνονται σε πραγματικό χρόνο με την ανάλυση Μεγάλων Δεδομένων. Στην παρακάτω εικόνα υπάρχουν κάποια παραδείγματα, όπως αυτό του χρηματιστηρίου, της διεξαγωγής διαστημικών αποστολών, τις πληροφορίες πρόγνωσης του καιρού και τα ιατρικά επίπεδα όπου μπορεί να παρακολουθείται μια συγκεκριμένη κατάσταση της υγείας των ασθενών.



Εικόνα 1.3 2 Analytical Big Data Technologies (Πηγή:<https://www.edureka.co/blog/top-big-data-technologies/>)

Κάποιες από τις Big Data Τεχνολογίες είναι οι εξής:

- Hadoop, MongoDB και RainStor, οι οποίες ανήκουν στο πεδίο Αποθήκευσης Δεδομένων.
- Presto, RapidMiner και Elasticsearch, οι οποίες ανήκουν στο πεδίο Εξόρυξης Δεδομένων.
- Spark, Kafka, Splunk, KNIME, R-Language, Flume και Blockchain, οι οποίες ανήκουν στο πεδίο Ανάλυσης Δεδομένων.
- Tableau και Plotly, οι οποίες ανήκουν στο πεδίο Οπτικοποίησης Δεδομένων.



Εικόνα 1.3 3 Big Data Technologies (Πηγή:<https://www.edureka.co/blog/top-big-data-technologies/>)

2 ΕΡΓΑΛΕΙΑ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΔΕΔΟΜΕΝΩΝ

2.1 APACHE HADOOP



Εικόνα 2.1 1 Apache Hadoop

Το Apache Hadoop είναι ένα framework ανοιχτού κώδικα το οποίο επιτρέπει στον χρήστη την αποθήκευση και επεξεργασία μεγάλων συνόλων δεδομένων σε συστάδες υπολογιστών χρησιμοποιώντας απλά μοντέλα προγραμματισμού. Είναι έτσι σχεδιασμένο για να μπορεί να κάνει scale up από μεμονωμένους διακομιστές σε χιλιάδες μηχανήματα, προσφέροντας



Εικόνα 2.1 2 Η έμπνευση του logo του Hadoop

στο καθένα υπολογιστική ισχύ και αποθηκευτικό χώρο τοπικά. Έχει σχεδιαστεί για να εντοπίζει και να χειρίζεται αποτυχίες στο επίπεδο των εφαρμογών, παρέχοντας μια διαθέσιμη υπηρεσία πάνω από μία συστάδα υπολογιστών και όχι να βασίζεται στο hardware για να το πετύχει αυτό. Επιπρόσθετα, αποτελείται από τέσσερις κύριες ενότητες, το Hadoop Common, το οποίο περιέχει εργαλεία ώστε να υποστηρίζονται οι υπόλοιπες λειτουργικές μονάδες, το HDFS (Hadoop Distributed File System), το οποίο είναι ένα καταναμημένο

σύστημα αρχείων που παρέχει πρόσβαση υψηλής απόδοσης σε δεδομένα εφαρμογών, το Hadoop YARN, το οποίο είναι ένα πλαίσιο για προγραμματισμό εργασιών και διαχείρισης πόρων σε μια συστάδα υπολογιστών και το Hadoop MapReduce, το οποίο είναι ένα σύστημα για παράλληλη επεξεργασία μεγάλων συνόλων δεδομένων. Τέλος, ιδρυτές του Hadoop είναι ο Doug Cutting και ο Mike Cafarella και συγκεκριμένα ο Doug Cutting ήταν εκείνος που σκέφτηκε το logo και το όνομα του Hadoop, εμπνευσμένος από ένα λούτρινο παιχνίδι του γιου του, έναν κίτρινο ελέφαντα που το φώναζε Hadoop [5].

2.1.1 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΤΡΟΠΟΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

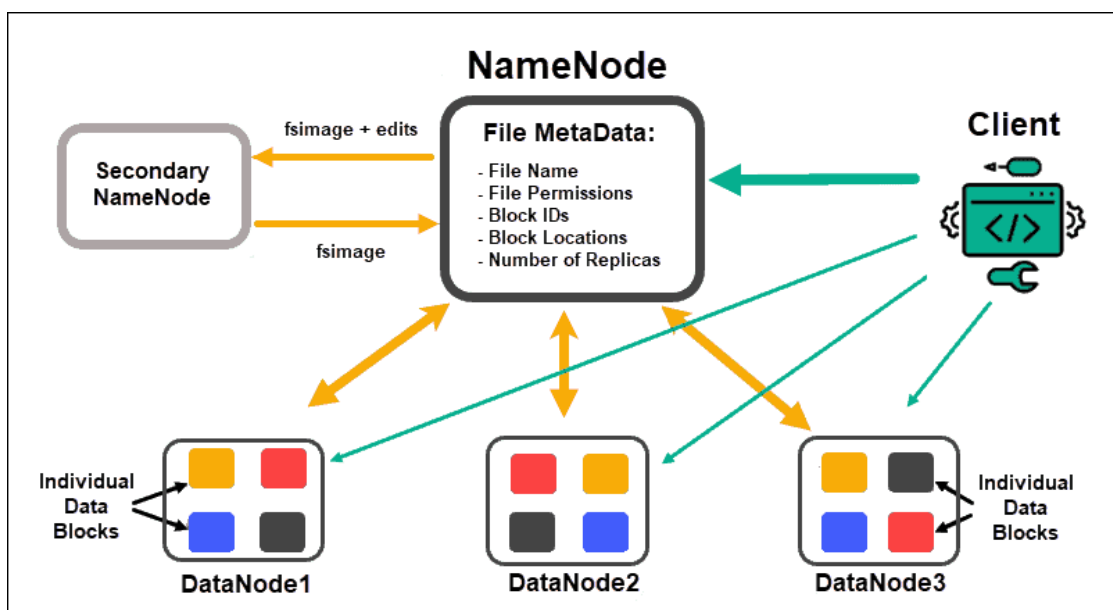
Το Apache Hadoop είναι κυρίως ανεπτυγμένο σε γλώσσα προγραμματισμού Java. Βέβαια υπάρχουν και διάφοροι άλλοι τρόποι προγραμματισμού, όπως είναι το Hadoop Streaming, Cascading, Pig, Scalding και Hive. Τα αρχεία παραμετροποίησης είναι σε μορφή XML. Η εγκατάστασή του επιτυγχάνεται σε λογισμικό Linux, Windows και MacOS, βέβαια ο συνηθέστερος τρόπος είναι σε Linux λογισμικό ή στα υπόλοιπα χρησιμοποιώντας ένα virtual machine. Υπάρχουν και διάφορες εταιρείες που προσφέρουν έτοιμο το περιβάλλον του Hadoop. Κάθε virtual machine περιέχει ένα μεγάλο αριθμό από όλα τα εργαλεία του Hadoop. Δύο τέτοιες εταιρείες είναι η Cloudera σε συνεργασία με τη Hortonworks και η MapR. Βέβαια απαιτούν αρκετά μεγάλη ποσότητα μνήμης, για παράδειγμα το Hortonworks HDP 2.6.5 έχει μέγεθος 15 GB. Μερικά στοιχεία του Apache Hadoop εμπνεύστηκαν από τις εργασίες της Google πάνω στο δικό της MapReduce αλγόριθμο και στο δικό τους σύστημα αρχείων, το Google File System.

2.1.2 ΒΑΣΙΚΑ ΥΠΟΣΥΣΤΗΜΑΤΑ ΟΙΚΟΣΥΣΤΗΜΑΤΟΣ HADOOP

Τα βασικά υποσυστήματα του οικοσυστήματος του Hadoop είναι τρία, το HDFS, το YARN και το MapReduce [6].

- **HDFS (Hadoop Distributed File System):** Το HDFS είναι ένα σύστημα κατανομημένων αρχείων που έχει σχεδιαστεί για να τρέχει πάνω από το σύστημα αρχείων ενός Linux συστήματος. Έχει πολλές ομοιότητες με τα υπάρχοντα κατανομημένα συστήματα αρχείων, ωστόσο οι διαφορές από άλλα τέτοια συστήματα

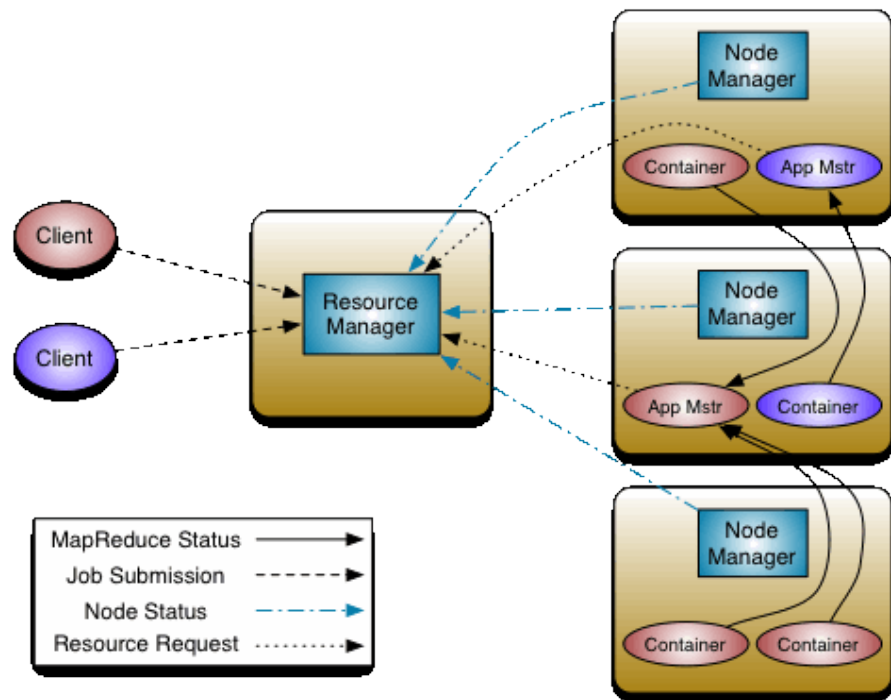
είναι σημαντικές. Το HDFS είναι το βασικότερο υποσύστημα του Hadoop, έχει εξαιρετικά μεγάλη ανοχή σε σφάλματα υλικού και έχει σχεδιαστεί για την ανάπτυξη σε υλικό χαμηλού κόστους. Επίσης, παρέχει πρόσβαση υψηλής απόδοσης σε δεδομένα εφαρμογών και είναι κατάλληλο για εφαρμογές που διαθέτουν μεγάλα σύνολα δεδομένων. Επιπρόσθετα, βοηθά στην αποθήκευση των δεδομένων σε διάφορους κόμβους και στη διατήρηση του αρχείου καταγραφής σχετικά με τα αποθηκευμένα δεδομένα. Η αρχιτεκτονική του είναι master/slave και αποτελείται από τρία στοιχεία, το NameNode, το Secondary NameNode και τα DataNodes. Το NameNode είναι ο κύριος κόμβος (master server) και δεν αποθηκεύει τα πραγματικά δεδομένα. Περιέχει μεταδεδομένα, με αποτέλεσμα να απαιτεί λιγότερη αποθήκευση και υψηλούς υπολογιστικούς πόρους. Το Secondary NameNode παρακολουθεί τις αλλαγές που πραγματοποιούνται στο NameNode και του παρέχει μία γρήγορη εκκίνηση. Όλα τα δεδομένα αποθηκεύονται στα DataNodes, με αποτέλεσμα να απαιτεί περισσότερους πόρους αποθήκευσης. Αυτά τα DataNodes είναι βασικό υλικό (όπως για παράδειγμα οι επιτραπέζιοι υπολογιστές) στο κατανεμημένο περιβάλλον και είναι ο λόγος για τον οποίο οι λύσεις με το Hadoop είναι αρκετά αποδοτικές. Τέλος, το HDFS υποστηρίζει την εργασία με πολύ μεγάλα δεδομένα χωρίζοντάς τα σε μπλοκ. Το μέγεθος των μπλοκ μπορεί να διαμορφωθεί κατά την εγκατάσταση και είναι από προεπιλογή 128 MB [7].



Εικόνα 2.1.2 1 HDFS Architecture (Πηγή: <https://phoenixnap.com/kb/apache-hadoop-architecture-explained>)

- **YARN (Yet Another Resource Negotiator):** Το YARN είναι ένα σύστημα το οποίο είναι υπεύθυνο για τον διαχωρισμό των λειτουργιών της διαχείρισης πόρων, του προγραμματισμού εργασιών καθώς και της παρακολούθησης εργασιών σε ξεχωριστά daemons. Η κατανομή των πόρων επιτυγχάνεται με το ResourceManager, ενώ το NodeManager είναι υπεύθυνο ανά σύστημα για την παρακολούθηση των πόρων (cpu, memory, disk, network) και για την αναφορά που θα στείλει στο ResourceManager προκειμένου να γίνει σωστά η κατανομή αυτών. Το NodeManager έχει ένα βασικό συστατικό, το ApplicationMaster. Το ApplicationMaster είναι υπεύθυνο για τη διαπραγμάτευση των πόρων από το ResourceManager και τη συνεργασία του με το NodeManager για την εκτέλεση και την παρακολούθηση των εργασιών. Το ResourceManager έχει δύο βασικά συστατικά, τον Scheduler και το ApplicationsManager. Ο Scheduler είναι υπεύθυνος για την κατανομή των πόρων σε διάφορες εφαρμογές που εκτελούνται εκείνη τη στιγμή. Βέβαια δεν προσφέρει καμία εγγύηση για την επανεκκίνηση αποτυχημένων εργασιών λόγω αστοχίας της εφαρμογής ή λόγω σφάλματος του υλικού, καθώς δεν είναι υπεύθυνος για την παρακολούθηση και τον εντοπισμό των εργασιών. Τέλος, εκτελεί τη λειτουργία προγραμματισμού βάσει των απαιτήσεων σε πόρους των εφαρμογών, όπου πραγματοποιείται με βάση τους πόρους που ενσωματώνουν στοιχεία όπως μνήμη, cpu, δίσκος και δίκτυο. Το ApplicationsManager είναι υπεύθυνο για την αποδοχή της εργασίας η οποία του υποβλήθηκε και για τη διαπραγμάτευση του πρώτου πλαισίου πόρων για την εκτέλεση της συγκεκριμένης εφαρμογής ApplicationMaster και την παροχή της υπηρεσίας επανεκκίνησης του ApplicationMaster σε περίπτωση αποτυχίας. Τέλος, είναι υπεύθυνο για τη

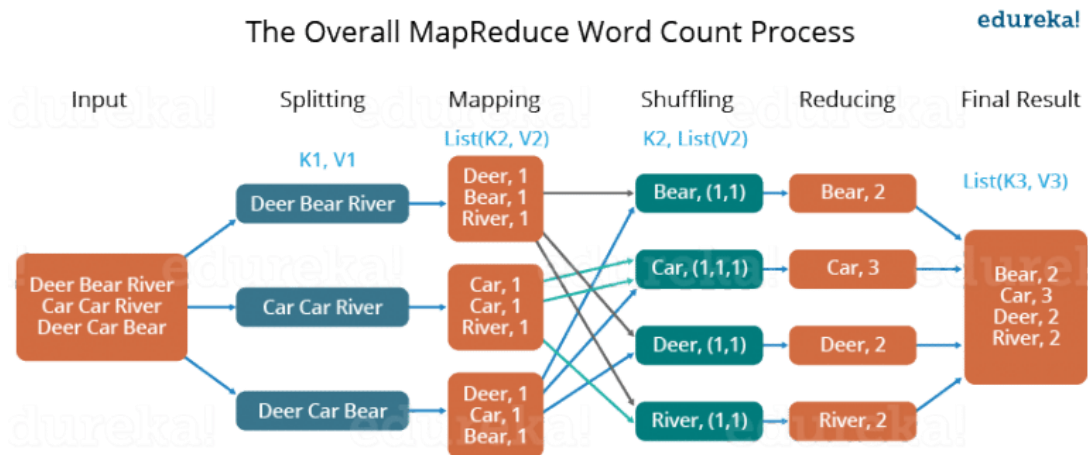
διαπραγμάτευση των κατάλληλων πόρων από τον Scheduler, ώστε να μπορεί να ανιχνεύσει την κατάστασή τους και να παρακολουθεί την πρόοδο της εκτέλεσης [8].



Εικόνα 2.1.2 2 YARN Architecture (Πηγή: <https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html>)

- MapReduce:** Το MapReduce είναι ένα framework που βοηθά να γράφονται εφαρμογές οι οποίες επεξεργάζονται τεράστιες ποσότητες δεδομένων παράλληλα σε μεγάλες συστάδες, παρέχοντας αξιοπιστία και ανεκτικότητα σε σφάλματα. Χωρίζει το σύνολο των δεδομένων εισόδου σε ανεξάρτητα κομμάτια, ώστε να δημιουργείται επεξεργασία αντιστοίχισης (Map) και μετά να ταξινομεί τις εξόδους ώστε να επιτευχθεί η εργασία της μείωσης (Reduce). Οι εργασίες εισόδου εξόδου αποθηκεύονται σε ένα σύστημα αρχείων [9]. Συγκεκριμένα μία εργασία MapReduce αποτελείται από τέσσερις φάσεις. Πρώτη φάση είναι αυτή του split, όπου τα δεδομένα χωρίζονται σε πολλούς κόμβους υπολογιστών, δεύτερη του map, όπου εκεί εφαρμόζεται μία λειτουργία αντιστοίχισης και υπάρχει το RecordReader το οποίο επεξεργάζεται κάθε εγγραφή εισόδου και δημιουργεί το αντίστοιχο ζεύγος key-value. Έπειτα, στην τρίτη φάση είναι το sort&shuffle, όπου η έξοδος των αντιστοιχίσεων ταξινομείται και κατανέμεται στους μειωτές και τέλος το reduce, όπου εφαρμόζεται η λειτουργία μείωσης στα δεδομένα και παράγεται μία έξοδος

[10]. Η όλη ιδέα του MapReduce ήταν της Google το 2004. Ένα γνωστό MR πρόβλημα είναι η καταμέτρηση λέξεων (WordCount). Σε ένα txt αρχείο βάζουμε διάφορες λέξεις και τρέχοντας τον αλγόριθμο βλέπουμε τις λέξεις του txt αρχείου και πόσες φορές υπάρχει η κάθε μία. Όπως φαίνεται και στην παρακάτω εικόνα, συγκεκριμένα διαιρούμε την είσοδο σε τρία κομμάτια και στη συνέχεια οριοθετούμε τις λέξεις σε καθέναν από τους mappers, δίνοντας μια τιμή κωδικοποίησης 1 σε κάθε λέξη. Δημιουργείται μία λίστα ζεύγους key-value όπου οι μεμονωμένες λέξεις και η τιμή τους είναι μία. Άρα για την πρώτη γραμμή, Deer Bear River, υπάρχουν τρία ζεύγη key-value ([Deer, 1] [Bear, 1] [River, 1]). Η διαδικασία του mapper παραμένει ίδια σε όλους τους κόμβους. Έπειτα περνάμε στη διαδικασία του ανακατέματος (shuffle) όπου γίνεται η ταξινόμηση και η αναδιάταξη ώστε όλες οι πλειάδες με το ίδιο κλειδί να αποστέλλονται στον αντίστοιχο reducer. Οπότε, κάθε reducer έχει ένα σύνολο κλειδιών και μία λίστα τιμών που αντιστοιχούν σε κάθε κλειδί, [Bear, (1,1)] [Car, (1,1,1)] [Deer, (1,1)] [River, (1,1)]. Κάθε reducer μετρά τις τιμές που υπάρχουν σε αυτή τη λίστα τιμών και λαμβάνει μια λίστα μορφής (1,1) για τη λέξη Bear. Μετρώντας τον αριθμό αυτών στην ίδια λίστα δίνει την τελική έξοδο ως Bear, 2. Τέλος, όλα τα ζεύγη key-value συλλέγονται και γράφονται στο αρχείο εξόδου [11].



Εικόνα 2.1.2 3 MapReduce WordCount Process (Πηγή: <https://www.edureka.co/blog/mapreduce-tutorial/>)

2.1.3 ΥΠΟΛΟΙΠΑ ΥΠΟΣΥΣΤΗΜΑΤΑ ΟΙΚΟΣΥΣΤΗΜΑΤΟΣ HADOOP

Το οικοσύστημα του Hadoop, εκτός από τα βασικά τρία υποσυστήματα HDFS, YARN και MapReduce, διαθέτει και κάποια άλλα. Κάποια από αυτά είναι το Hive, το Pig, το HBase, το Zookeeper, το Ambari, το Lucene/Solr, το Oozie, το Flume και το Sqoop [6] [12].

- **Hive:** Το Hive διευκολύνει στην αναζήτηση και τη διαχείριση μεγάλων συνόλων δεδομένων, όπου βρίσκονται σε κατανεμημένο χώρο αποθήκευσης και παρέχει έναν μηχανισμό για την αναζήτηση δεδομένων, χρησιμοποιώντας μία γλώσσα προγραμματισμού τύπου SQL, την HQL (Hive Query Language). Επιπρόσθετα, έχει δύο βασικά στοιχεία, το Hive Command Line και τους JDBC (Java Database Connectivity)/ODBC (Object Database Connectivity) drivers. Συγκεκριμένα, το Hive Command Line χρησιμοποιείται για την εκτέλεση εντολών HQL, ενώ το JDBC και το ODBC χρησιμοποιούνται για τη δημιουργία σύνδεσης από την αποθήκευση δεδομένων. Τέλος, λόγω της επεκτασιμότητας όπου έχει, εκτός ότι μπορεί να επεξεργάζεται μεγάλα σύνολα δεδομένων, μπορεί να τα επεξεργάζεται και σε πραγματικό χρόνο.
- **Pig:** Το Pig χωρίζεται σε δύο μέρη, τη γλώσσα Pig Latin (γλώσσα τύπου SQL) και το Pig Runtime για το περιβάλλον εκτέλεσης. Συγκεκριμένα, ο μεταγλωττιστής μετατρέπει τη Pig Latin σε MapReduce, παράγοντας ένα διαδοχικό σύνολο MapReduce εργασιών. Επίσης, το Pig παρέχει μία πλατφόρμα για τη δημιουργία ροής δεδομένων για ETL (Extract-Transform-Load) διαδικασίες, έτσι ώστε να επεξεργάζεται και να αναλύει μεγάλα σύνολα δεδομένων.
- **HBase:** Το HBase είναι ένα έργο ανοιχτού κώδικα, όπου δεν σχετίζεται με κάποια κατανεμημένη βάση δεδομένων. Στην ουσία είναι μία NoSQL βάση δεδομένων. Υποστηρίζει όλους του τύπους δεδομένων και χειρίζεται οτιδήποτε υπάρχει μέσα στο οικοσύστημα του Hadoop. Σχεδιάστηκε για να λειτουργεί πάνω από το HDFS και παρέχει δυνατότητες όπως το BigTable. Το HBase είναι γραμμένο σε γλώσσα προγραμματισμού Java και οι εφαρμογές του μπορούν να γραφτούν σε API REST, Avro και Thrift.

- **Zookeeper:** Το Zookeeper είναι ο συντονιστής οποιασδήποτε Hadoop εργασίας που περιλαμβάνει συνδυασμό διάφορων υπηρεσιών μέσα στο οικοσύστημα. Εξοικονομεί αρκετό χρόνο εκτελώντας συγχρονισμό δεδομένων, συντήρηση διαμόρφωσης των υπηρεσιών και ομαδοποίηση. Παρόλο που είναι μία απλή υπηρεσία, μπορεί και χρησιμοποιείται για τη δημιουργία ισχυρών λύσεων.
- **Ambari:** Το Ambari είναι ένα έργο όπου σαν στόχο έχει να κάνει πιο εύχρηστο το οικοσύστημα του Hadoop. Παρέχει τροφοδοσία στο Hadoop cluster, δίνοντας βήμα προς βήμα τη διαδικασία εγκατάστασης των υπηρεσιών του Hadoop σε μία συστάδα υπολογιστών. Διαχειρίζεται επίσης το cluster, παρέχοντας μία κεντρική υπηρεσία διαχείρισης για εκκίνηση, διακοπή και επαναδιαμόρφωση των υπηρεσιών. Τέλος, παρέχοντας έναν πίνακα εργαλείων παρακολουθεί την υγεία και την κατάσταση του cluster και με το πλαίσιο Amber Alert ειδοποιεί τον χρήστη όποτε απαιτείται προσοχή.
- **Lucene/Solr:** Το Lucene και το Solr είναι δύο υπηρεσίες για την αναζήτηση και την ευρετηρίαση στο οικοσύστημα του Hadoop. Συγκεκριμένα, το Lucene βασίζεται στη γλώσσα προγραμματισμού Java, όπου βοηθά και στον ορθογραφικό έλεγχο. Το Solr από την άλλη είναι μία πλήρης εφαρμογή που δημιουργήθηκε γύρω από το Lucene.
- **Oozie:** Το Oozie είναι ένα από τα διαθέσιμα εργαλεία του Hadoop για τον προγραμματισμό ροών εργασιών. Υπάρχουν δύο είδη εργασιών Oozie, η ροή εργασιών και ο συντονιστής. Η ροή εργασιών Oozie, πρόκειται για ένα διαδοχικό σύνολο ενεργειών που πρέπει να εκτελεστούν, ενώ ο συντονιστής Oozie, πρόκειται για τις εργασίες που ενεργοποιούνται όταν τα δεδομένα διατίθενται σε αυτό.
- **Flume:** Το Flume είναι μία υπηρεσία που βοηθά στην αξιοποίηση μη δομημένων και ημιδομημένων δεδομένων στο HDFS. Προσφέρει μία αξιόπιστη και διανεμημένη λύση ως προς τη συλλογή, τη συγκέντρωση, αλλά και τη μετακίνηση μεγάλου όγκου συνόλων δεδομένων. Επιπρόσθετα, βοηθά και στην αξιοποίηση δεδομένων συνεχούς ροής από διάφορες πηγές, όπως κυκλοφορία δικτύου, μέσα κοινωνικής δικτύωσης, email, αρχεία καταγραφής και διάφορα άλλα στο HDFS.

- **Sqoop:** Το Sqoop, όπως και το Flume, είναι μία υπηρεσία που βοηθά στην αξιοποίηση δεδομένων. Κύρια διαφορά τους είναι ότι το Sqoop μπορεί να εισάγει και να εξάγει δομημένα δεδομένα από RDBMS ή επιχειρησιακές αποθήκες δεδομένων στο HDFS και αντίστροφα.

2.2 APACHE SPARK



Εικόνα 2.2 1 Apache Spark

Το Apache Spark είναι μία μηχανή επεξεργασίας και ανάλυσης δεδομένων ανοιχτού κώδικα και χρησιμοποιείται για την αποθήκευση, την επεξεργασία και ανάλυση δεδομένων σε πραγματικό χρόνο σε διάφορες συστάδες υπολογιστών χρησιμοποιώντας απλές κατασκευές προγραμματισμού. Επεκτείνει το MapReduce και μπορεί να προγραμματιστεί σε διάφορες γλώσσες προγραμματισμού (θα αναφερθούν παρακάτω). Επίσης, περιλαμβάνει βιβλιοθήκες για μηχανική εκμάθηση, επεξεργασία ροής δεδομένων και επεξεργασία γράφων. Ξεκίνησε το 2009 ως ερευνητικό έργο στο Πανεπιστήμιο της Καλιφόρνια στο AMPLab του UC Berkley από την Matei Zaharia και το 2010 πήρε άδεια από το BSD. Το 2013 δωρίστηκε στο Apache Software Foundation και άλλαξε την άδειά του σε Apache 2.0 και τον Φεβρουάριο του 2014 έγινε πρωτεύον έργο της Apache. Τέλος, τον Νοέμβριο του 2014 η εταιρεία Databricks δημιούργησε ένα ρεκόρ στην ταξινόμηση μεγάλου όγκου χρησιμοποιώντας το Spark [13].

2.2.1 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΤΡΟΠΟΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

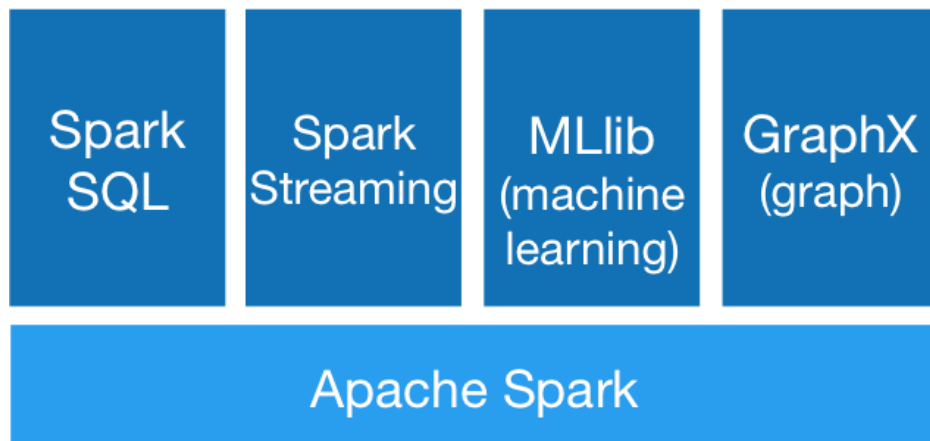
Το Spark μπορεί να προγραμματιστεί σε τέσσερις γλώσσες προγραμματισμού, Java, Python, Scala και R και ορισμένα αρχεία παραμετροποίησής του είναι σε μορφή XML. Η εγκατάστασή του επιτυγχάνεται σε λογισμικό Linux, Windows και MacOS. Μπορεί να τρέξει σε αρκετά συστήματα, όπως Hadoop, Apache Mesos, Kubernetes, standalone ή σε cloud και να έχει πρόσβαση σε διάφορες πηγές δεδομένων. Εκτός από τις βιβλιοθήκες που

έχει, αποτελείται και από το Spark Core, το οποίο είναι υπεύθυνο για την παροχή κατανεμημένης μετάδοσης εργασιών, προγραμματισμού και I/O λειτουργιών. Επίσης, το Spark Core χρησιμοποιεί ένα κατανεμημένο σύνολο δεδομένων τα RDDs (Resilient Distributed Dataset), τα οποία μοιάζουν με τους πίνακες των βάσεων δεδομένων. Επιπρόσθετα, τα RDDs υποστηρίζουν κατανεμημένη αποθήκευση δεδομένων στις μνήμες των μηχανημάτων μιας συστάδας υπολογιστών, ώστε να επιτυγχάνεται ανοχή σε σφάλματα υλικού καταγράφοντας το ιστορικό των μετασχηματισμών που εφαρμόζονται στα δεδομένα και να επιτυγχάνεται και υψηλή απόδοση με παραλληλισμό επεξεργασίας στους κόμβους των συστάδων [13].



Εικόνα 2.2.1 1 Συστήματα όπου τρέχει το Spark (Πηγή:<https://spark.apache.org/>)

2.2.2 TO STACK TOY SPARK



Εικόνα 2.2.2 1 To Stack του Spark (Πηγή: <https://spark.apache.org/>)

Το stack του Spark αποτελείται από τέσσερις βιβλιοθήκες, την Spark SQL, Spark Streaming, MLlib για Machine Learning και GraphX για γράφους. Μάλιστα αυτές οι βιβλιοθήκες μπορούν να συνδυαστούν και σε μία ίδια εφαρμογή [14].

