Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Συλλογή εργαστηριακών ασκήσεων στο Hadoop** **Collection of Hadoop lab exercises**
Ονοματεπώνυμο Φοιτητή	**Κωνσταντίνος Μαρκούτης**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 12043**
Υπεύθυνος Καθηγητής	**Χρήστος Δουληγέρης, Καθηγητής**
Επιβλέπων	**Δημήτριος Καλλέργης, Διδάκτωρ**

Ημερομηνία Παράδοσης **Μήνας Έτος**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Χρήστο Δουληγέρη, για την συμβολή και την καθοδήγησή του στην επιλογή του θέματος, και τον διδάκτορα κ. Δ.Ν. Καλλέργη για τις συμβουλές του και τη συνεργασία στη δημιουργία και συγγραφή αυτής της μεταπτυχιακής διατριβής.

1 Contents

Περίληψη

Η μεταπτυχιακή αυτή διατριβή, είναι ουσιαστικά μια συλλογή εργαστηριακών ασκήσεων για την κατανόηση των λειτουργιών του Hadoop. Το Hadoop είναι ένα πλαίσιο το οποίο προσφέρει γρηγορότερη και αποδοτικότερη διαχείριση δεδομένων, και αυτό είναι ορατό κυρίως σε περιπτώσεις δεδομένων μεγάλου όγκου. Μπορεί να χρησιμοποιηθεί πολύ εύκολα σε περιβάλλον Linux, ενώ υπάρχει και η δυνατότητα υλοποίησης σε Windows, τόσο μέσω command prompt, όσο και με τη χρήση I.D.E. όπως ο Eclipse. Στη συγκεκριμένη διατριβή είδαμε τα απαραίτητα θεωρητικά για την κατανόηση των ασκήσεων, και στη συνέχεια την πρακτική εφαρμογή τους. Κάναμε εγκατάσταση του Hadoop σε συστοιχία ενός κόμβου και ακολούθως σε συστοιχία πολλών κόμβων και πως μπορούμε να επιλύσουμε κάποια γνωστά προβλήματα όπως ο Καταμετρητής λέξεων ενός κειμένου. Οι ασκήσεις αυτές προέρχονται από πανεπιστήμια του εξωτερικού, αλλά υπάρχει και η παρουσία ενός ελληνικού ερευνητικού τμήματος.

Abstract

This dissertation is actually a collecion of laboratory exercises to help students comprehend the various functions of Hadoop. Hadoop is a framework that offers faster and more efficient data management, something that is visible in cases of big data. This framework can be used easily on Linux Operating System, but also on Windows, either via command prompt, or by using I.D.E. such as Eclipse. In this work we viewed the necessary theoretical aspects of Hadoop, and then the in practice implementation. We installed Hadoop in a single – node cluster, in a multi – node cluster, and finally we solved famous problems, such as WordCount. This collection of exercises derives from a pool of foreign institutions, but also from a domestic lab.

1. ΕΙΣΑΓΩΓΗ

1.1. Πρόλογος

Αυτή η μεταπτυχιακή διατριβή αποτελείται απο παραδείγματα πειραματικών εφαρμογών του Hadoop. Το Hadoop είναι μια πλατφόρμα που χρησιμοποιεί τα στοιχεία του υπολογιστικού νέφους για πιο γρήγορη και σίγουρη επεξεργασία δεδομένων. Τα παρακάτω παραδείγματα έχουν προκύψει απο έρευνα σε πανεπιστήμια της Ελλάδας και του εξωτερικού. Είναι εργαστηριακές ασκήσεις που έχουν γίνει για την κατανόηση των λειτουργιών του Hadoop. Η εργασία αυτή λοιπόν είναι μια συγκεντρωτική έρευνα με θεωρητική και πρακτική προσέγγιση στο Hadoop, ένα χρήσιμο και πρωτοπωριακό εργαλείο.

1.2. Σκοπός

Θα γίνει μια θεωρητική αναφορά στα τεχνικά χαρακτηριστικά του Hadoop, και στη συνέχεια θα παρουσιάσουμε τις εργαστηριακές ασκήσεις που έχουν γίνει με τον κώδικα για την επίλυσή τους.

1.3. Δομή πτυχιακής εργασίας

Συνοπτικά μπορούμε να πούμε οτι η δομή της εργασίας είναι:

Στο 1^{ο} κεφάλαιο υπάρχει μια σύντομη εισαγωγή, για να αντιληφθεί ο αναγνώστης ποιό είναι το θέμα της εργασίας, και με ποιο τρόπο θα γίνει η ανάλυση του θέματος.

Στο 2^{ο} κεφάλαιο υπάρχει η θεωρία πάνω στα χαρακτηριστικά του Hadoop, τον τρόπο λειτουργίας του, και κάποια πλεονεκτήματα της χρήσης του.

Στο 3^{ο} κεφάλαιο παρουσιάζονται αναλυτικά οι εργαστηριακές ασκήσεις που έχουν γίνει με θέμα το Hadoop.

2. ΘΕΩΡΗΤΙΚΑ

2.1. Γενικά για το Hadoop

Πώς προέκυψε το Hadoop

Στις αρχές του 2000, η Google αντιμετώπισε ένα πρόβλημα. Έπρεπε να βρει ένα τρόπο να διαχειριστεί την συνεχώς αυξανόμενη πληροφορία. Αυτό το πέτυχε με το *MapReduce*, ένα σύστημα επεξεργασίας δεδομένων που επιτρέπει τη διεργασία να διαχωριστεί σε πολλούς εξυπηρετητές και η επεξεργασία να γίνει παράλληλα. Ένας προγραμματιστής, ο Doug Cutting αντιμετώπισε παρόμοια προβήματα και στηρίχθηκε στο *MapReduce*, αλλά άλλαξε τη δομή των δεδομένων και τη διαδικασία της επεξεργασίας, και δημιουργησε ένα νέο πρόγραμμα, το Hadoop. Η Yahoo! με τη σειρά της αντιμετώπιζε παρόμοια προβλήματα, και αναγνώρισε στο Hadoop τη λύση. Τώρα, πλέον, το Hadoop είναι ένα project ανοιχτού κώδικα, ελεύθερο σε χρήση και βελτίωση που λειτουργεί υπο το Apache Software Foundation.[1]

2.2. Χαρακτηριστικά

Το Hadoop αποτελείται απο δύο χαρακτηριστικά: ένα σύστημα αποθήκευσης αρχείων και ένα σύστημα επεξεργασίας δεδομένων. Το σύστημα αποθήκευσης λέγεται *Hadoop Distributed File System (HDFS)*. Παρέχει δυνατότητα αποθήκευσης με χαμηλό κόστος, ενώ μπορεί να ανιχνεύσει προβλήματα στο δίσκο και στον εξυπηρετητή. Αποθηκεύει τα αρχεία στους server της συστοιχίας, χωρίζοντας τα σε μπλόκ. Κάθε μπλόκ αντιγράφεται σε πάνω απο ένα server, για να διασφαλιστεί η ασφάλεια και η γρήγορη επεξεργασία. Το *HDFS* ελέγχει την ακεραιότητα των δεδομένων με τη συνάρτηση *checksum* και σε περίπτωση αποτυχίας δίσκου ή εξυπηρετητή τα δεδομένα αντιγράφονται σε άλλο κόμβο της συστοιχίας. Το άλλο βασικό χαρακτηριστικό του Hadoop είναι το σύστημα επεξεργασίας δεδομένων, *MapReduce*. Το *MapReduce* έχει ένα προγραμματιστή διεργασιών, που αποφασίζει ποιος server θα τρέξει τη διεργασία, ενώ εκτελεί πολλές εργασίες παράλληλα. Το *MapReduce* για να βρει τα μπλόκ που απαρτίζουν ένα αρχείο χρησιμοποιεί το *NameNode*. Το *MapReduce* με τη σειρά του αποτελείται απο δυο διεργασίες. Η διεργασία *Map* επιτρέπει στο χρήστη να γράψει κώδικα που θα τρέξει κατευθείαν στο *DataNode server*, και περιλαμβάνει απλούς αλγορίθμους (π.χ. συχνότητα εμφάνισης μιας λέξης, αναγνώριση προτύπου, μηχανική μάθηση). Στο τέλος αυτής της διεργασίας, τα αποτελέματα συλλέγονται απο τον *Reducer*. Καθήκον του είναι να συλλέξει τα δεδομένα και να τα γράψει στο *HDFS*. Αξίζει να σημειωθεί οτι αν μια διεργασία ή ένα τμήμα της εκτελείται αργά, το *MapReduce* μπορεί να την αναθέσει σε άλλο *server* για να επιταχύνει τη διεργασία.
Θεωρητικά μπορούμε να δούμε τη διαδικασία *MapReduce* σαν πέντε βήματα παράλληλης επεξεργασίας:

1. Προετοιμασία της συνάρτησης εισόδου `Map()`: το σύστημα *MapReduce* αναθέτει τη διαδικασία Map σε επεξεργαστές, και σε κάθε επεξεργαστή αναθέτει μια τιμή - κλειδί, έστω *K1*. Αυτή η τιμή είναι ουσιαστικά και ο κωδικός της εργασίας που θα τρέξει σε αυτόν. Επίσης προμηθεύει τον επεξεργαστή με όλες τις απαραίτητες πληροφορίες για να τρέξει τη διεργασία.
2. Εκτέλεση του κώδικα για τη συνάρτηση `Map()` που του δίνει ο χρήστης: η συνάρτηση `Map()` τρέχει μια φορά για κάθε κλειδί *K1*, και παράγει μια έξοδο, στην οποία δίνουμε την τιμή *K2*.
3. Η έξοδος της `Map()` «ανακατεύεται» (*shuffle*) στους επεξεργαστές *Reduce*: το σύστημα *MapReduce* αναθέτει τη διαδικασία *Reduce* σε επεξεργαστές, και σε κάθε επεξεργαστή αναθέτει μια τιμή *K2*, για να επεξεργαστεί τα αντίστοιχα δεδομένα.
4. Εκτέλεση του κώδικα `Reduce()` που δίνει ο χρήστης: η συνάρτηση `Reduce()` εκτελείται μια φορά για κάθε τιμή *K2* που παράγεται απο τη συνάρτηση `Map()`.
5. Παραγωγή του τελικού αποτελέσματος: το σύστημα *MapReduce* συλλέγει τα αποτελέσματα της `Reduce()` και τα ταξινομεί ανάλογα με την τιμή *K2* για να δημιουργήσει το τελικό αποτέλεσμα. [1, 10, 11, 12]

2.3. Στοιχεία των διεργασιών

Οι διεργασίες του *MapReduce* πρέπει να περιλαμβάνουν κάποια χαρακτηριστικά για να λειτουργούν πετυχημένα. Κατ αρχάς οι αλγόριθμοι πρέπει να τρέχουν γρήγορα στη δομή του Hadoop, να μπορούν δηλαδή να διαβάζουν ένα μικρό αριθμό εγγραφών στο ίδιο *DataNode* για να υπολογίσουν το αποτέλεσμα. Το *MapReduce* εκτελεί κώδικα Java, αλλά μπορεί να χρησιμοποιήσει και άλλες γλώσσες όπως C++, PHP, Python και Pearl. Είναι μια καλή επιλογή επίσης να χρησιμοποιηθούν αλγόριθμοι που εξετάζουν κάθε μια εγγραφή

για να υπολογίσουν το αποτέλεσμα. Ο όγκος των δεδομένων δεν είναι πρόβλημα γιατί οι διεργασίες τρέχουν παράλληλα στους εξυπηρετητές *DataNode*. Το *MapReduce* είναι κατάλληλο για batch processing, και όχι για εργασίες αλληλεπίδρασης. Αυτό βέβαια σημαίνει οτι μπορεί να διαχειρίζεται μέχρι και petabyte δεδομένων και να απαντά σε ερωτήματα χρηστών κάτι που προηγουμένως ήταν αδύνατο.

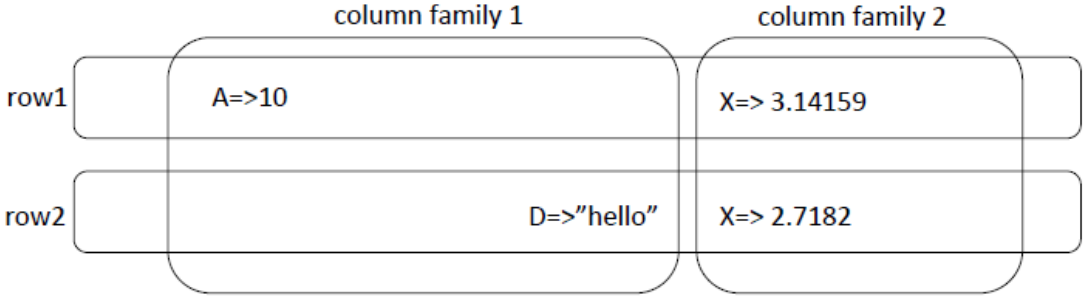
2.4. Πλεονεκτήματα του Hadoop

Το Hadoop έχει κάποια σημαντικά πλεονεκτήματα στη χρήση του. Μπορεί να αποθηκεύσει και να χειρίζεται πολλά είδη δεδομένων. Επίσης είναι ανοιχτού λογισμικού, που σημαίνει οτι η αποθήκευση και η επεξεργασία γίνονται με χαμηλό κόστος. Επίσης προσφέρει στο χρήστη τη δυνατότητα παράλληλης επεξεργασίας με τη χρήση αλγορίθμων κατευθείαν στο χώρο που υπάρχουν τα δεδομένα.

Εικόνα 2.4.1: Με μπλε χρώμα απεικονίζεται η διεργασία όπως χωρίζεται σε μπλοκ απο το HDFS, και με κόκκινο το MapReduce συγκεντρώνει τα αποτελέσματα.[1]

2.5. HBase

Ένας άλλος τρόπος αποθήκευσης δεδομένων μεγάλου όγκου είναι η *HBase*. Χρησιμοποιείται απο το Hadoop καθώς μπορεί να διαχειρίζεται εκατομμύρια γραμμές και στήλες (π.χ. τα 50 μεγαλύτερα αντικείμενα απο δισεκατομμύρια εγγραφές). Τα δεδομένα οργανώνονται σε πίνακες. Κάθε γραμμή χαρακτηρίζεται απο ένα κλειδί, ενώ οι στήλες οργανώνονται σε ομάδες. Μέσα σε κάθε ομάδα υπάρχουν οι στήλες και οι τιμές τους.[2]



Εικόνα 2.5.1: Κάθε σειρά χαρακτηρίζεται απο ένα κλειδί. Οι στήλες οργανώνονται σε οικογένειες, και σε κάθε οικογένεια υπάρχουν οι στήλες και οι τιμές τους.[2]

2.6 Οδηγός

Όπου στο κείμενο υπάρχει γραμματοσειρά Courier New σημαίνει οτι είναι γραμμές κώδικα.

Με Italic είναι γραμμένα ονόματα εργαλείων, προγραμμάτων, και ορισμοί.

Με Bold είναι γραμμένα ονόματα αρχείων και γενικά ότι χρήζει επισήμανσης στον αναγνώστη.

Όπου υπάρχει πλαίσιο κειμένου σημαίνει ότι γίνεται εισαγωγή κώδικα σε τερματικό ή είναι αποτέλεσμα εκτέλεσης κώδικα.

3. ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΣΤΟ HADOOP

Στην ενότητα αυτή θα γίνει η παρουσίαση των εργαστηριακών ασκήσεων που έχουν γίνει πάνω στο Hadoop. Οι ασκήσεις αυτές έχουν γίνει σε πανεπιστήμια της Ελλάδας και του εξωτερικού, για την εμπέδωση των χαρακτηριστικών και των δυνατοτήτων του και αποτελούν αποτέλεσμα έρευνας στις ιστοσελίδες και e-class των σχολών.