- **Spark SQL:** Το Spark SQL χρησιμοποιείται για επεξεργασία δομημένων δεδομένων και παρέχει ένα API, το DataFrames που μπορεί να λειτουργήσει σαν κατανεμημένη μηχανή SQL ερωτήσεων, με αποτέλεσμα να είναι προσβάσιμο σε περισσότερους χρήστες. Παρέχεται επίσης και ένα module, το spark session, το οποίο χρησιμεύει στη σύνδεση και στην παραμετροποίηση του cluster δικτύου μας. Το Spark SQL τρέχει πάνω από το Spark Core, επιτρέποντας στους χρήστες να εισάγουν σχεσιακά δεδομένα από πίνακες Hive και αρχεία Parquet. Μπορεί ακόμα να συνδυαστεί και με το υπόλοιπο stack, όπως για παράδειγμα την ενσωμάτωση επεξεργασίας SQL ερωτημάτων με μηχανική εκμάθηση (Machine Learning). Επιπρόσθετα, παρέχει και το επεκτάσιμο εργαλείο βελτιστοποίησης, το Catalyst, το οποίο βοηθά στην υποστήριξη ενός ευρέος φάσματος πηγών δεδομένων και αλγορίθμων στα Big Data [15].
- **Spark Streaming:** Το Spark Streaming προστέθηκε στο Apache Spark το 2013. Είναι μία επέκταση του βασικού API του Spark, όπου χρησιμοποιείται για streaming analytics. Συγκεκριμένα χρησιμοποιείται για τη ροή νέων δεδομένων σε πραγματικό χρόνο και επιτρέπει ισχυρές διαδραστικές αναλυτικές εφαρμογές, τόσο σε ροή όσο

και σε ιστορικά δεδομένα, έχοντας και τα χαρακτηριστικά ευκολίας στη χρήση και στην ανοχή σφαλμάτων υλικού του Spark. Μπορεί και ενσωματώνεται με δημοφιλείς πηγές δεδομένων, όπως είναι το HDFS, το Flume, το Kafka και το Twitter. Πραγματοποιεί επεξεργασία δεδομένων με τη χρήση σύνθετων αλγορίθμων που εκφράζονται με λειτουργίες υψηλού επιπέδου, όπως map και reduce και έπειτα αυτά τα δεδομένα μπορούν να προωθηθούν σε συστήματα αρχείων, βάσεις δεδομένων και dashboards. Γενικά, αντί να επεξεργάζεται τα δεδομένα ροής ως μία εγγραφή κάθε φορά, τα διαχωρίζει σε μικρές παρτίδες, με αποτέλεσμα να δέχεται δεδομένα παράλληλα. Αποθηκεύει τα δεδομένα στη μνήμη των workers και στη συνέχεια εκτελούνται σύντομες εργασίες για την επεξεργασία αυτών των παρτίδων και την παραγωγή των αποτελεσμάτων στα συστήματα που αναφέρθηκαν προηγουμένως (συστήματα αρχείων, βάσεις δεδομένων, dashboards). Αυτό που καταφέρνει το Spark, είναι να αναθέτει τις εργασίες στους workers, βάσει της τοποθεσίας των δεδομένων και των διαθέσιμων πόρων, με αποτέλεσμα την καλύτερη εξισορρόπηση φορτίου και την ταχύτερη αποκατάσταση σφαλμάτων υλικού. Τέλος, κάθε παρτίδα υλικού είναι ένα RDD, ένα κατανεμημένο σύνολο δεδομένων δηλαδή, όπου επιτρέπει την επεξεργασία των δεδομένων ροής χρησιμοποιώντας οποιονδήποτε κώδικα ή βιβλιοθήκη του Spark [16].

- **MLlib (Machine Learning):** Το MLlib είναι μία επεκτάσιμη βιβλιοθήκη μηχανικής εκμάθησης (Machine Learning), η οποία παρέχει αλγόριθμους υψηλής ποιότητας και ταχύτητα εκκίνησης. Χρησιμοποιείται για έλεγχο υποθέσεων, συσταδοποίηση, ανάλυση κύριων συνιστωσών κλπ. Επιπρόσθετα, μπορεί να χρησιμοποιηθεί σε Java, Scala και Python ως μέρος των εφαρμογών του Spark. Παρέχει εργαλεία όπως [17],
 - ML Αλγόριθμοι, όπου είναι κοινοί αλγόριθμοι μάθησης (ταξινόμηση, παλινδρόμηση, ομαδοποίηση).
 - Pipelines, όπου είναι εργαλεία για κατασκευή, αξιολόγηση και συντονισμό των ML Pipelines.
 - Βοηθητικά προγράμματα, όπως γραμμική άλγεβρα, στατιστικά στοιχεία και διαχείριση δεδομένων.

Το MLlib επίσης, υποστηρίζει και αρκετούς τύπους δεδομένων, όπως για παράδειγμα τοπικά διανύσματα και πίνακες που αποθηκεύονται σε ένα μόνο μηχάνημα και κατανεμημένους πίνακες όπου υποστηρίζονται από ένα ή περισσότερα RDDs. Τα τοπικά διανύσματα και οι τοπικοί πίνακες είναι απλά μοντέλα δεδομένων, παρόλα αυτά λειτουργούν ως δημόσιες διεπαφές [18]. Τέλος, σύμφωνα με τη Databricks το MLlib είναι έως και 100 φορές πιο γρήγορο από το MapReduce [19].

- **GraphX (graph):** Το GraphX χρησιμοποιείται για την ανάλυση γράφων και έχει δημιουργηθεί πάνω από το Spark επιτρέποντας στους χρήστες να αλληλοεπιδρούν διαδραστικά. Επεκτείνει τα RDD με ένα γράφημα ανθεκτικών κατανεμημένων ιδιοτήτων. Το συγκεκριμένο γράφημα, είναι ένα κατευθυνόμενο πολύγραμμα, όπου έχει πολλές άκρες παράλληλα. Ένα χαρακτηριστικό τους είναι πως μοιράζονται την ίδια πηγή και την κορυφή προορισμού. Επιπρόσθετα, μπορούν να δημιουργήσουν και πολλαπλές σχέσεις μεταξύ των ίδιων κορυφών. Όπως τα RDD, έτσι και τα γραφήματα ιδιοτήτων είναι επίσης αμετάβλητα, κατανεμημένα και ανεκτικά σε σφάλματα υλικού. Ο χρήστης έχει τη δυνατότητα να δημιουργήσει ένα νέο γράφημα, αλλάζοντας τις τιμές ή η δομή του. Ακόμα, μπορεί να επαναχρησιμοποιήσει τα μέρη του αρχικού γραφήματος στο νέο γράφημα, με αποτέλεσμα τη μείωση του κόστους αυτής της λειτουργικής δομής δεδομένων [20]. Τέλος, το GraphX συνοδεύεται από μία βιβλιοθήκη κοινών αλγορίθμων, όπως για παράδειγμα οι PageRank και Triangle Count. Το PageRank μετρά τη σημασία κάθε κορυφής σε ένα γράφημα, υποθέτοντας ότι ένα άκρο από το x στο y αντιπροσωπεύει μία έγκριση της σημασίας του y στο x . Για παράδειγμα, αν ένας χρήστης στο Instagram ακολουθείται από πολλούς άλλους χρήστες, τότε θα κατατάσσεται σε υψηλή θέση. Το Triangle Count είναι ένας αλγόριθμος μέτρησης τριγώνων που καθορίζει τον αριθμό των τριγώνων που διέρχονται από κάθε κορυφή, παρέχοντας ένα μέτρο ομαδοποίησης. Ο υπολογισμός των αριθμών των τριγώνων του συνόλου δεδομένων πραγματοποιείται από το PageRank. Επιπρόσθετα, το Triangle Count απαιτεί τα άκρα να έχουν κανονικό προσανατολισμό και το γράφημα να χωρίζεται χρησιμοποιώντας το Graph.partitionBy. Υπάρχουν ακόμα και κάποιοι άλλοι αλγόριθμοι, όπως για παράδειγμα οι Label propagation και SVD++ [21].

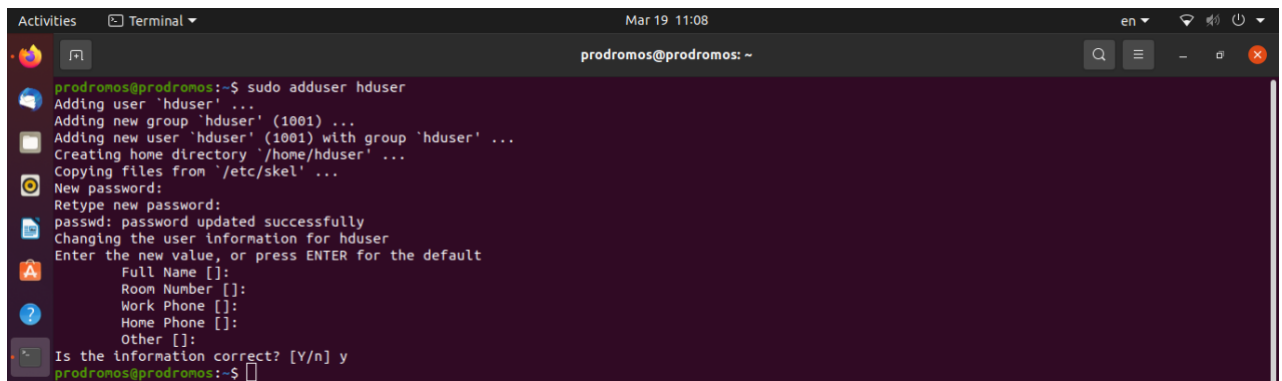
3 ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

Η εγκατάσταση του λογισμικού σε single node μορφή παρουσιάστηκε σε ένα macOS σύστημα με επεξεργαστή 1,6 GHz Dual-Core Intel Core i5, με 4 Threads και μία μνήμη RAM 8 GB 2133 MHz LPDDR3. Για την εγκατάσταση και την εκτέλεση των Apache Hadoop και Apache Spark χρησιμοποιήθηκε ένα virtual machine (μέσω του προγράμματος virtual box) με λειτουργικό Linux Ubuntu 20.04.2 LTS, τύπο 64-bit και μνήμη 4 GB. Οι εγκαταστάσεις σε multi node μορφή πραγματοποιήθηκαν από τον συνάδερφο Βασίλη Νάστο σε 5 υπολογιστές του εργαστηρίου του τμήματος, με λειτουργικό Linux Ubuntu 20.04.2 LTS, επεξεργαστή Intel Core i5-6500 CPU @ 3.20 GHz, μνήμη RAM 8 GB και δύο δίσκους, έναν με 256 GB SSD και έναν με 500 GB HDD .

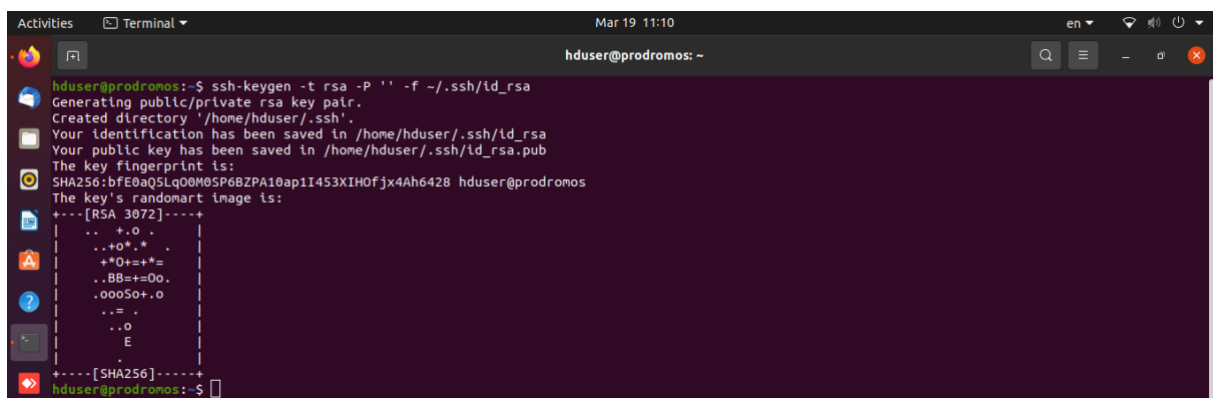
3.1 ΕΓΚΑΤΑΣΤΑΣΗ APACHE HADOOP

3.1.1 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ SINGLE NODE

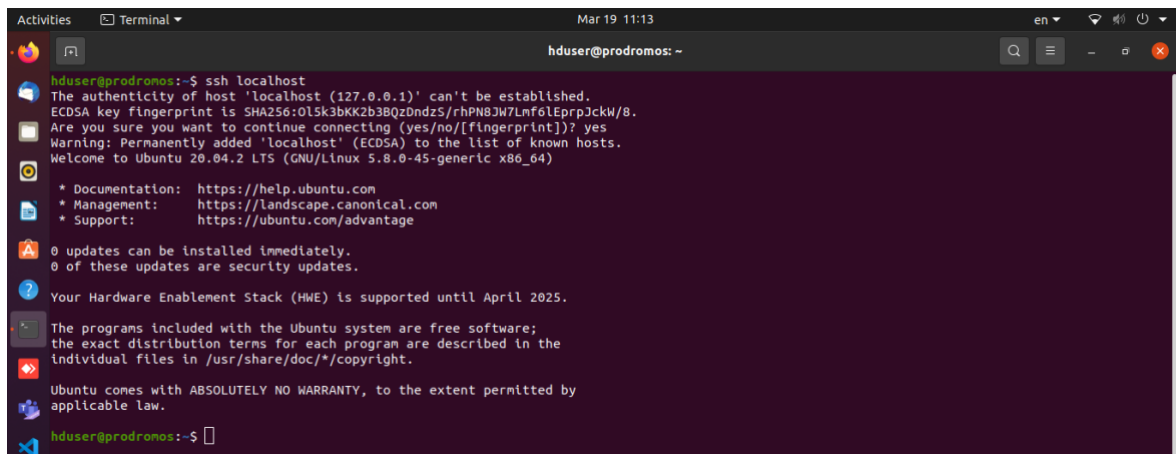
Πρώτα από όλα χρειάζεται να γίνει μία ενημέρωση του συστήματος από το terminal χρησιμοποιώντας την εντολή **“sudo apt update”**. Στη συνέχεια πρέπει να γίνει η εγκατάσταση μας JAVA OPENJDK 8 χρησιμοποιώντας την εντολή **“sudo apt install openjdk-8-jdk -y”** και με την εντολή **“java -version; Javac -version”** ελέγχουμε ότι έχει κατέβει η έκδοση που θέλουμε. Έπειτα, χρειάζεται να δημιουργήσουμε έναν Non-Root χρήστη για το περιβάλλον του Hadoop. Εγκαθιστούμε οπότε έναν OpenSSH server και έναν πελάτη με την εντολή **“sudo apt install openssh-server openssh-client -y”** και φτιάχνουμε έναν Hadoop χρήστη (στη δικιά μας περίπτωση τον ονομάσαμε hduser) με την εντολή **“sudo adduser hduser”**.



Έχοντας δημιουργήσει τον νέο χρήστη δίνοντας την εντολή **“su – hduser”** μπαίνουμε μέσα σε αυτόν. Τώρα πρέπει να ενεργοποιήσουμε ένα ssh χωρίς κωδικό πρόσβασης για τον νέο χρήστη. Όπως φαίνεται παρακάτω στην Εικόνα 3.1.1 2 δημιουργούμε ένα ζεύγος κλειδιών ssh και ορίζουμε την τοποθεσία που πρέπει να αποθηκευτεί δίνοντας την εντολή **“ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa”**.

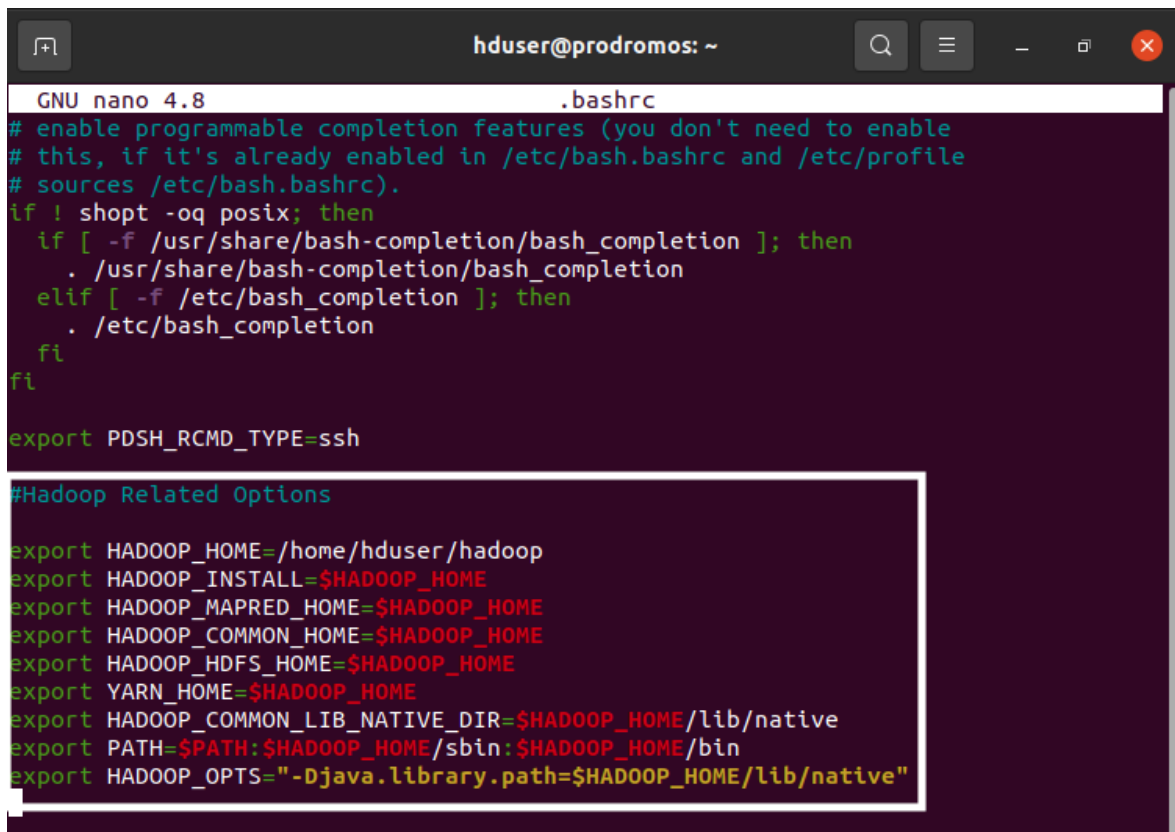


Έπειτα, με την εντολή **“cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys”** πραγματοποιούμε την εξουσιοδότηση του νέου κλειδιού στον κατάλογο ssh και με την εντολή **“chmod 0600 ~/.ssh/authorized_keys”** ορίζουμε τα δικαιώματα για τον χρήστη. Πλέον ο νέος χρήστης έχει τη δυνατότητα ssh χωρίς να χρειάζεται να εισάγει κωδικό πρόσβασης κάθε φορά. Για την επαλήθευση ότι όλα έχουν δημιουργηθεί σωστά δίνουμε την εντολή **“ssh localhost”** και αν δεν υπάρχει κάποιο λάθος θα εμφανιστεί το παρακάτω μήνυμα.



Εικόνα 3.1.1 3 ssh localhost

Μετά την παραπάνω προεργασία είμαστε έτοιμοι να εγκαταστήσουμε το Apache Hadoop. Αυτό μπορεί να γίνει με δύο τρόπους, είτε πηγαίνοντας στη σελίδα <https://hadoop.apache.org/> και από την ενότητα Download διαλέξουμε την binary έκδοση που προτιμάμε (στην προκειμένη περίπτωση το Hadoop-3.2.1) ή δίνοντας την εντολή **“wget https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz”**. Έπειτα με την εντολή **“tar xzf hadoop-3.2.1.tar.gz”** αποσυμπιέζουμε το Hadoop αρχείο. Ωστε να είναι πιο εύκολη η πλοήγηση αργότερα στους φακέλους του Hadoop δίνουμε την εντολή **“mv ./hadoop-3.2.1 ./hadoop”** και έτσι μετονομάζουμε το αρχείο σε hadoop. Για να μην υπάρχει αργότερα πρόβλημα όταν θα πάμε να εκκινήσουμε τους κόμβους, πηγαίνουμε στο κέλυφος bashrc με την εντολή **“sudo nano .bashrc”** ώστε να τροποποιήσουμε το προεπιλεγμένο rcmd pdsh σε ssh και τοποθετούμε την εντολή **“export PDSH_RCMD_TYPE=ssh”**. Βγαίνοντας από το bashrc πρέπει να δώσουμε την εντολή **“source ~/.bashrc”** ώστε να εφαρμοστούν οι αλλαγές που πραγματοποιήσαμε. Έπειτα θα φτιάξουμε το Hadoop σε Single-Node ή αλλιώς Pseudo-Distributed Mode. Μεταβαίνουμε ξανά στο bashrc με την εντολή **“sudo nano .bashrc”** και κάνουμε export τις εντολές που φαίνονται στην Εικόνα 3.1.1 4 (Δεν ξεχνάμε ότι βγαίνοντας από το bashrc δίνουμε την εντολή **“source ~/.bashrc”**).



```
GNU nano 4.8                                .bashrc
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

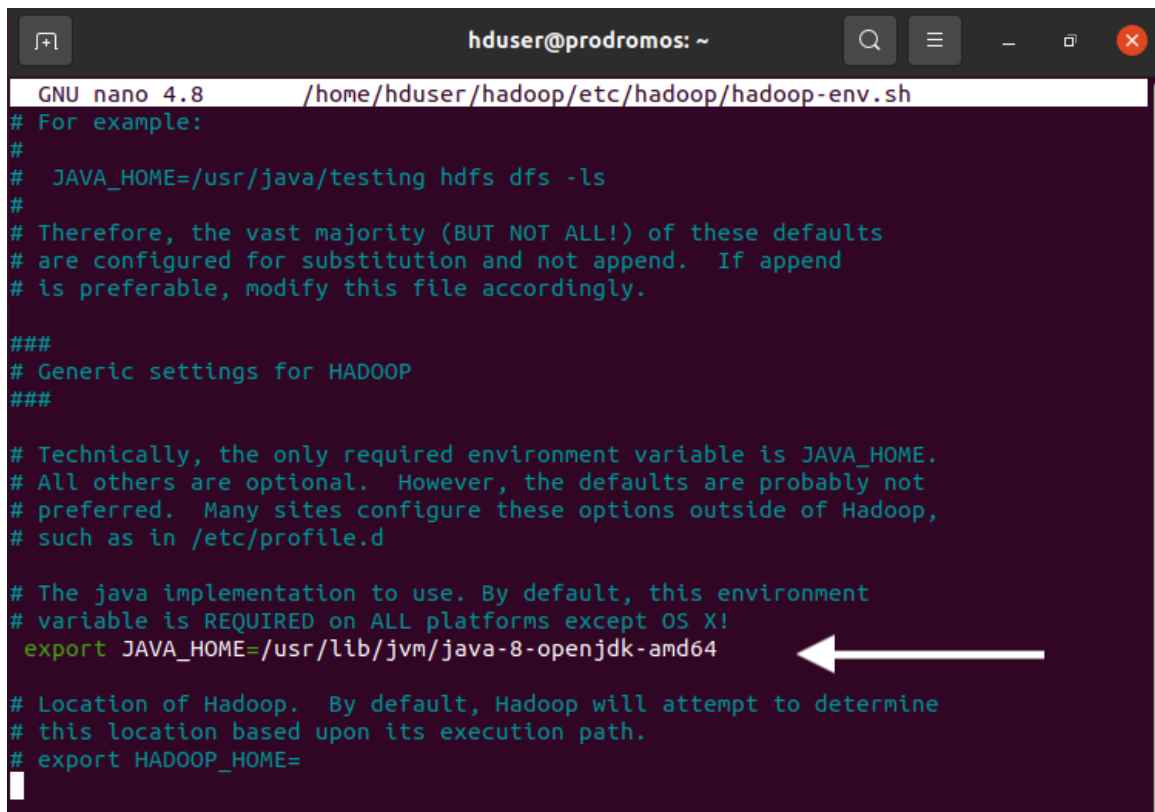
export PDSH_RCMD_TYPE=ssh

#Hadoop Related Options

export HADOOP_HOME=/home/hduser/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Εικόνα 3.1.1 4 Hadoop Related Options

Μετά πρέπει να γίνει η επεξεργασία του αρχείου `hadoop-env.sh`, διότι αυτό το αρχείο χρησιμεύει ως κύριο αρχείο για τη διαμόρφωση των ρυθμίσεων που σχετίζονται με το YARN, το HDFS και το MapReduce. Για να μπούμε στο αρχείο δίνουμε την εντολή “**sudo nano \$HADOOP_HOME/etc/hadoop/hadoop-env.sh**” και βρίσκουμε τη γραμμή που λέει **#export JAVA_HOME=** και βάζουμε την τοποθεσία της JAVA (“**export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64**”) όπως φαίνεται στην Εικόνα 3.1.1



```
GNU nano 4.8 /home/hduser/hadoop/etc/hadoop/hadoop-env.sh
# For example:
#
# JAVA_HOME=/usr/java/testing hdfs dfs -ls
#
# Therefore, the vast majority (BUT NOT ALL!) of these defaults
# are configured for substitution and not append. If append
# is preferable, modify this file accordingly.
###
# Generic settings for HADOOP
###

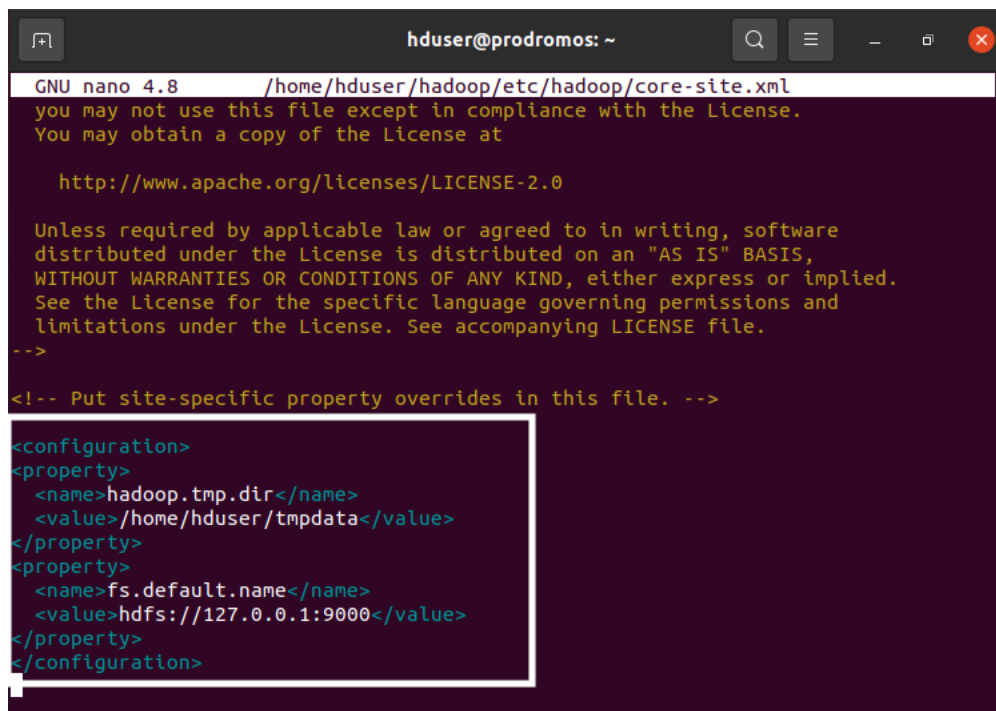
# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
# Location of Hadoop. By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=
```

Εικόνα 3.1.1 5 Παραμετροποίηση αρχείου `hadoop-env.sh`

Μόλις τελειώσουμε και με την παραμετροποίηση του `hadoop-env.sh` αρχείου πάμε να πραγματοποιήσουμε επεξεργασία στα αρχεία `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml` και `yarn-site.xml`.

- **core-site.xml:** Το αρχείο `core-site.xml` καθορίζει τις ιδιότητες του HDFS και Hadoop Core. Για να ρυθμιστεί το Hadoop σε Pseudo-Distributed λειτουργία πρέπει να καθοριστεί η διεύθυνση URL για το NameNode και τον προσωρινό κατάλογο που χρησιμοποιεί το Hadoop για τη διαδικασία του MapReduce. Δίνοντας την εντολή “`sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml`” μεταβαίνουμε μέσα στο αρχείο και τοποθετούμε τις εντολές παραμετροποίησης που φαίνονται στην Εικόνα 3.1.1 6 (σημαντικό ότι θα πρέπει να δημιουργηθεί και ένας κατάλογος `tmpdata` για τα προσωρινά δεδομένα, πραγματοποιείται με την εντολή “`mkdir tmpdata`”).



```
GNU nano 4.8 /home/hduser/hadoop/etc/hadoop/core-site.xml
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

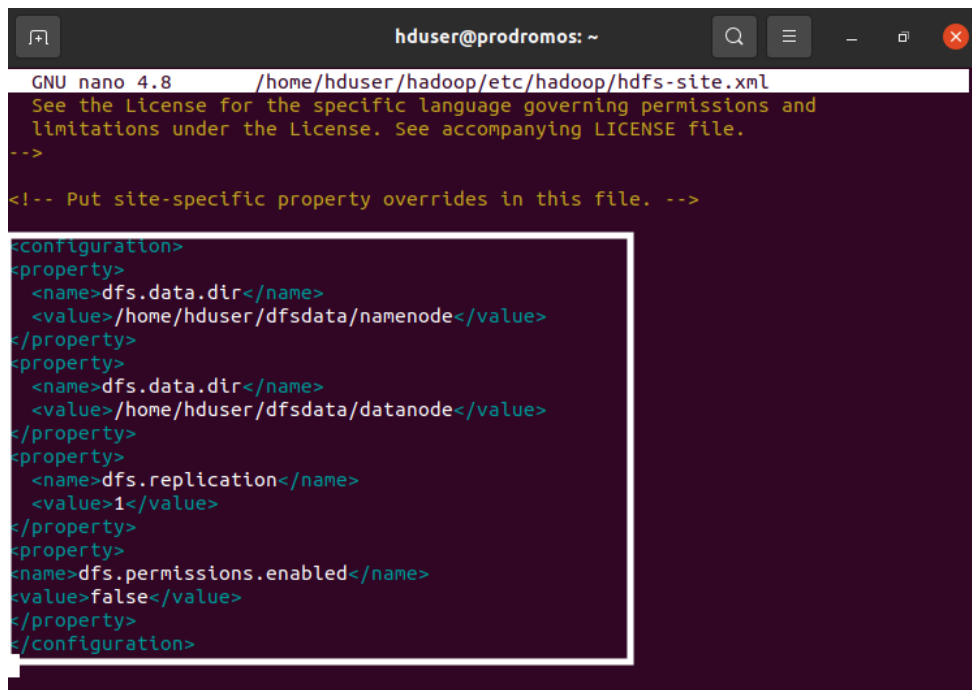
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hduser/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

Εικόνα 3.1.1 6 Παραμετροποίηση core-site.xml

- **hdfs-site.xml:** Το αρχείο hdfs-site.xml χρησιμεύει για τη θέση αποθήκευσης των μεταδεδομένων κόμβων, αρχείων fsimage και επεξεργασίας αρχείων καταγραφής. Μέσα στο αρχείο καθορίζουμε τους καταλόγους αποθήκευσης NameNode και DataNode. Δίνοντας την εντολή “**sudo nano \$HADOOP_HOME/etc/hadoop/hdfs-site.xml**” μεταβαίνουμε μέσα στο αρχείο και τοποθετούμε τις εντολές παραμετροποίησης που φαίνονται στην Εικόνα 3.1.1 7 (σημαντικό ότι θα πρέπει να δημιουργηθούν και δύο κατάλογοι dfsdata, ένα για το namenode και ένα για το datanode).



```
GNU nano 4.8 /home/hduser/hadoop/etc/hadoop/hdfs-site.xml
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

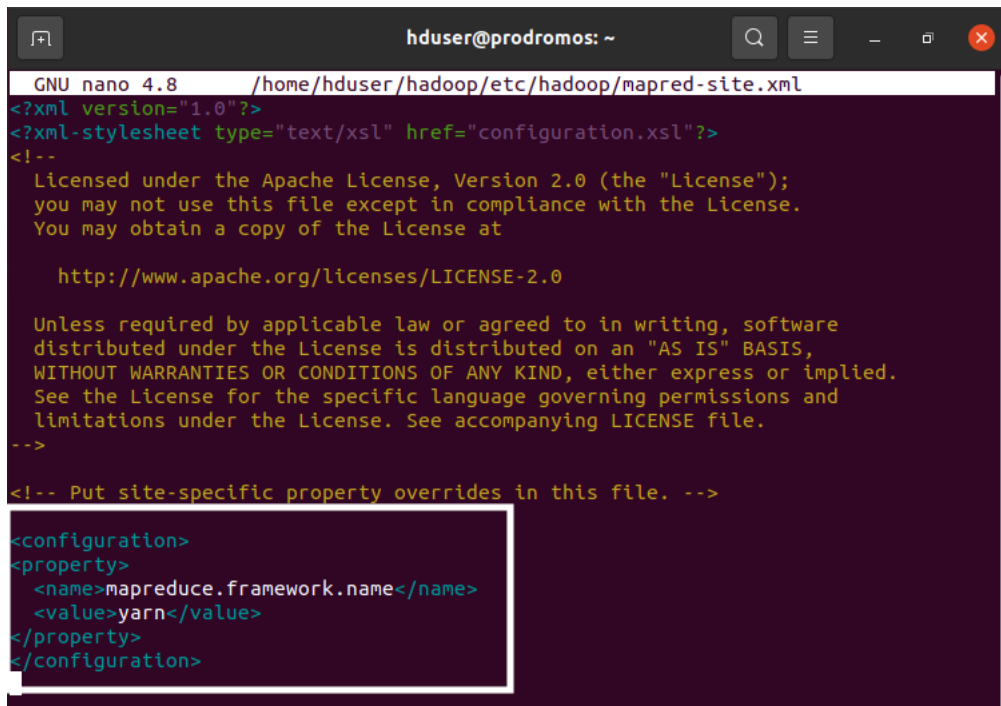
<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hduser/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hduser/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
<name>dfs.permissions.enabled</name>
<value>>false</value>
</property>
</configuration>
```

Εικόνα 3.1.1 7 Παραμετροποίηση hdfs-site.xml

- **mapred-site.xml:** Για την επεξεργασία του αρχείου mapred-site.xml χρησιμοποιούμε την εντολή “sudo nano \$HADOOP_HOME/etc/hadoop/mapred-

site.xml” και προσθέτουμε τις εντολές παραμετροποίησης της Εικόνας 3.1.1 8 ώστε να αλλάξει η προεπιλεγμένη τιμή ονόματος πλαισίου MapReduce σε νήματα.



```
GNU nano 4.8 /home/hduser/hadoop/etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl">
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

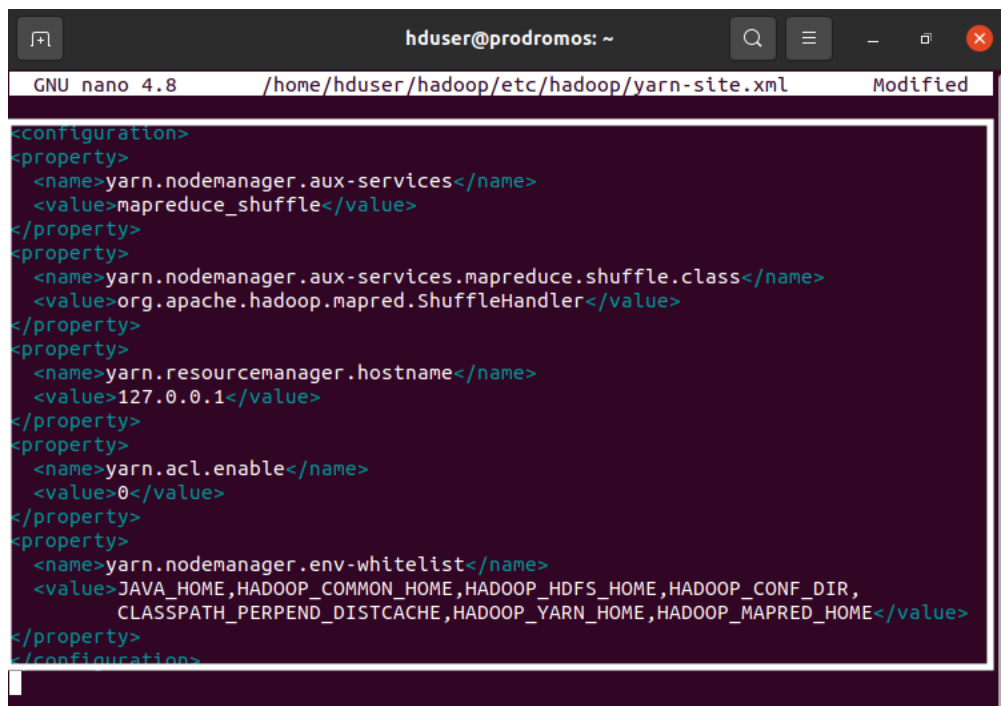
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```

Εικόνα 3.1.1 8 Παραμετροποίηση mapred-site.xml

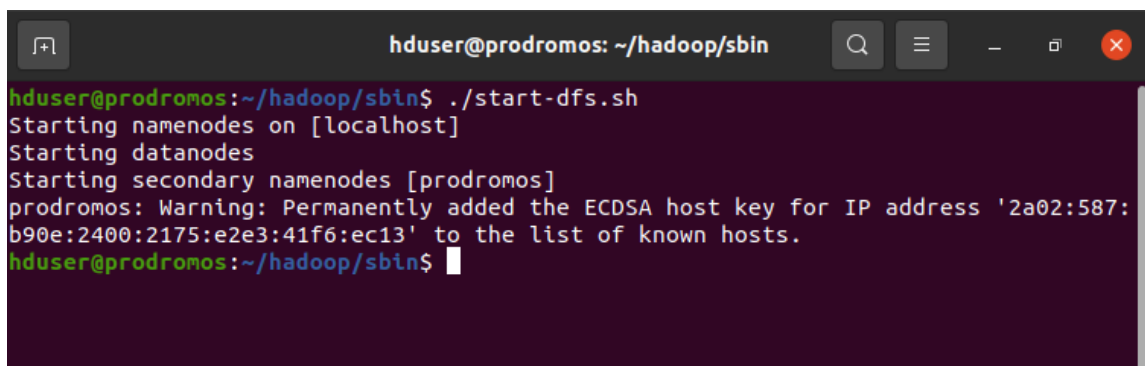
- **yarn-site.xml:** Το αρχείο yarn-site.xml χρησιμοποιείται για τον καθορισμό ρυθμίσεων σχετικά με το YARN. Περιέχει διαμορφώσεις για το NodeManager, Resource Manager, Containers και Application Master. Για την επεξεργασία του χρησιμοποιούμε την εντολή “**sudo nano \$HADOOP_HOME/etc/hadoop/yarn-site.xml**” και προσθέτουμε τις εντολές παραμετροποίησης της Εικόνας 3.1.1 9.



```
GNU nano 4.8 /home/hduser/hadoop/etc/hadoop/yarn-site.xml Modified
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,
    CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
</configuration>
```

Εικόνα 3.1.1 9 Παραμετροποίηση yarn-site.xml

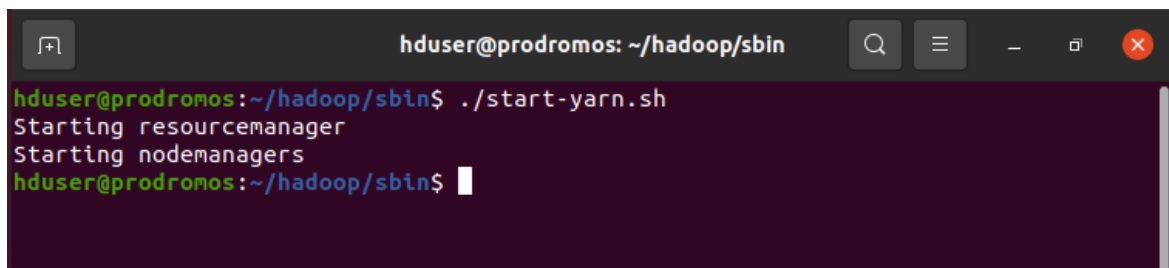
Μετά τις παραμετροποιήσεις των παραπάνω αρχείων πρέπει να γίνει μία μορφοποίηση για το NameNode, πριν ξεκινήσει τις υπηρεσίες του Hadoop για πρώτη φορά. Αυτό επιτυγχάνεται με την εντολή “**hdfs namenode -format**”. Έπειτα πρέπει να κατευθυνθούμε στον φάκελο `hadoop/sbin` και να εκτελέσουμε την εντολή “**./start-dfs.sh**” ώστε να ξεκινήσει το NameNode και το DataNode.



```
hduser@prodromos: ~/hadoop/sbin
hduser@prodromos:~/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [prodromos]
prodromos: Warning: Permanently added the ECDSA host key for IP address '2a02:587:
b90e:2400:2175:e2e3:41f6:ec13' to the list of known hosts.
hduser@prodromos:~/hadoop/sbin$
```

Εικόνα 3.1.1 10 Εκκίνηση namenode, datanode και secondary namenodes

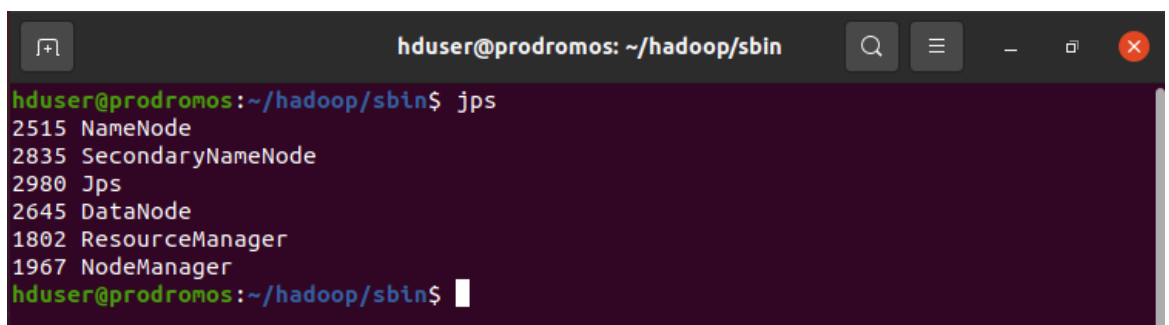
Μόλις ξεκινήσουν και τρέχουν οι namenode, datanode και secondary namenode, δίνουμε την εντολή “**./start-yarn.sh**” ώστε να ξεκινήσουν και τα resourcemanager και nodemanagers του YARN.

A terminal window titled 'hduser@prodromos: ~/hadoop/sbin' with search, menu, and window control icons. The command './start-yarn.sh' has been executed, resulting in the output: 'Starting resourcemanager' and 'Starting nodemanagers'. The prompt is now 'hduser@prodromos:~/hadoop/sbin\$' with a cursor.

```
hduser@prodromos:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hduser@prodromos:~/hadoop/sbin$
```

Εικόνα 3.1.1 11 Εκκίνηση resourcemanager και nodemanages

Τέλος, πληκτρολογώντας την εντολή “**jps**” βλέπουμε πως όλα τρέχουν σωστά και δεν υπάρχει κανένα λάθος.

A terminal window titled 'hduser@prodromos: ~/hadoop/sbin' with search, menu, and window control icons. The command 'jps' has been executed, listing several Java processes: NameNode (PID 2515), SecondaryNameNode (PID 2835), Jps (PID 2980), DataNode (PID 2645), ResourceManager (PID 1802), and NodeManager (PID 1967). The prompt is now 'hduser@prodromos:~/hadoop/sbin\$' with a cursor.

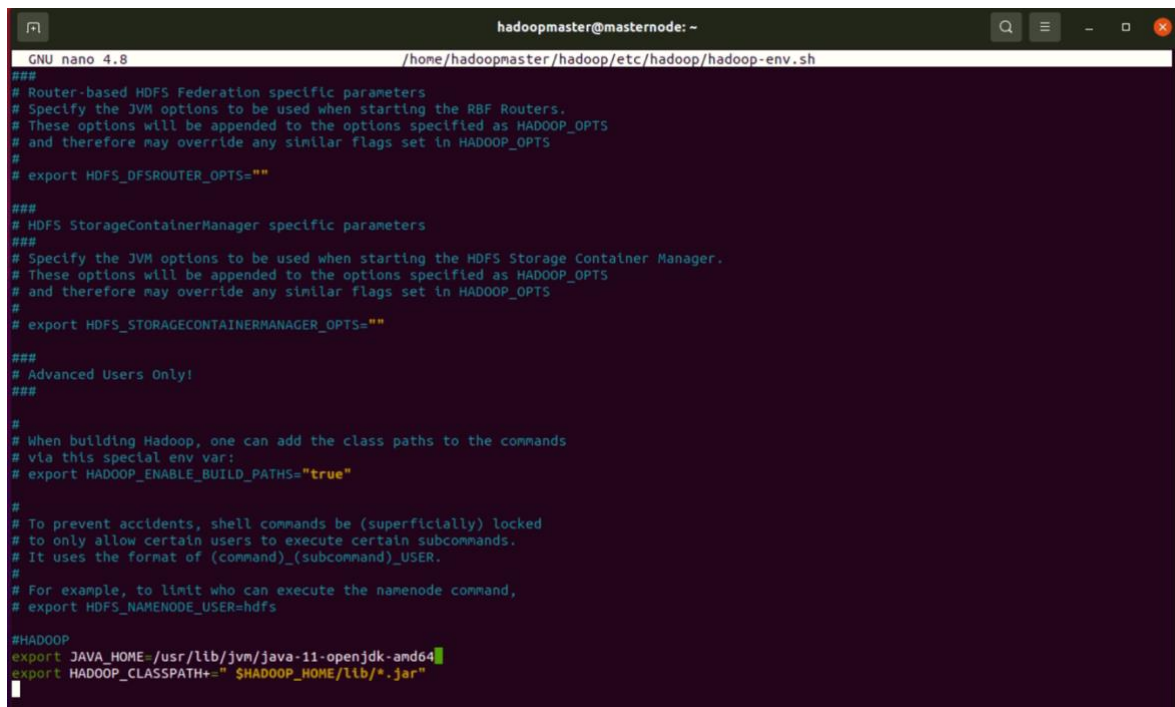
```
hduser@prodromos:~/hadoop/sbin$ jps
2515 NameNode
2835 SecondaryNameNode
2980 Jps
2645 DataNode
1802 ResourceManager
1967 NodeManager
hduser@prodromos:~/hadoop/sbin$
```

Εικόνα 3.1.1 12 Επαλήθευση πως όλα τρέχουν σωστά

3.1.2 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ MULTI NODE

Αρχικά με την εντολή “**sudo apt install ssh**” εγκαθιστούμε το ssh και με την εντολή “**sudo apt install pdsh**” εγκαθιστούμε το pdsh. Για να μην υπάρχει αργότερα πρόβλημα όταν θα πάμε να εκκινήσουμε τους κόμβους, πηγαίνουμε στο κέλυφος bashrc με την εντολή “**sudo nano .bashrc**” ώστε να τροποποιήσουμε το προεπιλεγμένο rcmd pdsh σε ssh και τοποθετούμε την εντολή “**export PDSH_RCMD_TYPE=ssh**”. Βγαίνοντας από το bashrc πρέπει να δώσουμε την εντολή “**source ~/.bashrc**” ώστε να εφαρμοστούν οι αλλαγές που πραγματοποιήσαμε. Για να δημιουργήσουμε μία σύνδεση χωρίς να χρειάζεται να βάζουμε συνέχεια κάποιον κωδικό πρόσβασης δημιουργούμε ένα ζεύγος κλειδιών ssh και ορίζουμε την τοποθεσία που πρέπει να αποθηκευτεί δίνοντας την εντολή “**ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa**”. Έπειτα, με την εντολή “**cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys**” πραγματοποιούμε την εξουσιοδότηση του νέου κλειδιού στον κατάλογο ssh. Για την επαλήθευση ότι όλα έχουν δημιουργηθεί σωστά δίνουμε την εντολή “**ssh localhost**”. Μετά

πρέπει να κάνουμε έναν έλεγχο εάν υπάρχει η java, αυτό γίνεται με την εντολή **“java -version; javac -version”**, αν δεν υπάρχει την εγκαθιστούμε με την εντολή **“sudo apt install openjdk-11-jdk”**. Τώρα είμαστε έτοιμοι να κατεβάσουμε το Hadoop αρχείο. Αυτό μπορεί να γίνει με δύο τρόπους, είτε πηγαίνοντας στη σελίδα <https://hadoop.apache.org/> και από την ενότητα Download διαλέξουμε την binary έκδοση που προτιμάμε (στην προκειμένη περίπτωση το Hadoop-3.2.1) ή δίνοντας την εντολή **“sudo wget http://apache.cs.utah.edu/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz”**. Έπειτα με την εντολή **“tar xzf hadoop-3.2.1.tar.gz”** αποσυμπιέζουμε το hadoop αρχείο. Ωστε να είναι πιο εύκολη η πλοήγηση αργότερα στους φακέλους του Hadoop δίνουμε την εντολή **“mv ./hadoop-3.2.1 ./hadoop”** και έτσι μετονομάζουμε το αρχείο σε hadoop. Επόμενο βήμα, είναι να πάμε στο περιβάλλον του hadoop, αυτό πραγματοποιείται με την εντολή **“nano ~/hadoop/etc/hadoop/hadoop-env.sh”** και να ορίσουμε το Java Home, με την εντολή **“export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/”**.



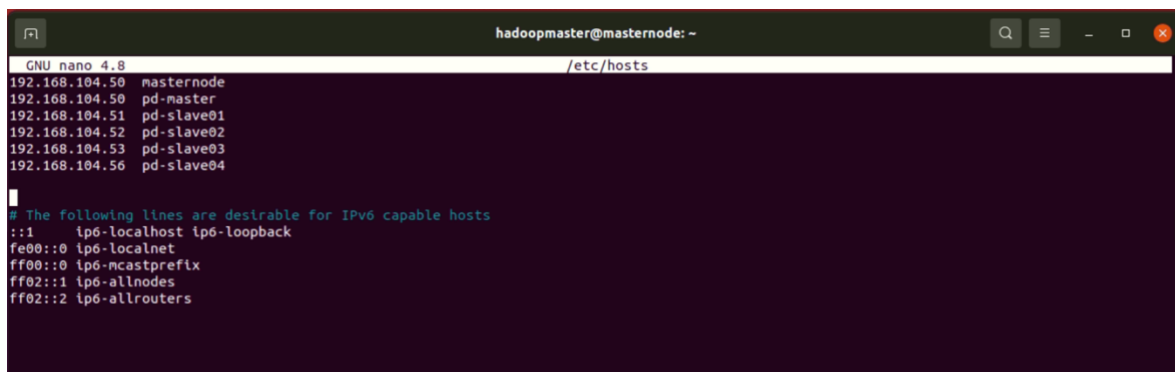
```
hadoopmaster@masternode: ~
GNU nano 4.8 /home/hadoopmaster/hadoop/etc/hadoop/hadoop-env.sh
###
# Router-based HDFS Federation specific parameters
# Specify the JVM options to be used when starting the RBF Routers.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_OFSROUTER_OPTS=""
###
# HDFS StorageContainerManager specific parameters
###
# Specify the JVM options to be used when starting the HDFS Storage Container Manager.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_STORAGECONTAINERMANAGER_OPTS=""
###
# Advanced Users Only!
###
#
# When building Hadoop, one can add the class paths to the commands
# via this special env var:
# export HADOOP_ENABLE_BUILD_PATHS="true"
#
# To prevent accidents, shell commands be (superficially) locked
# to only allow certain users to execute certain subcommands.
# It uses the format of (command)_(subcommand)_USER.
#
# For example, to limit who can execute the namenode command,
# export HDFS_NAMENODE_USER=hdfs
#HADOOP
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
export HADOOP_CLASSPATH+=" $HADOOP_HOME/lib/*.jar"
```

Εικόνα 3.1.2 1 Παραμετροποίηση αρχείου hadoop-env.sh

Έπειτα με την εντολή **“sudo mv hadoop /usr/local/hadoop”** θα μετακινήσουμε το hadoop στον κατάλογο /usr/local και με την εντολή **“sudo nano /etc/environment”** θα μεταβούμε στο περιβάλλον του συστήματος ώστε να βάλουμε το παρακάτω path “

PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin" JAVA_HOME="/usr/lib/jvm/j

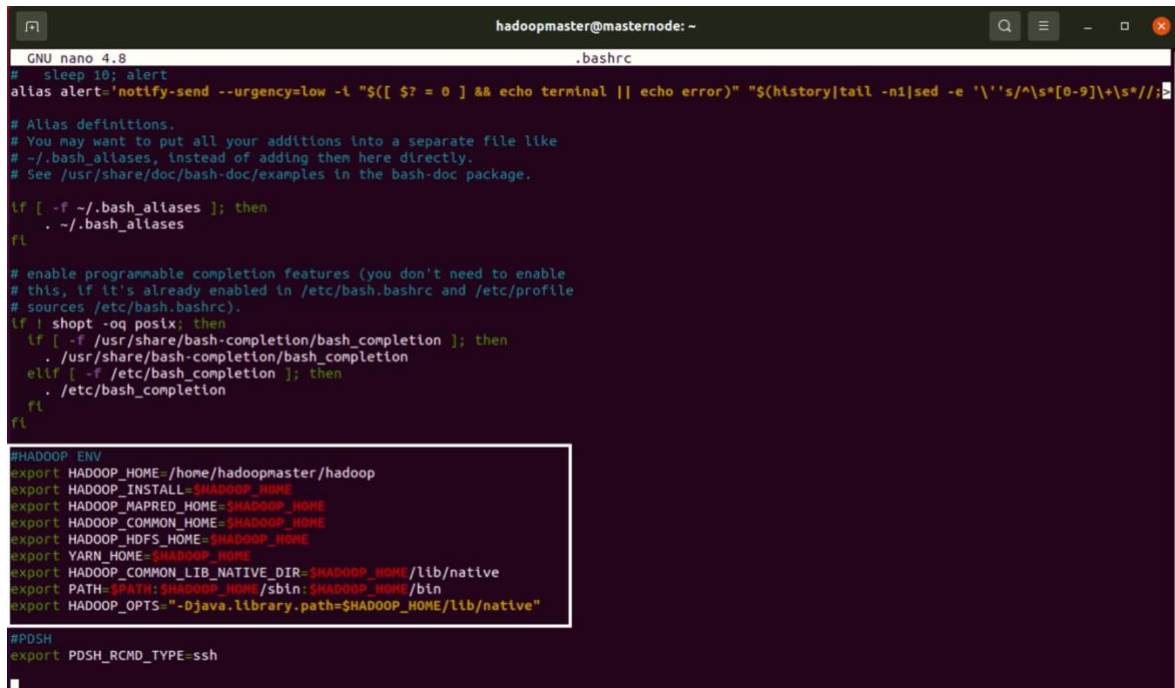
ava-11-openjdk-amd64/jre" ". Τώρα με την εντολή **"sudo adduser hadoopuser"** θα δημιουργήσουμε έναν hadoop χρήστη και στη συνέχεια θα δώσουμε τις παρακάτω τέσσερις εντολές **"sudo usermod -aG hadoopuser hadoopuser"**, **"sudo chown hadoopuser:root -R /usr/local/hadoop/"**, **"sudo chmod g+rwX -R /usr/local/hadoop/"**, **"sudo adduser hadoopuser sudo"**. Επόμενο βήμα είναι, με την εντολή **"ip addr"** να βρούμε την ip του συστήματός μας και έπειτα με την εντολή **"sudo nano /etc/hosts"** να μεταβούμε στο αρχείο των hosts και να ορίσουμε τα ψευδώνυμα master/slaves για κάθε συσκευή και τις αντίστοιχες ip.



```
hadoopmaster@masternode: ~  
GNU nano 4.8 /etc/hosts  
192.168.104.50 masternode  
192.168.104.50 pd-master  
192.168.104.51 pd-slave01  
192.168.104.52 pd-slave02  
192.168.104.53 pd-slave03  
192.168.104.56 pd-slave04  
  
# The following lines are desirable for IPv6 capable hosts  
::1 ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters
```

Εικόνα 3.1.2 2 Ανάθεση ονομάτων master/slaves στις IP

Έπειτα αντιγράφουμε το ssh κλειδί σε όλους τους χρήστες με τις εντολές **"ssh-copy-id hadoopuser@hadoop-master"**, **"ssh-copy-id hadoopuser@hadoop-slave1"**, **"ssh-copy-id hadoopuser@hadoop-slave2"**, συνεχίζουμε τη διαδικασία για όσους slaves έχουμε. Μεταβαίνουμε μετά στο bashrc με την εντολή **"sudo nano .bashrc"** και κάνουμε export τις παρακάτω εντολές (βγαίνοντας από το bashrc δίνουμε την εντολή **"source ~/.bashrc"** ώστε να εφαρμοστούν οι αλλαγές)

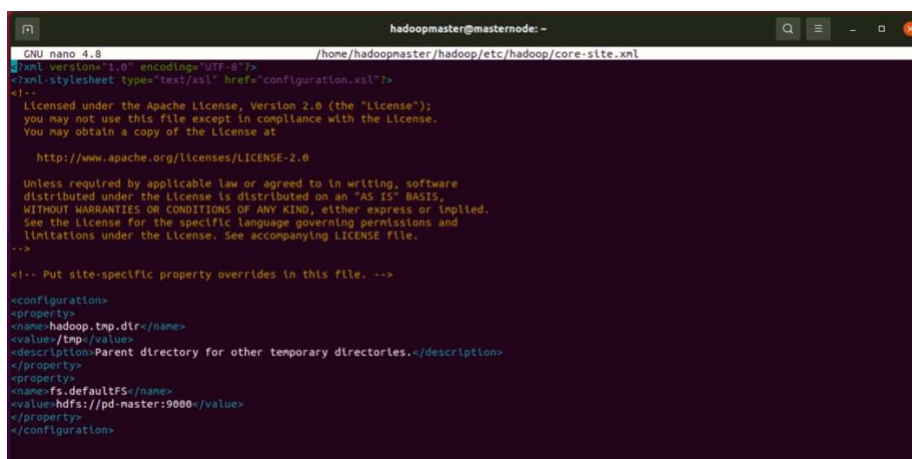


```
hadoopmaster@masternode: ~  
GNU nano 4.8 .bashrc  
# sleep 10; alert  
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;2  
# Alias definitions.  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.  
  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
if ! shopt -oq posix; then  
    if [ -f /usr/share/bash-completion/bash_completion ]; then  
        . /usr/share/bash-completion/bash_completion  
    elif [ -f /etc/bash_completion ]; then  
        . /etc/bash_completion  
    fi  
fi  
  
#HADOOP ENV  
export HADOOP_HOME=/home/hadoopmaster/hadoop  
export HADOOP_INSTALL=$HADOOP_HOME  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"  
  
#PDSH  
export PDSH_RCMD_TYPE=ssh
```

Εικόνα 3.1.2 3 Παραμετροποίηση bashrc αρχείου

Τώρα ήρθε η ώρα των παραμετροποιήσεων των xml αρχείων και συγκεκριμένα των core-site.xml, hdfs-site.xml, mapred-site.xml και yarn-site.xml.

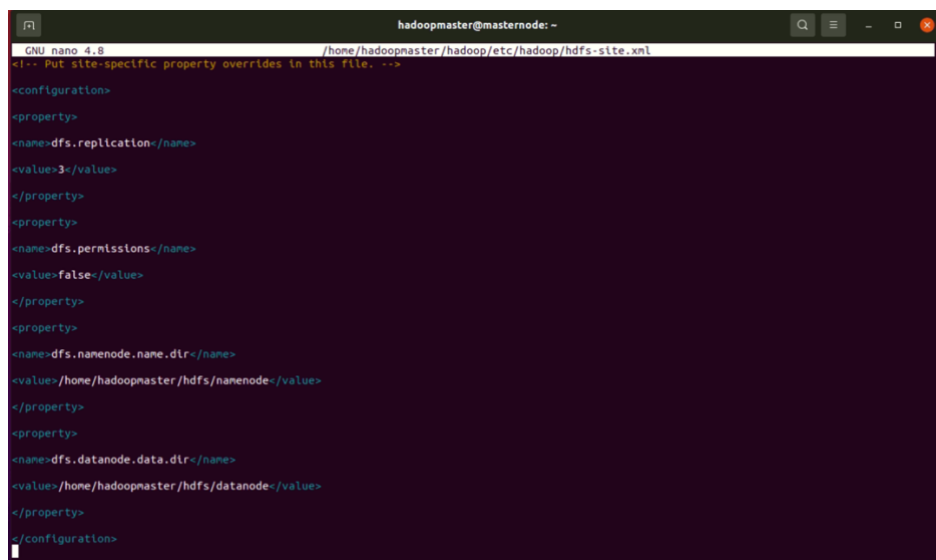
- **core-site.xml:** Το αρχείο core-site.xml καθορίζει τις ιδιότητες του HDFS και Hadoop Core. Δίνοντας την εντολή
“**sudo nano \$HADOOP_HOME/etc/hadoop/core-site.xml**” μεταβαίνουμε μέσα στο αρχείο και τοποθετούμε τις εντολές παραμετροποίησης που φαίνονται στην Εικόνα 3.1.2 4, ώστε να ρυθμίσουμε το port για το hadoop-master.