3.1 Παράδειγμα εγκατάστασης Hadoop σε περιβάλλον Windows

3.1.1 Περιγραφή

Αυτό το παράδειγμα δείχνει πώς θα γίνει η εγκατάσταση του Hadoop σε περιβάλλον Windows.[5]

3.1.2 Θεωρητική προσέγγιση

Αρχικά, θα εγκαταστήσουμε το Cygwin. Το Cygwin είναι ένα πρόγραμμα το οποίο δίνει στο χρήστη εργαλεία GNU και ανοιχτού κώδικα, για να λειτουργεί τα Windows με τρόπο παρόμοιο αυτών των Linux. [13]

Προαπαιτούμενα

- Εγκατάσταση Cygwin (εγκατάσταση των tar, vim)
- Εγκατάσταση του JDK7 (jdk-7u7-windows-i586.exe)
- Εγκατάσταση κάποιου notepad (π.χ. notepad++)
- `hadoop-0.20.2.tar.gz` (οι επόμενες εκδόσεις δεν λειτουργούν σε Windows)

3.1.3 Πειραματική προσέγγιση

Στήσιμο του Hadoop:

Βήμα 1ο

Αντιγραφή του `hadoop-0.20.2.tar.gz` στο Home Directory:

```
cp              /cygdrive/c/Documents\              and\
Settings/Administrator/My\  Documents/Downloads/hadoop-
0.20.2.tar.gz ~/
```

Βήμα 2ο

Κάνουμε Extract

```
tar xvzf hadoop-0.20.2.tar.gz
```

Βήμα 3ο

Στη συνέχεια κάνουμε export:

```
export
JAVA_HOME=/cygdrive/c/Progra~1/Java/jdk1.7.0_07/
```

Βήμα 4ο

Δοκιμή

`bin/hadoop`

Παράδειγμα 1ο:

```
$ mkdir input
$ cp conf/*.xml input
$ bin/hadoop jar hadoop-0.20.2-examples.jar grep input
output 'dfs[a-z.]+'
$ cat output/*
```

Παράδειγμα 2ο:

Στήσιμο του Java compiler στο Cygwin:

```
$ cd /usr/local/bin
$    ln    -s    /cygdrive/c/”Program
Files”/Java/jdk1.7.0_07/bin/javac.exe
$    ln    -s    /cygdrive/c/”Program
Files”/Java/jdk1.7.0_07/bin/jar.exe
$ rehash
```

```
$ cd ~/hadoop-0.20.2
$ mkdir wordcount
$ javac –classpath "hadoop-0.20.2-core.jar;lib/commons-cli-
1.2.jar”
src/examples/org/apache/hadoop/examples/WordCount.java –d wordcount
$ jar -cvf WordCount.jar -C wordcount/ .
$ bin/hadoop jar WordCount.jar
org/apache/hadoop/examples/WordCount input out
```

3.1.4 Διαφορετική προσέγγιση

Μπορούμε όμως να βρούμε μια διαφορετική προσέγγιση αυτού του προβλήματος.[17]

Προγραμματιστές που ασχολούνται με Hadoop έχουν χρησιμοποιήσει Windows Server 2008 ή Windows Server 2008 R2.

Επιλογή έκδοσης Java

Οι εκδόσεις Java που έχουν ελεγχθεί για τη συμβατότητα τους είναι οι 1.6 και 1.7.. Αρχικά πρέπει να σιγουρευτούμε οτι η μεταβλητή JAVA_HOME δεν περιέχει κενά.

Κατεβάζουμε το Hadoop

Κατεβάζουμε την έκδοση 2.2.

Εκτέλεση

Απο το command prompt εκτελούμε:

```
mvn package -Pdist,native-win -DskipTests -Dtar
```

Εγκατάσταση

Κάνουμε extract το Hadoop στο C.

Έναρξη μεταβλητών περιβάλλοντος

Εκτελούμε

```
c:\deploy\etc\hadoop\hadoop-env.cmd
```

Δημιουργία συστήματος αρχείων

```
%HADOOP_PREFIX%\bin\hdfs dfs -format
```

Έναρξη HDFS Daemons

```
%HADOOP_PREFIX%\sbin\start-dfs.cmd
```

Έναρξη YARN Daemons

```
%HADOOP_PREFIX%\sbin\start-yarn.cmd
```

3.2 Δημιουργία κόμβων Hadoop σε γραμμές εντολών Linux και παραδείγματα σε Java

3.2.1 Περιγραφή

Το Hadoop προϋποθέτει τη γνώση βασικών εννοιών και εντολών του Linux. Σε αυτό το παράδειγμα θα δούμε κάποιες εντολές που χρησιμοποιούνται για την κατασκευή κόμβων Hadoop, και γενικά το πώς θα το υλοποιήσουμε σε περιβάλλον Linux.[7]

3.2.2 Θεωρητική προσέγγιση

Σε αυτό το παράδειγμα χρησιμοποιούνται εντολές Linux. Κάποιες βασικές εντολές που χρησιμοποιούνται μπορούμε να τις αναφέρουμε συγκεντρωτικά στον παρακάτω πίνακα:

Εντολή	Επεξήγηση
ls	Δείχνει τα περιεχόμενα του τρέχοντος καταλόγου

ls -a	Δείχνει κρυμένα αρχεία και καταλόγους
cp filename /path/dir_name	Αντιγράφει το filename στο path/dir_name
mv filename /path/dir_name	Μετακινεί το filename στο path/dir_name
cat filename	Δείχνει τα περιεχόμενα του κάθε filename
cd ..	Ανεβαίνει ένα κατάλογο πάνω
mkdir dir_name	Δημιουργεί φάκελο dir_name
grep string /path/dir_name	Βρίσκει όλα τα αρχεία που περιέχουν /path/dir_name
tar -xzf filename.tgz	Αποσυμπιέζει ένα αρχείο tgz
tar -xzf filename.tar.gz	Αποσυμπιέζει ένα αρχείο tar.tgz
tar -xjf filename.tar.bz2	Αποσυμπιέζει ένα αρχείο tar.bz2
startx	Εκκίνηση του x μηχανήματος
su	Είσοδος στο σύστημα σαν root απο τον τρέχο λογαριασμό
Exit	Εξοδος απο την κονσόλα
-conf <configuration file>	Καθορίζει ένα configuration file εφαρμογής
-fs <local|namenode:port>	Καθορίζει namenode
-jt <local|jobtracker:port>	Καθορίζει jobtracker
-archiveName NAME	Όνομα αρχείου που θα δημιουργηθεί
src	Αρχικός φάκελος
dest	Φάκελος προορισμού που θα περιέχει το αρχείο
jar	Εκτέλεση jar αρχείου
-input <path>	Φάκελος εισόδου
-output <path>	Φάκελος εξόδου
-map	Java map class
-partitioner	Java partitioner
-reduce	Java Reduce class
-writer	Java RecordWriter
-program <executable>	Εκτελέσιμο URI
-reduces <num>	Αριθμός μειώσεων
hadoop jobtracker	Εκτέλεση jobtracker
hadoop namenode	Εκτέλεση namenode
hadoop tasktracker	Εκτέλεση tasktracker

3.2.3 Πειραματική προσέγγιση

Δημιουργία κόμβων Hadoop

Βήμα 1ο

Δίνουμε την εντολή για να ξεκινήσει η συστοιχία του Hadoop.

```
$ bin/start-dfs.sh
```

Βήμα 2ο

Στο Home directory του Hadoop εκτελούμε:

```
$ bin/hadoop fs
```

και παίρνουμε το ακόλουθο μήνυμα:

```
hadoop@                 :/opt/hadoop$ bin/hadoop fs
Usage: java FsShell
           [-ls <path>]
           [-lsr <path>]
           [-du <path>]
           [-dus <path>]
           [-count[-q] <path>]
           [-mv <src> <dst>]
           [-cp <src> <dst>]
           [-rm [-skipTrash] <path>]
           [-rmr [-skipTrash] <path>]
           [-expunge]
           [-put <localsrc> ... <dst>]
           [-copyFromLocal <localsrc> ... <dst>]
           [-moveFromLocal <localsrc> ... <dst>]
           [-get [-ignoreCrc] [-crc] <src> <localdst>]
           [-getmerge <src> <localdst> [addnl]]
           [-cat <src>]
           [-text <src>]
           [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
           [-moveToLocal [-crc] <src> <localdst>]
           [-mkdir <path>]
           [-setrep [-R] [-w] <rep> <path/file>]
           [-touchz <path>]
           [-test -[ezd] <path>]
           [-stat [format] <path>]
           [-tail [-f] <file>]
           [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
           [-chown [-R] [OWNER][:[GROUP]] PATH...]
           [-chgrp [-R] GROUP PATH...]
           [-help [cmd]]
```

Βήμα 3ο

Ανεβάζουμε τα δεδομένα στο HDFS:

```
$ bin/hadoop fs -mkdir input
$ bin/hadoop fs -put conf/ input/
```

Για να ελέγξουμε αν τα δεδομένα είναι μέσα στο φάκελο `input`:

```
$ bin/hadoop fs -ls input/
```

```
hadoop@[illegible]:/opt/hadoop$ bin/hadoop fs -ls input
Found 13 items
-rw-r--r--   1 hadoop supergroup       3936 2010-10-05 14:25 /user/hadoop/i
nput/capacity-scheduler.xml
-rw-r--r--   1 hadoop supergroup        535 2010-10-05 14:25 /user/hadoop/i
nput/configuration.xsl
-rw-r--r--   1 hadoop supergroup        464 2010-10-05 14:25 /user/hadoop/i
nput/core-site.xml
-rw-r--r--   1 hadoop supergroup       2237 2010-10-05 14:25 /user/hadoop/i
nput/hadoop-env.sh
-rw-r--r--   1 hadoop supergroup       1245 2010-10-05 14:25 /user/hadoop/i
nput/hadoop-metrics.properties
-rw-r--r--   1 hadoop supergroup       4190 2010-10-05 14:25 /user/hadoop/i
nput/hadoop-policy.xml
-rw-r--r--   1 hadoop supergroup        259 2010-10-05 14:25 /user/hadoop/i
nput/hdfs-site.xml
-rw-r--r--   1 hadoop supergroup       2815 2010-10-05 14:25 /user/hadoop/i
nput/log4j.properties
-rw-r--r--   1 hadoop supergroup        277 2010-10-05 14:25 /user/hadoop/i
nput/mapred-site.xml
-rw-r--r--   1 hadoop supergroup         10 2010-10-05 14:25 /user/hadoop/i
nput/masters
-rw-r--r--   1 hadoop supergroup         10 2010-10-05 14:25 /user/hadoop/i
nput/slaves
-rw-r--r--   1 hadoop supergroup       1243 2010-10-05 14:25 /user/hadoop/i
nput/ssl-client.xml.example
-rw-r--r--   1 hadoop supergroup       1195 2010-10-05 14:25 /user/hadoop/i
nput/ssl-server.xml.example
```

εκτελούμε την διεργασία *MapReduce*:

```
$ bin/hadoop jar hadoop-0.20.2-examples.jar wordcount
input output
```

Για να πάρουμε τα δεδομένα απο το HDFS έχουμε δύο επιλογές:

1. `$ bin/hadoop fs -cat output/part-r-00000`
2. `$ bin/hadoop fs -copyToLocal output/ output/`

Και βρίσκουμε το αποτέλεσμα στο μονοπάτι `/opt/hadoop/`.

Βήμα 4ο

Μπορούμε να διαγράψουμε ένα φάκελο απο το HDFS, για παράδειγμα το αρχείο `output/part-r-00000`:

```
$ bin/hadoop fs -rm output/part-r-00000
$ bin/hadoop fs -ls output/
```

Και στη συνέχεια να διαγράψουμε το φάκελο εξόδου:

```
$ bin/hadoop fs -rmr output
$ bin/hadoop fs –ls
```

Δημιουργία προγράμματος σε Java για αλληλεπίδραση με το HDFS:

Δημιουργούμε ένα φάκελο στο HDFS

Βήμα 1ο

Αλλάζουμε το `.profile` στο `home directory` του Hadoop

Προσθέτουμε το CLASSPATH που χρησιμοποιείται απο το Java Compiler

```
$ vim ~/.profile
```

Και στο τέλος του αρχείου προσθέτουμε τις γραμμές κώδικα:

```
CLASSPATH=/opt/hadoop/hadoop-0.20.2-core.jar
export CLASSPATH
```

τέλος, κάνουμε ξανά log in:

```
$ exit
$ su – hadoop
```

Βήμα 2ο

Ανοίγουμε ένα αρχείο java, έστω το ex01.java:

```
$ vim ex01.java
```

Και εισάγουμε τις γραμμές κώδικα:

```
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
public class ex01 {
public static void main(String[] args){
if(args.length < 2){
System.out.print("Need two arguments\n");
System.out.flush();
return;
```

```
System.out.print("Creat File: " + args[0] + "\n");
System.out.print("Content: " + args[1] + "\n");
System.out.flush();
try {
Configuration conf = new Configuration();
FileSystem hdfs = FileSystem.get( conf );
Path filepath = new Path( args[0] );
FSDataOutputStream output = hdfs.create( filepath );
byte[] buff = args[1].getBytes();
output.write(buff, 0, buff.length);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Βήμα 3ο

Κάνουμε compile το αρχείο:

```
$ javac ex01.java
```

Βήμα 4ο

Εκτελούμε το πρόγραμμα:

Ελέγχουμε τα περιεχόμενα του test στο HDFS:

```
$ bin/hadoop ex01 test “Hello World”
```

```
$ bin/hadoop fs -cat test
```

Ανεβάζουμε ένα αρχείο στο HDFS

Βήμα 1ο

Δημιουργούμε ένα αρχείο π.χ. το `ex02.java`

```
$ vim ex02.java
```

Και εισάγουμε τις γραμμές κώδικα:

```
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
public class ex02 {
public static void main( String[] args){
if ( args.length < 2){
System.out.print("Need two arguments\n");
System.out.flush();
return;
}
System.out.print("Source File: " +args[0]+ "\n");
System.out.print("Destination File: " +args[1]+ "\n");
System.out.flush();
try {
Configuration conf = new Configuration();
FileSystem hdfs = FileSystem.get(conf);
Path srcPath = new Path(args[0]);
Path dstPath = new Path(args[1]);
hdfs.copyFromLocalFile(srcPath, dstPath);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Βήμα 2ο

Κάνουμε compile το πρόγραμμα:

```
$ javac ex02.java
```

Βήμα 3ο

Εκτελούμε το πρόγραμμα:

```
$ bin/hadoop ex02 conf/ input2/
```

Ελέγχουμε τα περιεχόμενα του test στο HDFS.

```
$ bin/hadoop fs -ls input2
```

3.2.4 Διαφορετική προσέγγιση

Η διαφορετική προσέγγιση που θα αναπτύξουμε αποτελείται απο το στήσιμο του Hadoop σε Linux, με γραμμή εντολών. Πρώτα δημιουργείται ένας κόμβος, και με βάση αυτόν μπορούμε να χτίσουμε περισσότερους.[17]

Java

Κατεβάζουμε την έκδοση 1.6 (ή νεότερη)

Προσθήκη χρήστη

Προετείνεται η προσθήκη χρήστη για να ξεχωρίσουμε την εγκατάστασή του απο άλλες εφαρμογές.
Εκτελούμε:

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduse
```

Διαμόρφωση SSH

```
user@ubuntu:~$ su – hduser
hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
The key's randomart image is:
[...snipp...]
hduser@ubuntu:~$
```

Δίνουμε πρόσβαση SSH στο τοπικό μας μηχάνημα:

```
hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys
```

Ελέγχουμε το SSH, κάνοντας σύνδεση στο τοπικό μας μηχάνημα ως *hduser*:

```
hduser@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is
d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known
hosts.
Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC
2010 i686 GNU/Linux
Ubuntu 10.04 LTS
[...snipp...]
hduser@ubuntu:~$
```

IPv6

Απενεργοποιούμε την IPv6, πηγαίνοντας στο αρχείο sysctl.conf και αλλάζοντας τις γραμμές:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Εγκατάσταση Hadoop

Αρχικά κατεβάζουμε μια έκδοση του Hadoop και κάνουμε extract σε ένα φάκελο, έστω τον `/usr/local/hadoop`.