```
hadoopmaster@masternode: ~  
GNU nano 4.8 /home/hadoopmaster/hadoop/etc/hadoop/core-site.xml  
<?xml version='1.0' encoding='UTF-8' ?>  
<?xml-stylesheet type='text/xsl' href='configuration.xsl'?>  
<!--  
Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at  
  
http://www.apache.org/licenses/LICENSE-2.0  
  
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the license for the specific language governing permissions and  
limitations under the License. See accompanying LICENSE file.  
-->  
  
<!-- Put site-specific property overrides in this file. -->  
  
<configuration>  
  <property>  
    <name>hadoop.tmp.dir</name>  
    <value>/tmp</value>  
    <description>Parent directory for other temporary directories.</description>  
  </property>  
  <property>  
    <name>fs.defaultFS</name>  
    <value>hdfs://pd-master:9000</value>  
  </property>  
</configuration>
```

Εικόνα 3.1.2 4 Παραμετροποίηση core-site.xml

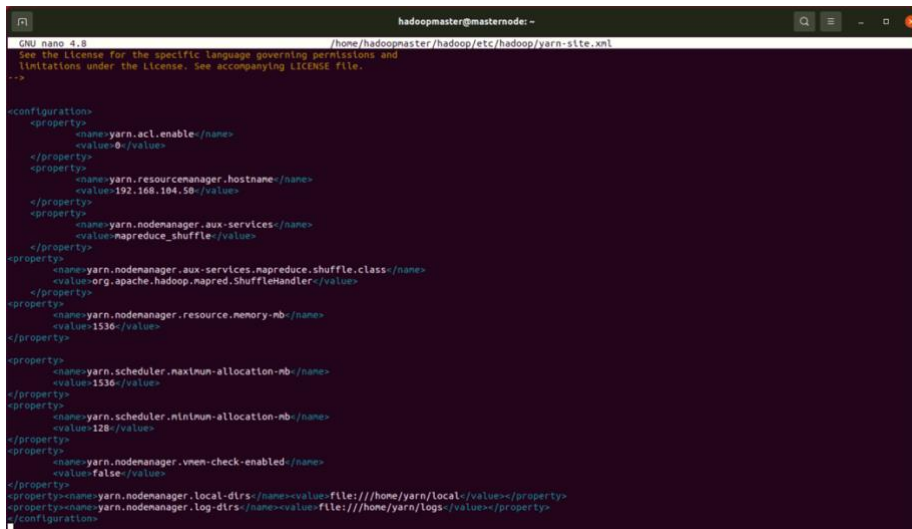
- **hdfs-site.xml:** Το αρχείο hdfs-site.xml χρησιμεύει για τη θέση αποθήκευσης των μεταδεδομένων κόμβων, αρχείων fsimage και επεξεργασίας αρχείων καταγραφής. Μέσα στο αρχείο καθορίζουμε τους καταλόγους αποθήκευσης NameNode και DataNode. Δίνοντας την εντολή **“sudo nano \$HADOOP_HOME/etc/hadoop/hdfs-site.xml”** μεταβαίνουμε μέσα στο αρχείο και τοποθετούμε τις εντολές παραμετροποίησης της Εικόνας 3.1.2 5 (σημαντικό ότι θα πρέπει να δημιουργηθούν και δύο κατάλογοι dfsdata, ένα για το namenode και ένα για το datanode).



```
hadoopmaster@masternode: ~  
GNU nano 4.8 /home/hadoopmaster/hadoop/etc/hadoop/hdfs-site.xml  
<!-- Put site-specific property overrides in this file. -->  
<configuration>  
  <property>  
    <name>dfs.replication</name>  
    <value>3</value>  
  </property>  
  <property>  
    <name>dfs.permissions</name>  
    <value>>false</value>  
  </property>  
  <property>  
    <name>dfs.namenode.name.dir</name>  
    <value>/home/hadoopmaster/hdfs/namenode</value>  
  </property>  
  <property>  
    <name>dfs.datanode.data.dir</name>  
    <value>/home/hadoopmaster/hdfs/datanode</value>  
  </property>  
</configuration>
```

Εικόνα 3.1.2 5 Παραμετροποίηση hdfs-site.xml

- **yarn-site.xml:** Το αρχείο yarn-site.xml χρησιμοποιείται για τον καθορισμό ρυθμίσεων σχετικά με το YARN. Περιέχει διαμορφώσεις για το NodeManager, Resource Manager, Containers και Application Master. Για την επεξεργασία του χρησιμοποιούμε την εντολή **“sudo nano \$HADOOP_HOME/etc/hadoop/yarn-site.xml”** και προσθέτουμε τις εντολές παραμετροποίησης της Εικόνας 3.1.2 6

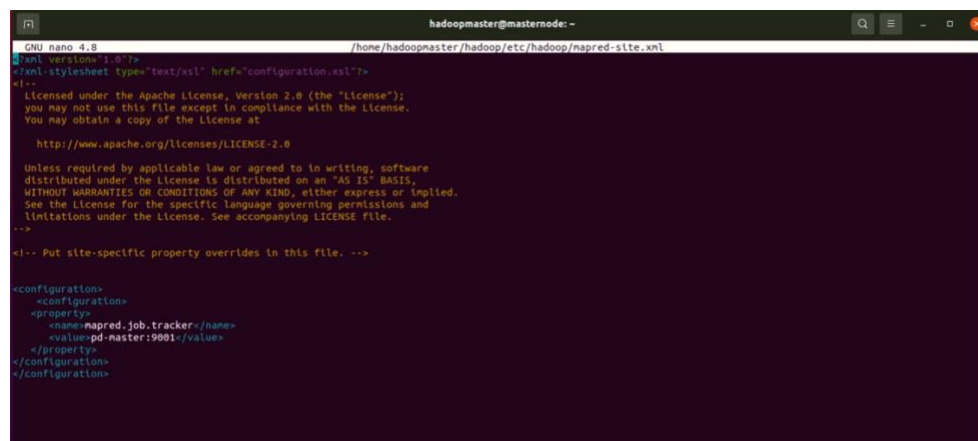


```
GNU nano 4.8 /home/hadoopmaster/hadoop/etc/hadoop/yarn-site.xml
Set the license for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<configuration>
  <property>
    <name>yarn.acl.enable</name>
    <value>0</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>192.168.104.50</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>128</value>
  </property>
  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>file:///home/yarn/local</value>
  </property>
  <property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:///home/yarn/logs</value>
  </property>
</configuration>
```

Εικόνα 3.1.2 6 Παραμετροποίηση yarn-site.xml

- **mapred-site.xml:** Για την επεξεργασία του αρχείου mapred-site.xml χρησιμοποιούμε την εντολή “**sudo nano \$HADOOP_HOME/etc/hadoop/mapred-site.xml**” και προσθέτουμε τις εντολές παραμετροποίησης που φαίνονται στην Εικόνα 3.1.2 7

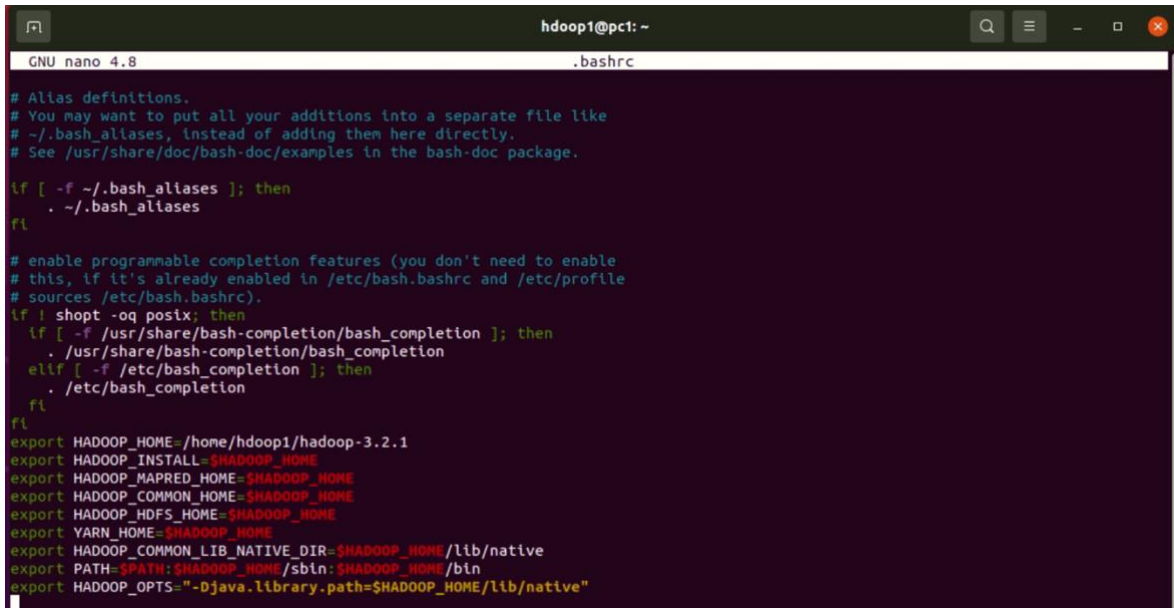


```
GNU nano 4.8 /home/hadoopmaster/hadoop/etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>pd-master:9001</value>
  </property>
</configuration>
```

Εικόνα 3.1.2 7 Παραμετροποίηση mapred-site.xml

Έπειτα, με την εντολή “**sudo nano ~/hadoop/etc/hadoop/workers**” θα πάμε στο αρχείο των workers και θα τοποθετήσουμε τις εντολές “**hadoop1@pd-slave01**”, “**hadoop1@pd-slave01**”, συνεχίζουμε τη διαδικασία για όσους slaves έχουμε. Μετά τις παραμετροποιήσεις των παραπάνω αρχείων πρέπει να γίνει μία μορφοποίηση για το NameNode, πριν ξεκινήσει τις υπηρεσίες του Hadoop για πρώτη φορά. Αυτό επιτυγχάνεται με την εντολή “**hdfs namenode -format**”. Έπειτα πρέπει να κατευθυνθούμε στον φάκελο `hadoop/sbin` και να εκτελέσουμε την εντολή “**./start-dfs.sh**” ώστε να ξεκινήσει το NameNode και το DataNode.

Καλό είναι να δώσουμε και την εντολή “**jps**” ώστε να δούμε πως τρέχουν σωστά και πως δεν υπάρχει κάποιο πρόβλημα. Μετά χρειάζεται να ορίσουμε τις εντολές της Εικόνας 3.1.2 8 για να ρυθμίσουμε το yarn.



```
GNU nano 4.8                                .bashrc

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

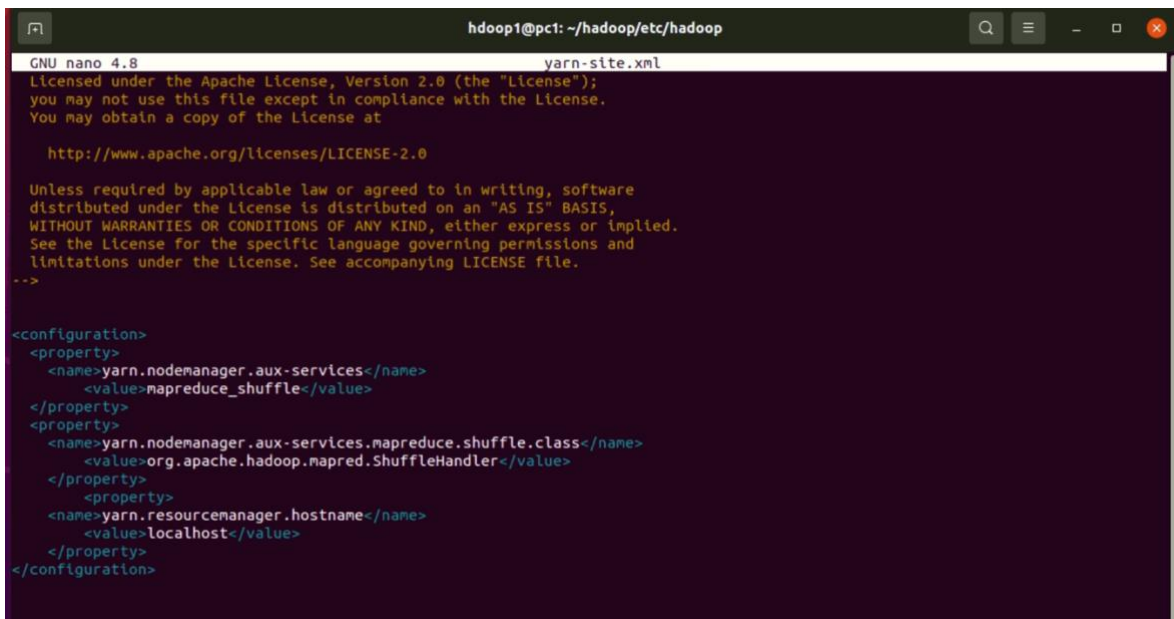
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

export HADOOP_HOME=/home/hadoop1/hadoop-3.2.1
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Εικόνα 3.1.2 8 Ρύθμιση Yarn μέσα στον 1ο slave

Έπειτα, πάμε στο yarn-site.xml των slaves, με την εντολή “**sudo nano hadoop/etc/hadoop/yarn-site.xml**” και προσθέτουμε τις εντολές της Εικόνας 3.1.2 9



```
GNU nano 4.8                                yarn-site.xml
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

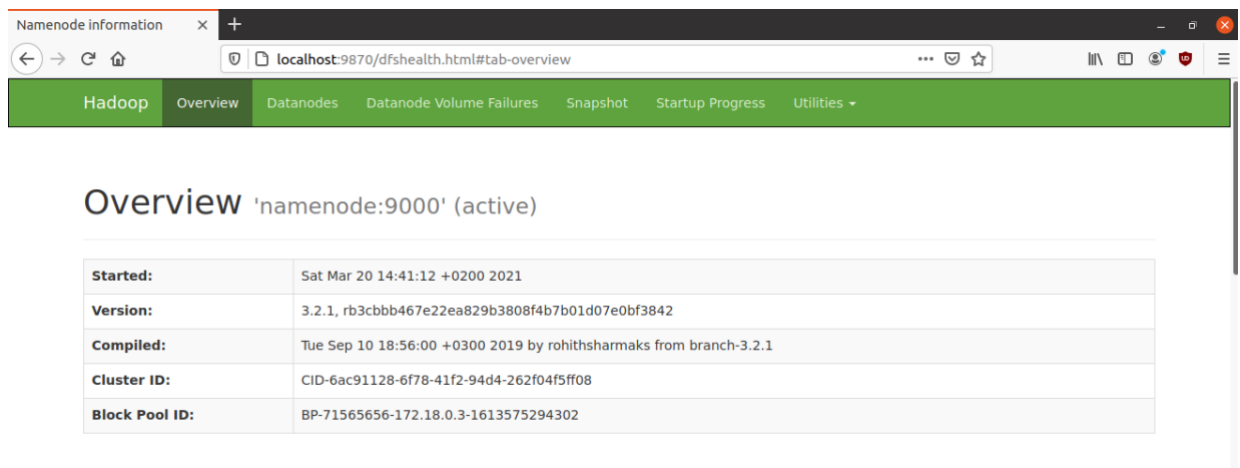
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>localhost</value>
  </property>
</configuration>
```

Εικόνα 3.1.2 9 Ρύθμιση yarn-site.xml στον 1ο slave

Τέλος, από τον φάκελο `hadoop/sbin` δίνουμε την εντολή `“./start-yarn.sh”` ώστε να ξεκινήσουν και τα `resourcemanager` και `nodemanagers` του YARN.

3.1.3 ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ

Για την πρόσβαση στο περιβάλλον του χρήστη στο Hadoop UI, πρέπει να ανοίξουμε τον φυλλομετρητή (browser) που χρησιμοποιούμε και να μεταβούμε στη διεύθυνση URL ή τη διεύθυνση IP του localhost. Ο προεπιλεγμένος αριθμός θύρας 9870 δίνει την πρόσβαση στο Hadoop NameNode UI και παίζουμε πληκτρολογώντας **`http://localhost:9870`**. Στην Εικόνα 3.1.3 1 βλέπουμε κάποιες πληροφορίες σχετικά με την ημερομηνία και την ώρα που μπήκαμε, την έκδοση του Hadoop, το πότε συντάχθηκε, το αναγνωριστικό συστήματος (Cluster ID) και το αναγνωριστικό μπλοκ ομάδας (Block Pool ID).



Εικόνα 3.1.3 10 Hadoop NameNode UI Overview

NameNode Storage

Storage Directory	Type	State
/hadoop/dfs/name	IMAGE_AND_EDITS	Active

DFS Storage Types

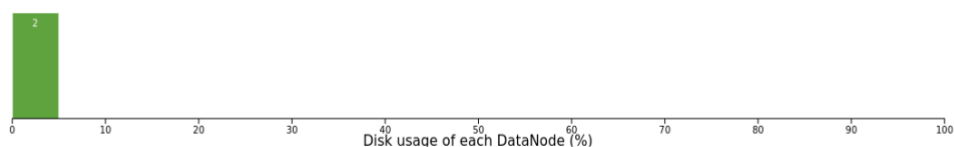
Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	913.91 GB	1.28 MB (0%)	818.92 GB (89.61%)	1.28 MB	2

Hadoop, 2019.

Εικόνα 3.1.3 11 NameNode Storage & DFS Storage Types

Στην παραπάνω εικόνα (Εικόνα 3.1.3 2), βλέπουμε κάποια στοιχεία για τον αποθηκευτικό χώρο του NameNode και τους τύπους αποθήκευσης του dfs. Έπειτα στην Εικόνα 3.1.3 3 απεικονίζονται διάφορα στοιχεία ως προς τη χρήση δίσκου κάθε DataNode, ποια Node βρίσκονται σε λειτουργία και διάφορα ακόμα.

Datanode usage histogram



In operation

Show entries

Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ 7ab139f3d91e:9866 (172.18.0.5:9866)	http://7ab139f3d91e:9864	1s	128m	456.96 GB	26	656 KB (0%)	3.2.1
✓ prodromos:9866 (172.18.0.1:9866)	http://prodromos:9864	1s	5m	456.96 GB	26	656 KB (0%)	3.2.1

Showing 1 to 2 of 2 entries

Previous **1** Next

Εικόνα 3.1.3 12 DataNode usage histogram

Στην Εικόνα 3.1.3 4 βλέπουμε τον κατάλογο αναζήτησης, όπου εδώ θα βρίσκονται οι εργασίες που θα δημιουργηθούν.

Browse Directory

Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwt	root	root	0 B	Feb 17 17:30	0	0 B	app-logs
drwxr-xr-x	root	supergroup	0 B	Feb 17 17:21	0	0 B	rmstate
drwx-----	root	supergroup	0 B	Feb 17 17:29	0	0 B	tmp
drwxr-xr-x	root	supergroup	0 B	Feb 25 12:59	0	0 B	user

Showing 1 to 4 of 4 entries

Previous 1 Next

Hadoop, 2019.

Εικόνα 3.1.3 13 Browse Directory

Η επόμενη θύρα είναι η 9864 και χρησιμοποιείται για πρόσβαση σε μεμονωμένους DataNodes. Για να μεταβούμε θα πρέπει να πληκτρολογήσουμε **http://localhost:9864**. Στην Εικόνα 3.1.3 5 απεικονίζεται το περιβάλλον παρακολούθησης του DataNode, όπου παρέχει κάποιες πληροφορίες για το DataNode, Block Pools και Volume.

DataNode Information x +

localhost:9864/datanode.html

DataNode on prodromos:9866

Cluster ID:	CID-6ac91128-6f78-41f2-94d4-262f04f5ff08
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842

Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
localhost:9000	BP-71565656-172.18.0.3-1613575294302	RUNNING	2s	8 minutes	243 B (64 MB)

Volume Information

Directory	StorageType	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
/home/hduser/dfsdata/datanode	DISK	656 KB	409.46 GB	0 B	0 B	26

Hadoop, 2019.

Εικόνα 3.1.3 14 Περιβάλλον παρακολούθησης DataNode

Τέλος το YARN Resource Manager βρίσκεται στη θύρα 8088 και πηγαίνουμε πληκτρολογώντας **http://localhost:8088**. Όπως φαίνεται και παρακάτω στην Εικόνα 3.1.3 6 εκεί είναι το περιβάλλον όπου δείχνει τις εργασίες που πραγματοποιούνται στο cluster.

The screenshot displays the Hadoop YARN Resource Manager web interface. The top navigation bar includes the Hadoop logo and the title 'All Applications'. The left sidebar contains a menu with options like 'Cluster', 'About', 'Nodes', 'Node Labels', 'Applications', 'NEW', 'NEW SAVING', 'SUBMITTED', 'ACCEPTED', 'RUNNING', 'FINISHED', 'KILLED', and 'Scheduler'. The main content area shows 'Cluster Metrics' with a table of application statistics. Below this, there are 'Cluster Nodes Metrics' and 'Scheduler Metrics' sections. The 'Applications' table is currently empty, displaying 'Showing 0 to 0 of 0 entries'.

Cluster Metrics													
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved			
0	0	0	0	0	0 B	8 GB	0 B	0	8	0			

Cluster Nodes Metrics						
Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
0	0	0	0	0	0	0

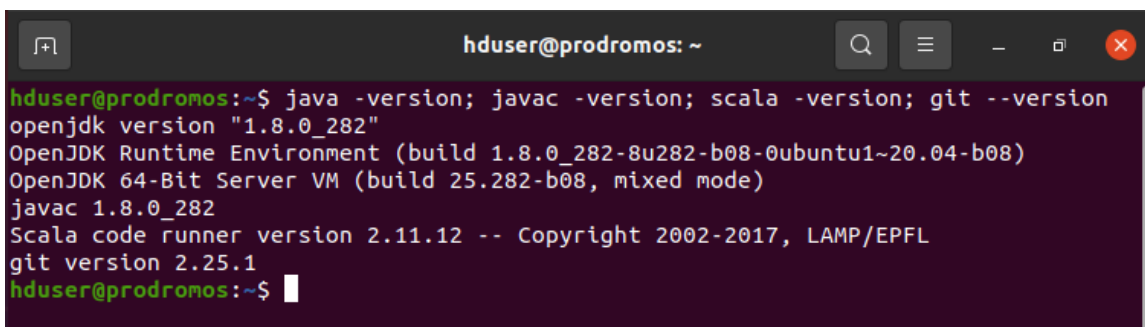
Scheduler Metrics		Scheduling Resource Type		Minimum Allocation		Maximum Allocation		Maximum Cluster Application Priority	
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>		<memory:8192, vCores:4>		0			
No data available in table									

Εικόνα 3.1.3 15 Περιβάλλον παρακολούθησης εργασιών cluster

3.2 ΕΓΚΑΤΑΣΤΑΣΗ APACHE SPARK

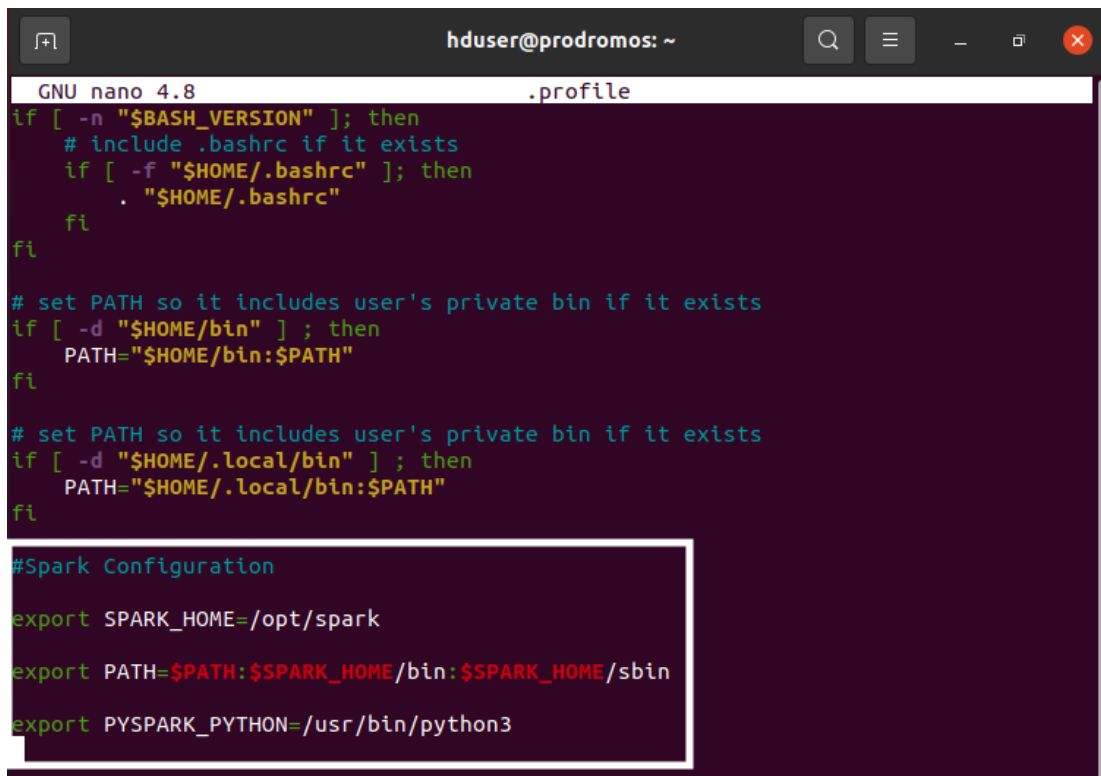
3.2.1 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ SINGLE NODE

Για την εγκατάσταση του Apache Spark πρέπει να κατεβάσουμε τρία πακέτα, Java, Scala και Git. Αυτό επιτυγχάνεται πληκτρολογώντας στο terminal την εντολή **“sudo apt install default-jdk scala git -y”**. Για την επιβεβαίωση ότι κατέβηκαν σωστά δίνουμε την εντολή **“java -version; javac -version; scala -version; git --version”** και αν δούμε το μήνυμα της Εικόνας 3.2.1 1 σημαίνει ότι όλα πήγαν σωστά.

A terminal window titled 'hduser@prodromos: ~' with a search icon, a menu icon, and window control buttons. The terminal shows the command 'hduser@prodromos:~\$ java -version; javac -version; scala -version; git --version' and its output: 'openjdk version "1.8.0_282"', 'OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~20.04-b08)', 'OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)', 'javac 1.8.0_282', 'Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL', and 'git version 2.25.1'. The prompt returns to 'hduser@prodromos:~\$'.

Εικόνα 3.2.1 1 Επιβεβαίωση έκδοσης Java, Scala & Git

Έπειτα πάμε να κατεβάσουμε το Apache Spark. Αυτό μπορεί να γίνει με δύο τρόπους, είτε πηγαίνοντας στη σελίδα <https://spark.apache.org/> και από την ενότητα Download διαλέξουμε την έκδοση που προτιμάμε (στην προκειμένη περίπτωση το Spark 3.1.1 με Pre-Built για Apache Hadoop 3.2) είτε δίνοντας την εντολή **“wget https://downloads.apache.org/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz”**. Μετά με την εντολή **“tar xvf spark-*”** αποσυμπιέζουμε το Spark αρχείο. Για να είναι πιο εύκολη η πλοήγηση αργότερα στους φακέλους του Spark δίνουμε την εντολή **“mv ./spark-3.0.1-bin-hadoop2.7 ./spark”** και έτσι μετονομάζουμε το αρχείο σε spark. Τέλος, μετακινούμε το αρχείο στον κατάλογο opt/spark με την εντολή **“sudo mv spark /opt/spark”**. Το επόμενο βήμα είναι η παραμετροποίηση του περιβάλλοντος του Spark. Με την εντολή **“sudo nano .profile”** πηγαίνουμε στο profile και κάνουμε export τις εντολές που απεικονίζονται στην Εικόνα 3.2.1 2 (Σημαντικό μόλις βγούμε από το profile να δώσουμε την εντολή **“source ~/.profile”** ώστε να εφαρμοστούν οι αλλαγές που πραγματοποιήσαμε).



```
GNU nano 4.8 .profile
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

#Spark Configuration

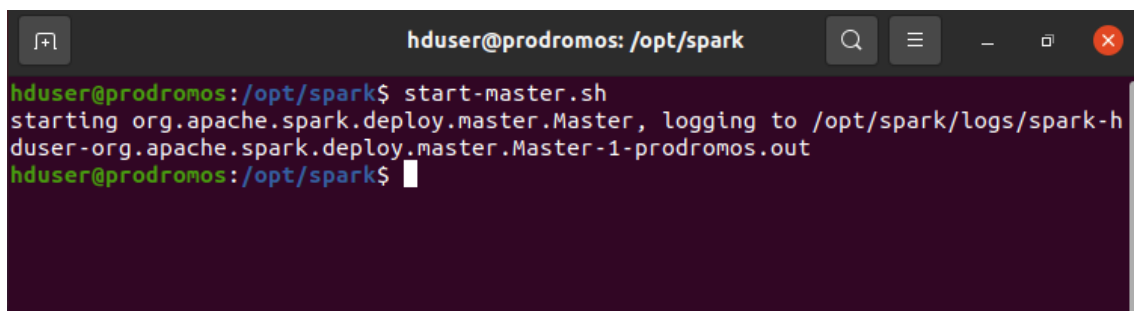
export SPARK_HOME=/opt/spark

export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

export PYSARK_PYTHON=/usr/bin/python3
```

Εικόνα 3.2.1 2 Spark Configuration

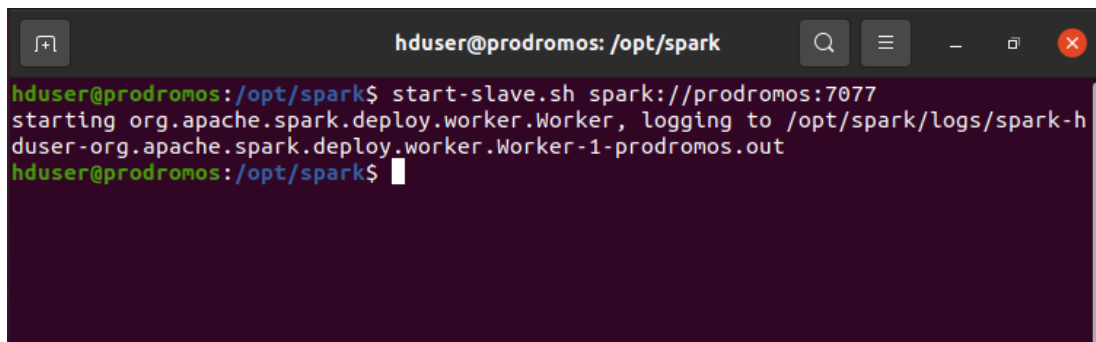
Μόλις τελειώσουμε την παραπάνω παραμετροποίηση μπορούμε να ξεκινήσουμε τον master server. Δίνουμε την εντολή **“cd ../../opt/spark”** για να μεταβούμε στον κατάλογο του Spark και πληκτρολογούμε την εντολή **“start-master.sh”**



```
hduser@prodromos: /opt/spark
hduser@prodromos:/opt/spark$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-hduser-org.apache.spark.deploy.master.Master-1-prodromos.out
hduser@prodromos:/opt/spark$
```

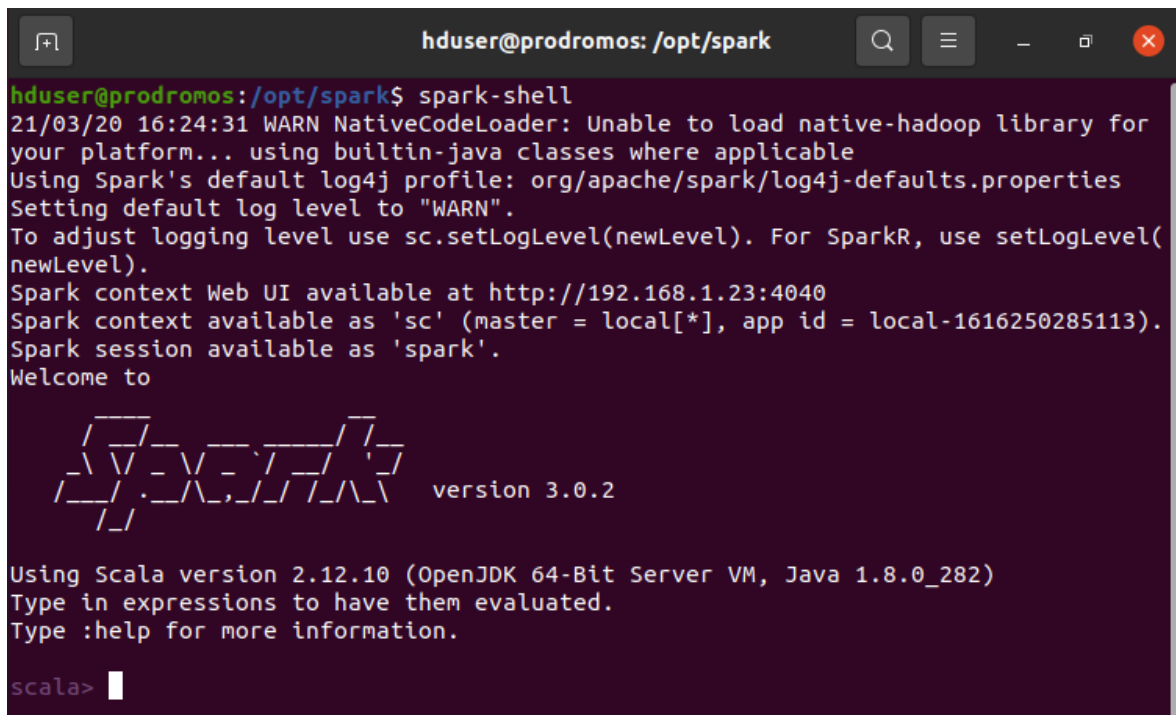
Εικόνα 3.2.1 3 Εκκίνηση master server

Για να ξεκινήσουμε τον slave server δίνουμε την εντολή **“start-slave.sh spark://prodromos:7077”** όπου το prodromos:7077 είναι το master port.