```
$ cd /usr/local
$ sudo tar xzf hadoop-1.0.3.tar.gz
$ sudo mv hadoop-1.0.3 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

Ενημέρωση bashrc

Προσθέτουμε τις παρακάτω γραμμές στο αρχείο `bashrc`:

```
export HADOOP_HOME=/usr/local/hadoop
export JAVA_HOME=/usr/lib/jvm/java-6-sun
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}
export PATH=$PATH:$HADOOP_HOME/bin
```

hadoop-env.sh

αλλάζουμε το:

```
export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

σε:

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

conf/*-site.xml

φτιάχνουμε το φάκελο `/app/hadoop/tmp`:

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
# ...and if you want to tighten up security, chmod from 755 to 750...
$ sudo chmod 750 /app/hadoop/tmp
```

conf/core-site.xml

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

conf/mapred-site.xml

```
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
```

conf/hdfs-site.xml

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is
created.
  The default is used if replication is not specified in create time.
  </description>
</property>
```

κάνουμε αρχικοποίηση του φακέλου `dfs.name.dir`

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

το αποτέλεσμα είναι:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop namenode -format
10/05/08 16:59:56 INFO namenode.NameNode: STARTUP_MSG:
10/05/08 16:59:56 INFO namenode.FSNamesystem: fsOwner=hduser,hadoop
10/05/08 16:59:56 INFO namenode.FSNamesystem: supergroup=supergroup
10/05/08 16:59:56 INFO namenode.FSNamesystem:
isPermissionEnabled=true
10/05/08 16:59:56 INFO common.Storage: Image file of size 96 saved in
0 seconds.
10/05/08 16:59:57 INFO common.Storage: Storage directory .../hadoop-
hduser/dfs/name has been successfully formatted.
10/05/08 16:59:57 INFO namenode.NameNode: SHUTDOWN_MSG:
hduser@ubuntu:/usr/local/hadoop$
```

έναρξη του κόμβου:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

παύση της συστοιχίας:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

3.3 Εγκατάσταση Hadoop και διεργασίας MapReduce σε Linux

3.3.1 Περιγραφή

Στο συγκεκριμένο παράδειγμα, θα εγκαταστήσουμε το Hadoop σε περιβάλλον Linux, αρχικά σε ένα κόμβο και στη συνέχεια σε περισσότερους. Επίσης θα δείξουμε μια διεργασία σε *MapReduce*. [4]

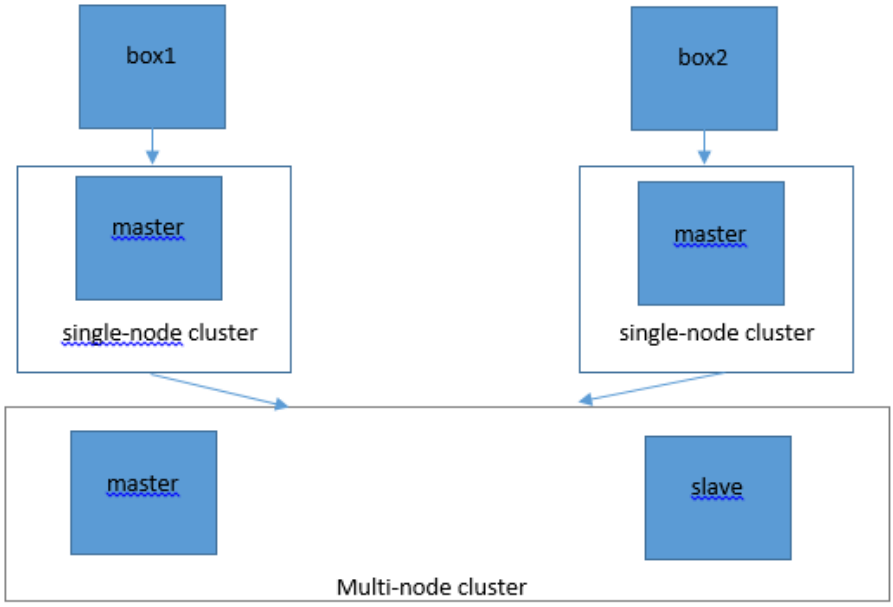
3.3.2 Θεωρητική προσέγγιση

Συστοιχία ενός κόμβου

Το πρώτο βήμα για την εγκατάσταση του Hadoop είναι η διαμόρφωση του συστήματος αρχείων του, που βρίσκεται στην κορυφή του συστήματος αρχείων της συστοιχίας. Αυτό θα γίνει την πρώτη φορά που δημιουργούμε μια συστοιχία Hadoop. Αν το κάνουμε κατά τη διάρκεια που εκτελείται ήδη μια συστοιχία, τότε θα διαγραφούν όλα τα δεδομένα.

Συστοιχία πολλών κόμβων

Θα κατασκευάσουμε μια συστοιχία πολλών κόμβων με δύο μηχανές Ubuntu. Ένας τρόπος για να γίνει αυτό είναι με την εγκατάσταση και δοκιμή του Hadoop σε δύο μηχανές Ubuntu. Στη συνέχεια, θα ενώσουμε αυτή τη συστοιχία απλών κόμβων σε μια συστοιχία πολλών κόμβων στην οποία μια μηχανή Ubuntu θα είναι ο *master* κόμβος και η άλλη ο *slave* κόμβος. Ο master κόμβος θα τρέχει *master daemons* σε κάθε στρώμα, *namenode* για το *HDFS*, και *JobTracker* για το *MapReduce*. Και οι δύο μηχανές θα τρέχουν τους slave *daemons*: *datanode* για το *HDFS* και *TaskTracker* για το *MapReduce*. Οι master daemons είναι υπεύθυνοι για το συντονισμό και τη διαχείριση των *slave daemons* οι οποίοι αργότερα θα υλοποιούν την αποθήκευση και την επεξεργασία των δεδομένων.



Εικόνα 3.3.2.1: Στη συστοιχία ενός κόμβου υπάρχει μόνο ένας master. Για τη δημιουργία συστοιχίας πολλών κόμβων, χρησιμοποιούμε πολλές συστοιχίες ενός κόμβου.[4]

3.3.3 Πειραματική προσέγγιση

Προαπαιτούμενα:
Κατ αρχάς θα κατεβάσουμε μια έκδοση του Hadoop (για παράδειγμα 0.6.0).
Βήμα 1ο

Καθορίζουμε ένα φάκελο για εγκατάσταση, έστω:

`/foo/bar/hadoop-install`

Βήμα 2ο

Κάνουμε untar. Τότε θα δημιουργηθεί ο φάκελος

`/foo/bar/hadoop-install/hadoop-0.6.0`

Η μεταβλητή `HADOOP_INSTALL=/foo/bar/hadoop-install` αναπαριστά το μονοπάτι για όλες τις εκδόσεις που έχουν εγκατασταθεί.

Το μονοπάτι `$HADOOP_INSTALL/hadoop -> hadoop-0.6.0.` μας δείχνει οτι χρησιμοποιείται η έκδοση 0.6.0.

Όμοια το `$HADOOP_INSTALL/hadoop/bin` δείχνει τα εργαλεία που θα χρησιμοποιηθούν, ενώ το `$HADOOP_INSTALL/hadoop/conf` δείχνει τα configuration αρχεία.

Δημιουργία κόμβου:

Για την δημιουργία ενός κόμβου θα ακολουθήσουμε μια διαδικασία. Τα αρχεία που θα χρειαστούν είναι τα παρακάτω:

Αρχείο hadoop-env.sh

Βήμα 1ο

Ανοίγουμε με κάποιον editor το αρχείο

`<HADOOP_INSTALL>/conf/hadoop-env.sh`

Βήμα 2ο

Θέτουμε τη μεταβλητή

`JAVA_HOME`

στο αρχείο `Sun JDK/JRE 1.5.0`

Δηλαδή:

```
export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

Αρχείο hadoop-site.xml

Όλες οι αλλαγές στο Hadoop μπορούν να γίνουν στο

`<HADOOP_INSTALL>/conf/hadoop-site.xml`

Από εδώ μπορούμε να τροποποιήσουμε το φάκελο που χρησιμοποιεί το Hadoop για να αποθηκεύσει τα αρχεία, τις θύρες που χρησιμοποιεί κλπ. Μπορούμε να αφήσουμε όλες τις επιλογές όπως είναι αλλά να αλλάξουμε τη μεταβλητή

`hadoop.tmp.dir`

Για παράδειγμα:

```
<property>
    <name>hadoop.tmp.dir</name>
   <value>/your/path/to/hadoop/tmp/dir/hadoop-
${user.name}</value>
   <description>A   base   for   other   temporary
directories.</description>
</property>
```

Συστοιχία ενός κόμβου:

Βήμα 1ο

Τρέχουμε την εντολή (σε Ubuntu):

```
~$ <HADOOP_INSTALL>/hadoop/bin/hadoop namenode –format
```

Βήμα 2ο

Αρχίζουμε τη συστοιχία:

```
~$ <HADOOP_INSTALL>/bin/start-all.sh
```

Αυτή η εντολή δημιουργεί *Namenode*, *Datanode*, *JobTracker* και *Tasktrackers*.

Βήμα 3ο

Για να σταματήσουμε του *daemons* που τρέχουν εκτελούμε τον κώδικα:

```
~$ <HADOOP_INSTALL>/bin/stop-all.sh
```

Συστοιχία πολλών κόμβων

Οι *master* και *slave* μπορούν να επικοινωνούν μεταξύ τους μέσω του δικτύου. Για να κλείσουμε τη συστοιχία του απλού κόμβου εκτελούμε:

```
<HADOOP_INSTALL>/bin/stop-all.sh
```

Διαχείριση αρχείων

Το αρχείο `conf/masters` διαχειρίζεται το *master node* της συστοιχίας. Στο αρχείο `<HADOOP_INSTALL>/conf/masters`

φαίνεται έτσι:

```
master
```

Το αρχείο `conf/slaves` περιέχει τους *hosts*, έναν σε κάθε γραμμή όπου θα τρέχουν οι *slave daemons* του Hadoop(*datanodes* και *tasktrackers*). Και η *master* αλλά και η *slave* μηχανή λειτουργούν σαν *slaves* του Hadoop, γιατί θέλουμε να αποθηκεύουν και να επεξεργάζονται δεδομένα. Στη *master* λοιπόν ενημερώνουμε το αρχείο:

`<HADOOP_INSTALL>/conf/slaves`

Και παίρνουμε αποτέλεσμα:

```
master
slave
```

Εφόσον αλλάξουμε το αρχείο `conf/hadoop-site.xml` σε κάθε μηχανή, θα αλλάξουμε και μερικές άλλες μεταβλητές. Αρχικά αλλάζουμε την `fs.default.name`(*master* του *HDFS*) που καθορίζει τη θύρα του *NameNode*. Στην περίπτωσή μας είναι η *master* μηχανή:

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:54310</value>
    <description>The name of the default file
system. . .
</property>
```

Στη συνέχεια αλλάζουμε τη μεταβλητή `mapred.job.tracker` (*slave* του *MapReduce*) που καθορίζει τη θύρα του *JobTracker*:

```
<property>
 <name>mapred.job.tracker</name>
<value>master:54311</value>
<description>The host and port that the MapReduce job
tracker runs at . . . </description>
</property>
```

Στη συνέχεια αλλάζουμε τη μεταβητή `dfs.replication`, η οποία καθορίζει σε πόσες μηχανές θα αντιγραφεί κάποιο αρχείο. Σε περίπτωση δηλαδή που θέσουμε τιμή μεγαλύτερη του αριθμού των *slave* μηχανών θα δούμε μηνύματα λάθους:

```
<property>
 <name>dfs.replication</name>
<value>2</value>
<description>Default block replication. .
.</description>
 </property>
```

Έναρξη της συστοιχίας πολλών κόμβων

Αρχικά πρέπει να διαμορφώσουμε το *HDFS* (format). Για να το κάνουμε αυτό, τρέχουμε την εντολή:

```
bin/hadoop namenode -format
```

Η έναρξη της συστοιχίας γίνεται σε δύο βήματα:

πρώτα ξεκινούν οι *daemons* του *HDFS*. Συγκεκριμένα ο *namenode* στο *master*, και οι *datanode* στους *slave*.

Στη συνέχεια ξεκινούν οι *daemons* του *MapReduce*. Ο *JobTracker* στο *master*, και οι *TaskTracker* σε όλους.

HDFS daemons

Τρέχουμε την εντολή:

`<HADOOP_INSTALL>/bin/start-dfs.sh`

Στη μηχανή που θέλουμε να τρέξει ο *namenode*. Αυτό θα φέρει το *HDFS* με το *namenode* να τρέχει στη μηχανή που τρέξαμε την αμέσως προηγούμενη εντολή, και το *datanode* στις μηχανές που περιγράφονται στο αρχείο `conf/slaves`. Θα τρέξουμε λοιπόν στο *master* την εντολή:

```
bin/start-dfs.sh
```

Στο *slave* μπορούμε να δούμε την πορεία της εντολής στο αρχείο

`<HADOOP_INSTALL>/logs/hadoop-hadoop-datanode-slave.log.`

```
hadoop@master:/usr/local/hadoop$ jps

14799 NameNode

15314 Jps

14880 DataNode

14977 SecondaryNameNode
```

Όπως και οι διαδικασίες:

```
hadoop@slave:/usr/local/hadoop$ jps
15183 DataNode
15616 Jps
```

MapReduce Daemons

Θα χρησιμοποιήσουμε την εντολή:

`<HADOOP_INSTALL>/bin/start-mapred.sh`

Στη μηχανή που θέλουμε να τρέξει ο *JobTracker*. Αυτό θα φέρει τη συστοιχία του *MapReduce* με τον *JobTracker* να τρέχει στη μηχανή που τρέξαμε την προηγούμενη εντολή, και τους *TaskTracker* στις μηχανές της λίστας `conf/slaves`. Στην περίπτωσή μας τρέχουμε στο *master* την εντολή:

```
bin/start-mapred.sh
```

Στο *slave* μπορούμε να δούμε την πορεία της εντολής στο αρχείο `<HADOOP_INSTALL>/logs/hadoop-hadoop-tasktracker-slave.log`.

Σε αυτό το σημείο τρέχουν οι ακόλουθες διεργασίες Java στο *master*:

```
hadoop@master:/usr/local/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
```

Ενώ στο *slave*:

```
hadoop@slave:/usr/local/hadoop$ jps
15183 DataNode
15897 TaskTracker
16284 Jps
```

Παύση της συστοιχίας

Αρχικά θα σταματήσουμε τους *daemons* του *MapReduce*. Τον *JobTracker* στο *Master* και τον *TaskTracker* στο *slave*. Στη συνέχεια σταματάμε τους daemons του *HDFS*. Ο *namenode* στον *master* και ο *datanode* στους *slaves*.