A terminal window titled 'hduser@prodromos: /opt/spark'. The prompt is 'hduser@prodromos:/opt/spark\$'. The user has entered 'start-slave.sh spark://prodromos:7077'. The output shows the Spark worker starting, logging to '/opt/spark/logs/spark-hduser-org.apache.spark.deploy.worker.Worker-1-prodromos.out', and then returning to the prompt 'hduser@prodromos:/opt/spark\$'.

Εικόνα 3.2.1 4 Εκκίνηση slave server

Μόλις τελειώσουμε με την παραμετροποίησης του Spark μπορούμε να αρχίσουμε να χρησιμοποιούμε το spark-shell, εάν θέλουμε να προγραμματίσουμε σε γλώσσα προγραμματισμού Scala ή το pyspark , εάν θέλουμε να προγραμματίσουμε σε γλώσσα προγραμματισμού Python. Το spark-shell ξεκινάει με την εντολή “**spark-shell**” και αν θέλουμε να ορίσουμε εμείς πόσοι πυρήνες θα χρησιμοποιηθούν θα πρέπει να συμπληρώσουμε την εντολή “**-master local[x]**”, όπου το *x* συμβολίζει τον αριθμό των πυρήνων. Για να βγούμε από το spark-shell πληκτρολογούμε “**:q**”. Το pyspark ξεκινάει με την εντολή “**pyspark**” και αν θέλουμε να βγούμε πληκτρολογούμε “**quit()**”. Στις Εικόνες 3.2.1 5 και 3.2.1 6 απεικονίζεται το spark-shell και pyspark.

A terminal window titled 'hduser@prodromos: /opt/spark'. The prompt is 'hduser@prodromos:/opt/spark\$'. The user has entered 'spark-shell'. The output shows various warnings and information about the Spark context, including the Web UI URL 'http://192.168.1.23:4040' and the Spark session ID 'spark'. It then displays the Spark logo and the version 'version 3.0.2'. Below that, it shows the Scala version 'Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_282)' and prompts the user to type in expressions to have them evaluated. The prompt is now 'scala>'.

Εικόνα 3.2.1 5 spark-shell

```
hduser@prodromos: /opt/spark
hduser@prodromos:/opt/spark$ pyspark
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/03/20 16:25:20 WARN NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(
newLevel).
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| |_) | |_| |
  ___) | |_) | | | |
  |___|_|_|\___|_|_|_|

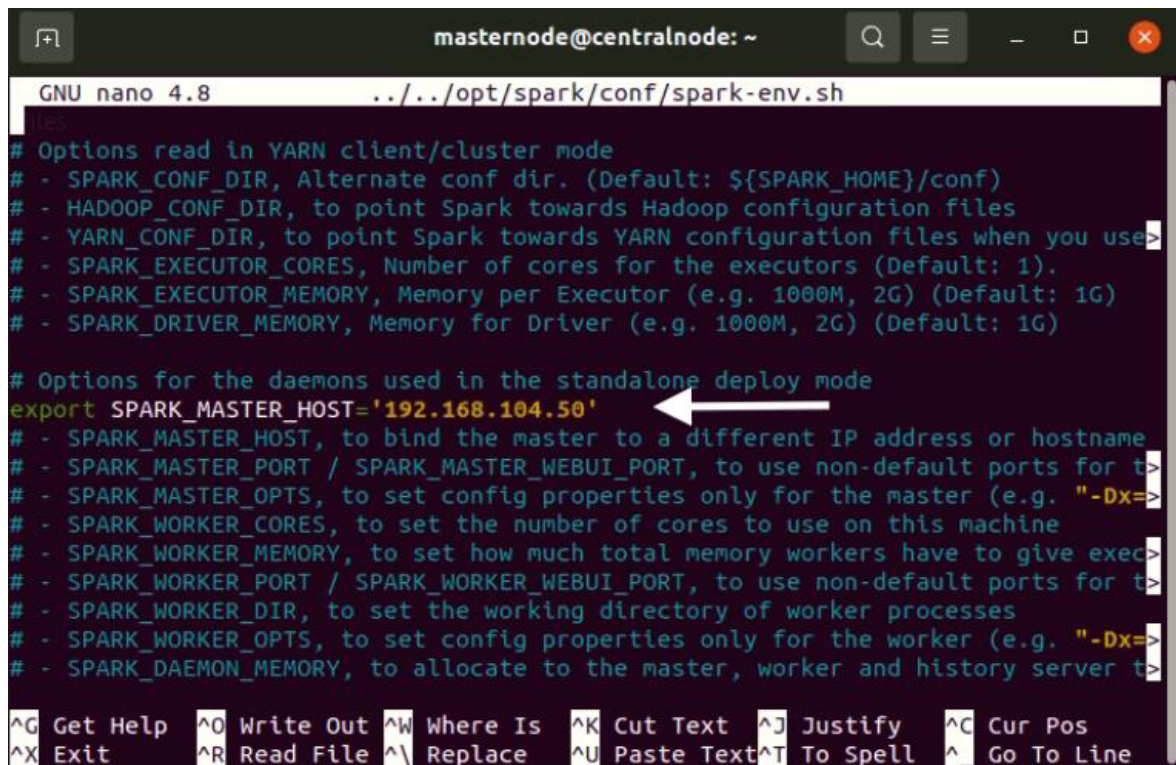
version 3.0.2

Using Python version 3.8.5 (default, Jan 27 2021 15:41:15)
SparkSession available as 'spark'.
>>> 
```

Εικόνα 3.2.1 6 pyspark

3.2.2 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΕ MULTI NODE

Αρχικά πρέπει να κάνουμε αντιγραφή των περιεχομένων των template αρχείων σε καινούργια shell αρχεία (.sh) με το ίδιο όνομα, αυτό γίνεται με την εντολή “**sudo nano spark-env.template spark-env.sh**”. Εκτελούμε επίσης τις εντολές “**sudo cp spark-defaults.conf.template spark-defaults.conf**” και “**sudo cp slaves.template slaves**”. Έπειτα θα φτιάξουμε τον Master. Για τη διαδικασία αυτή, πρώτα από όλα πρέπει να γίνει επεξεργασία των μεταβλητών του Spark περιβάλλοντος. Πηγαίνουμε οπότε στον φάκελο conf όπου υπάρχει η εγκατάσταση του Spark, δίνοντας την εντολή “**sudo nano ../../opt/spark/conf/spark-env.sh**”. Έπειτα πρέπει να γίνει η ρύθμιση για την ip του master host. Πηγαίνουμε οπότε στο αρχείο spark-env.sh και ορίζουμε το όνομα του master host μαζί με την ip του.



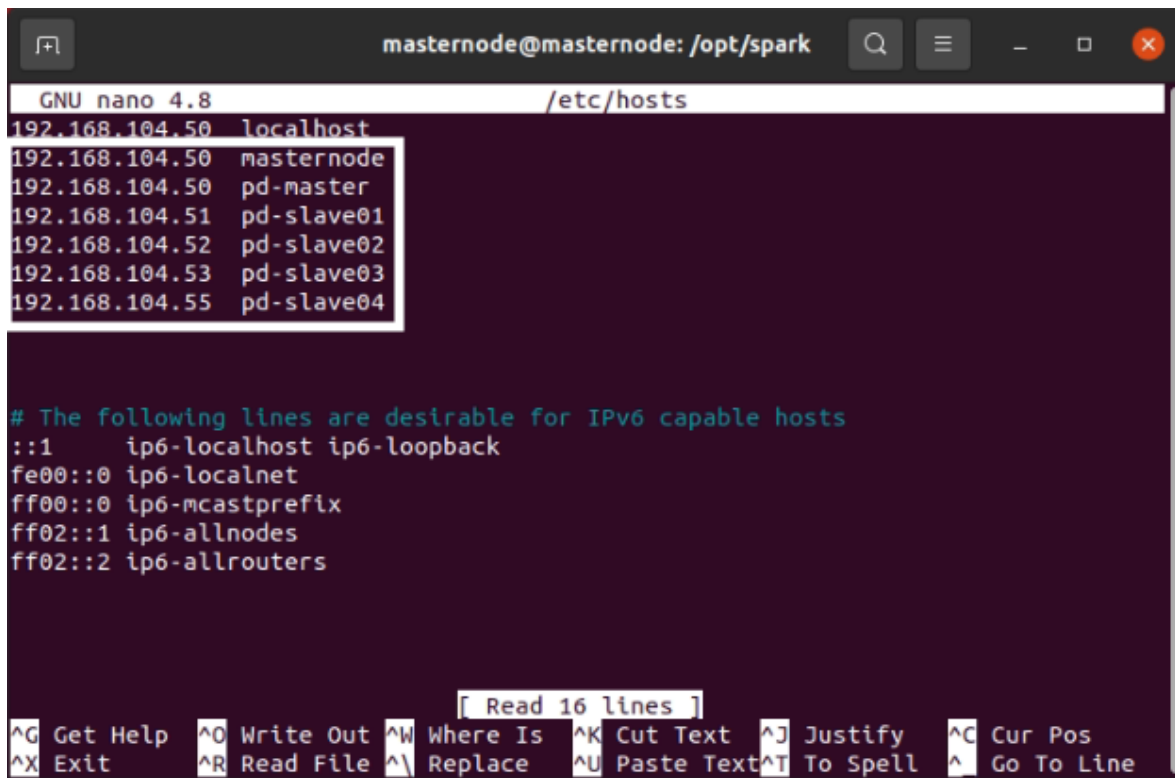
```
GNU nano 4.8 ../../opt/spark/conf/spark-env.sh
# Options read in YARN client/cluster mode
# - SPARK_CONF_DIR, Alternate conf dir. (Default: ${SPARK_HOME}/conf)
# - HADOOP_CONF_DIR, to point Spark towards Hadoop configuration files
# - YARN_CONF_DIR, to point Spark towards YARN configuration files when you use
# - SPARK_EXECUTOR_CORES, Number of cores for the executors (Default: 1).
# - SPARK_EXECUTOR_MEMORY, Memory per Executor (e.g. 1000M, 2G) (Default: 1G)
# - SPARK_DRIVER_MEMORY, Memory for Driver (e.g. 1000M, 2G) (Default: 1G)

# Options for the daemons used in the standalone deploy mode
export SPARK_MASTER_HOST='192.168.104.50'
# - SPARK_MASTER_HOST, to bind the master to a different IP address or hostname
# - SPARK_MASTER_PORT / SPARK_MASTER_WEBUI_PORT, to use non-default ports for t>
# - SPARK_MASTER_OPTS, to set config properties only for the master (e.g. "-Dx=>
# - SPARK_WORKER_CORES, to set the number of cores to use on this machine
# - SPARK_WORKER_MEMORY, to set how much total memory workers have to give exec>
# - SPARK_WORKER_PORT / SPARK_WORKER_WEBUI_PORT, to use non-default ports for t>
# - SPARK_WORKER_DIR, to set the working directory of worker processes
# - SPARK_WORKER_OPTS, to set config properties only for the worker (e.g. "-Dx=>
# - SPARK_DAEMON_MEMORY, to allocate to the master, worker and history server t>

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Εικόνα 3.2.2 1 Παραμετροποίηση αρχείου spark-env.sh

Στο επόμενο βήμα πρέπει να ορίσουμε για κάθε μία ip και από ένα όνομα. Δίνοντας την εντολή “**sudo nano ../../etc/hosts**” μεταβαίνουμε μέσα στο αρχείο και βάζουμε τα ονόματα master/slaves για τις ip που αντιστοιχούν. Η ίδια διαδικασία πρέπει να γίνει και στους υπολογιστές που θα χρησιμοποιηθούν ως workers. Για να βρούμε τις ip δίνουμε την εντολή “**ip addr**” και η ip που βρίσκουμε (στην περίπτωσή μας 192.168.104.50) θα είναι ο master. Έπειτα με την εντολή “**sudo reboot**” γίνεται επανεκκίνηση του συστήματος ώστε να εφαρμοστούν οι αλλαγές.



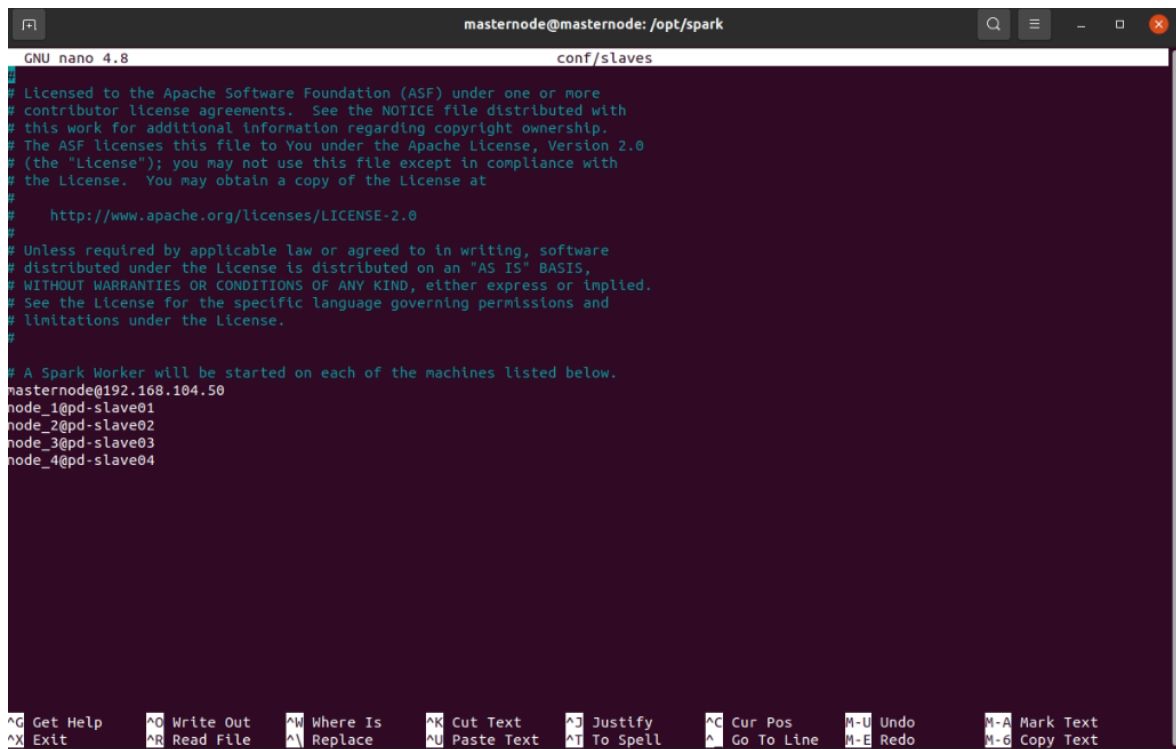
```
masternode@masternode: /opt/spark
GNU nano 4.8 /etc/hosts
192.168.104.50 localhost
192.168.104.50 masternode
192.168.104.50 pd-master
192.168.104.51 pd-slave01
192.168.104.52 pd-slave02
192.168.104.53 pd-slave03
192.168.104.55 pd-slave04

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

[ Read 16 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Εικόνα 3.2.2 2 Ανάθεση ονομάτων master/slaves στις IP

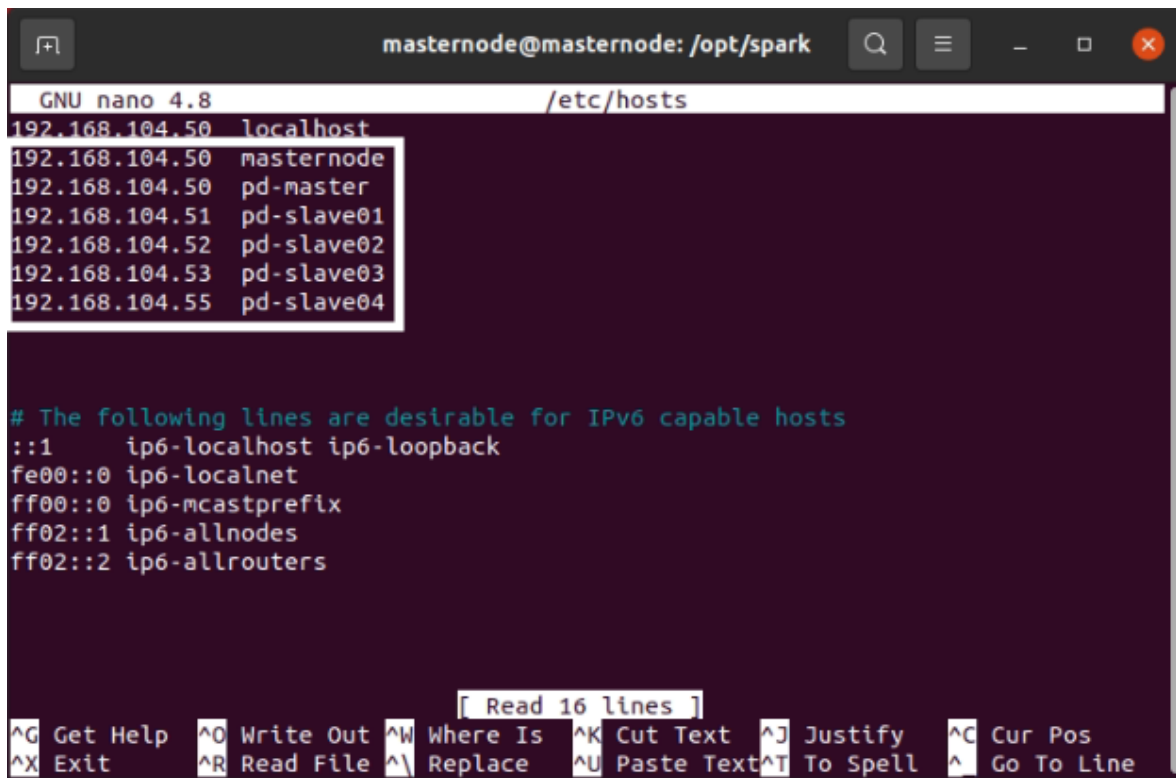
Μετά πρέπει να ελέγξουμε εάν έχουμε εγκατεστημένη τη Java και τη Scala. Αυτό πραγματοποιείται με την εντολή **“java -version; javac -version; scala -version”** (στο Single Node τις είχαμε εγκαταστήσει). Έπειτα, με την εντολή **“ssh”** ελέγχουμε εάν το ssh είναι ρυθμισμένο στο μηχανήμά μας. Εάν δεν υπάρχει το εγκαθιστούμε με την εντολή **“sudo apt-get install openssh-server openssh-client”**. Μετά με την εντολή **“ssh-keygen -t rsa -P ”** δημιουργούμε ένα νέο ζεύγος κλειδιών και με την εντολή **“cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys”** εξουσιοδοτούμε το νέο κλειδί. Ως τελικό βήμα, πρέπει να αντιγράψουμε το περιεχόμενο του .ssh/id_rsa.pub στο .ssh/authorized_keys όλων των workers. Αυτό επιτυγχάνεται με τις εντολές **“ssh-copy-id user@pd-master”** και **“ssh-copy-id user@pd-slave1”**. Με την εντολή **“ssh user@pd-slave1”** μπαίνουμε μέσα στον πρώτο slave, ώστε να ελέγξουμε τη σύνδεση με τους slaves. Για να επιστρέψουμε πίσω στον master δίνουμε την εντολή **“exit”**. Έπειτα, πρέπει να προσθέσουμε τη διαδρομή με την τοποθεσία που βρίσκεται το Spark. Δίνουμε την εντολή **“sudo nano ~/.bashrc”** για να μπούμε στο bashrc και κάνουμε **“export PATH=\$PATH:/opt/spark/bin”**. Βγαίνοντας δίνουμε την εντολή **“source ~/.bashrc”** για να εφαρμοστεί η αλλαγή. Μετά με την εντολή **“sudo nano ../../opt/spark/conf/slaves”** μπαίνουμε στο slaves και εκεί προσθέτουμε τους workers, ώστε ο master node να μπορεί να τους αναγνωρίζει.



```
masternode@masternode: /opt/spark
GNU nano 4.8 conf/slaves
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# A Spark Worker will be started on each of the machines listed below.
masternode@192.168.104.50
node_1@pd-slave01
node_2@pd-slave02
node_3@pd-slave03
node_4@pd-slave04
```

Εικόνα 3.2.2 3 Ορισμός των workers

Μόλις τελειώσουμε με τη ρύθμιση του Master, πρέπει να κάνουμε τη ρύθμιση ενός worker. Αρχικά πραγματοποιούμε αντιγραφή των template αρχείων παρόμοια με την αντιγραφή που έγινε στον κεντρικό κόμβο. Έπειτα πηγαίνουμε στο αρχείο `spark-env.sh` με την εντολή **“`sudo nano ../../opt/spark/conf/spark-env.sh`”** και ορίζουμε τον master host μαζί με την ip του κάνοντας, **“`export SPARK_MASTER_HOST = ‘YOUR_MASTER_HOST_IP’`”**. Επιπρόσθετα, πρέπει να τοποθετήσουμε τα μηχανήματα στο αρχείο `hosts`. Δίνουμε την εντολή **“`sudo nano ../../etc/hosts`”** για να μπούμε στο αρχείο και προσθέτουμε τις ip με τα ονόματα των υπολογιστών που απεικονίζονται στην Εικόνα 3.2.2 4, ώστε να φτιάξουμε ίδια ονόματα για τα μηχανήματα του cluster network με αυτά του master node. Εάν θέλουμε παραπάνω από έναν worker επαναλαμβάνουμε τη διαδικασία και για τους υπόλοιπους. Μόλις τελειώσουμε και τη ρύθμιση του worker μπορούμε να ξεκινήσουμε το cluster network, δίνοντας την εντολή **“`../../opt/spark/sbin/start-all.sh`”**

A screenshot of a terminal window titled 'masternode@masternode: /opt/spark'. The terminal shows the GNU nano 4.8 editor editing the /etc/hosts file. The file content is as follows:

```
192.168.104.50 localhost
192.168.104.50 masternode
192.168.104.50 pd-master
192.168.104.51 pd-slave01
192.168.104.52 pd-slave02
192.168.104.53 pd-slave03
192.168.104.55 pd-slave04

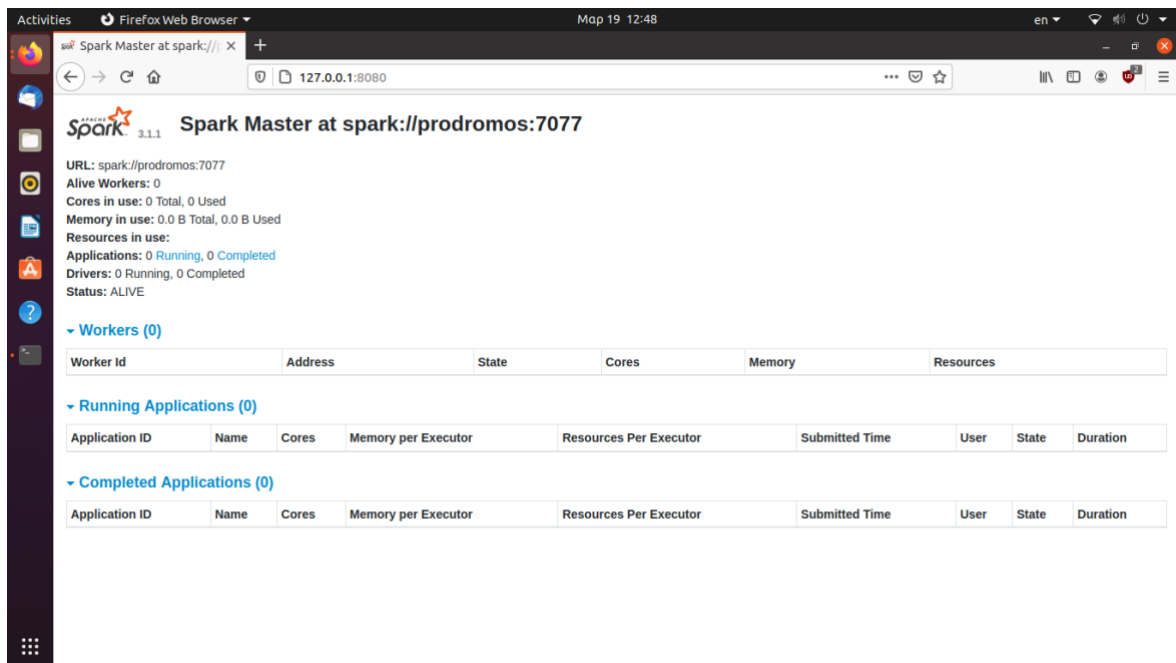
# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
```

The bottom of the terminal shows the nano editor's status bar with various keyboard shortcuts like '^G Get Help', '^O Write Out', etc. A status line above the status bar indicates '[Read 16 lines]'.

Εικόνα 3.2.2 4 Διαμόρφωση host αρχείου

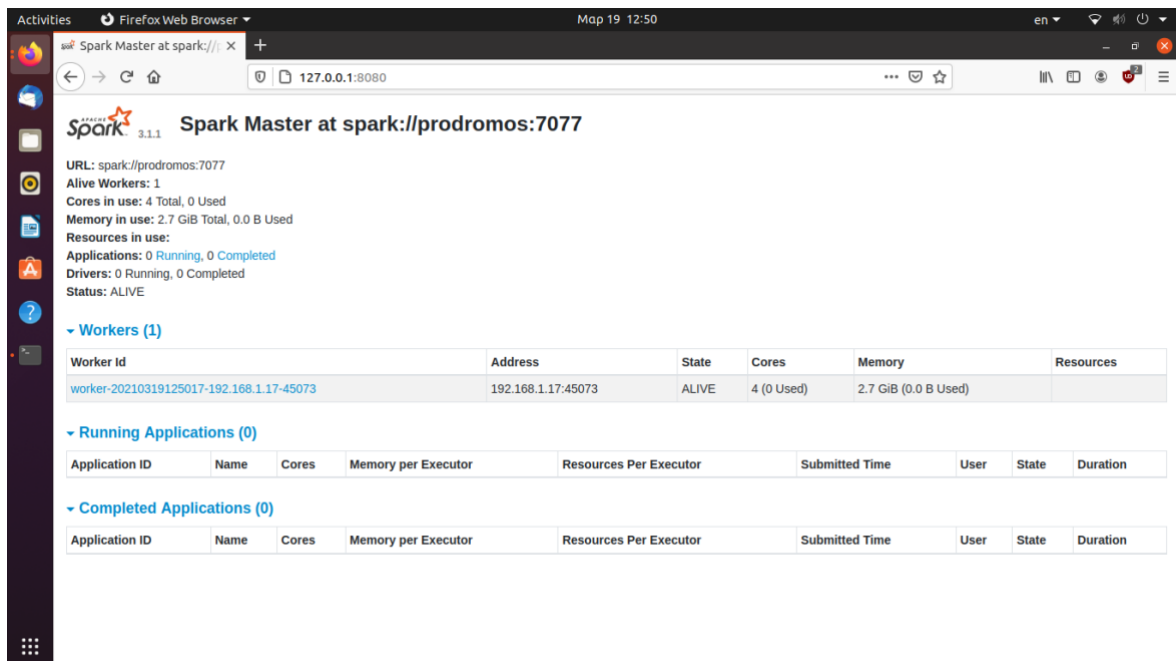
3.2.3 ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ

Για την πρόσβαση στο περιβάλλον του χρήστη του Spark Web, πρέπει να ανοίξουμε τον φυλλομετρητή (browser) που χρησιμοποιούμε και να μεταβούμε στη διεύθυνση URL ή τη διεύθυνση IP του localhost στη θύρα 8080. Πληκτρολογώντας οπότε **http://127.0.0.1:8080/** μπαίνουμε στο περιβάλλον. Έχοντας ξεκινήσει τον master server αυτό που αντικρίζουμε στο περιβάλλον χρήστη και διαχείρισης είναι η παρακάτω εικόνα, η Εικόνα 3.2.3 1, όπου φαίνεται η έκδοση του Spark που χρησιμοποιούμε, το URL, πόσοι workers είναι ενεργοποιημένοι, πόσοι πυρήνες, μνήμη και πόροι χρησιμοποιούνται. Επιπρόσθετα απεικονίζονται οι εφαρμογές και οι drivers που τρέχουν και έχουν ολοκληρωθεί και η κατάσταση του Spark.



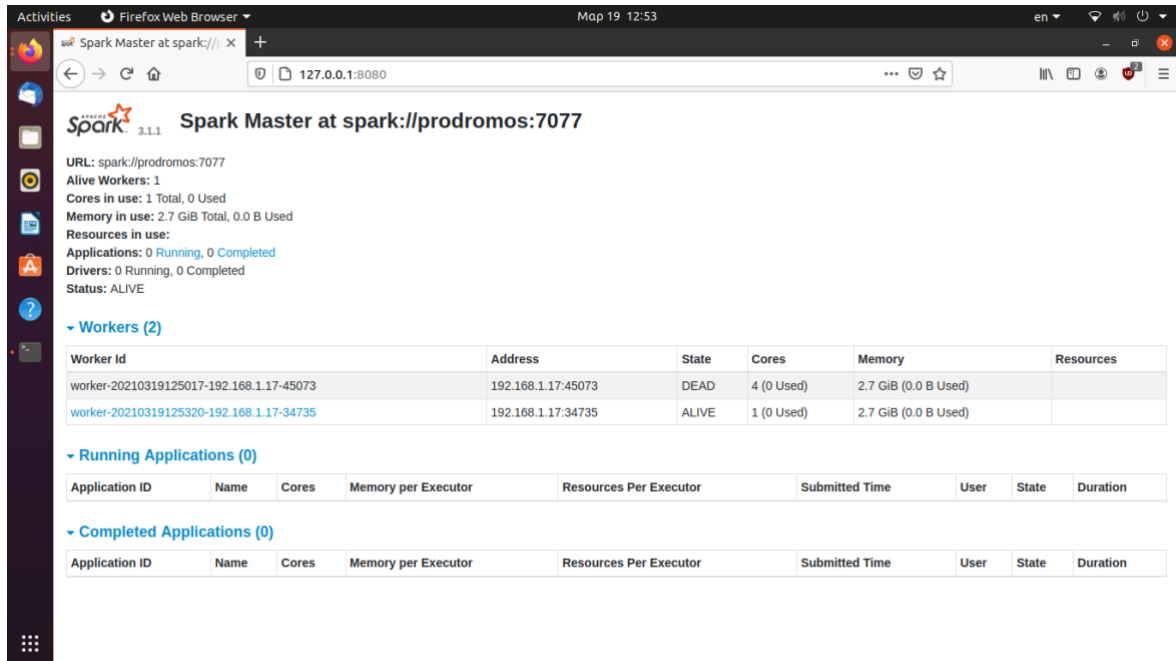
Εικόνα 3.2.3 1 Περιβάλλον χρήση και διαχείρισης Spark (*start-master.sh*)

Ενεργοποιώντας και τον slave server (όπως φαίνεται στην Εικόνα 3.2.3 2) βλέπουμε στο περιβάλλον να έχει συμπληρωθεί ένας worker με το ID του, τη διεύθυνσή του, την κατάστασή του, τους πυρήνες και την μνήμη που χρησιμοποιεί.