MapReduce Daemons

Θα χρησιμοποιήσουμε την εντολή

`<HADOOP_INSTALL>/bin/stop-mapred.sh`

στη μηχανή του *JobTracker*

Αυτή θα κλείσει τη συστοιχία του *MapReduce* σταματώντας τον *JobTracker* που τρέχει στη μηχανή που τρέξαμε την προηγούμενη εντολή και τους *TaskTracker* στη λίστα του αρχείου `conf/slaves`. Τρέχουμε την εντολή:

```
bin/stop-mapred.sh
```

στο *master*

σε αυτό το σημείο τρέχουν οι διεργασίες Java στο *master*:

```
hadoop@master:/usr/local/hadoop$ jps
14799 NameNode
18386 Jps
14880 DataNode
14977 SecondaryNameNode
```

Ενώ στο *slave*:

```
hadoop@slave:/usr/local/hadoop$ jps
15183 DataNode
18636 Jps
```

HDFS Daemons

Θα χρησιμοποιήσουμε την εντολή:

`<HADOOP_INSTALL>/bin/stop-dfs.sh`

στον κόμβο του *namenode*. Αυτή θα κλείσει το *HDFS* σταματώντας τον *namenode* κόμβο στη μηχανή που τρέξαμε την προηγούμενη εντολή, και τους *datanodes* στη λίστα που υπάρχει στο αρχείο `conf/slaves`. Τρέχουμε στο *master*:

```
bin/stop-dfs.sh
```

Οι διεργασίες java που εκτελούνται στο *master* είναι:

```
hadoop@master:/usr/local/hadoop$ jps
18670 Jps
```

Ενώ στο *slave*:

```
hadoop@slave:/usr/local/hadoop$ jps
18894 Jps
```

Παράδειγμα διεργασίας MapReduce

Στο παράδειγμα αυτό θα πάρουμε ένα αρχείο text και θα μετρήσουμε τις λέξεις που υπάρχουν σε αυτό. Αρχικά κατεβάζουμε το eBook *«The Notebooks of Leonardo Da Vinci»*, σε μορφή *text*, κωδικοποίηση us-ascii, και το αποθηκεύουμε σε ένα φάκελο π.χ. `/tmp/Gutenberg`.

Βήμα 1ο

Επανεκινούμε το Hadoop:

```
~$ <HADOOP_INSTALL>/bin/start-all.sh
```

Βήμα 2ο

Αντιγράφουμε τα αρχεία στο *HDFS*:

```
/usr/local/hadoop$ bin/hadoop dfs -copyFromLocal
/tmp/source destination
```

Βήμα 3ο

Εκτελούμε το *MapReduce* με την εντολή:

```
/usr/local/hadoop$ bin/hadoop hadoop-example
wordcount destination output
```

Διαβάζουμε όλα τα αρχεία στο φάκελο "*destination*" του *HDFS* και αποθηκεύουμε το αποτέλεσμα στο "*output*".

Βήμα 4ο

Μπορούμε να αντιγράψουμε το αποτέλεσμα απο το *HDFS* στο δικό μας σύστημα αρχείων, ή πολύ απλά να τα διαβάσουμε κατ'ευθείαν απο το HDFS:

`/usr/local/hadoop$ bin/hadoop dfs –cat output/part-00000`

```
/usr/local/hadoop$ mkdir /tmp/output
/usr/local/hadoop$ bin/hadoop dfs –copyToLocal
output/part-00000 /tmp/output
```

3.4 Μετρητής λέξεων σε κείμενο

3.4.1 Τίτλος εφαρμογής

Το συγκεκριμένο παράδειγμα, *Word Count*, είναι το πρόβλημα καταμέτρησης του πλήθους των λέξεων σε ένα κείμενο. Η καταμέτρηση γίνεται όταν υπάρχει η ανάγκη οι λέξεις του κειμένου να μείνουν μέσα σε κάποια όρια. Αυτό συμβαίνει κυρίως στις περιπτώσεις πανεπιστημιακών εργασιών, διαφήμισης, δημοσιογραφικών κειμένων και λογοτεχνίας.

Αρχικά το πρόγραμμα θα τρέξει σε "*Standalone mode*", δηλαδή θα τρέξει στο τοπικό μας μηχάνημα. Αυτό σημαίνει ότι δεν χρησιμοποιούνται καθόλου *daemons*(*JobTracker, TaskTracker, NameNode, DataNode*) και όλες οι διεργασίες εκτελούνται σε μια εικονική μηχανή Java. Γενικά μπορούμε να πούμε ότι αυτό το mode μπορεί να χρησιμοποιηθεί για να εκτελέσει το πρόγραμμα στο στάδιο της ανάπτυξης, να το δοκιμάσει και να το απασφαλματώσει.

Στη συνέχεια το πρόγραμμα θα τρέξει σε "*Pseudo Distributed Mode*". Σε αυτή την περίπτωση όλες οι διεργασίες εκτελούνται σε ένα κόμβο. Σε αυτό τον κόμβο εκτελούνται όλοι οι daemons. Αυτό βέβαια σημαίνει ότι τα δεδομένα εισόδου πρέπει να εισαχθούν στο *HDFS*, το οποίο αποθηκεύεται τοπικά, κάτι που απαιτεί περισσότερο χρόνο μεταφοράς για τα δεδομένα εισόδου και εξόδου, αλλά και περισσότερο αποθηκευτικό χώρο. Αυτό το mode χρησιμοποιείται για να προσομοιώσει τη συστοιχία σε μικρότερη κλίμακα.[1]

3.4.2 Θεωρητική προσέγγιση

Στο συγκεκριμένο παράδειγμα βλέπουμε ένα εργαστήριο για τη δημιουργία προγράμματος σε *MapReduce*. Αρχικά είναι χρήσιμο να γνωρίζει τη λειτουργία των συναρτήσεων *Map* και *Reduce*. Θεωρητικά μπορούμε να δούμε τη διαδικασία *MapReduce* σαν πέντε βήματα παράλληλης επεξεργασίας:

1. Προετοιμασία της συνάρτησης εισόδου `Map()`: το σύστημα *MapReduce* αναθέτει τη διαδικασία Map σε επεξεργαστές, και σε κάθε επεξεργαστή αναθέτει μια τιμή-κλειδί, έστω *K1*. Αυτή η τιμή είναι ουσιαστικά και ο κωδικός της εργασίας που θα τρέξει σε αυτόν. Επίσης προμηθεύει τον επεξεργαστή με όλες τις απαραίτητες πληροφορίες για να τρέξει τη διεργασία.
2. Εκτέλεση του κώδικα για τη συνάρτηση `Map()` που του δίνει ο χρήστης: η συνάρτηση `Map()` τρέχει μια φορά για κάθε κλειδί *K1*, και παράγει μια έξοδο, στην οποία δίνουμε την τιμή *K2*.
3. Η έξοδος της `Map()` «ανακατεύεται» (*shuffle*) στους επεξεργαστές *Reduce*: το σύστημα *MapReduce* αναθέτει τη διαδικασία *Reduce* σε επεξεργαστές, και σε κάθε επεξεργαστή αναθέτει μια τιμή *K2*, για να επεξεργαστεί τα αντίστοιχα δεδομένα.
4. Εκτέλεση του κώδικα `Reduce()` που δίνει ο χρήστης: η συνάρτηση `Reduce()` εκτελείται μια φορά για κάθε τιμή *K2* που παράγεται απο τη συνάρτηση `Map()`.
5. Παραγωγή του τελικού αποτελέσματος: το σύστημα *MapReduce* συλλέγει τα αποτελέσματα της `Reduce()` και τα ταξινομεί ανάλογα με την τιμή *K2* για να δημιουργήσει το τελικό αποτέλεσμα. [1, 10, 11, 12]

3.4.3 Πειραματική προσέγγιση

Για το συγκεκριμένο παράδειγμα, το οποίο είναι απο το εργαστήριο του Rice University μπορούμε να βρούμε τις απαραίτητες πηγές στο:

`https://wiki.rice.edu/confluence/display/PARPROG/COMP322`

Δημιουργία περιβάλλοντος

Βήμα 1ο

Αντιγράφουμε το αρχείο tar στο home directory:

```
cp /home/yz17/comp322-lab12.tar
```

Βήμα 2ο

Στο directory `lab_12/` τρέχουμε την εντολή για να τρέξουν τα *mpiJava*:

```
tar -xvf comp322-lab12.tar
```

τώρα θα μπορούμε να δούμε το directory hadoop-1.0.3, στο οποίο θα δουλεύουμε στη συνέχεια.

Γράφουμε το πρόγραμμα

Βήμα 1ο

Βρίσκουμε το αρχείο `WordCount.java` και συμπληρώνουμε τον κώδικα για τις συναρτήσεις `map()` και `reduce()`, όπως υποδεικνύουν τα σχόλια TODO.

Βήμα 2ο

Κάνουμε compile το πρόγραμμα με την εντολή:

```
./compileWordCount.sh
```

Θα δούμε το αρχείο `WordCount.jar`

Εκτέλεση προγράμματος

Εκτέλεση σε Standalone mode:

Βήμα 1ο

Πηγαίνουμε στο φάκελο `hadoop-1.0.3`.

Αλλάζουμε σε *Standalone mode* με την εντολή:

```
./changeToStandAlone.sh
```

Βήμα 2ο

Τρέχουμε το πρόγραμμα *WordCount* με την εντολή:

```
bin/hadoop jar WordCount.jar WordCount inputFiles
output
```

Αξίζει να σημειώσουμε οτι η *Standalone mode* χρησιμεύει για αποσφαλμάτωση, άρα στην περίπτωση αυτή δεν γίνεται εκκίνηση του *HDFS*.

Για να δούμε το αποτέλεσμα εκτελούμε:

```
cat output/*
```

Εκτέλεση σε Pseudo – Distributed mode:

Αφού ουσιαστικά κάναμε έναν έλεγχο σωστής λειτουργίας του προγράμματος μπορούμε πλέον να δημιουργήσουμε ένα πλήρες σύστημα *Hadoop MapReduce* με πολλούς πυρήνες.

Βήμα 1ο

Μεταφερόμαστε απο *Standalone mode* σε *Pseudo-Distributed mode* με την εντολή:

```
./changeToPD.sh
```

Βήμα 2ο

Εκτελούμε το πρόγραμμα *WordCount* με την εντολή:

```
./runWordCountPD.sh
```

Καταγράφουμε το χρόνο εκτέλεσης του προγράμματος απο το script. Το Hadoop επιτρέπει στο χρήστη να κάνει παραμετροποιήσεις, συγκεκριμένα πόσα map tasks μπορούν να τρέχουν παράλληλα, δηλαδή πόσοι πυρήνες. Αυτό γίνεται απο το `mapred.tasktracker.map.tasks.maximum`

Για να το κάνουμε αυτό ανοίγουμε το `conf/mapred-site.xml` με ένα text editor και αλλάζουμε την τιμή του `mapred.tasktracker.map.tasks.` σε 2.

Βήμα 3ο

Τρέχουμε το πρόγραμμα και καταγράφουμε το χρόνο εκτέλεσης:

`./runWordCountPD.sh`

Αυτός ο χρόνος εκτέλεσης είναι για την περίπτωση των δύο map tasks παράλληλα.

Επαναλαμβάνουμε τη διαδικασία και αλλάζουμε την τιμή της παραμέτρου σε 4 και 8 και καταγράφουμε τους αντίστοιχους χρόνους στο αρχείο `lab 12 written.txt`. Σε αυτό το αρχείο έχουν καταγραφεί οι χρόνοι εκτέλεσης των παραδειγμάτων μας.

3.4.4 Διαφορετική προσέγγιση

Βέβαια, το πρόβλημα καταμέτρησης λέξεων υπήρχε εδω και πολλές δεκαετίες. Το Hadoop μπορεί να προσφέρει ένα διαφορετικό τρόπο επίλυσης αλλά αξίζει να δούμε πώς αντιμετωπίστηκε στο παρελθόν. Ο πιο συνηθισμένος τρόπος, ο οποίος χρησιμοποιείται και σήμερα στους περιηγητές Ιστού, είναι με κώδικα Javascript. Παρακάτω δείχνουμε ένα παράδειγμα κώδικα που μετρά τις λέξεις σε ένα κείμενο που εισάγει ο χρήστης:

```
<form method="POST" name="wordcount">
  <script language="JavaScript">
function countit(){
var formcontent=document.wordcount.wordcount2.value
formcontent=formcontent.split(" ")
document.wordcount.wordcount3.value=formcontent.length
}
</script>
<table border="0" cellspacing="0" cellpadding="0">
    <tr>
      <td         width="100%"><textarea         rows="12"
name="wordcount2" cols="60" wrap="virtual"></textarea></td>
    </tr>
    <tr>
      <td     width="100%"><div     align="right"><p><input
type="button" value="Calculate Words"
      onClick="countit()">        <input        type="text"
name="wordcount3" size="20"></p>
      <div   align="center"><center><p><font   face="arial"
size="-2">This free script provided by</font>
      <font     face="arial,     helvetica"     size="-2"><a
href="http://javascriptkit.com">JavaScript
      Kit</a></font></p>
      </center></div></div></td>
    </tr>
  </table>
</form>
```

3.5 Κατανόηση λειτουργιών Hadoop και εκτέλεση απλού παραδείγματος

3.5.1 Περιγραφή παραδείγματος

Στο παρακάτω παράδειγμα θα δούμε πώς κάνουμε εγκατάσταση του Hadoop σε *command line* και στον *Eclipse*, ενώ και στις δυο περιπτώσεις θα δούμε κάποια παραδείγματα που μας βοηθούν να κατανοήσουμε τις λειτουργίες και τα χαρακτηριστικά του.[8]

3.5.2 Θεωρητική προσέγγιση

Στο συγκεκριμένο παράδειγμα θα χρησιμοποιήσουμε μια εικονική μηχανή (Virtual Machine). Η εικονική μηχανή είναι μια εκτέλεση μιας μηχανής σε λογισμικό. Οι εικονικές μηχανές χωρίζονται σε δύο κατηγορίες:

1. Εικονική μηχανή συστήματος, η οποία παρέχει μια ολοκληρωμένη πλατφόρμα που υποστηρίζει την εκτέλεση οποιουδήποτε λειτουργικού συστήματος. Σε αυτή την περίπτωση εξομοιώνουν ένα λειτουργικό σύστημα, ενώ δεν υπάρχει διαθέσιμο το αντίστοιχο hardware ή στην καλύτερη χρήση των πηγών του υπολογιστή σε θέματα βελτίωσης απόδοσης ή μείωσης κατανάλωσης ενέργειας.
2. Εικονική μηχανή διαδικασιών, που σχεδιάστηκε για να εκτελεί ένα μόνο πρόγραμμα, δηλαδή μια συγκεκριμένη εργασία. Αυτές οι μηχανές συνήθως γράφονται για να υποστηρίζουν περισσότερες απο μια γλώσσες για τη μεταφερσιμότητα του προγράμματος.

3.5.3 Πειραματική προσέγγιση

Είναι απαραίτητο αρχικά να έχουμε εγκαταστήσει ένα *Java Developer Kit (JDK)*, έστω την έκδοση 6, και μια έκδοση των Linux, όπου εκεί θα δουλεύουμε. Πιο αναλυτικά:

Εγκατάσταση VMware Player

Βήμα 1ο

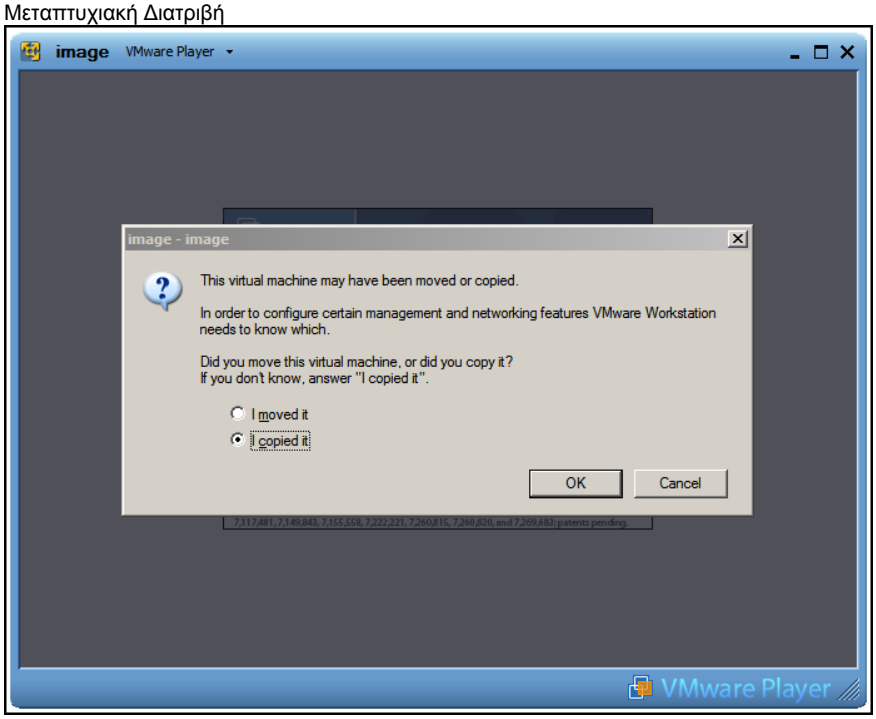
Κατεβάζουμε μια έκδοση του *VMware Player* (έστω την 2.5) για 32 - bit Windows και Linux, και κάνουμε την εγκατάστασή της.

Βήμα 2ο

Αντιγράφουμε το *Hadoop Virtual Machine* στο σκληρό μας δίσκο. Είναι ο φάκελος `hadoop-0.20.0-vmware` σε μορφή *zip*, που περιέχει τα εξής αρχεία: ένα αρχείο `.vmdk` που έχει την εικόνα του σκληρού δίσκου της εικονικής μηχανής, και ένα `.vmx` που περιέχει τις πληροφορίες για να ξεκινήσουμε την εικονική μηχανή. Αφού κάνουμε *unzip* το φάκελο `vmware`, κάνουμε διπλό κλικ στο `.vmx`.

Βήμα 3ο

Στη συνέχεια θα επιλέξουμε την επιλογή ότι κάναμε αντιγραφή της εικόνας, όπως φαίνεται στην εικόνα 4.6.1.1:



Εικόνα 3.5.3.1: όταν εκκινούμε για πρώτη φορά το *VMware Player* επιλέγουμε ότι έχουμε αντιγράψει την εικόνα [8]

Τότε ο *VMware Player* θα δημιουργήσει νέα αναγνωριστικά για την εικονική μηχανή. Αφού επιλέξουμε *OK*, θα γίνει η εκκίνηση των Linux. Θα πάρει μια διεύθυνση IP που δεν χρησιμοποιείται και θα ζητήσει από το χρήστη να συνδεθεί.

Λογαριαμοί χρηστών

Η εικονική μηχανή έχει προκαθορισμένους δύο λογαριασμούς χρηστών, "*root*" και "*guest*", με τον ίδιο κωδικό. Ο λογαριασμός *guest* έχει δυνατότητες διαχείρισης συστήματος, π.χ. να κλείσει την εικονική μηχανή. Για να μπούμε σαν *guest* κάνουμε κλικ μέσα στην οθόνη της εικονικής μηχανής, και ο απαραίτητος κωδικός είναι *hadoop*. Για να μπούμε σαν *root* ο κωδικός είναι *root*. Για να επιστρέψουμε στα Windows πατάμε *CTR+ALT*.

Εκτέλεση διεργασίας Hadoop

Αν χρησιμοποιούμε το *VMware Player* κάνουμε *log in* σαν *guest* και θα ξεκινήσουμε στο *home directory*:

`/home/guest`

Αν πληκτρολογήσουμε άσσους θα δούμε ένα φάκελο με το όνομα `hadoop/` και *scripts* για τη διαχείριση του *server*. Για να ξεκινήσουμε το Hadoop εκτελούμε τις παρακάτω γραμμές κώδικα:

```
~$ ./stop-hadoop
~$ ./start-hadoop
```

Αν εμφανιστεί κάποιο μήνυμα που να ρωτάει αν θέλουμε να συνδεθούμε στον τρέχοντα *host* πληκτρολογούμε "*yes*".

Για να ελέγξουμε αν στήσαμε σωστά το Hadoop θα τρέξουμε ένα παράδειγμα:

```
~$ cd hadoop
~/hadoop$ bin/hadoop jar hadoop-0.20.0-examples.jar pi 10
1000000
```

Αυτό θα μας δώσει την ακόλουθη έξοδο:

```
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
 ...
Wrote input for Map #10
Starting Job
INFO mapred.FileInputFormat: Total input paths to process: 10
INFO mapred.JobClient: Running job: job_200806230804_0001
INFO mapred.JobClient: map 0% reduce 0%
INFO mapred.JobClient: map 10% reduce 0%
...
INFO mapred.JobClient: map 100% reduce 100%
INFO mapred.JobClient: Job complete: job_200806230804_0001
...
Job Finished in 25.841 second Estimated value of PI is 3.141688
```

το οποίο υπολογίζει την τιμή του π μέχρι μερικά ψηφία με βάση κάποια δείγματα. Υπάρχουν όμως και περιπτώσεις που ο *server* του VM Hadoop δεν λειτουργεί. Τότε μπορούμε να ακολουθήσουμε την παρακάτω διαδικασία:

Για να ελέγξουμε αν έγινε σωστή εκκίνηση:

```
~$ cd hadoop/bin
~/hadoop/bin$ hadoop dfsadmin -report
Datanodes available: 1 (1 total, 0 dead)
```

Αν δεν δούμε αυτή την εικόνα , κάτι δεν έγινε σωστά. Τότε μπορούμε να δοκιμάσουμε:

```
~/hadoop/bin$ hadoop dfsadmin –safemode leave
~/hadoop/bin$ hadoop dfsadmin –report
```

Εγκατάσταση του Eclipse

Θα χρησιμοποιήσουμε την έκδοση 3.3.1, την "*Europa*", που είναι συμβατή με το Hadoop.

Εγκατάσταση Hadoop MapReduce

Το Hadoop έχει ένα *plugin* για τον *Eclipse* που διευκολύνει την ανάπτυξη προγραμμάτων *MapReduce*. Στο φάκελο `hadoop-0.20.0/contrib/eclipse-plugin` θα βρούμε το `hadoop-0.20.0-eclipse-plugin.jar`, το οποίο αντιγράφουμε στο `plugins/`.

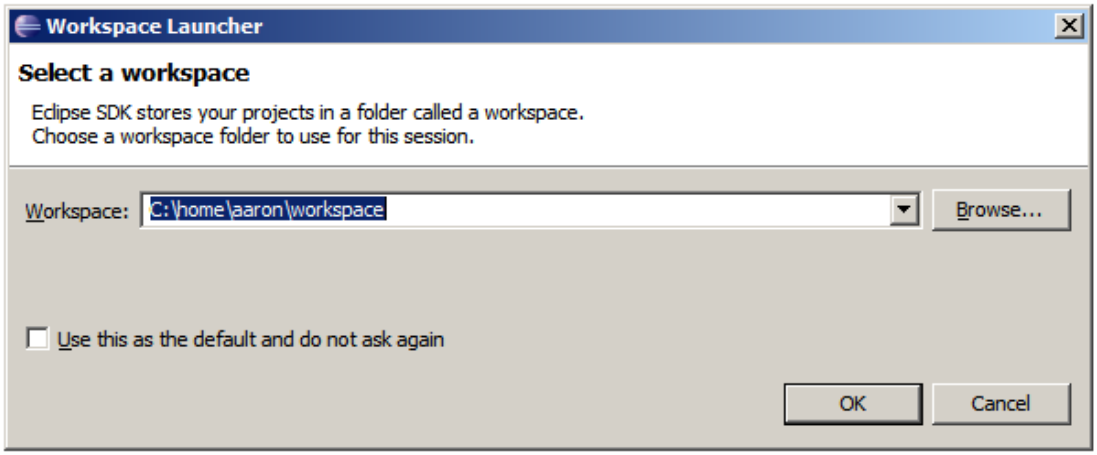
Αντίγραφο του Hadoop

Ενώ τρέχουμε προγράμματα *MapReduce* στην εικονική μηχανή, θα τα κάνουμε *compile* στην *host*. Συνεπώς χρειαζόμαστε τα αρχεία *jar*, και θα πρέπει να αντιγράψουμε στο δίσκο μας το `/hadoop-0.20.0`.

Εκτέλεση του Eclipse

Επιλέγουμε με διπλό κλικ το `eclipse.exe` για να ξεκινήσει. Θα αποθηκεύει τα *project* και τις ρυθμίσεις τους στον φάκελο *workspace*.

Όταν ξεκινήσουμε τον *Eclipse* θα μας ζητήσει ένα φάκελο για *workspace*. Επιλέγουμε ένα και πατάμε *OK*.



Εικόνα 3.5.3.2: επιλέγουμε το φάκελο για workspace [8]

Διαμόρφωση του Plug – in για το MapReduce

Βήμα 1ο

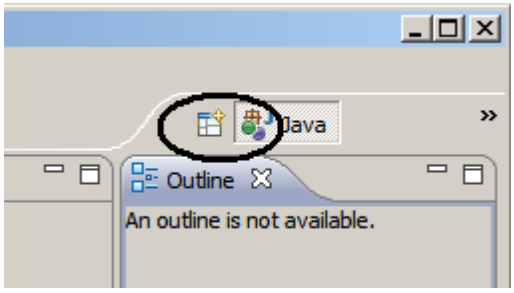
Εκκίνηση του *Eclipse* και επιλογή *workspace*. Επιλέγουμε "*Go to the Workbench*", που είναι η κύρια οθόνη και μπορούμε να γράψουμε κώδικα, να εκκινήσουμε τα προγράμματα και να διαχειριστούμε τα *project*.

Βήμα 2ο

Ανοίγουμε την εικονική μηχανή. Επιλέγουμε με διπλό κλικ το αρχείο `.vmx` για να ξεκινήσει η εικονική μηχανή και η διαδικασία εκκίνησης των Linux.

Βήμα 3ο

Αλλάζουμε στην οθόνη του *MapReduce*. Στο *workbench* επιλέγουμε "*Open Perspective*":

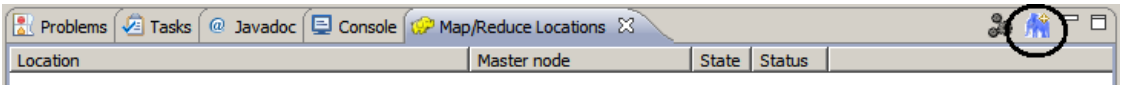


Εικόνα 3.5.3.2: αλλάζουμε σε οθόνη MapReduce.[8]

Επιλέγουμε "*Other*", και στο νέο παράθυρο "*Map/Reduce*". Στο μενού επιλέγουμε *Window - Show View - Other*. Στο "*MapReduce Tools*" επιλέγουμε "*Map/Reduce Locations*". Αυτό θα δημιουργήσει μια νέα καρτέλα στο κάτω μέρος της οθόνης, δίπλα απο τα "*Problems*" και "*Tasks*".

Βήμα 4ο

Προσθέτουμε τον εξυπηρετητή. Στην ταμπέλα "*Map/Reduce Locations*" κάνουμε κλικ στο εικονίδιο του ελέφαντα και προσθέτουμε νέο *server* στον *Eclipse*.



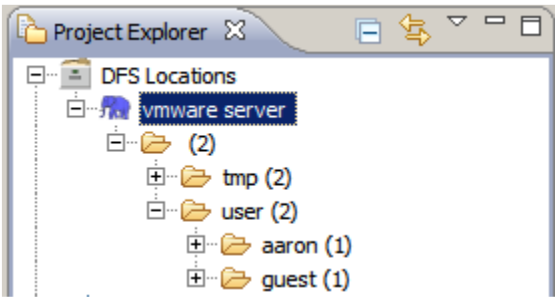
Εικόνα 3.5.3.3: προσθήκη νέου server[8]

Θα μας ζητηθεί να ρυθμίσουμε διάφορες παραμέτρους του *server*. Για να συνδεθούμε στην εικόνα του *VMware* οι τιμές είναι:

```
Location name: (π.χ. "VMware server")
Map/Reduce Master Host: (η διεύθυνση IP στην αρχή)
Master Port: 54311
DFS Master Port: 54310
User name: guest
```

Στη συνέχεια κάνουμε κλικ στο "*Advanced*". Επιλέγουμε το *hadoop.job.ugi*. Αυτό περιέχει τα πιστοποιητικά μας για *login* στα Windows. Επιλέγουμε την πρώτη τιμή της λίστας που είναι το username και το αλλάζουμε σε "*guest*".

Τελειώνοντας επιλέγουμε "*Finish*". Ο *server* θα φαίνεται τώρα στην ταμπέλα "*Map/Reduce Locations*". Αν ελέγξουμε το "*Project Explorer*", θα δούμε οτι υπάρχει η δυνατότητα να δούμε το *HDFS*.



Εικόνα 3.5.3.4: τα ορατά αρχεία του HDFS[8]

Αλληλεπίδραση με το HDFS

Το **VMware** μας δίνει ένα κόμβο για να τρέξουμε τις εφαρμογές **MapReduce**. Μπορούμε να δούμε τα περιεχόμενα του **HDFS** όπως αναφέραμε πάνω.

Παράδειγμα

Κατ αρχάς πρέπει να δημιουργήσουμε το *project*. Επιλέγουμε *File - New – Project*. Στη συνέχεια "*Map/Reduce Project*" και *Next*. Δίνουμε ένα όνομα στο *project* π.χ. "*WordCount*" και τη βιβλιοθήκη εγκατάστασης του Hadoop "*Configure Hadoop install directory...*". επιλέγουμε το φάκελο που έχουμε το Hadoop και εκεί θα βρούμε το αρχείο `hadoop-0.20.0-core.jar`. Αν δημιουργήσουμε ένα project *MapReduce* θα έχει τις βιβλιοθήκες jar. Τελικά επιλέγουμε "*Finish*".

Δημιουργία των source files

Το πρόγραμμα χρειάζεται τρεις κλάσεις. *Mapper*, *Reducer*, και *Driver*. Ο *Driver* υποδεικνύει στο Hadoop πως να τρέχει τη διεργασία *MapReduce*. Ο *Mapper* και ο *Reducer* δρούν στα δεδομένα.

Κάνουμε δεξί κλικ στο "*src*" και επιλέγουμε *New – Other*.