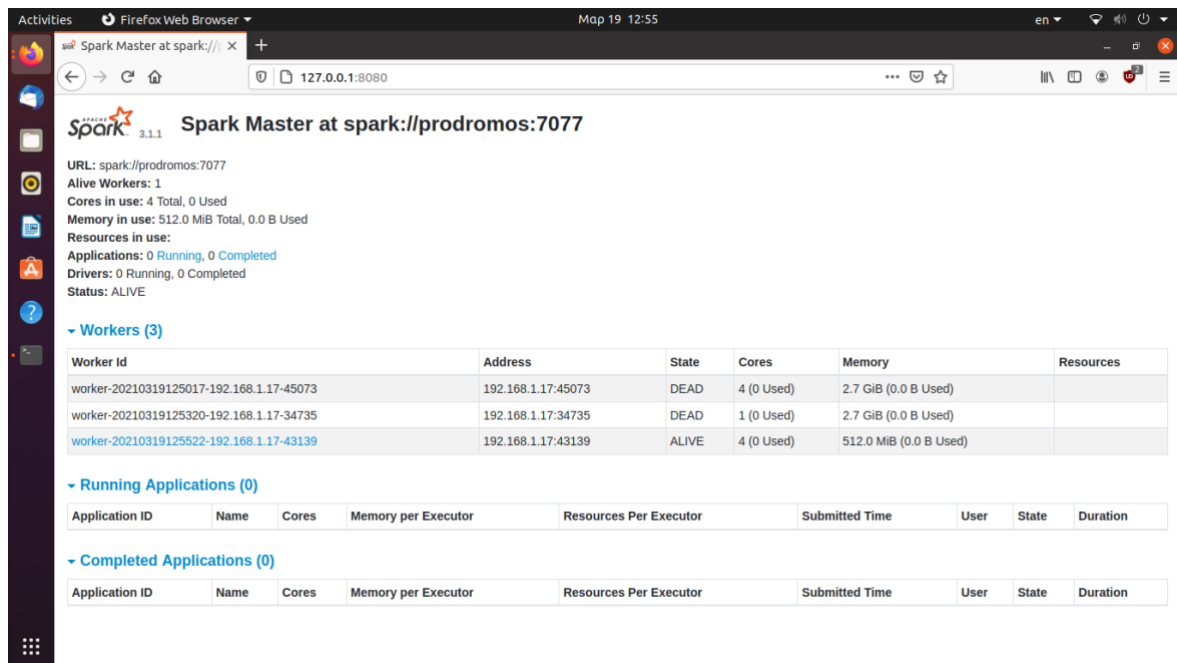
Εικόνα 3.2.3 2 Περιβάλλον χρήσης και διαχείρισης Spark (*start-slave.sh*)

Κατά την εκκίνηση η προεπιλεγμένη ρύθμιση του worker είναι η χρήση όλων των διαθέσιμων πυρήνων. Μία επιλογή ακόμα, είναι να καθορίσουμε οι ίδιοι την κατανομή πόρων για τους workers. Αυτό επιτυγχάνεται με την εντολή “**start-slave.sh -c 1 spark://prodromos:7077**”, όπου το -c 1 σημαίνει πόσοι πυρήνες θέλουμε να χρησιμοποιηθούν.



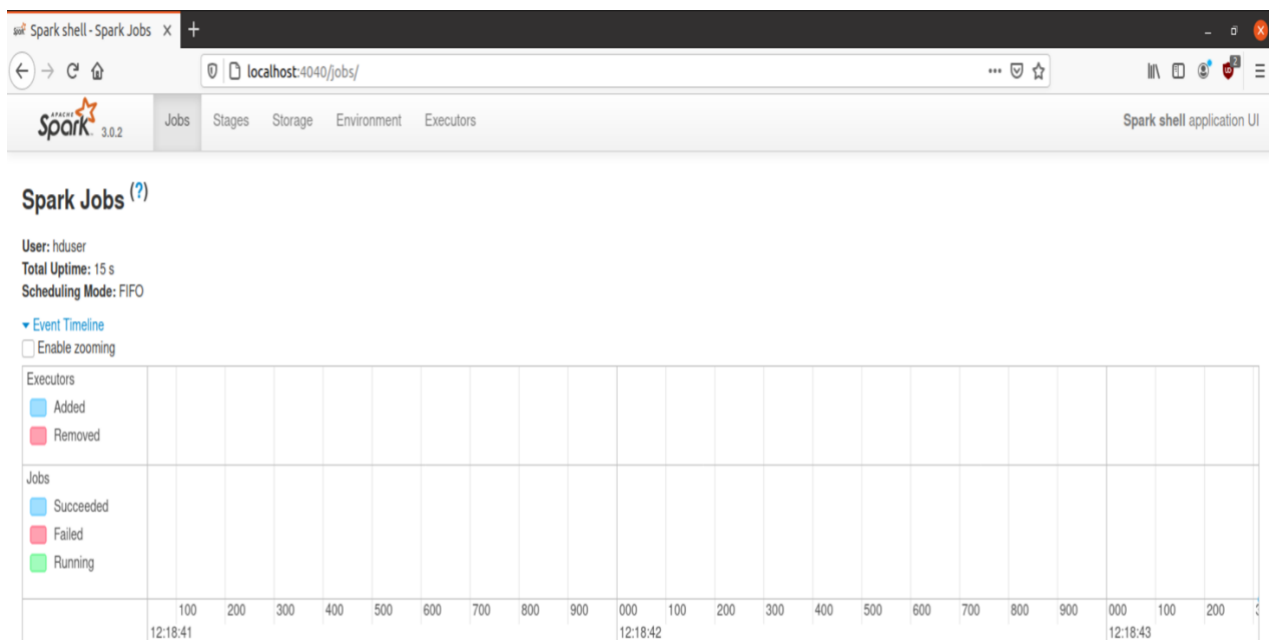
Εικόνα 3.2.3 3 Εκκίνηση worker χρησιμοποιώντας 1 πυρήνα

Με τον ίδιο τρόπο μπορούμε να ορίσουμε πόση μνήμη θέλουμε να διαθέσουμε. Αυτό γίνεται με την εντολή “**start-slave.sh -m 512M spark://prodromos:7077**”, όπου το -m 512M σημαίνει ότι θέλουμε να διαθέσουμε 512 MB. Με την προεπιλεγμένη ρύθμιση χρησιμοποιούμε όση μνήμη διαθέτει το μηχάνημα που έχουμε μείον 1 GB.

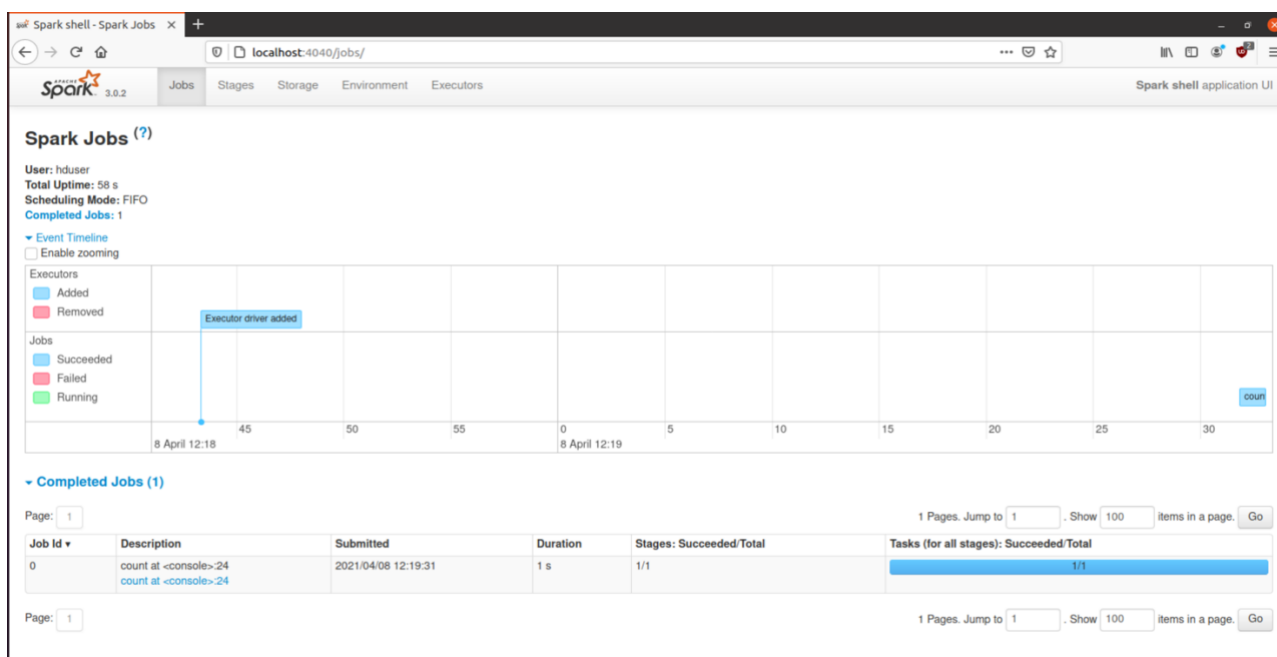


Εικόνα 3.2.3 4 Εκκίνηση worker χρησιμοποιώντας 512 MB

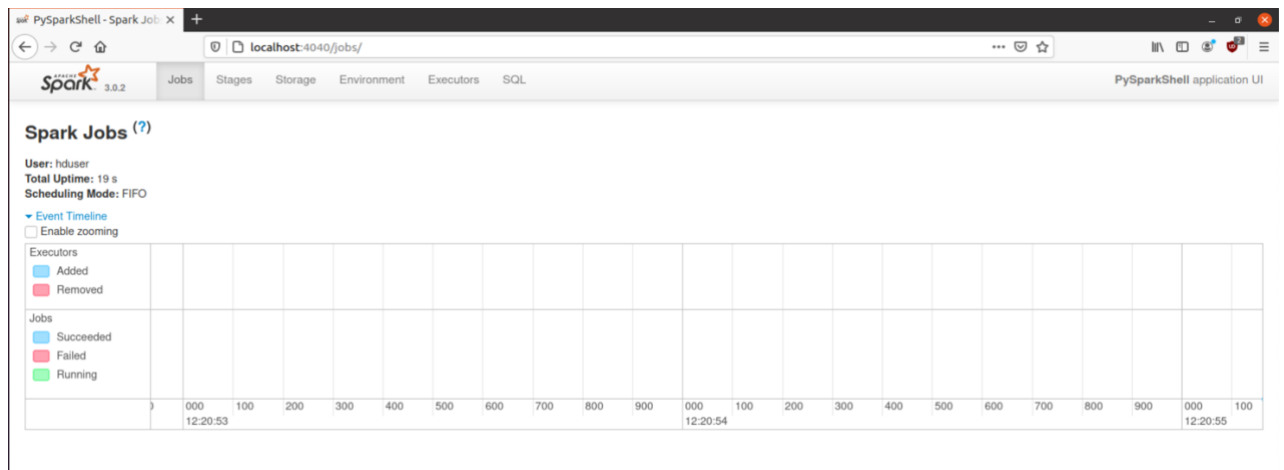
Εκτός από το παραπάνω περιβάλλον, υπάρχουν άλλα δύο, αυτά του spark-shell και του ryspark. Για να μεταβούμε σε αυτά πληκτρολογούμε στον φυλλομετρητή (browser) τη διεύθυνση **http://127.0.0.1:4040/**. Εκεί παρουσιάζονται αρχικά το όνομα του χρήστη, ο συνολικός χρόνος που είναι μέσα στο shell, η μέθοδος για την εξυπηρέτηση των εργασιών, η οποία είναι η FIFO (First In First Out) και οι ολοκληρωμένες εργασίες που έγιναν. Παρακάτω θα παρουσιαστούν κάποιες εικόνες (Εικόνα 3.2.3.5, Εικόνα 3.2.3.6, Εικόνα 3.2.3.7, Εικόνα 3.2.3.8) από το περιβάλλον πριν και αφότου εκτελέσουμε μία εργασία.



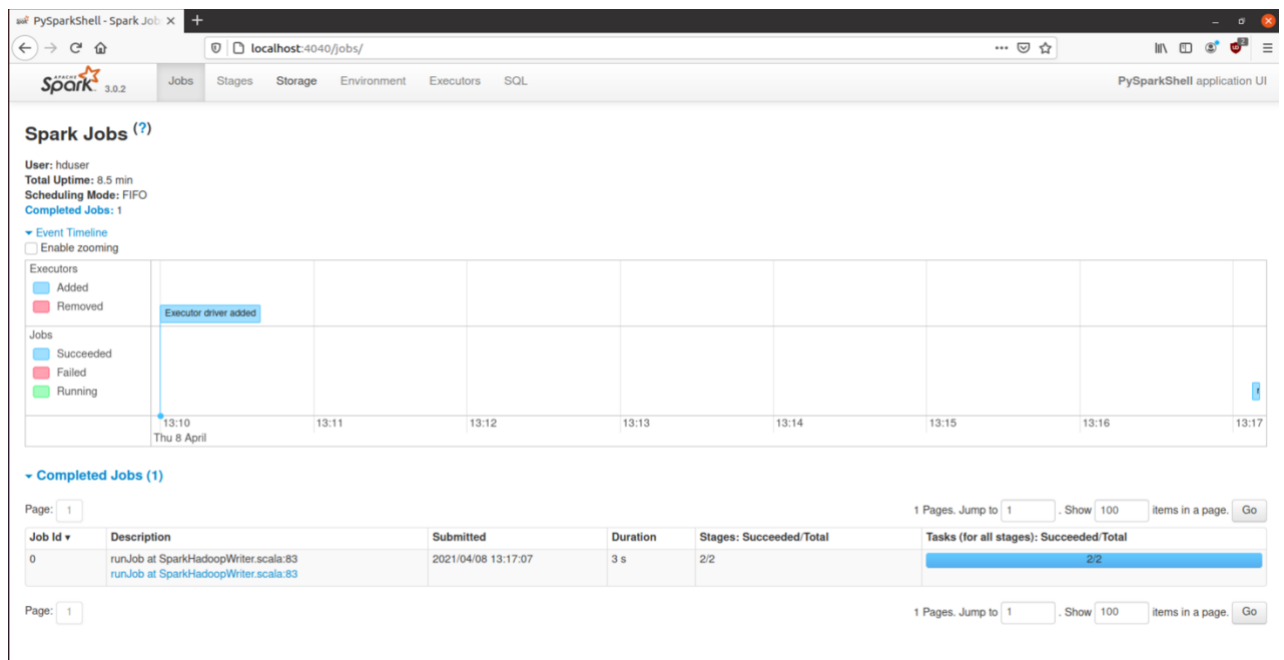
Εικόνα 3.2.3 5 Περιβάλλον χρήστη και διαχείρισης spark-shell (χωρίς να έχει εκτελεστεί κάποια εργασία)



Εικόνα 3.2.3 6 Περιβάλλον χρήστη και διαχείρισης spark-shell (έχοντας εκτελεστεί μία εργασία)

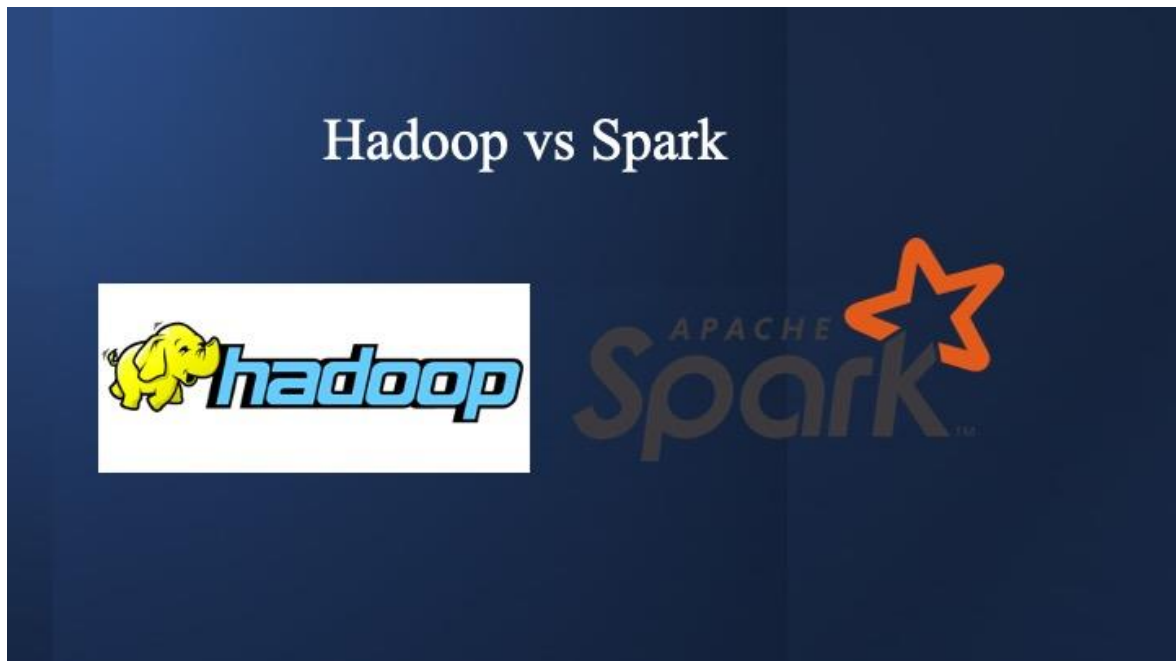


Εικόνα 3.2.3 7 Περιβάλλον χρήστη και διαχείρισης pyspark (χωρίς να έχει εκτελεστεί κάποια εργασία)



Εικόνα 3.2.3 8 Περιβάλλον χρήστη και διαχείρισης pyspark (έχοντας εκτελεστεί μία εργασία)

4 ΣΥΓΚΡΙΣΗ HADOOP ΜΕ SPARK



Εικόνα 4 1 Hadoop VS Spark

Καθώς παραπάνω αναφέρθηκαν τα εργαλεία μοντελοποίησης δεδομένων Apache Hadoop και Apache Spark, τώρα ήρθε η ώρα να γίνει μία συγκριτική μελέτη μεταξύ τους. Η σύγκριση αυτή θα πραγματοποιηθεί με βάση τα κριτήρια της Εκτέλεσης, του Κόστους, της Ανοχής τους σε σφάλματα υλικού, της Επεξεργασίας Δεδομένων, της Ευκολίας στη Χρήση & Υποστήριξης Γλώσσας, της Επεκτασιμότητας, της Ασφάλειας, της Μηχανικής Εκμάθησης και τέλος του Προγραμματισμού και Διαχείρισης Πόρων [22].

- **Performance (Εκτέλεση):**

- **Hadoop:** Το Hadoop παρόλη την πρόσβαση στα δεδομένα που είναι αποθηκευμένα τοπικά στο HDFS (αποτέλεσμα την ενίσχυση της συνολικής απόδοσης) είναι γενικά αργό, διότι εκτελεί εργασίες στο δίσκο. Συγκεκριμένα αποθηκεύει τα δεδομένα σε πολλές διαφορετικές τοποθεσίες και στη συνέχεια τα επεξεργάζεται σε παρτίδες (batches) χρησιμοποιώντας το MapReduce.

- **Spark:** Το Spark αποθηκεύει τα δεδομένα σε RAM και σύμφωνα με την Apache αυτό του προσφέρει τη δυνατότητα να είναι 100 φορές πιο γρήγορο από το Hadoop με το MapReduce. Σημαντική «νίκη» του Spark ήταν όταν κατάφερε να είναι 3 φορές πιο γρήγορο στην ταξινόμηση των δεδομένων σε δίσκους και χρειάστηκε 10 φορές λιγότερους κόμβους για την επεξεργασία 100 TB δεδομένων στο HDFS. Έτσι κατάφερε να δημιουργήσει παγκόσμιο ρεκόρ το 2014.

- **Cost (Κόστος):**

- **Hadoop:** Το Hadoop, όπως και το Spark, είναι λογισμικά ανοιχτού κώδικα και είναι δωρεάν. Παράγοντας του κόστους οπότε είναι το κόστος υποδομής, συντήρησης και ανάπτυξης. Στο Hadoop λόγω της αποθήκευσης των δεδομένων στο δίσκο, το κόστος εκτέλεσης είναι σχετικά χαμηλό. Επιπρόσθετα, χρησιμοποιεί υπολογιστικά συστήματα χαμηλού κόστους
- **Spark:** Το Spark όπως αναφέρθηκε και προηγουμένως είναι κι αυτό λογισμικό ανοιχτού κώδικα και να είναι δωρεάν. Βέβαια, εξαρτάται από υπολογισμούς στη μνήμη για την επεξεργασία δεδομένων σε πραγματικό χρόνο και έτσι η εκκίνηση κόμβων με μεγάλη ποσότητα μνήμης RAM αυξάνει σημαντικά το κόστος

- **Fault Tolerance (Ανοχή σε σφάλματα υλικού):**

- **Hadoop:** Το Hadoop είναι ένα σύστημα όπου διαθέτει αρκετά μεγάλη ανοχή σε σφάλματα υλικού. Αναπαράγει τα δεδομένα στους κόμβους και σε περίπτωση κάποιου προβλήματος το σύστημα συνεχίζει την εργασία δημιουργώντας τα μπλοκ που λείπουν από άλλες τοποθεσίες. Ο κύριος κόμβος παρακολουθεί την κατάσταση των δευτερευόντων κόμβων και αν κάποιος δεν ανταποκρίνεται σε pings από έναν κύριο, ο κύριος κόμβος εκχωρεί τις εκκρεμείς εργασίες σε άλλον.

- **Spark:** Το Spark είναι κι αυτό ένα σύστημα όπου διαθέτει αρκετά μεγάλη ανοχή σε σφάλματα υλικού, αλλά ο τρόπος προσέγγισης σε αυτά είναι διαφορετικός. Συγκεκριμένα, χρησιμοποιεί μπλοκ RDD και παρακολουθεί τον τρόπο δημιουργίας του αμετάβλητου συνόλου δεδομένων. Έπειτα, επανεκκινεί τη διαδικασία εάν τυχόν υπάρξει κάποιο πρόβλημα. Επιπρόσθετα, μπορεί να αναδημιουργήσει δεδομένα σε ένα cluster χρησιμοποιώντας παρακολούθηση DAG (Directed Acyclic Graph) κατά την ροή εργασιών.

- **Data Processing (Επεξεργασία Δεδομένων):**

- **Hadoop:** Το Hadoop για την επεξεργασία δεδομένων χρησιμοποιεί το MapReduce (το οποίο δεν απαιτεί μεγάλη ποσότητα μνήμης RAM για τη διαχείριση μεγάλου όγκου δεδομένων) με σκοπό να χωρίσει ένα μεγάλο σύνολο δεδομένων σε μία συστάδα για παράλληλη ανάλυση. Το Hadoop βασίζεται σε απλό υλικό για την αποθήκευση και κατά συνέπεια είναι πιο κατάλληλο για γραμμική επεξεργασία δεδομένων.
- **Spark:** Το Spark χρησιμοποιεί τα RDDs, όπου το μέγεθός τους συνήθως είναι μεγάλο για να το χειριστεί ένας κόμβος και έτσι τα χωρίζει στους πλησιέστερους κόμβους ώστε να εκτελεί τις λειτουργίες παράλληλα. Επιπρόσθετα, παρέχει τη δυνατότητα της ανάλυσης δεδομένων σε πραγματικό χρόνο.

- **Ease of Use & Language Support (Ευκολία στη Χρήση & Υποστήριξη Γλώσσας):**

- **Hadoop:** Το MapReduce του Hadoop δεν έχει διαδραστική λειτουργία και είναι περίπλοκο. Το framework του βασίζεται σε γλώσσα προγραμματισμού Java και η σύνταξη του κώδικα για MapReduce εργασίες πραγματοποιείται με Java ή Python.

- **Spark:** Το Spark είναι πιο φιλικό προς τον χρήστη διότι παρέχει υποστήριξη για API σε αρκετές γλώσσες προγραμματισμού. Η μητρική του γλώσσα είναι η Scala, αλλά υποστηρίζει και άλλες, όπως Java, Python, R και Spark SQL, με αποτέλεσμα ο χρήστης να διαλέγει όποια γλώσσα προγραμματισμού προτιμάει. Δίνει τη δυνατότητα των spark-shell και pyspark, τα λεγόμενα repl (Read Eval Print Loop), όπου εκεί μπορεί ο χρήστης να κάνει ανάλυση δεδομένων αλληλεπιδραστικά με Scala ή Python. Επιπρόσθετα, υπάρχει και το script, spark-submit όπου χρησιμοποιείται για την εκκίνηση εφαρμογών σε μία συστάδα υπολογιστών. Τέλος, δίνει τη δυνατότητα στον προγραμματιστή να επαναχρησιμοποιεί τον ήδη υπάρχοντα κώδικα και έτσι να κερδίζει σημαντικό χρόνο στην ανάπτυξη εφαρμογών.

- **Scalability (Επεκτασιμότητα):**

- **Hadoop:** Το Hadoop λόγω του HDFS στην επεξεργασία μεγάλων δεδομένων, παρέχει μία εύκολη κλιμάκωση προσθέτοντας κόμβους και δίσκους για την αποθήκευση. Μπορεί και υποστηρίζει δεκάδες χιλιάδες κόμβους, χωρίς να είναι γνωστό κάποιο συγκεκριμένο όριο.
- **Spark:** Επειδή το Spark βασίζεται στη μνήμη RAM για υπολογισμούς, είναι λιγότερο εύκολο να κλιμακωθεί. Μπορεί και υποστηρίζει χιλιάδες κόμβους σε ένα cluster και συγκεκριμένα μερικοί από τους επιβεβαιωμένους αριθμούς περιλαμβάνουν 8000 μηχανήματα με petabytes δεδομένων.

- **Security (Ασφάλεια):**

- **Hadoop:** Το Hadoop είναι εξαιρετικά ασφαλές καθώς λειτουργεί με πολλαπλές μεθόδους ελέγχου ταυτότητας και ελέγχου πρόσβασης. Συγκεκριμένα, για τον έλεγχο ταυτότητας υποστηρίζει Kerberos και LDAP.
- **Spark:** Το Spark δεν θεωρείται καθόλου ασφαλές καθώς η ασφάλεια είναι απενεργοποιημένη από προεπιλογή. Για να θεωρηθεί ασφαλές πρέπει να ενσωματώσει το Hadoop ώστε να μπορεί να χρησιμοποιήσει τις δικές του

μεθόδους.

- **Machine Learning (Μηχανική Εκμάθηση):**

- **Hadoop:** Το Hadoop χρησιμοποιεί τη βιβλιοθήκη Mahout για επεξεργασία δεδομένων, ομαδοποίησης και ταξινόμησης. Επίσης, μία νεότερη είναι η Samsara, η οποία χρησιμοποιεί μία γλώσσα DSL με υποστήριξη Scala και μπορεί να χρησιμοποιηθεί για αλγεβρικές λειτουργίες στη μνήμη και επιτρέπει στους χρήστες να γράφουν δικό τους αλγόριθμο.
- **Spark:** Το Spark χρησιμοποιεί μία ενσωματωμένη βιβλιοθήκη την MLlib, όπου του προσφέρει επαναληπτικούς υπολογισμούς στη μνήμη. Περιλαμβάνει εργαλεία για την εκτέλεση παλινδρόμησης, ταξινόμησης, αξιολόγησης και πολλά άλλα ακόμη. Το Spark με τη βιβλιοθήκη MLlib αποδείχτηκε αρκετά πιο γρήγορο από το Hadoop με τη βιβλιοθήκη Mahout.

- **Scheduling & Resource Management (Προγραμματισμός & Διαχείριση Πόρων):**

- **Hadoop:** Το Hadoop δεν διαθέτει ενσωματωμένο προγραμματιστή, με αποτέλεσμα να χρησιμοποιεί εξωτερικές λύσεις για τον προγραμματισμό και τη διαχείριση πόρων. Το Oozie είναι ένα από τα διαθέσιμα εργαλεία για τον προγραμματισμό ροών εργασιών και το YARN για τη διαχείριση πόρων.
- **Spark:** Το Spark διαθέτει ενσωματωμένα εργαλεία για κατανομή πόρων, προγραμματισμό και παρακολούθηση. Κύριο εργαλείο είναι το DAG, το οποίο είναι υπεύθυνο για τον διαχωρισμό των λειτουργιών σε στάδια.

5 ΕΦΑΡΜΟΓΕΣ

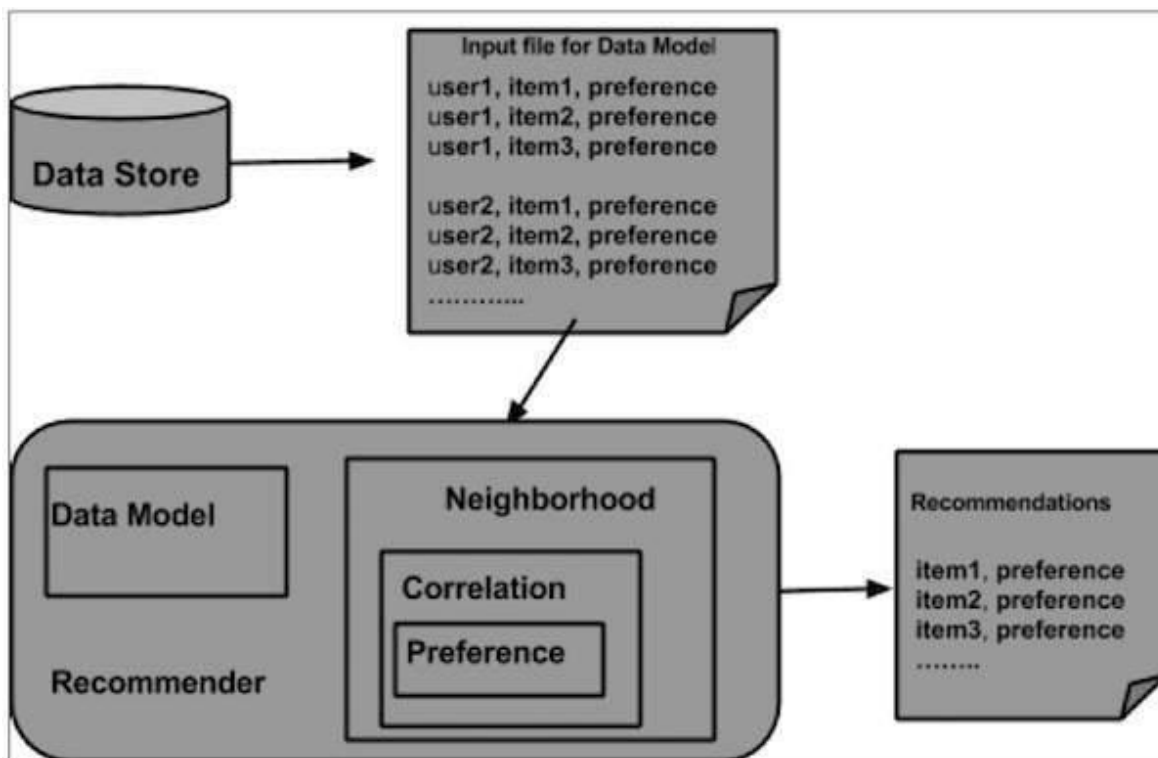
5.1 ΕΦΑΡΜΟΓΕΣ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ ΗΑΔΟΟΡ

Κατά τη διάρκεια της πτυχιακής εργασίας, δημιουργήθηκαν για το Apache Hadoop δύο εφαρμογές. Ένα για συστάσεις ταινιών με τη χρήση του Apache Mahout και ένα για ανάλυση βαθμολογίας με τη χρήση Hadoop Streaming (Οι κώδικες των εφαρμογών υπάρχουν στο ΠΑΡΑΡΤΗΜΑ μετά τη Βιβλιογραφία).