Στο παράθυρο που ανοίγει, στην ταμπέλα "*Map/Reduce*" μπορούμε να δημιουργήσουμε κλάσεις *Mapper*, *Reducer*, και *Driver* με βοήθεια απο τον *Eclipse*. Δημιουργούμε αντίστοιχα τις κλάσεις *WordCountMapper*, *WordCountReducer*, και *WordCount*. Ο κώδικας για αυτές τις κλάσεις είναι:

WordCountMapper.java:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WordCountMapper extends MapReduceBase
 implements Mapper<LongWritable, Text, Text, IntWritable> {
private final IntWritable one = new IntWritable(1);
private Text word = new Text();

public   void   map(WritableComparable   key,   Writable   value,
OutputCollector output, Reporter reporter) throws IOException {

String  line  =  value.toString();  StringTokenizer  itr  =  new
StringTokenizer(line.toLowerCase());
while(itr.hasMoreTokens())       {       word.set(itr.nextToken());
output.collect(word, one);
}
}
}
```

WordCountReducer.java:

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WordCountReducer extends MapReduceBase
 implements Reducer<Text, IntWritable, Text, IntWritable>
{

public   void   reduce(Text   key,   Iterator   values,
OutputCollector   output,   Reporter   reporter)   throws
IOException {
 int sum = 0;
while (values.hasNext()) {
 IntWritable value = (IntWritable) values.next(); sum +=
value.get(); // process value
}
 output.collect(key, new IntWritable(sum));
 }
}
```

WordCount.java:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
public class WordCount {
public static void main(String[] args) {
 JobClient client = new JobClient();
 JobConf conf = new JobConf(WordCount.class);
// specify output types
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
//      specify      input      and      output      dirs
FileInputPath.addInputPath(conf,   new   Path("input"));
FileOutputPath.addOutputPath(conf, new Path("output"));
// specify a mapper
conf.setMapperClass(WordCountMapper.class);
//             specify             a             reducer
conf.setReducerClass(WordCountReducer.class);
conf.setCombinerClass(WordCountReducer.class);
client.setConf(conf); try { JobClient.runJob(conf);
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Έναρξη της διεργασίας

Αφού γράψαμε τον κώδικα, πρέπει να τον εκτελέσουμε. Έχουμε δημιουργήσει ένα φάκελο με το όνομα `input` στο `/user/guest` στο *HDFS*. Κάνουμε δεξί κλίκ στο `WordCount.java`, και στο παράθυρο επιλέγουμε *Run As - Run On Hadoop*. Ένα άλλο παράθυρο θα μας ρωτήσει την τοποθεσία που θα τρέχει το Hadoop και επιλέγουμε το *VMware server*.

Στην κονσόλα του *Eclipse* μπορούμε να δούμε την πρόοδο του Hadoop. Θα είναι περίπου έτσι:

```
08/06/25 12:14:22 INFO mapred.FileInputFormat: Total input paths
to process : 3
08/06/25 12:14:23 INFO mapred.JobClient: Running job:
job_200806250515_0002
08/06/25 12:14:24 INFO mapred.JobClient: map 0% reduce 0% 08/06/25
12:14:31 INFO mapred.JobClient: map 50% reduce 0% 08/06/25
12:14:33 INFO mapred.JobClient: map 100% reduce 0% 08/06/25
12:14:42 INFO mapred.JobClient: map 100% reduce 100% 08/06/25
12:14:43 INFO mapred.JobClient: Job complete:
job_200806250515_0002
08/06/25 12:14:43 INFO mapred.JobClient: Counters: 12
08/06/25 12:14:43 INFO mapred.JobClient: Job Counters
08/06/25 12:14:43 INFO mapred.JobClient: Launched map tasks=4
08/06/25 12:14:43 INFO mapred.JobClient: Launched reduce tasks=1
08/06/25 12:14:43 INFO mapred.JobClient: Data-local map tasks=4
08/06/25 12:14:43 INFO mapred.JobClient: Map-Reduce Framework
08/06/25 12:14:43 INFO mapred.JobClient: Map input records=211
08/06/25 12:14:43 INFO mapred.JobClient: Map output records=1609
```

```
08/06/25 12:14:43 INFO mapred.JobClient: Map input bytes=11627
08/06/25 12:14:43 INFO mapred.JobClient: Map output bytes=16918
08/06/25 12:14:43 INFO mapred.JobClient: Combine input records=1609
08/06/25 12:14:43 INFO mapred.JobClient: Combine output records=682
08/06/25 12:14:43 INFO mapred.JobClient: Reduce input groups=568
08/06/25 12:14:43 INFO mapred.JobClient: Reduce input records=682
08/06/25 12:14:43 INFO mapred.JobClient: Reduce output records=568
```

Στον explorer του *DFS* κάνουμε δεξί κλικ στο `/user/` και επιλέγουμε "*Refresh*". Θα δούμε ένα φάκελο "`output`" που να έχει το αρχείο `part-00000`. Σε αυτό θα είναι τα αποτελέσματα της διεργασίας μας.

3.6 Παράδειγμα εγκατάστασης Hadoop σε command line και εκτέλεση απλού παραδείγματος

3.6.1 Περιγραφή παραδείγματος

Αυτό το εργαστήριο δείχνει πώς γίνεται η εγκατάσταση του Hadoop σε command line, ενώ θα δούμε τις βασικές εντολές του και το αποτέλεσμα που δίνουν.[9]

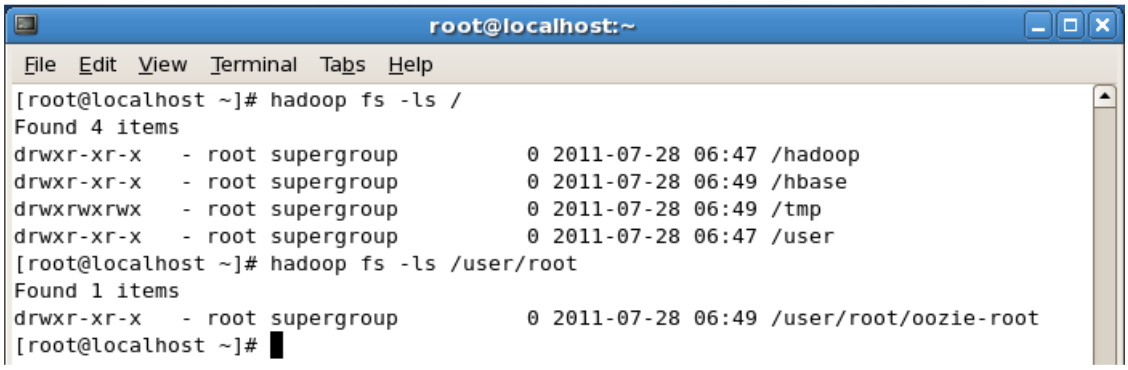
3.6.2 Θεωρητική προσέγγιση

Τα θεωρητικά στοιχεία για την κατανόηση του παραδείγματος αναφέρθηκαν παραπάνω, στην ενότητα 3.4.

3.6.3 Πειραματική προσέγγιση

Μια από τις βασικές εντολές είναι η "`hadoop fs`". Επίσης για να δούμε τα περιεχόμενα του φακέλου `/user/root` εκτελούμε:

```
> hadoop fs –ls /
> hadoop fs –ls /user/root
```



Εικόνα 3.6.3.1: τα αρχεία του φακέλου root[9]

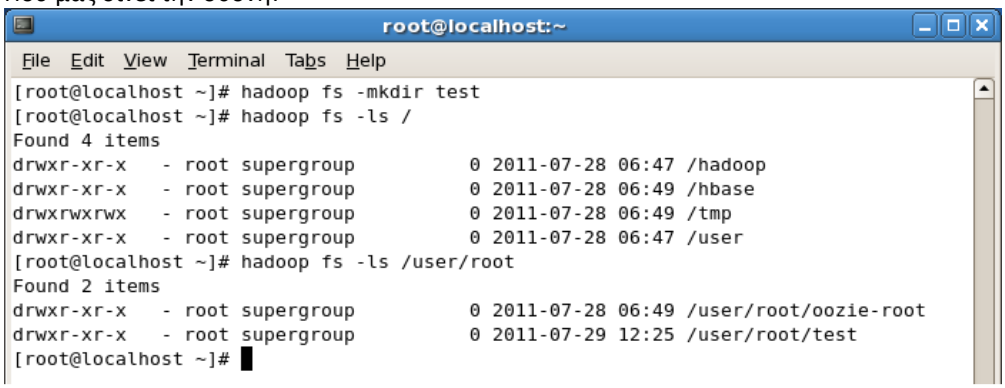
Αν θέλουμε να δημιουργήσουμε ένα φάκελο, έστω τον “test”:

```
> hadoop fs –mkdir test
```

Και για να δούμε το αποτέλεσμα:

```
> hadoop fs –ls /
> hadoop fs –ls /user/root
```

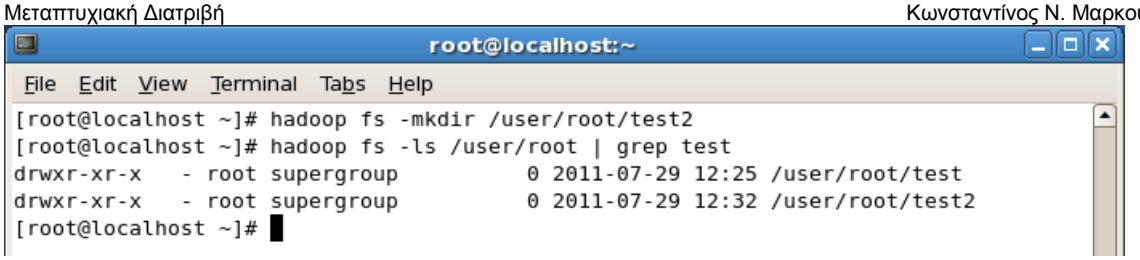
Που μας δίνει την οθόνη:



Εικόνα 3.6.3.2: φτιάχνουμε το φάκελο “test”

Μπορούμε να χρησιμοποιήσουμε το “`grep`” για να δούμε για παράδειγμα ποιες γραμμές κώδικα έχουν τη λέξη “test”:

```
> hadoop fs –mkdir /user/root/test2
> hadoop fs –ls /user/root | grep test
```



Εικόνα 3.6.3.3: εντολή grep

Για να μετακινήσουμε αρχεία μεταξύ του Linux και του HDFS με τις εντολές "`put`" "`get`". Για να μετακινήσουμε ένα αρχείο στο Hadoop:

```
> hadoop fs -put /BigDataUniversity/README README
> hadoop fs -ls /user/root
```

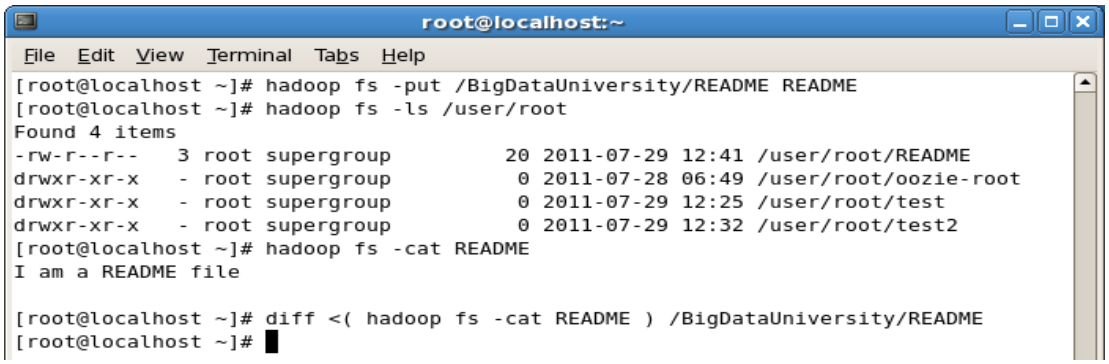
Αυτό θα μετακινήσει το αρχείο στο `/user/root/README`. Για να δούμε τα περιεχόμενά του εκτελούμε:

```
> hadoop fs -cat README
```

Επιπλέον μπορούμε να δούμε αν είναι ίδιο με το αρχικό:

```
> diff <( hadoop fs -cat README ) /BigDataUniversity/README
```

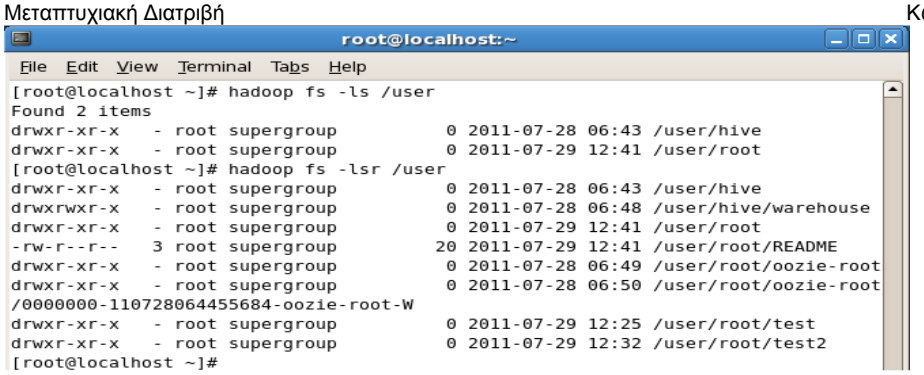
Αν αυτή η εντολή δεν δώσει κάποιο αποτέλεσμα, τότε το αρχείο είναι το ίδιο.



Εικόνα 3.6.3.3: η εντολή diff δεν δίνει αποτέλεσμα, αρα τα αρχεία ειναι ίδια

Μπορούμε επίσης να χρησιμοποιήσουμε αναδρομικά τις εντολές του *HDFS*. Για να το κάνουμε αυτό προσθέτουμε το "r" (ενώ στα Linux αυτό γίνεται με το -R). Για παράδειγμα για να χρησιμοποιήσουμε μια αναδρομική λίστα εκτελούμε:

```
> hadoop fs -ls /user
> hadoop fs -lsr /user
```



Εικόνα 3.6.3.4: αναδρομικές εντολές

Για να δούμε το μέγεθος των αρχείων μας χρησιμοποιούμε τις εντολές “`du`”, “`dus`”:

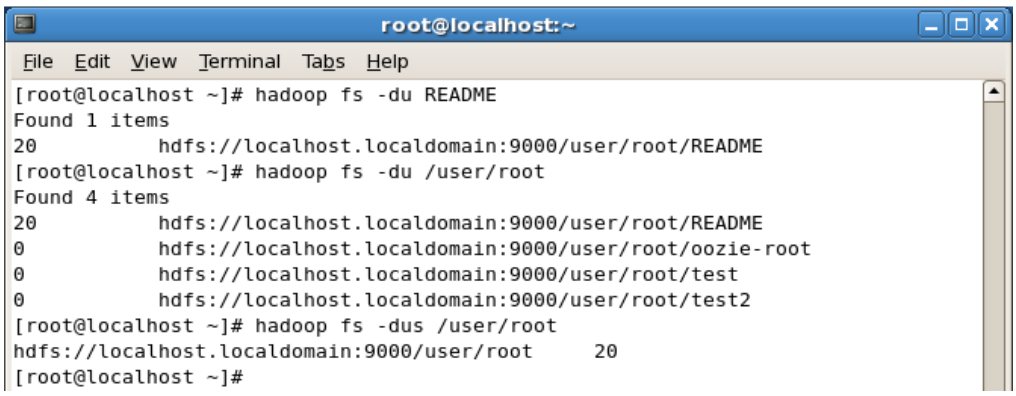
```
> hadoop fs -du README
```

Για να δούμε το μέγεθος των αρχείων χωριστά στο `/user/root`:

```
> hadoop fs -du /user/root
```

Ενώ για να δούμε το συνολικό μέγεθος των αρχείων:

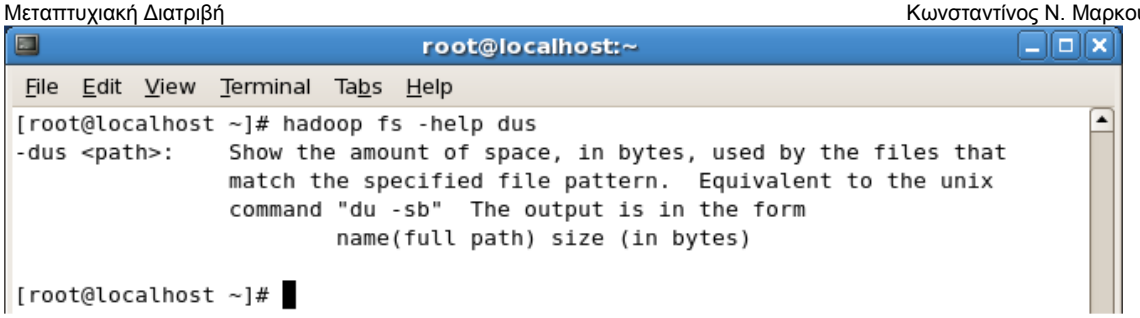
```
> hadoop fs -dus /user/root
```



Εικόνα 3.6.3.5: το μέγεθος των αρχείων

Αν θέλουμε να μάθουμε πληροφορίες για μια εντολή αρκεί να προσθέσουμε το “*help*”. Για να δούμε για παράδειγμα πληροφορίες για την “`dus`”:

```
> hadoop fs -help dus
```



Εικόνα 3.6.3.6: εντολή "dus"

3.7 Παράδειγμα εγκατάστασης Hadoop και HBase

3.7.1 Περιγραφή παραδείγματος

Αυτό το παράδειγμα προέρχεται απο εργαστηριακές ασκήσεις που έγιναν στο Ε.Μ.Π., για την εγκατάσταση του Hadoop. Χρησιμοποιήθηκε το Okeanos, ένα project που είναι στην υπηρεσία της ελληνικής ακαδημαϊκής κοινότητας.[6]

3.7.2 Θεωρητική προσέγγιση

Στο θεωρητικό μέρος αναφερθήκαμε γενικά στην *HBase*. Καλό θα ήταν να αναφερθούμε σε αυτό το σημείο εκτενέστερα για να κατανοήσουμε αυτή την άσκηση καλύτερα. Η *HBase* είναι μια ΒΔ ανοιχτού κώδικα, μη σχεσιακή και είναι γραμμένη σε Java στα πρότυπα του *Big Table* της Google. Είναι ένα project του Apache Hadoop και εκτελείται ένα επίπεδο πάνω από το *HDFS*. Προσφέρει ανθεκτικότητα στα σφάλματα αποθήκευσης δεδομένων, συμπίεση δεδομένων, ενώ οι πίνακες της μπορούν να χρησιμοποιηθούν ως είσοδος/έξοδος του *MapReduce*. [15]

Σε αυτό το παράδειγμα χρησιμοποιείται το *Okeanos*, το οποίο είναι μια υπηρεσία *υπολογιστικού νέφους* για την ελληνική ακαδημαϊκή έρευνα. Προσφέρει εικονικές μηχανές, δίκτυα και αποθηκευτικό χώρο, ενώ προσφέρεται στους φοιτητές στην Ελλάδα και στα μέλη ερευνητικών πανεπιστημιακών προγραμμάτων. Όλη η διαδικασία γίνεται διαδικτυακά, που σημαίνει ότι υπάρχει η δυνατότητα συνεχούς πρόσβασης στις εικονικές μηχανές και φυσικά παύση τους όταν τελειώσει το οποιοδήποτε project. [16]

Σε αυτό το παράδειγμα θα δούμε την εγκατάσταση του Hadoop και του *HBase* σε περιβάλλον *Debian*. Μαζί με τις οδηγίες εγκατάστασης θα δούμε παραδείγματα χρήσης των εργαλειών από το *command line* και από το *Eclipse*. Η δομή του οδηγού έχει ως εξής: Αρχικά περιγράφουμε την διαδικασία δημιουργίας εικονικών μηχανών με λειτουργικό *debian* είτε στο υπολογιστικό νέφος *Okeanos* είτε στο πρόγραμμα *VirtualBox*. Στη συνέχεια βλέπουμε τα βήματα εγκατάστασης του Hadoop σε δύο κόμβους. Μετά την εγκατάσταση περιγράφουμε τα εργαλεία ελέγχου της σωστής λειτουργίας και εκτελούμε ορισμένα παραδείγματα του *MapReduce* και *HDFS* είτε σε command line είτε σε προγραμματιστικό περιβάλλον. Τέλος, την ίδια διαδικασία κάνουμε και για την κατανεμημένη *NoSQL* ΒΔ *HBase*.

3.7.3 Πειραματική προσέγγιση

Εικονικές μηχανές

Okeanos

Βήμα 1ο

Μέσω του Okeanos (`http://cyclades.okeanos.grnet.gr/ui/`) δημιουργούμε 2 εικονικές μηχανές (*VM*). Επιλέγουμε σαν image το *debian_ATDS_2014* (στα *public images*), επιλέγουμε *flavor (cores, ram, disk* και *storage Archipelago*) επιλέγουμε τη σύνδεση χωρίς τη χρήση κλειδιού ssh και σημειώνουμε το *password* που θα μας εμφανιστεί κατά τη δημιουργία των *VMs*. Σε περίπτωση που δεν το σημειώσουμε, δεν θα έχουμε πρόσβαση στα *VM*.

Προσοχή: Στην επιλογή *Network* δεν επιλέγουμε κάποιο *network* μέχρι τη δημιουργία των *VMs* (εκτός από το προεπιλεγμένο IPv6 public network).

Βήμα 2ο

Μόλις έχουν δημιουργηθεί τα *VMs* (μπορούμε να δούμε την κατάσταση τους στην κονσόλα machines), επιλέγουμε την καρτέλα IP. Δημιουργούμε νέα διεύθυνση IP (αν δεν έχουμε ήδη) και επιλέγουμε την επιλογή *attach* (εμφανίζεται μόλις τοποθετήσουμε το ποντίκι επάνω στην νέα IP). Στο μενού που εμφανίζεται, επιλέγουμε το ένα από τα 2 *VMs*. Το *VM* αυτό θα θεωρείται ο *master* στο εξής και το δεύτερο *VM* θα θεωρείται ο *slave*. Για να δούμε αν έχει συνδεθεί επιτυχώς ο *master* εκτελούμε:

```
ping
```

(από *terminal* Linux ή *command prompt* Windows) στην IPv4 που δημιουργήσαμε προηγουμένως. Μπορεί να χρειαστούν αρκετά λεπτά μέχρι να πάρει ο *master* την IPv4 που του δώσαμε.

Βήμα 3ο

Στη συνέχεια, δημιουργούμε ένα νέο ιδιωτικό δίκτυο από την καρτέλα *networks* του Okeanos επιλέγοντας *New Network*. Δίνουμε ένα όνομα (πχ, atds) με *Network Subnet* 192.168.0.0/24. Στο νέο μας δίκτυο, επιλέγουμε την επιλογή *Connect machine* και συνδέουμε και τα δυο *VMs* που φτιάξαμε προηγουμένως. Τώρα ο *master* έχει 2 IP διευθύνσεις: Μία *public* με την οποία μπορούμε να συνδεόμαστε και μία *private* με την οποία μπορεί να επικοινωνεί με τον *slave*.

Αντίστοιχα ο slave έχει μία IP διεύθυνση: αυτή που τον συνδέει στο ιδιωτικό δίκτυο.

Για να δοκιμάσουμε ότι όλα λειτουργούν όπως θα έπρεπε κάνουμε *ssh* στον *master* (χρησιμοποιώντας κάποιον *ssh client*) στην *public* IP (της μορφής 83.212.x.x). Δίνουμε το *password* του *master* και στη συνέχεια κάνουμε *ssh* στην *private* IP του *slave*. Αν μπορούμε να συνδεθούμε επιτυχώς χωρίς προβλήματα, τότε τα *VMs* έχουν δημιουργηθεί σωστά και μπορούμε να προχωρήσουμε στην προετοιμασία των *VMs* για την εγκατάσταση Hadoop και *HBase*.

Virtual Box

Αν δεν θέλουμε να χρησιμοποιήσουμε τον Okeanos, μπορούμε να δημιουργήσουμε δύο εικονικά μηχανήματα τοπικά χρησιμοποιώντας το *VirtualBox*.

Βήμα 1ο

Κατεβάζουμε από το site του εργαστηρίου του Ε.Μ.Π. (`http://cslab.ece.ntua.gr/courses/atds/files/fall2013_14/debian.vdi`) το image των *VMs* που πρόκειται να δημιουργήσουμε.

Βήμα 2ο

Δημιουργούμε ένα νέο *VM* τύπου Linux Debian x64 (έστω *master* το όνομά του), ρυθμίζουμε *RAM* και αριθμό *cores* και στην επιλογή του δίσκου, χρησιμοποιούμε το *debian.vdi* (Use an existing virtual hard drive file).

Βήμα 3ο

Με τη βοήθεια του *Virtual Media Manager* κάνουμε *clone* το *image* που μόλις κατεβάσαμε σε ένα νέο *image* με όνομα *slave* τύπου *vdi* (δεξί κλικ στο δίσκο, copy και στον οδηγό χρησιμοποιούμε τύπο *vdi dynamically allocated* με όνομα *slave*).

Μόλις ολοκληρωθεί η διαδικασία, δημιουργούμε και το *VM slave* με την ιδια διαδικασία που ακολουθήσαμε για τον *master*.

Πριν εκκινήσουμε τα *VMs*, πρέπει να ρυθμίσουμε το δίκτυό τους έτσι ώστε να ανήκουν στο ίδιο υποδίκτυο και, κατ' επέκταση, να μπορούν να έχουν άμεση πρόσβαση. Κάνουμε δεξί κλικ στον *master*, επιλέγουμε *Settings*, επιλέγουμε την καρτέλα *Network* και στο πεδίο *Attached to* στον *Adapter 1*, βάζουμε την επιλογή *Bridged Adapter*. Επιλέγουμε *OK* και επαναλαμβάνουμε τη διαδικασία για τον *slave*.

Είμαστε έτοιμοι τώρα να εκκινήσουμε τις εικονικές μηχανές. Κάνοντας διπλό κλικ επάνω σε κάθε *VM* ή επιλέγοντας *Start*, θα δούμε τα μηχανήματα να εκκινούν. Όταν η εκκίνηση ολοκληρωθεί ζητείται *username* και *password*. Βάζουμε **root** για *username* και **password** για *password*.

Προετοιμασία

Απαραίτητη προϋπόθεση για την εγκατάσταση του Hadoop και της HBase είναι να μπορούν τα VMs να επικοινωνήσουν μεταξύ τους μέσω ssh χωρίς password.

Βήμα 1ο

Κατασκευή ζεύγους κλειδιών ssh με τον ακόλουθο τρόπο (στον master):

`sshkeygen` (δημιουργείται ένα ζεύγος κλειδιών σε και όλα τα prompts πατάμε Enter)

```
cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
chmod 700 /root/.ssh/
chmod 600 /root/id_rsa
```

Βήμα 2ο

Καλό θα είναι οι *IPs* των εικονικών μηχανών να είναι εγκατασταθούν στο `/etc/hosts` αρχείο κάθε εικονικής μηχανής, για να μην αναφερόμαστε στα *VM* με το *domain name* τους ή την *IP* τους. Για να πληροφορηθούμε για την *IP* κάθε μηχανής, αρκεί να δώσουμε στο τερματικό την εντολή

```
ipconfig
```

Αν, για παράδειγμα, ο *master* έχει *IP* 192.168.0.2 και ο *slave* έχει 192.168.0.3 τότε βάζουμε στο `/etc/hosts` αρχείο τις ακόλουθες γραμμές κώδικα (τις συμπληρώνουμε στο τέλος του αρχείου):

```
192.168.0.2 master
192.168.0.3 slave
```

Βήμα 3ο

Για να χρησιμοποιήσουμε τα ίδια *ssh κλειδιά* και στους δύο υπολογιστές (για λόγους ευκολίας), πρέπει να τα αντιγράψουμε και στον *slave*. Αυτό θα επιτευχθεί με τις ακόλουθες εντολές:

```
cd /root/
```

(αν δεν είμαστε ήδη)

```
scp r .ssh slave:/root/
```

(θα μας ζητήσει password)

```
ssh slave
```

(αν όλα πήγαν καλά, θα πρέπει να συνδεθούμε χωρίς password)

Βήμα 4ο

Τέλος, ορίζουμε τα *hostnames* των *VMs*, για να γνωρίζουμε ανά πάσα χρονική στιγμή σε ποιο *VM* βρισκόμαστε. Δίνουμε την εντολή:

`hostname master` (στον master)

`hostname slave` (στον slave)

Αξίζει να σημειώσουμε ότι η δεύτερη βασική προϋπόθεση για την εγκατάσταση των Hadoop και *HBase* είναι η εγκατάσταση *JVM* στους κόμβους. Το *JVM* είναι προεγκατεστημένο στο *Image* που δόθηκε και για αυτό αγνοούμε το κομμάτι της εγκατάστασης του *JVM*.

Εγκατάσταση Hadoop

Για την εγκατάσταση ενός Hadoop *cluster*, τα παρακάτω βήματα θα πρέπει να εκτελεστούν σε όλους τους υπολογιστές του *cluster*. Για το παράδειγμα αυτό της εγκατάστασης, θα θεωρήσουμε το εξής *setup*:

Βήμα 1ο

Έχουμε ένα *cluster* δύο κόμβων, όπου ο ένας κόμβος θα χρησιμοποιείται σαν *datanode* και *tasktracker*, ενώ ο άλλος κόμβος σαν *namenode*, *jobtracker* και συγχρόνως και *datanode* και *tasktracker*. Επιλέγουμε το *installation package* από ένα *repository* (`http://www.apache.org/dyn/closer.cgi/hadoop/common/`) και το κατεβάζουμε στον κατάλογο `/opt`.

```
# wget
http://apache.mirrors.tds.net/hadoop/common/hadoop1.2.1
/hadoop1.2.1.tar.gz
# cd /opt
# tar xvzf hadoop1.2.1.tar.gz
# cd hadoop1.2.1
```

Βήμα 2ο

Στη συνέχεια πρέπει να βάλουμε τις κατάλληλες ρυθμίσεις στα *configuration files* του Hadoop. Το Hadoop ρυθμίζεται από τρία βασικά xml αρχεία τα οποία βρίσκονται στον κατάλογο *conf* (`coresite.xml`, `hdfssite.xml` και `mapredsite.xml`) και από τα αρχεία *masters* και *slaves*.

Το αρχείο `coresite.xml` περιέχει την διεύθυνση του κεντρικού *master* του Hadoop.

Το αρχείο `hdfssite.xml` περιέχει ρυθμίσεις για το κατανεμημένο *HDFS* σύστημα ενώ

το `mapred-site.xml` ρυθμίζει το περιβάλλον εκτέλεσης *MapReduce*.

Τα αρχεία `masters` και `slaves` περιέχουν τα *dns* των Hadoop *masters* και Hadoop *slaves*.