5.1.1 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ ΜΑΗΟΥΤ

Το Apache Mahout είναι ένα εκ των υποσυστημάτων του οικοσυστήματος του Hadoop και είναι ένα έργο ανοιχτού κώδικα, όπου χρησιμοποιείται κυρίως για τη δημιουργία κλιμακούμενων αλγορίθμων μηχανικής εκμάθησης. Δημοφιλείς τεχνικές είναι η ομαδοποίηση, η ταξινόμηση και η σύσταση. Μπορεί και λειτουργεί σε κατανεμημένο περιβάλλον, διότι οι αλγόριθμοί του γράφονται στην κορυφή του Hadoop και προσφέρει στον χρήστη ένα έτοιμο πλαίσιο για την εκτέλεση εργασιών εξόρυξης δεδομένων σε δεδομένα με μεγάλο όγκο. Επιπρόσθετα, επιτρέπει στις εφαρμογές να πραγματοποιούν ανάλυση μεγάλων συνόλων δεδομένων αρκετά γρήγορα και αποτελεσματικά. Το Mahout χρησιμοποιείται από αρκετά γνωστές εταιρείες, όπως Adobe, Facebook, LinkedIn, Twitter και Yahoo. Συγκεκριμένα το Twitter χρησιμοποιεί το Mahout για μοντελοποίηση των ενδιαφερόντων του χρήστη και το Yahoo για την εξόρυξη προτύπων [23]. Ως προς την τεχνική των συστάσεων, έχουμε διάφορους μηχανισμούς, συστάσεις με βάσει τον χρήστη, το αντικείμενο και πολλά άλλα ακόμη. Ο τρόπος με τον οποίο λειτουργεί είναι σχετικά απλός, δίνεις ένα αρχείο κειμένου με προτιμήσεις διάφορων χρηστών για κάποια αντικείμενα και η έξοδος αυτού είναι οι εκτιμώμενες προτιμήσεις ενός συγκεκριμένου χρήστη για άλλα αντικείμενα. Για παράδειγμα, έχουμε μία πλατφόρμα με ταινίες και θέλουμε να εφαρμόσουμε τις δυνατότητες του Mahout, ώστε να δημιουργήσουμε μια μηχανή συστάσεων. Η μηχανή αυτή αναλύει προηγούμενες ταινίες που έχουν παρακολουθήσει οι χρήστες και τους προτείνει νέες βάσει με αυτές που έχουν δει. Η μηχανή αυτή αποτελείται από το Μοντέλο Δεδομένων (DataModel), την Ομοιότητα των χρηστών

(UserSimilarity), την ομοιότητα των αντικειμένων (ItemSimilarity), τον γειτονικό χρήστη (UserNeighborhood) και τις συστάσεις (Recommender). Στην Εικόνα 5.1.1 1 φαίνεται η αρχιτεκτονική της μηχανής των συστάσεων [24].



Εικόνα 5.1.1 1 Αρχιτεκτονική Μηχανής Συστάσεων (Πηγή:
https://www.tutorialspoint.com/mahout/mahout_recommendation.htm)

Τώρα ως προς την τεχνική της ομαδοποίησης. Με την ομαδοποίηση πραγματοποιείται η οργάνωση των στοιχείων ή των αντικειμένων μιας συλλογής σε ομάδες, βάσει την ομοιότητα μεταξύ των αντικειμένων. Η ομαδοποίηση χρησιμοποιείται σε αρκετές εφαρμογές, όπως έρευνα αγοράς, αναγνώριση προτύπων και ανάλυση δεδομένων. Μπορεί να βοηθήσει τους εμπόρους να ανακαλύψουν ξεχωριστές ομάδες στη βάση των πελατών τους. Επιπρόσθετα, χρησιμοποιείται και σε εφαρμογές ανίχνευσης εξωτερικών παραμέτρων, όπως για παράδειγμα η ανίχνευση απάτης με πιστωτικές κάρτες. Οι αλγόριθμοι ομαδοποίησης που υποστηρίζει το Mahout είναι οι εξής, Canopy και K-means. Η ομαδοποίηση Canopy είναι μία γρήγορη τεχνική, όπου τα αντικείμενα αντιμετωπίζονται ως σημεία. Επίσης, χρησιμοποιείται και ως αρχικό βήμα σε άλλους αλγορίθμους όπως ο K-means. Ο K-means είναι ένας αλγόριθμος συστάδων και το K καθορίζει τον αριθμό των συστάδων όπου πρόκειται να διαιρεθούν τα δεδομένα [25]. Τέλος, η τεχνική της ταξινόμησης χρησιμοποιεί γνωστά δεδομένα για να καθορίσει πώς τα νέα δεδομένα πρέπει να ταξινομηθούν σε ένα σύνολο ήδη υπάρχουσών κατηγοριών. Για παράδειγμα, το Spotify

χρησιμοποιεί την ταξινόμηση για την προετοιμασία των λιστών αναπαραγωγής. Επίσης, πάροχοι υπηρεσιών αλληλογραφίας χρησιμοποιούν την τεχνική αυτή ώστε να αποφασίσουν εάν θα ταξινομηθεί η αλληλογραφία στα ανεπιθύμητα. Συγκεκριμένα, ο αλγόριθμος αναλύει τις συνήθειες των χρηστών στην επισήμανση των μηνυμάτων ως ανεπιθύμητα κι έτσι μετά μπορεί και αποφασίζει το που θα τα τοποθετήσει στο μέλλον αυτόματα, στα εισερχόμενα ή στα ανεπιθύμητα [26]. Τώρα ήρθε η ώρα να δείξουμε και την εφαρμογή που δημιουργήσαμε. Για την εφαρμογή μας έχουμε εγκαταστήσει το mahout στο Hadoop, έχουμε κατεβάσει τα δεδομένα που χρειαζόμαστε από το MovieLens και συγκεκριμένα το ml-100k.zip [27], όπου από το οποίο θα χρησιμοποιήσουμε το u.data, το u.user και το u.item. Το u.data αποτελείται από τις πλειάδες, user_id, movie_id, rating και timestamp, το u.user από τις πλειάδες user_id, age, gender, occupation και zip_code και το u.item από τις πλειάδες movie_id, title, release_date, video_release_data, imdb_url, cat_unknown, cat_action, cat_adventure, cat_animation, cat_children, cat_comedy, cat_crime, cat_documentary, cat_drama, cat_fantasy, cat_film_noir, cat_horror, cat_musical, cat_mystery, cat_romance, cat_sci-fi, cat_thriller, cat_war, cat_western. Για αρχή θα χρειαστεί με την εντολή **hadoop fs -put u.data u.data** να φορτώσουμε τα δεδομένα στο hdfs και ύστερα με την εντολή **hadoop jar mahout-mr-0.13.0-job.jar org.apache.mahout.cf.taste.hadoop.item.RecommenderJob -s SIMILARITY_COOCURRENCE -input u.data -output output** να εκτελέσουμε το πρόγραμμα, όπου βρίσκεται μέσα στον φάκελο του mahout και συγκεκριμένα στο path org/apache/mahout/cf/taste/hadoop/item. Το όρισμα **-s SIMILARITY_COOCURRENCE** συμβολίζει τον τύπο προσομοίωσης των αντικειμένων που θα χρησιμοποιηθεί. Επιπρόσθετα, με το SIMILARITY COOCURRENCE δύο ταινίες θα είναι παρόμοιες εάν εμφανίζονται συχνά μαζί στις αξιολογήσεις του χρήστη. Για να βρούμε οπότε τις ταινίες όπου το σύστημα θα προτείνει στον χρήστη, πρέπει πρώτα να βρούμε 10 ταινίες παρόμοιες με αυτές που έχει αξιολογήσει. Το Mahout έπειτα θα υπολογίσει τις προτάσεις εκτελώντας μερικές MapReduce εργασίες, όπως φαίνεται στις Εικόνες 5.1.1 2 και 5.1.1 3.

The screenshot shows the Hadoop cluster management interface. On the left, there's a sidebar with a 'Cluster' menu and a 'Tools' menu. The main area displays 'All Applications' with a table of application metrics. The table has columns for ID, User, Name, Application Type, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, and Allocated CPU. The table lists six applications, all of which are in a 'FINISHED' state with a 'SUCCEEDED' final status. The applications are: application_1624284276694_0002, application_1624284276694_0008, application_1624284276694_0007, application_1624284276694_0006, application_1624284276694_0005, and application_1624284276694_0004. The applications are all of type 'MAPREDUCE' and are running on the 'default' queue. The applications are all of priority 0 and have a 'SUCCEEDED' final status. The applications are all of type 'MAPREDUCE' and are running on the 'default' queue. The applications are all of priority 0 and have a 'SUCCEEDED' final status.

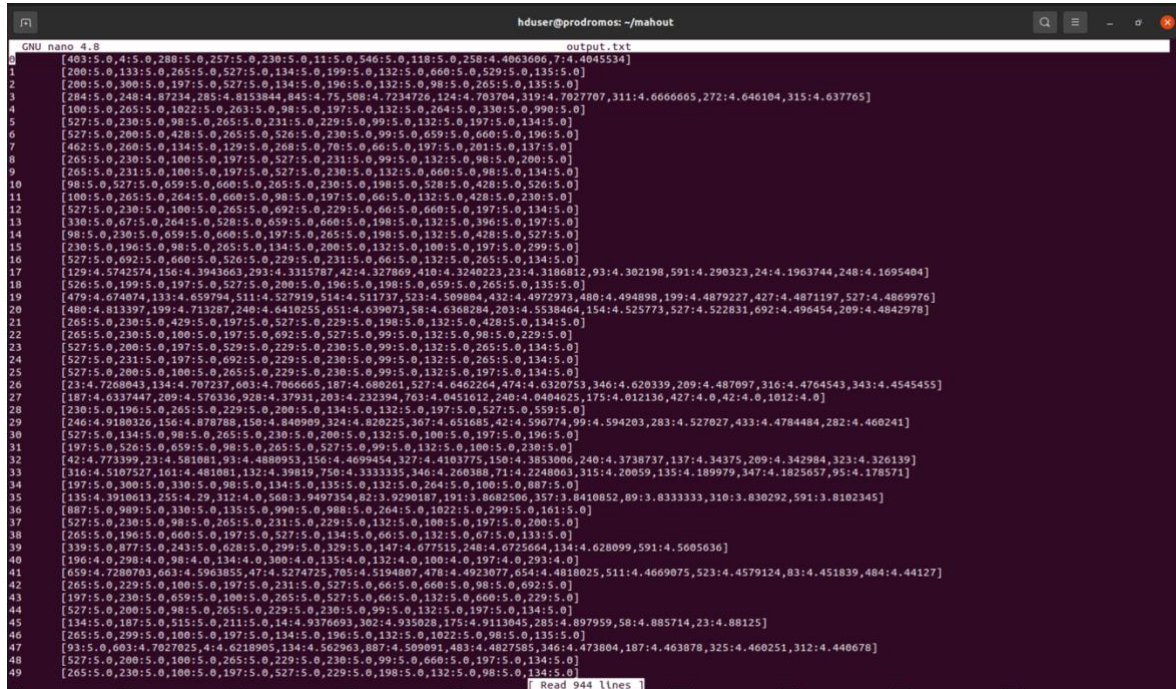
Εικόνα 5.1.1 2 Hadoop Jobs (1)

The screenshot shows the Hadoop cluster management interface. On the left, there's a sidebar with a 'Cluster' menu and a 'Tools' menu. The main area displays 'All Applications' with a table of application metrics. The table has columns for ID, User, Name, Application Type, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU, Allocated Memory, Reserved CPU, Reserved Memory, % of Queue, % of Cluster, Progress, Tracking UI, and Blacklisted Nodes. The table lists six applications, all of which are in a 'FINISHED' state with a 'SUCCEEDED' final status. The applications are: application_1624284276694_0002, application_1624284276694_0008, application_1624284276694_0007, application_1624284276694_0006, application_1624284276694_0005, and application_1624284276694_0004. The applications are all of type 'MAPREDUCE' and are running on the 'default' queue. The applications are all of priority 0 and have a 'SUCCEEDED' final status.

Εικόνα 5.1.1 3 Hadoop Jobs (2)

Μόλις ολοκληρωθούν αυτές οι εργασίες, κάθε χρήστης θα έχει τις 10 ταινίες που είναι πιθανό να του αρέσουν. Επόμενο βήμα είναι να αντιγράψουμε και να συγχωνεύσουμε τα αρχεία από το hdfs στο τοπικό σύστημα αρχείων. Αυτό επιτυγχάνεται με την εντολή **hadoop fs -getmerge output output.txt**. Το txt αρχείο θα είναι όπως στην Εικόνα 5.1.1 4. Κάθε γραμμή συμβολίζει τη σύσταση για τον κάθε χρήστη. Συγκεκριμένα, ο πρώτος αριθμός είναι

το user id και τα 10 ζεύγη αριθμών αντιπροσωπεύουν ένα movie id και ένα score. Για παράδειγμα, για τον χρήστη με id=4 οι 10 προτάσεις ταινιών είναι αυτές με id=100, 265, 1022, 263, 98, 197, 132, 264, 330, 990.



```
GNU nano 4.8 output.txt
1 [403:5.0,4:5.0,288:5.0,257:5.0,238:5.0,11:5.0,546:5.0,118:5.0,258:4.4863686,744.4845534]
2 [200:5.0,133:5.0,265:5.0,527:5.0,134:5.0,199:5.0,132:5.0,668:5.0,529:5.0,135:5.0]
3 [200:5.0,300:5.0,197:5.0,527:5.0,134:5.0,196:5.0,132:5.0,98:5.0,265:5.0,135:5.0]
4 [284:5.0,248:4.87234,285:4.8153844,845:4.75,508:4.7234726,124:4.703704,319:4.7027707,311:4.6666665,272:4.646104,315:4.637765]
5 [100:5.0,265:5.0,1022:5.0,263:5.0,98:5.0,197:5.0,132:5.0,264:5.0,330:5.0,990:5.0]
6 [527:5.0,230:5.0,98:5.0,265:5.0,231:5.0,229:5.0,99:5.0,132:5.0,197:5.0,134:5.0]
7 [527:5.0,200:5.0,428:5.0,265:5.0,526:5.0,230:5.0,99:5.0,659:5.0,660:5.0,196:5.0]
8 [462:5.0,200:5.0,134:5.0,129:5.0,268:5.0,70:5.0,66:5.0,197:5.0,201:5.0,137:5.0]
9 [265:5.0,230:5.0,100:5.0,197:5.0,527:5.0,231:5.0,99:5.0,132:5.0,98:5.0,200:5.0]
10 [265:5.0,231:5.0,100:5.0,197:5.0,527:5.0,231:5.0,99:5.0,132:5.0,98:5.0,134:5.0]
11 [98:5.0,527:5.0,659:5.0,660:5.0,265:5.0,230:5.0,198:5.0,528:5.0,428:5.0,527:5.0]
12 [100:5.0,265:5.0,264:5.0,660:5.0,98:5.0,197:5.0,66:5.0,132:5.0,428:5.0,230:5.0]
13 [527:5.0,230:5.0,100:5.0,265:5.0,692:5.0,229:5.0,66:5.0,660:5.0,197:5.0,134:5.0]
14 [138:5.0,67:5.0,264:5.0,528:5.0,659:5.0,660:5.0,198:5.0,132:5.0,196:5.0,197:5.0]
15 [98:5.0,230:5.0,659:5.0,660:5.0,197:5.0,265:5.0,198:5.0,132:5.0,428:5.0,527:5.0]
16 [230:5.0,196:5.0,98:5.0,265:5.0,134:5.0,200:5.0,132:5.0,100:5.0,197:5.0,299:5.0]
17 [527:5.0,692:5.0,660:5.0,526:5.0,229:5.0,231:5.0,66:5.0,132:5.0,265:5.0,134:5.0]
18 [129:4.5742574,156:4.3943663,293:4.3315787,42:4.327869,410:4.3248223,23:4.3186812,93:4.302198,591:4.290323,24:4.1963744,248:4.1695404]
19 [526:5.0,199:5.0,197:5.0,527:5.0,200:5.0,196:5.0,198:5.0,659:5.0,265:5.0,135:5.0]
20 [470:4.674874,133:4.659794,511:4.527919,514:4.511737,523:4.509804,432:4.4972273,480:4.494898,199:4.4879227,427:4.4871197,527:4.4869976]
21 [480:4.813397,199:4.713287,240:4.6410255,651:4.639073,58:4.6368284,203:4.5538464,154:4.525773,527:4.522831,692:4.496454,209:4.4842978]
22 [265:5.0,230:5.0,429:5.0,197:5.0,527:5.0,229:5.0,198:5.0,132:5.0,428:5.0,134:5.0]
23 [265:5.0,230:5.0,100:5.0,197:5.0,692:5.0,527:5.0,99:5.0,132:5.0,98:5.0,229:5.0]
24 [527:5.0,200:5.0,107:5.0,629:5.0,229:5.0,230:5.0,99:5.0,132:5.0,265:5.0,134:5.0]
25 [527:5.0,231:5.0,197:5.0,692:5.0,229:5.0,230:5.0,99:5.0,132:5.0,265:5.0,134:5.0]
26 [527:5.0,200:5.0,100:5.0,265:5.0,229:5.0,230:5.0,99:5.0,132:5.0,197:5.0,134:5.0]
27 [23:4.7268843,134:4.707237,603:4.7066665,187:4.688261,527:4.6462264,474:4.6320753,346:4.620339,209:4.487097,316:4.4764543,343:4.4545455]
28 [187:4.6337447,209:4.576336,928:4.37931,203:4.232394,763:4.0451612,248:4.0404625,175:4.012136,427:4.0,42:4.0,1012:4.0]
29 [230:5.0,196:5.0,265:5.0,229:5.0,200:5.0,134:5.0,132:5.0,197:5.0,559:5.0]
30 [246:4.9188326,156:4.878788,150:4.840999,324:4.820225,367:4.651685,42:4.596774,99:4.594203,283:4.527027,433:4.4784484,282:4.460241]
31 [527:5.0,134:5.0,98:5.0,265:5.0,230:5.0,200:5.0,132:5.0,100:5.0,197:5.0,196:5.0]
32 [197:5.0,526:5.0,659:5.0,98:5.0,265:5.0,527:5.0,99:5.0,132:5.0,100:5.0,230:5.0]
33 [42:4.773399,23:4.581081,93:4.4888953,156:4.4699454,327:4.4803775,150:4.3833806,248:4.3738737,137:4.34375,209:4.342984,323:4.326139]
34 [316:4.5107527,161:4.481081,132:4.39819,750:4.333333,346:4.260388,71:4.2248063,315:4.20059,135:4.189979,347:4.1825657,95:4.178571]
35 [197:5.0,300:5.0,330:5.0,98:5.0,134:5.0,135:5.0,132:5.0,264:5.0,100:5.0,887:5.0]
36 [135:4.3910613,255:4.29,312:4.0,568:3.9497354,82:3.9298187,191:3.8682506,357:3.8410852,89:3.8333333,310:3.830292,591:3.8102345]
37 [887:5.0,989:5.0,330:5.0,135:5.0,990:5.0,988:5.0,264:5.0,1022:5.0,299:5.0,161:5.0]
38 [527:5.0,230:5.0,98:5.0,265:5.0,231:5.0,229:5.0,99:5.0,132:5.0,100:5.0,197:5.0,200:5.0]
39 [265:5.0,196:5.0,660:5.0,197:5.0,527:5.0,134:5.0,66:5.0,132:5.0,67:5.0,133:5.0]
40 [339:5.0,877:5.0,243:5.0,628:5.0,299:5.0,329:5.0,147:4.677515,248:4.6725664,134:4.628099,591:4.5605636]
41 [196:4.0,298:4.0,98:4.0,134:4.0,300:4.0,135:4.0,132:4.0,100:4.0,197:4.0,293:4.0]
42 [659:4.7288703,683:4.5963855,474:4.5274725,705:4.5194807,478:4.4923077,654:4.4818025,511:4.4669075,523:4.4579124,83:4.451839,484:4.44127]
43 [265:5.0,229:5.0,100:5.0,197:5.0,231:5.0,527:5.0,66:5.0,660:5.0,98:5.0,692:5.0]
44 [197:5.0,230:5.0,659:5.0,100:5.0,265:5.0,527:5.0,66:5.0,132:5.0,660:5.0,229:5.0]
45 [527:5.0,200:5.0,98:5.0,265:5.0,229:5.0,230:5.0,99:5.0,132:5.0,197:5.0,134:5.0]
46 [134:5.0,187:5.0,515:5.0,211:5.0,14:4.9376093,302:4.935028,175:4.9113045,285:4.897959,58:4.885714,23:4.88125]
47 [265:5.0,299:5.0,100:5.0,197:5.0,134:5.0,196:5.0,132:5.0,1022:5.0,98:5.0,135:5.0]
48 [93:5.0,603:4.7027025,4:4.6218905,134:4.562963,887:4.509001,483:4.4827585,346:4.473804,187:4.463878,325:4.460251,312:4.440678]
49 [527:5.0,200:5.0,100:5.0,265:5.0,229:5.0,230:5.0,99:5.0,660:5.0,197:5.0,134:5.0]
50 [265:5.0,230:5.0,100:5.0,197:5.0,527:5.0,229:5.0,198:5.0,132:5.0,98:5.0,134:5.0]
```

Εικόνα 5.1.1 4 output.txt

Για να δούμε όμως και ποιες είναι ακριβώς αυτές οι ταινίες, γράφτηκε και ένα πρόγραμμα σε γλώσσα προγραμματισμού python. Για το πρόγραμμα χρησιμοποιήθηκε το αρχείο u.data για τη λίστα των ταινιών, το αρχείο u.item για τη λίστα των τίτλων των ταινιών και το output.txt για τη λήψη της λίστας των προτεινόμενων ταινιών για τον χρήστη. Για την εκτέλεση του προγράμματος για τις συστάσεις στον χρήστη με id=4, δώσαμε την εντολή **python3 mahout_recommendations.py 4 u.data u.item output.txt**. Το αποτέλεσμα φαίνεται στην Εικόνα 5.1.1 5.

```
hduser@prodromos: ~/mahout
hduser@prodromos:~/mahout$ python3 mahout_recommendations.py 4 u.data u.item output.txt
Reading Movies Descriptions
Reading Rated Movies
Reading Recommendations
Rated Movies
-----
Mimic (1997), rating=5
Ulee's Gold (1997), rating=5
Incognito (1997), rating=5
One Flew Over the Cuckoo's Nest (1975), rating=4
Event Horizon (1997), rating=4
Client, The (1994), rating=3
Liar Liar (1997), rating=5
Scream (1996), rating=4
Star Wars (1977), rating=5
Wedding Singer, The (1998), rating=5
Starship Troopers (1997), rating=4
Air Force One (1997), rating=5
Conspiracy Theory (1997), rating=3
Contact (1997), rating=5
Indiana Jones and the Last Crusade (1989), rating=3
Desperate Measures (1998), rating=5
Seven (Se7en) (1995), rating=4
Cop Land (1997), rating=5
Lost Highway (1997), rating=5
Assignment, The (1997), rating=5
Blues Brothers 2000 (1998), rating=5
Spawn (1997), rating=2
Wonderland (1997), rating=5
In & Out (1997), rating=5
-----
Recommended Movies
-----
Fargo (1996), score=5.0
Hunt for Red October, The (1990), score=5.0
Fast, Cheap & Out of Control (1997), score=5.0
Steel (1997), score=5.0
Silence of the Lambs, The (1991), score=5.0
Graduate, The (1967), score=5.0
Wizard of Oz, The (1939), score=5.0
Mimic (1997), score=5.0
187 (1997), score=5.0
Anna Karenina (1997), score=5.0
-----
hduser@prodromos:~/mahout$
```

Εικόνα 5.1.1 5 Αποτέλεσμα συστάσεων για τον χρήστη με id=4

5.1.2 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ HADOOP STREAMING

Το Hadoop Streaming είναι ένα βοηθητικό πρόγραμμα όπου υπάρχει μέσα στο Hadoop και επιτρέπει τη δημιουργία και την εκτέλεση Map Reduce εργασιών με οποιαδήποτε εκτελέσιμο ή script ως mapper ή και reducer. Συγκεκριμένα βρίσκεται μέσα στον φάκελο `hadoop/share/hadoop/tools/lib`. Στο παράδειγμά μας χρησιμοποιήσαμε 1 σύνολο δεδομένων από το MovieLens (ml-100k.zip) [27], το `u.data`, όπου το αποθηκεύσαμε μέσα στο `hdfs` και αποτελείται από τέσσερις πλειάδες, USER ID, MOVIE ID, RATING και TIMESTAMP. Ο κώδικας γράφτηκε σε γλώσσα προγραμματισμού `python` και επιστρέφει τις βαθμολογίες των χρηστών για τις ταινίες και τον αριθμό όπου πραγματοποιήθηκαν οι αξιολογήσεις. Επιπρόσθετα χρησιμοποιήθηκε και το `mrjob`, το οποίο είναι ένα πακέτο της `python` και βοηθά να γράφονται και να εκτελούνται εργασίες χρησιμοποιώντας Hadoop Streaming. Παρακάτω στις Εικόνες 5.1.2 1 και 5.1.2 2 φαίνεται η εντολή που δόθηκε για να εκτελεστεί το πρόγραμμα και το αποτέλεσμά του.


```
hduser@prodromos: ~  
hduser@prodromos:~$ python3 RatingsBreakdown.py -r hadoop --hadoop-streaming-jar hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar u.data  
No configs found; falling back on auto-configuration  
No configs specified for hadoop runner  
Looking for hadoop binary in /home/hduser/hadoop/bin...  
Found hadoop binary: /home/hduser/hadoop/bin/hadoop  
Using Hadoop version 3.2.1  
Creating temp directory /tmp/RatingsBreakdown.hduser.20210603.172107.683205  
uploading working dir files to hdfs:///user/hduser/tmp/mrjob/RatingsBreakdown.hduser.20210603.172107.683205/files/wd...  
Copying other local files to hdfs:///user/hduser/tmp/mrjob/RatingsBreakdown.hduser.20210603.172107.683205/files/  
Running step 1 of 1...
```

Εικόνα 5.1.2 1 Εκτέλεση RatingsBreakdown.py

```
job output is in hdfs:///user/hduser/tmp/mrjob/RatingsBreakdown.hduser.20210603.172107.683205/output  
Streaming final output from hdfs:///user/hduser/tmp/mrjob/RatingsBreakdown.hduser.20210603.172107.683205/output...  
STDERR: 2021-06-03 20:22:26,908 INFO sasl.SaslDataTransferClient: SASL encryption trust check: LocalHostTrusted = false, remoteHostTrusted = false  
"1" 6110  
"2" 11370  
"3" 27145  
"4" 34174  
"5" 21201  
Removing HDFS temp directory hdfs:///user/hduser/tmp/mrjob/RatingsBreakdown.hduser.20210603.172107.683205...  
Removing temp directory /tmp/RatingsBreakdown.hduser.20210603.172107.683205...  
hduser@prodromos:~$
```

Εικόνα 5.1.2 2 Βαθμολογίες & οι φορές όπου πραγματοποιήθηκαν οι αξιολογήσεις

The screenshot shows the Hadoop All Applications web interface. On the left is a navigation menu with options like 'Cluster', 'About', 'Nodes', 'Node Labels', 'Applications', and 'Tools'. The main area displays 'Cluster Metrics' with a table showing 1 app submitted, 0 pending, 1 running, and 0 completed. Below this is a 'Cluster Nodes Metrics' table showing 1 active node and 0 decommissioning nodes. The 'Scheduler Metrics' section shows the Capacity Scheduler. At the bottom, a table lists running applications, with one entry for 'application_1623166786203_0001' in a 'RUNNING' state.

Cluster Metrics											
	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total				
1	0	1	0	3	4 GB	8 GB					

Cluster Nodes Metrics											
	Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Un						
1	0	0	0	0	0						

Scheduler Metrics											
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation								
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>								

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers
application_1623166786203_0001	hduser	streamjob8148687457891500668.jar	MAPREDUCE	default	0	Tue Jun 8 18:44:46 +0300 2021	Tue Jun 8 18:44:51 +0300 2021	N/A	RUNNING	UNDEFINED	3

Εικόνα 5.1.2 3 Hadoop All Applications UI (έχοντας εκτελεστεί το παράδειγμα) [1^η]

...

🔒

☆

⬇

⌵

📄

🔍

🔴

☰

Logged in as: dr.who

Memory Reserved	VCores Used	VCores Total	VCores Reserved
0 B	3	8	0

Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
0	0	0

Maximum Allocation	Maximum Cluster Application Priority
1024>	0

Search:

App Name	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
ApplicationMaster	3	4096	0	0	50.0	50.0			0

First

Previous

1

Next

Last

Εικόνα 5.1.2 4 Hadoop All Applications UI (έχοντας εκτελεστεί το παράδειγμα) [2η]

5.2 ΕΦΑΡΜΟΓΗ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ SPARK

Κατά τη διάρκεια της πτυχιακής εργασίας, δημιουργήθηκε για το Apache Spark μία εφαρμογή σύστασης ταινιών με τη χρήση της μεθόδου του Collaborative Filtering. Το Collaborative Filtering βασίζεται στη βιβλιοθήκη ALS (Alternating Least Squares) και είναι ένας αλγόριθμος που χρησιμοποιείται για τη δημιουργία συστάσεων. Συγκεκριμένα, συγκεντρώνει προβλέψεις που σχετίζονται με τα ενδιαφέροντα ενός χρήστη, συλλέγοντας παράλληλα προτιμήσεις από πολλούς άλλους χρήστες. Έχει εφαρμοστεί επίσης στο Spark MLlib, ώστε να εκτελεί μεγάλα σύνολα δεδομένων γρήγορα [28]. Στη δικιά μας περίπτωση, χρησιμοποιήσαμε 2 σύνολα δεδομένων από το MovieLens (ml-latest.zip) [29], το movies.csv και το ratings.csv, όπου μέσα έχουν διάφορες ταινίες και κριτικές και διαλέξαμε έναν χρήστη, όπου με βάσει κάποιες ταινίες που έχει δει και κριτικές που έχει κάνει, το σύστημά μας να μπορεί να του προτείνει 10 νέες ταινίες. Η υλοποίηση έγινε με το pyspark και δημιουργήθηκε στατικά με html και javascript μία σελίδα για την εμφάνιση των αποτελεσμάτων. Όπως φαίνεται και στην Εικόνα 5.2 1, για την εκτέλεση του προγράμματος χρειάστηκε να μεταβούμε στον φάκελο όπου αποθηκεύσαμε το πρόγραμμα μέσα στο spark και αυτό έγινε με την εντολή “ **cd ../../opt/spark/Movie_Recommendation/'Final App'/'** ” και έπειτα δώσαμε την εντολή “ **../../bin/spark-submit user_recommendation.py** ” ώστε να υποβάλλουμε το πρόγραμμα. Όπως φαίνεται από την εντολή εκτέλεσης του προγράμματος, η υποβολή στο cluster έγινε με το script spark-submit.

```
masternode@masternode: /opt/spark/ Movie_Recommendation/Final App
masternode@masternode:/opt/spark/ Movie_Recommendation/Final App$ ls
Input.txt      Movie_App_Logger.log  __pycache__  masterfile.mst  recout.py      user_recommendation.py.save
MovieRecOut.html  SparkObj.py          master.py    movies.py       user_recommendation.py
masternode@masternode:/opt/spark/ Movie_Recommendation/Final App$ ../../bin/spark-submit user_recommendation.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/opt/spark/jars/spark-unsafe_2.12-3.0.2.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
+2021-06-08 17:56:12,971
+INFO
+Session with Apache Spark Cluster has begun

+2021-06-08 17:56:13,504
+INFO
+Files open

+2021-06-08 17:56:40,751
+INFO
+Movies:58098-Ratings:27753444

+2021-06-08 17:56:40,751
+INFO
+Data Transformation had been made

+2021-06-08 17:56:41,029
+DEBUG
+Database has been created-Recs:58098

+2021-06-08 17:59:47,826
+DEBUG
+Algorithm executed and recommendations ready for viewing

+2021-06-08 17:59:47,834
+DEBUG
+Evaluation executed after 70.24787386785583 seconds

+2021-06-08 17:59:47,834
+DEBUG
+Root Mean Square Error:0.8419725599729025
```

Εικόνα 5.2 1 Εκτέλεση προγράμματος

 **Spark Master at spark://192.168.104.50:7077**

URL: spark://192.168.104.50:7077

Alive Workers: 5

Cores in use: 16 Total, 0 Used

Memory in use: 26.7 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 1 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (5)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210608175403-192.168.104.51-36871	192.168.104.51:36871	ALIVE	3 (0 Used)	6.0 GiB (0.0 B Used)	
worker-20210608175403-192.168.104.52-33559	192.168.104.52:33559	ALIVE	4 (0 Used)	6.7 GiB (0.0 B Used)	
worker-20210608175403-192.168.104.53-40957	192.168.104.53:40957	ALIVE	3 (0 Used)	6.0 GiB (0.0 B Used)	
worker-20210608175403-192.168.104.56-40671	192.168.104.56:40671	ALIVE	4 (0 Used)	4.0 GiB (0.0 B Used)	
worker-20210608175521-192.168.104.50-37921	192.168.104.50:37921	ALIVE	2 (0 Used)	4.0 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210608175611-0000	Movie Recommendation	16	3.0 GiB		2021/06/08 17:56:11	masternode	FINISHED	7.3 min

Εικόνα 5.2 2 Spark UI (έχοντας εκτελεστεί η εφαρμογή)

Movie Recommendation System For		
User 567		
User Ratings		
MOVIE	RATING	
Ace Ventura: When Nature Calls (1995)	Comedy	
Leaving Las Vegas (1995)	Drama/Romance	
Dangerous Minds (1995)	Drama	
Clueless (1995)	Comedy/Romance	
Mortal Kombat (1995)	Action/Adventure/Fantasy	
"Birdcage The (1996)"	Comedy	
Bad Boys (1995)	Action/Comedy/Crime/Drama/Thriller	
Batman Forever (1995)	Action/Adventure/Comedy/Crime	
"Net The (1995)"	Action/Crime/Thriller	
Nine Months (1995)	Comedy/Romance	
Boys on the Side (1995)	Comedy/Drama	
Circle of Friends (1995)	Drama/Romance	
Disclosure (1994)	Drama/Thriller	
Dumb & Dumber (Dumb and Dumber) (1994)	Adventure/Comedy	
Exit to Eden (1994)	Comedy	
Houseguest (1994)	Comedy	
Little Women (1994)	Drama	
Legends of the Fall (1994)	Drama/Romance/War/Western	
Milk Money (1994)	Comedy/Romance	
Pulp Fiction (1994)	Comedy/Crime/Drama/Thriller	
"Santa Clause The (1994)"	Comedy/Drama/Fantasy	
Tommy Boy (1995)	Comedy	
While You Were Sleeping (1995)	Comedy/Romance	
Ace Ventura: Pet Detective (1994)	Comedy	
"Client The (1994)"	Drama/Mystery/Thriller	
Forrest Gump (1994)	Comedy/Drama/Romance/War	
Four Weddings and a Funeral (1994)	Comedy/Romance	
"Lion King The (1994)"	Adventure/Animation/Children/Drama/Musical/IMAX	

Εικόνα 5.2 3 Ταινίες που έχει δει ο χρήστης με ID=567 (i)

"Mask The (1994)"	Action/Comedy/Crime/Fantasy
True Lies (1994)	Action/Adventure/Comedy/Romance/Thriller
When a Man Loves a Woman (1994)	Drama/Romance
City Slickers II: The Legend of Curly's Gold (1994)	Adventure/Comedy/Western
Dave (1993)	Comedy/Romance
"Firm The (1993)"	Drama/Thriller
Jurassic Park (1993)	Action/Adventure/Sci-Fi/Thriller
"Man Without a Face The (1993)"	Drama
Philadelphia (1993)	Drama
Sleepless in Seattle (1993)	Comedy/Drama/Romance
"Brady Bunch Movie The (1995)"	Comedy
Home Alone (1990)	Children/Comedy
Ghost (1990)	Comedy/Drama/Fantasy/Romance/Thriller
Aladdin (1992)	Adventure/Animation/Children/Comedy/Musical
Terminator 2: Judgment Day (1991)	Action/Sci-Fi
Dances with Wolves (1990)	Adventure/Drama/Western
Batman (1989)	Action/Crime/Thriller
Snow White and the Seven Dwarfs (1937)	Animation/Children/Drama/Fantasy/Musical
Pretty Woman (1990)	Comedy/Romance
TOTAL RATINGS	47

Εικόνα 5.2 4 Ταινίες που έχει δει ο χρήστης με ID=567 (ii)

Movie Recommendation System For		
User 567		
567 Recommendations		
MOVIE	GENRE	
Desirable Teacher (1981)	Comedy	
Shala (2011)	Drama	
Duelle (une quarantaine) (1976)	Drama/Fantasy/Mystery	
Peculiarities of the National Ice Fishing (2007)	Comedy	
Celestial Subway Lines/Salvaging Noise (2005)	Documentary	
"Wedding March The (1928)"	Drama/Romance	
Retrieval (2006)	(no genres listed)	
Trail of the Screaming Forehead (2007)	Comedy/Sci-Fi	
DeadHeads (2011)	Adventure/Comedy/Horror	
WUSA (1970)	Drama	

Εικόνα 5.2 5 Ταινίες που προτείνει το σύστημα στον χρήστη με ID=567

6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα συμπεράσματα που προέκυψαν κατά την εκπόνηση της εργασίας είναι τα εξής. Αρχικά, ως προς τις εγκαταστάσεις των δύο αυτών εργαλείων παρατηρούμε πως η εγκατάσταση του Apache Hadoop παρουσιάζει μεγαλύτερο βαθμό πολυπλοκότητας από την αντίστοιχη εγκατάσταση του Apache Spark. Το Hadoop έχει αρκετές παραμετροποιήσεις όπου πρέπει να δημιουργηθούν, έχει το `core-site.xml`, το `hdfs-site.xml`, το `mapred-site.xml` και το `yarn-site.xml`. Επίσης, χρειάζεται να δημιουργηθούν και κάποιοι φάκελοι (`tmpdata`, `dfsdata/namenode`, `dfsdata/datanode`), αλλιώς θα υπάρξει σφάλμα κατά την εκκίνηση των κόμβων. Αντίθετα στο Spark, μετά τους ορισμούς όπου πρέπει να γίνουν στο `profile`, ο χρήστης είναι έτοιμος να ξεκινήσει τους κόμβους χωρίς να χρειάζεται να κάνει κάτι άλλο. Παρόλα αυτά, υπάρχει και η λύση από τις εταιρείες MapR και Cloudera σε συνεργασία με τη Hortonworks, δίνοντας έτοιμο το περιβάλλον των εργαλείων σε `virtual machines`. Και τα δύο εργαλεία είναι πολύ χρήσιμα ως προς την επεξεργασία Μεγάλων Δεδομένων. Συνδυάζοντάς τα ειδικά, πιστεύω είναι ο καλύτερος τρόπος διαχείρισής τους. Όπου υστερεί το ένα το άλλο εργαλείο έρχεται και το συμπληρώνει. Το Hadoop εκτελεί τις εργασίες στον δίσκο με αποτέλεσμα να είναι σχετικά αργό, σε αντίθεση με το Spark όπου αποθηκεύει τα δεδομένα σε RAM. Το Hadoop χρησιμοποιεί υπολογιστικά συστήματα χαμηλού κόστους, ενώ το Spark εξαρτάται από υπολογισμούς στη μνήμη για την επεξεργασία δεδομένων σε πραγματικό χρόνο, με αποτέλεσμα η εκκίνηση των κόμβων με μεγάλη ποσότητα μνήμης RAM να αυξάνει σημαντικά το κόστος. Ως προς την ανοχή σε σφάλματα υλικού, το Hadoop αναπαράγει τα δεδομένα σε κόμβους και σε περίπτωση κάποιου προβλήματος το σύστημα συνεχίζει την εργασία δημιουργώντας τα μπλοκ που λείπουν από άλλες τοποθεσίες. Το Spark αντίθετα μπορεί να αναδημιουργήσει δεδομένα σε ένα cluster χρησιμοποιώντας παρακολούθηση DAG. Το Hadoop είναι αρκετά ασφαλές, σε αντίθεση με το Spark που έχει από προεπιλογή την ασφάλεια κλειστή. Τέλος, το Hadoop παρέχει μία εύκολη κλιμάκωση προσθέτοντας κόμβους και δίσκους για την αποθήκευση, ενώ το Spark είναι λιγότερο εύκολο να κλιμακωθεί, αφού βασίζεται στη μνήμη RAM για υπολογισμούς. Επιπρόσθετα, ένα από τα πλεονεκτήματά τους είναι ότι μπορούν να εκτελεστούν από υπολογιστές με hardware το οποίο δεν πρέπει να είναι απαραίτητα υψηλών προδιαγραφών. Μπορούν εύκολα να εκτελεστούν και μέσα από `virtual machines`. Στην περίπτωση μας πραγματοποιήθηκαν αρκετά πειράματα τοπικά σε VM. Εν κατακλείδι, μπορεί κάποιος να κάνει αρκετά πράγματα με αυτά τα εργαλεία, όπως η δημιουργία εφαρμογών συστάσεων

ταινιών (όπως δείξαμε κι εμείς), εφαρμογές ανίχνευσης κάποιας ηλεκτρονικής απάτης και γενικά οτιδήποτε έχει να κάνει με Μεγάλα Δεδομένα, δεν είναι τυχαίο εξάλλου που χρησιμοποιούνται και από τόσο μεγάλες εταιρείες. Όποιος θέλει να ασχοληθεί με τον κλάδο των Big Data, θα ήταν πολύ ενδιαφέρον και αρκετά χρήσιμο για εκείνον να ασχοληθεί και με το Hadoop και το Spark.

BIBΛΙΟΓΡΑΦΙΑ

- [1] D. Gewirtz, “Volume, velocity, and variety: Understanding the three V’s of big data.” 2018, [Online]. Available: <https://www.zdnet.com/article/volume-velocity-and-variety-understanding-the-three-vs-of-big-data/>.
- [2] C. P. Kasula, “Netflix Recommender System — A Big Data Case Study.” 2020, [Online]. Available: <https://towardsdatascience.com/netflix-recommender-system-a-big-data-case-study-19cfa6d56ff5>.
- [3] S. Gutta, “Data Science: The 5 V’s of Big Data.” 2020, [Online]. Available: <https://suryagutta.medium.com/the-5-vs-of-big-data-2758bfcc51d>.
- [4] R. Kiran, “Top Big Data Technologies that you Need to know.” 2020, [Online]. Available: <https://www.edureka.co/blog/top-big-data-technologies/>.
- [5] Apache Software Foundation, “Apache Hadoop.” 2021, [Online]. Available: <http://hadoop.apache.org/>.
- [6] H. V. Karambelkar, *Apache Hadoop 3 Quick Start Guide*. Packt Publishing, 2018.
- [7] Apache Software Foundation, “HDFS Architecture.” 2021, [Online]. Available: <https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>.
- [8] Apache Software Foundation, “Apache Hadoop YARN.” 2021, [Online]. Available: <https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html>.
- [9] Apache Software Foundation, “MapReduce Tutorial.” 2021, [Online]. Available: <https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>.
- [10] D. Kvasnicka and G. Roda, “BIG DATA ON VSC,” p. 63, 2021.
- [11] R. Kiran, “MapReduce Tutorial – Fundamentals of MapReduce with MapReduce Example.” 2020, [Online]. Available: <https://www.edureka.co/blog/mapreduce-tutorial/>.
- [12] S. Sinha, “Hadoop Ecosystem: Hadoop Tools for Crunching Big Data.” 2020, [Online]. Available: <https://www.edureka.co/blog/hadoop-ecosystem>.
- [13] Databricks, “Apache Spark.” 2021, [Online]. Available: <https://databricks.com/glossary/what-is-apache-spark>.
- [14] S. Penchikala, *BIG-DATA PROCESSING WITH APACHE SPARK*. C4Media, 2018.
- [15] DataFlair, “Apache Spark SQL Tutorial – Quick Introduction Guide.” 2021, [Online]. Available: <https://data-flair.training/blogs/apache-spark-sql/>.
- [16] DataFlair, “Spark Streaming Tutorial for Beginners.” 2021, [Online]. Available: <https://data-flair.training/blogs/apache-spark-streaming-tutorial/>.
- [17] Apache Spark, “Machine Learning Library (MLlib) Guide.” 2021, [Online]. Available: <https://spark.apache.org/docs/latest/ml-guide.html>.
- [18] DataFlair, “Spark MLlib Data Types | Apache Spark Machine Learning.” 2021, [Online]. Available: <https://data-flair.training/blogs/spark-mllib-data-types/>.
- [19] Databricks, “Apache Spark Ecosystem.” 2021, [Online]. Available: <https://databricks.com/spark/about>.
- [20] DataFlair, “GraphX API in Apache Spark: An Introductory Guide.” 2021, [Online]. Available: <https://data-flair.training/blogs/graphx-api-spark/>.
- [21] Apache Spark, “Graphx Programming Guide.” 2021, [Online]. Available: <https://spark.apache.org/docs/latest/graphx-programming-guide.html>.
- [22] phoenixNAP, “Hadoop vs Spark - Detailed Comparison.” 2020, [Online]. Available: <https://phoenixnap.com/kb/hadoop-vs-spark>.
- [23] tutorialspoint, “Mahout - Quick Guide.” 2021, [Online]. Available:

- https://www.tutorialspoint.com/mahout/mahout_quick_guide.htm.
- [24] tutorialspoint, “Mahout - Recommendation.” 2021, [Online]. Available: https://www.tutorialspoint.com/mahout/mahout_recommendation.htm.
- [25] tutorialspoint, “Mahout - Clustering.” 2021, [Online]. Available: https://www.tutorialspoint.com/mahout/mahout_clustering.htm.
- [26] tutorialspoint, “Mahout - Classification.” 2021, [Online]. Available: https://www.tutorialspoint.com/mahout/mahout_classification.htm.
- [27] grouplens, “MovieLens 100K Dataset.” 2016, [Online]. Available: <https://grouplens.org/datasets/movielens/100k/>.
- [28] S. Rosaria, “Movie Recommendations with Spark Collaborative Filtering.” 2018, [Online]. Available: <https://www.datanami.com/2018/11/02/movie-recommendations-with-spark-collaborative-filtering/>.
- [29] grouplens, “MovieLens Latest Datasets.” 2018, [Online]. Available: <https://grouplens.org/datasets/movielens/latest/>.

ΠΑΡΑΡΤΗΜΑ

Α. ΚΩΔΙΚΑΣ ΑΡΑΧΕ ΗΑΔΟΟΡ

Α.Α ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ ΣΥΣΤΑΣΕΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΑΡΑΧΕ ΜΑΗΟΥΤ

- Mahout_recommendations.py

```
import sys

# Έλεγχος του αριθμού των ορισμάτων τα οποία έδωσε ο χρήστης κατά την εκτέλεση της εντολής καταχώρησης της
εργασίας στο cluster
if len(sys.argv) != 5:
    print("Arguments: userId userDataFilename movieFilename recommendationFilename")
    sys.exit(1)

# Εκχώρηση δεδομένων σε 4 μεταβλητές
userId, userDataFilename, movieFilename, recommendationFilename = sys.argv[1:]

print("Reading Movies Descriptions")
# Άνοιγμα αρχείου
movieFile = open(movieFilename)
# Δημιουργία dictionary
movieById = {}
# Προσέλαση του movieFile
for line in movieFile:
    # Αποθήκευση πληροφοριών ανά ταινία
    tokens = line.split("|")
    movieById[tokens[0]] = tokens[1:]
movieFile.close()

print("Reading Rated Movies")
# Άνοιγμα αρχείου
userDataFile = open(userDataFilename)
# Δημιουργία λίστας
ratedMovieIds = []
for line in userDataFile:
```

```

# Split τα δεδομένα με βάση το κενό
tokens = line.split("\t")
if tokens[0] == userId:
    ratedMovies.append((tokens[1],tokens[2]))
userDataFile.close()

print("Reading Recommendations")
# Βρίσκει τις αξιολογήσεις ανά ταινία
recommendationFile = open(recommendationFilename)
recommendations = []
for line in recommendationFile:
    tokens = line.split("\t")
    if tokens[0] == userId:
        movieIdAndScores = tokens[1].strip("\n").split(",")
        recommendations = [ movieIdAndScore.split(":") for movieIdAndScore in movieIdAndScores ]
        break
recommendationFile.close()

print("Rated Movies")
print("-----")
for movieId, rating in ratedMovies:
    print("%s, rating=%s" % (movieById[movieId][0], rating))
print("-----")
# Έναρξη προτάσεων ανά ταινία
print("Recommended Movies")
print("-----")
for movieId, score in recommendations:
    print("%s, score=%s" % (movieById[movieId][0], score))
print("-----")

```

Α.Β ΚΩΔΙΚΑΣ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΑΝΑΛΥΣΗΣ ΒΑΘΜΟΛΟΓΙΑΣ ΜΕ ΤΗ ΧΡΗΣΗ HADOOP STREAMING

- RatingsBreakdown.py

```

from mrjob.job import MRJob
from mrjob.step import MRStep

class RatingsBreakdown(MRJob):
    # Define single mapreduce phase
    def steps(self):

```

```

return [
    MRStep(mapper=self.mapper_get_ratings,
           reducer=self.reducer_count_ratings)
]

# Define mapper to break up lines of data on tab, return key value pair of rating and 1
def mapper_get_ratings(self, _, line):
    (userID, movieID, rating, timestamp) = line.split("\t")
    yield rating, 1

# Reduce key values, return rating and rating count
def reducer_count_ratings(self, key, values):
    yield key, sum(values)

if __name__ == '__main__':
    RatingsBreakdown.run()

```

B. ΚΩΔΙΚΑΣ ΑΡΑΧΕ SPARK

Β.Α ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ ΣΥΣΤΑΣΕΩΝ ΤΑΙΝΙΩΝ ΜΕ ΤΗ ΧΡΗΣΗ COLLABORATIVE FILTERING, ALS ΚΑΙ MLLIB

- user_recommendation.py (Movie Recommendation System)

```
import pyspark as ps
from pyspark.sql import SparkSession, DataFrame
import pyspark.sql as sql
from pyspark import SparkConf, SparkContext, RDD
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
import master as ms
import os
import sys
from time import time
import webbrowser
import logging as log
import SparkObj as so
import recout as rc

#Create a logger
logger=log.getLogger('Movie Recommendation Logger')
logger.setLevel(log.DEBUG)
fh=log.FileHandler('Movie_App_Logger.log','w')
sh=log.StreamHandler()
formatter=log.Formatter('%(asctime)s\n'%(levelname)s\n'%(message)s\n\n')
fh.setFormatter(formatter)
sh.setFormatter(formatter)
logger.addHandler(fh)
logger.addHandler(sh)

#Handler functions Συναρτήσεις για επεξεργασία δεδομένων
# Επιστρέφει λίστα από τις λέξεις από τις οποίες έχουν προκύψει
# έπειτα από διαχωρισμό ενός αλφαριθμητικού με βάσει το κόμμα (,)
def SplitCsv(s):
    return [x for x in s.split(',')]

def CreateTuple(m): # Μετατροπή της λίστας σε πλειάδα
```

```

        return (m[0],".join(m[1:len(m)-1]),m[len(m)-1])

def RatingTuple(s): # Κατασκευή υποπλειάδας με βάσει μια πλειάδα
    return (s[0],s[1],s[2])
    #Throw timestamp

def DbModel(elem): # Μετατροπή πλειάδας σε αντίστοιχη πλειάδα διαφορετικού μήκους
    return (elem[0],[elem[1],elem[2]])
def RemoveFiles():
    pages=os.path.join(".", 'Static_Pages')
    #os.removedirs(pages)
    for x in os.listdir(pages):
        os.remove(os.path.join(pages,x))

#-----

#Spark Session Initiate and Configure Spark Parameters
# Παραμετροποίηση πόρων εφαρμογής
conf=SparkConf().setAppName('Movie
Recommendation').setMaster(ms.master().Master()).set('spark.executor.memory','3G').set('spark.ui.showCons
oleProcess',True)
# Συνεδρία με cluster.
# Δημιουργεί ή αν υπάρχει συνεδρία με το cluster εκχωρεί την εφαρμογή στην υπάρχουσα συνεδρία
session=SparkSession.builder.config(conf=conf).getOrCreate()
sc=session.sparkContext
logger.info('Session with Apache Spark Cluster has began')

#Open Datasets and transform them
moviespath=os.path.join('.', 'ml-latest', 'movies.csv')
ratingspath=os.path.join('.', 'ml-latest', 'ratings.csv')

#In order to create an instict DataFrame
#movieFrame=sc.read.csv(moviespath)
#EratingsFrame=sc.read.csv(ratingspath)
#Alternative-->txt,json

#Open File Data
movies_raw_data=sc.textFile(moviespath) # Κατασκευή RDD
ratings_raw_data=sc.textFile(ratingspath) # Κατασκευή RDD
logger.info('Files open')

```

```

mheaderstemplate=movies_raw_data.take(1)[0] # Παίρνει το 1ο στοιχείο του RDD (επικεφαλίδες)
rheaderstemplate=ratings_raw_data.take(1)[0] # Παίρνει το 1ο στοιχείο του RDD (επικεφαλίδες)
# filtering των δεδομένων. Απαλοιφή πρώτης στήλης
movies=movies_raw_data.filter(lambda x:x!=mheaderstemplate).map(lambda x:SplitCsv(x)).map(lambda
y:CreateTuple(y))
# filtering των δεδομένων. Απαλοιφή πρώτης στήλης
ratings=ratings_raw_data.filter(lambda x:x!=rheaderstemplate).map(lambda x:SplitCsv(x)).map(lambda
y:RatingTuple(y))
mheaders=mheaderstemplate.split(',') # Βρίσκουμε τις επικεφαλίδες του αρχείου movies.csv (movieId, name,
genre)
rheaders=rheaderstemplate.split(',') # Βρίσκουμε τις επικεφαλίδες του αρχείου ratings.csv (userId, movieId,
rating, timestamp)
rheaders.remove('timestamp') # Αφαίρεση στήλης timespamp (userId, movieId, rating)
logger.info(f'Movies:{movies.count()}-Ratings:{ratings.count()}')
logger.info('Data Transformation had been made')
#create a PseudoDatabase for finding a movie name
moviedb=movies.map(lambda k:(k[0],[k[1],k[2]])).collectAsMap() #moviedb['500'][0]-->NM [1]-->genre
logger.debug(f'Database has been created-Recs:{len(moviedb)}')

#create the appropriate DataFrames
movieFrame=movies.toDF(schema=mheaders) # Μετατρέπουμε το rdd σε dataframe
# Επεξεργασία των υπάρχοντων δεδομένων του dataset χωρίς να αλλάζει τον συνολικό αριθμό των δεδομένων
ratingsFrame=ratings.map(lambda elem:(int(elem[0]),int(elem[1]),float(elem[2]))).toDF(schema=rheaders)

#Split Data into Training and Test Set
train,test=ratingsFrame.randomSplit([0.7,0.3])

#Start working with the Data and execute alternating least square algorithm
als=ALS() # Κατασκευή αντικειμένου τύπου als
# Ορισμός παραμέτρων
als.setMaxIter(7).setRegParam(0.01).setUserCol(rheaders[0]).setItemCol(rheaders[1]).setRatingCol(rheaders[
2]).setColdStartStrategy('drop').setImplicitPrefs(False)
model=als.fit(train) # Εκπαίδευση αλγορίθμου
prediction=model.transform(test) # Παραγωγή προβλέψεων
# Αξιολόγηση το μοντέλου πρόβλεψης που φτιάξαμε με χρήση του μέτρου Root Mean Square Error
evaluator=RegressionEvaluator(metricName='rmse',labelCol='rating',predictionCol='prediction')
start=time()
rmse=evaluator.evaluate(prediction) # Αξιολόγηση με βάσει τις προβλέψεις
evaluationtime=time()-start
logger.debug('Algorithm executed and recommendations ready for viewing')

```

```

logger.debug(f'Evaluation executed after {evaluationtime} seconds')
logger.debug(f'Root Mean Square Error:{rmse}')

#----- App Starting -----
userId=567 # User Id
n=10 #Number of Recommendations
ratingsFrame.createOrReplaceTempView('ratings')
# Εκτέλεση sql ερωτήματος
userDf=session.sql(f'SELECT userId FROM ratings WHERE userId={userId}').distinct()
userr=model.recommendForUserSubset(userDf,n)
userrecommendations=usererr.rdd.collectAsMap() #DF-->RDD-->Dictionary
logger.info(f'Recommendations for {len(userrecommendations)} users made')

##### Recommendations
#####

logger.info(f'User sign in:{userId}')
table='<tr><th>Movie</th><th>Genre</th></tr>'
# Εκτέλεση sql ερωτήματος για εύρεση αξιολογήσεων του χρήστη που επιλέξαμε
userinfo=session.sql(f'SELECT movieid,rating FROM ratings WHERE userId={int(userId)}')
# Πρόσθεση των αξιολογήσεων στον html κώδικα
userrates='<tr><th>MOVIE</th><th>RATING</th></tr>'
message=""
serialout=""
for y in userinfo.collect():
    userrates+=f'<tr><td>{moviedb[str(y[0])][0]}</td><td>{moviedb[str(y[0])][1]}</td></tr>'
    message+=f'<li>{y[0]},{y[1]}</li>'
userrates+=f'<tr style=\"color:white; font-size:22px; background-color:#a410c9;\"><td>TOTAL RATINGS</td><td>{userinfo.count()}</td></tr>'
logger.info(f'Total user Ratings:{userinfo.count()}')

userRecs=""
print("\t\tPosibilities")
for x in userrecommendations[int(userId)]:
    userRecs+=f'<tr><td>{moviedb[str(x[0])][0]}</td><td>{moviedb[str(x[0])][1]}</td></tr>'
    print(x)
    serialout+=f'<li>{x[0]}-->{x[1]}</li>'

##### Output Production
#####

code=rc.htmloutput()
logger.debug(str(code))

```

```

codeout=code.SliderShow(userId,userrates,userRecs)
logger.info(codeout)
obj=so.SparkObject(ps.__version__,sc.appName,sc.master,serialout,message,sc.getConf().getAll())
logger.info('Static Page for User {userid} made')
obj.Show(userId)
##### App Stop #####
RemoveFiles()
logger.info('Application Exceeded')
session.stop()

```

- recout.py (Αρχείο για τη δημιουργία στατικής ιστοσελίδας με σκοπό την αναπαράσταση των αποτελεσμάτων)

```

from datetime import datetime
import webbrowser as wb
import os

class htmloutput:
    def __init__(self):
        self.time=datetime.now().strftime('%Y/%M/%D %H:%M;%S')
        self.htmlcode=""

    def SliderShow(self,userid,message,recommendations):
        backpath=os.path.join('../','Images','usersrec.jpg')
        self.htmlcode=f"""
<html>
<head>
<title>{userid} recommendations</title>
<style>
"""+""""
* {box-sizing:border-box}
.slideshow-container {
max-width: 1000px;
position: relative;
margin: auto;
}
.prev, .next {
cursor: pointer;
position: absolute;
top: 50%;
width: auto;

```



```

margin-top: -22px;
padding: 16px;
color: white;
font-weight: bold;
font-size: 18px;
transition: 0.6s ease;
border-radius: 0 3px 3px 0;
user-select: none;
}

.next {
right: 0;
border-radius: 3px 0 0 3px;
}

.prev:hover, .next:hover {
background-color: rgba(0,0,0,0.8);
}

.text {
color: ;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
font-size: 21px;
width: 100%;
text-align: center;
}

.dot {
cursor: pointer;
height: 15px;
width: 15px;
margin: 0 2px;
background-color: #bbb;
border-radius: 50%;
display: inline-block;
transition: background-color 0.6s ease;
}

.active, .dot:hover {
background-color: #717171;
}

```

```

}

.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

@-webkit-keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

"""+""

body{
  font-family:calibri;
  font-size:18px;
  """+f""

  background-image:url(\'{backpath}\');
  """+""

  background-repeat:no-repeat;
  background-size:cover;
  background-attachment:fixed;
  color:white;
}

table{
  background-color:white;
  color:black;
  font-size:19px;
  width:70%;
  text-align:center;
}

th{
  background-color:black;
  color:white;
  font-size:21px;

```

```

}
hr{
border-top:2px solid red;
}
</style>
</head>
<body>
    """+f""
    <center>
        <h1 style=\"margin-top:100px; width:80% font-weight:bold; text-decoration:underline; color:black;\"><img
src=\"https://beta.cloudnesil.com/wp-content/uploads/2018/12/apa>
        <hr>
        <h3 style=\"width:80%; font-weight:bold;\">Movies Evaluated by user {userid}</h3>
        <hr style=\"border-top: 2px solid green;\">
        <div class=\"slideshow-container\">
            <div class=\"mySlides Fade\">
                <table>
                    {message}
                </table>
                <br>
                <div class=\"text\">{userid} Ratings</div>
            </div>
            <div class=\"mySlides Fade\">
                <table>
                    {recommendations}
                </table>
                <br>
                <div class=\"text\">{userid} Recommendations</div>
            </div>
        </div>
    </center>
    <script>
        ""
        self.htmlcode+=\"\"\"{
        var slideIndex = 1;
        showSlides(slideIndex);
        // Next/previous controls
        function plusSlides(n)
        {
            showSlides(slideIndex += n);

```

```

}

function currentSlide(n) {
    showSlides(slideIndex = n);
}

function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("dot");
    if (n > slides.length) {slideIndex = 1}
    if (n < 1) {slideIndex = slides.length}
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
}

</script>
</body>
</html>
"""

outpath=os.path.join('..','Static Pages',str(userid)+'.html')
if not os.path.exists(os.path.join('..','Static Pages')):
    os.mkdir(os.path.join('..','Static Pages'))
with open(outpath,'w') as F:
    F.write(self.htmlcode)
return f'File Output Created At {outpath}'

def __str__(self):
    return str(self.time)+'-->Html Slider Object Has been Created!!!!!'

```