Περισότερες πληροφορίες για τα αρχεία υπάρχουν εδώ:

`http://hadoop.apache.org/docs/r1.2.1/cluster_setup.html#Configuring+the+Hadoop+Daemons`

Στον παραπάνω σύνδεσμο φαίνεται οτι οι παράμετροι που μπορούν να ρυθμιστούν είναι πάρα πολλές, δίνοντας μεγάλη ευελιξία στον τρόπο λειτουργίας του συστήματος. Σε αυτή την ενότητα θα δούμε τις βασικές ρυθμίσεις που πρέπει να γίνουν, ώστε να λειτουργήσει το Hadoop. Τα παρακάτω αρχεία είναι τα πιο βασικά αρχεία ρυθμίσεων του Hadoop.

hadoopenv.sh:

Ανοίγουμε το αρχείο `hadoopenv.sh` κι ενημερώνουμε την τιμή του `JAVA_HOME`:

```
# vim conf/hadoopenv.sh
export
JAVA_HOME=/usr/lib/jvm/java1.7.0openjdkamd64/
```

coresite.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
ref="configuration.xsl"?>
<!--Put site specific property overrides in this
file. -->
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/opt/tmp_dir</value>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
```

Hdfs-site.xml

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl"
href="configuration.xsl"?>
<!--Put site-specific property overrides in this file. -
->
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.block.size</name>
<value>67108864</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>/opt/hdfsnames</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/opt/hdfsdata</value>
</property>
<property>
<name>dfs.support.append</name>
<value>true</value>
</property>
</configuration>
```

mapredsite.xml

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl"
href="configuration.xsl"?>
<!--Put site-specific property overrides in this file. -
->
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>master:9001</value>
</property>
<property>
<name>mapred.map.child.java.opts</name>
<value>Xmx2g</
value>
</property>
<property>
<name>mapred.reduce.child.java.opts</name>
<value>-Xmx2g</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>/opt/tmp_dir</value>
</property>
<property>
<name>mapred.jobtracker.taskScheduler</name>
<value>org.apache.hadoop.mapred.FairScheduler</value>
</property>
</configuration>
```

Το αρχείο `masters` περιέχει το όνομα του *master*, και το αρχείο `slaves` τα ονόματα των *slaves* (το ένα κάτω από το άλλο) έτσι τα κάνουμε:

```
# echo “master” > conf/masters
# echo “slave” > conf/slaves;echo “master” > slaves
```

Εκκίνηση Hadoop

Πλέον είμαστε έτοιμοι να ξεκινήσουμε το Hadoop.

Βήμα 1ο

Συνδεόμαστε με *ssh* στον *master*:

```
$ ssh root@master
```

Βήμα 2ο

Μεταφερόμαστε στον κατάλογο της εγκατάστασης του Hadoop:

```
# cd /opt/hadoop1.2.1
```

Βήμα 3ο

Την πρώτη φορά που θα χρησιμοποιήσουμε το *cluster* μας, πρέπει να κάνουμε *format* τον *namenode* του *HDFS*:

```
# bin/hadoop namenode format
```

Βήμα 4ο

Ξεκινάμε το Hadoop και το HDFS:

```
# bin/startall.sh
```

Το *output* που θα δούμε μετά την τελευταία εντολή, θα πρέπει να μοιάζει κάπως έτσι:

```
starting namenode, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-namenode-
snf-252660.out
master: starting datanode, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-datanode-
snf-252660.out
slave: starting datanode, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-datanode-
snf-252661.out
master: starting secondarynamenode, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-
secondarynamenode-snf-252660.out
starting jobtracker, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-
jobtracker-snf-252660.out
slave: starting tasktracker, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-
tasktracker-snf-252661.out
master: starting tasktracker, logging to
/opt/hadoop1.2.1/libexec/../logs/hadoop-root-
tasktracker-snf-252660.out
```

Το σύστημα έχει ξεκινήσει, και στα παρακάτω url μπορεί κανείς να βλέπει την κατάσταση του *cluster*:

`http://master:50070`: Το url αυτό δείχνει την κατάσταση του *NameNode*, καθώς και τα περιεχόμενα του αποθηκευτικού συστήματος.

`http://master:50030`: Αυτό το url δείχνει την κατάσταση του *MapReduce cluster* μέσω του *jobtracker*.

Χρήσιμες εντολές και αρχεία

- Δίνοντας την εντολή jps σε ένα *terminal*

```
# jps
```

μπορούμε να δούμε τις διεργασίες *java* που τρέχουν. Η εντολή μας δίνει και το *pid* της διεργασίας, οπότε κάνοντας

```
kill 9 <pid>
```

σκοτώνουμε κάποια διεργασία.

- Ο κατάλογος `logs/` που βρίσκεται στον κατάλογο του `Hadoop installation` περιέχει τα *logfiles* όλων των υπηρεσιών του Hadoop. Είναι χρήσιμος καθώς εκεί μπορούμε να βλέπουμε πιθανά λάθη/προβλήματα που προκύπτουν.

- Στον κατάλογο `/tmp` υπάρχουν τα *lock files* των προγραμμάτων του Hadoop (με κατάληξη .pid). Καλό είναι, εάν έχουμε τερματίσει τον *Hadoop cluster* με βίαιο τρόπο (πχ με `kill 9 <pid>`), να τα σβήνουμε.

MapReduce στο Eclipse

Δημιουργία προγράμματος MapReduce

Παρακάτω θα δούμε πώς μπορούμε να δημιουργήσουμε από την αρχή και να

τρέξουμε ένα *MapReduce* πρόγραμμα. Θα παρουσιάσουμε τον κώδικα για το πρόβλημα του *WordCount*. Στο παράδειγμα αυτό, για τη δημιουργία του προγράμματος, θα χρησιμοποιήσουμε το *Eclipse IDE*.

Βήμα 1ο

Μεταφορτώνουμε και εξάγουμε το `hadoop installation` στο

`Home folder` μας, ακριβώς όπως κάναμε και στα *virtual machines* (εδώ δεν χρειάζεται να φτιάξουμε τα *conf files*).

Βήμα 2ο

Ανοίγουμε το Eclipse κι ακολουθούμε τη διαδικασία:

File > *New*.. > *Project*

και από τα προτεινόμενα επιλέγουμε *Java Project*.

Στο παράθυρο που θα ανοίξει, δίνουμε όνομα στο project και πατάμε *Finish*.

Βήμα 3ο

Μόλις δημιουργηθεί το project, κάνουμε δεξί κλικ πάνω στο όνομα του κι επιλέγουμε:

New.. > *Class*

Βήμα 4ο

Στο παράθυρο που θα ανοίξει, ονομάζουμε την κλάση μας *WordCount* και πατάμε *Finish*.

Βήμα 5ο

Ανοίγουμε το αρχείο της κλάσης κι αντιγράφουμε το παρακάτω τμήμα κώδικα:

```
import java.io.IOException;
import java.util.*;
```

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class WordCount {
public static class Map extends Mapper<LongWritable, Text, Text,
IntWritable> {
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
word.set(tokenizer.nextToken());
context.write(word, one);
}
}
}
public static class Reduce extends Reducer<Text, IntWritable, Text,
IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context
```

```
context)
throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
context.write(key, new IntWritable(sum));
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "wordcount");
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.waitForCompletion(true);
}
}
```

Βλέπουμε ότι το *Eclipse* μας δείχνει ότι υπάρχει λάθος στα *import* μας, καθώς δεν αναγνωρίζει τις βιβλιοθήκες του Hadoop. Για να ξεπεράσουμε αυτό το πρόβλημα, πρέπει να ρυθμίσουμε σωστά το *build path* του project, ώστε να επιλυθούν οι εξαρτήσεις:

Βήμα 1ο

Κάνουμε δεξί κλικ πάνω στο όνομα του project κι επιλέγουμε:

Build Path > *Configure Build Path*

Βήμα 2ο

Στο tab με τα *Libraries*, επιλέγουμε το *Add external JARs*.

Βήμα 3ο

Στο παράθυρο που ανοίγει μεταφερόμαστε στον κατάλογο που έχουμε κάνει extract το *hadoop installation* και επιλέγουμε το `hadoopcore1.2.1.jar` και πατάμε *OK*.

Για να μεταφέρουμε την εφαρμογή μας στο *Hadoop cluster* και να την τρέξουμε, πρέπει πρώτα να την μετατρέψουμε σε μορφή *jar*.

Βήμα 1ο

Δεξί κλικ στο όνομα του *project* > *Export as*.. > *Java* > *JAR*

Ονομάζουμε το jar `wordcount.jar` και το αποθηκεύουμε στο `Home folder` μας.

Βήμα 2ο

Μεταφέρουμε την εφαρμογή μας στο *Hadoop cluster* και την τρέχουμε. Αυτό θα γίνει με τη βοήθεια του *scp utility*.

```
$ scp ~/wordcount.jar root@master:~/
```

Βήμα 3ο

Τέλος, συνδεόμαστε στον master και τρέχουμε την εφαρμογή μας.

```
$ ssh root@master
# cd /opt/hadoop1.2.1/
bin
# ./hadoop jar ~/wordcount.jar
```

Εγκατάσταση HBase v0.94.16

Μπορούμε να το βρούμε πληροφορίες και tutorial απο τα site:
`http://hbase.apache.org/book/quickstart.html`
`http://hbase.apache.org/book/book.html`
Για να κατεβάσουμε το *HBase* εκτελούμε τον κώδικα:

```
wget
http://mirrors.gigenet.com/apache/hbase/hbase-
0.94.16/hbase-0.94.16.tar.gz
tar xvzf hbase-0.94.16.tar.gz
cd hbase-0.94.16
```

Σύνδεση Hadoop με HBase

Βήμα 1ο

Αντιγραφή του `hadoop-core.jar` στο *lib folder* της *HBase*:

```
cp                  /opt/hadoop1.2.1/hadoopcore1.2.1.jar
/opt/hbase0.94.16/lib/
rm /opt/hbase0.94.16/lib/hadoopcore1.0.4.jar
```

Βήμα 2ο

Αντιγραφή του `HBase.jar` στο *lib folder* του *Hadoop*:

```
cp                  /opt/hbase0.94.16/hbase0.94.16.jar
/opt/hadoop1.2.1/lib/
```

Βήμα 3ο

Αντιγραφή του `hdfssite.xml` στο *conf folder* της *HBase*:

```
cp                   /opt/hadoop1.2.1/conf/hdfssite.xml
/opt/hbase0.94.16/conf/
```

Βήμα 4ο

Αντιγραφή του `hbasesite.xml` στο *conf folder* του Hadoop:

```
cp /opt/hbase0.94.16/conf/hbasesite.xml
/opt/hadoop1.2.1/conf/
```

Διαμόρφωση HBase

```
Conf folder: /opt/hbase0.94.16/conf/
```

regionservers:

Περιέχει ανά γραμμή τα *dns* ονόματα των *regionservers*. Στην περίπτωσή μας:

```
master
slave
```

hbasesite.xml:

```
<configuration>
<property>
<name>hbase.cluster.distributed</name>
<value>true</value>
</property>
<property>
<name>hbase.rootdir</name>
<value>hdfs://master:9000/hbase</value>
</property>
<property>
<name>hbase.zookeeper.quorum</name>
<value>master</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/opt/zookeeper</value>
</property>
<property>
<name>dfs.support.append</name>
<value>true</value>
</property>
</configuration>
```

hbaseenv.sh:

Uncomment:

```
1. export JAVA_HOME=/usr/lib/jvm/java7openjdkamd64
```

Ορίζει το *folder* στο οποίο βρίσκεται το *jdk* που θα χρησιμοποιήσει η *Hbase*.

```
2. export HBASE_MANAGES_ZK=true
```

Δηλώνει ότι η *HBase* θα σηκώσει από μόνη της *zookeeper* και δεν θα συνδεθεί σε ένα υπάρχον *zookeeper quorum*.

```
3. export HBASE_HEAPSIZE=1000
```

Εδώ ορίζουμε το *heap size* των *regionserver* (default 1GB).

Αντιγραφή του *HBase folder* σε όλους τους κόμβους της συστοιχίας

```
scp –r hbase-0.94.16 master:/opt/
scp –r hbase-0.94.16 slave:/opt/
```

Παραδείγματα

Έναρξη HBase:

`bin/starthbase.sh`

Αν έχουν όλα σηκωθεί σωστά θα πρέπει το *jps* στον *master* να μας δείχνει τις διεργασίες:

```
HQuorumPeer
HRegionServer
HMaster
```

Στον *slave* πρέπει να υπάρχει μόνο η διεργασία:

```
HquorumPeer
```

Επίσης μπορούμε να δούμε το *Web Interface* της *HBase* από το:

`http://master:60010/masterstatus`

Τερματισμός HBase:

`bin/stophbase.sh`

Παράδειγμα Shell:

```
root@snf252660:/opt/hbase-0.94.16#bin/hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.16, r1557241, Fri Jan 10 20:43:03 UTC 2014
hbase(main):002:0> create 'test', 'cf'
0 row(s) in 3.3060 seconds
hbase(main):003:0> list
TABLE
test
1 row(s) in 0.1350 seconds
hbase(main):004:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.3850 seconds
hbase(main):005:0> put 'test', 'row2', 'cf:a', 'value2'
0 row(s) in 0.0080 seconds
hbase(main):006:0> scan 'test'
ROW COLUMN+CELL
```

```
row1 column=cf:a, timestamp=1390917599242, value=value1
row2 column=cf:a, timestamp=1390917607120, value=value2
2 row(s) in 0.1640 seconds
hbase(main):007:0> disable 'test'
0 row(s) in 1.5450 seconds
hbase(main):008:0> drop 'test'
0 row(s) in 1.3030 seconds
hbase(main):009:0> list
TABLE
0 row(s) in 0.0530 seconds
hbase(main):009:0> exit
```

Παράδειγμα Java:

```
import java.io.IOException;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Get;
```

```
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;
// Does a Put, Get and a Scan against an hbase table.
public class MyLittleHBaseClient {
public static void main(String[] args) throws IOException {
/* You need a configuration object to tell the client where to
connect. When you create a HBaseConfiguration, it reads in whatever
you've set into your hbase-site.xml and in hbase-default.xml, as long
as these can be found on the CLASSPATH */
```

```
Configuration config = HBaseConfiguration.create();
/* This instantiates an HTable object that connects you to
the "myLittleHBaseTable" table.*/
HTable table = new HTable(config, "myLittleHBaseTable");
```

```
/* To add to a row, use Put. A Put constructor takes the name of the row you want to insert into as a byte array. In HBase, the Bytes class has utility for converting all kinds of java types to byte arrays. In the below, we are converting the String "myLittleRow" into a byte array to use as a row key for our update. Once you have a Put instance, you can adorn it by setting the names of columns you want to update on the row, the timestamp to use in your update, etc.If no timestamp, the server applies current time to the edits. */
Put p = new Put(Bytes.toBytes("myLittleRow"));
```

```
/* To set the value you'd like to update in the row 'myLittleRow',specify the column family, column qualifier, and value of the table cell you'd like to update. The column family must already exist in your table schema. The qualifier can be anything. All must be specified as byte arrays as hbase is all about byte arrays. Lets pretend the table 'myLittleHBaseTable' was created with a family 'myLittleFamily'. */
p.add(Bytes.toBytes("myLittleFamily"),
Bytes.toBytes("someQualifier"),
Bytes.toBytes("Some Value"));
/* Once you've adorned your Put instance with all the updates you want to make, to commit it do the following (The HTable#put method takes the Put instance you've been building and pushes the changes you made into hbase)*/
table.put(p);
/* Now, to retrieve the data we just wrote. The values that come back are Result instances. Generally, a Result is an object that will package up the hbase return into the form you find most palatable.*/
```

```
Get g = new Get(Bytes.toBytes("myLittleRow"));
Result r = table.get(g);
byte [] value = r.getValue(Bytes.toBytes("myLittleFamily"),
Bytes.toBytes("someQualifier"));
/* If we convert the value bytes, we should get back 'Some Value', the value we inserted at this location.*/
String valueStr = Bytes.toString(value);
System.out.println("GET: " + valueStr);
```

```
/* Sometimes, you won't know the row you're looking for. In this
case,you use a Scanner. This will give you cursorlike
interface to the contents of the table. To set up a Scanner, do like
you did above making a Put and a Get, create a Scan. Adorn it with
column names, etc.*/
Scan s = new Scan();
```

```
s.addColumn(Bytes.toBytes("myLittleFamily"),
Bytes.toBytes("someQualifier"));
ResultScanner scanner = table.getScanner(s);
try {
/* Scanners return Result instances. Now, for the actual iteration.
One way is to use a while loop like so:*/
for (Result rr = scanner.next(); rr != null; rr = scanner.next()) {
/* print out the row we found and the columns we were looking for */
System.out.println("Found row: " + rr);
}
/* The other approach is to use a foreach loop. Scanners are iterable!
for (Result rr : scanner) {
System.out.println("Found row: " + rr); */
}
} finally {
/* Make sure you close your scanners when you are done!
Thats why we have it inside a try/finally clause*/
scanner.close();
}
}
}
/* example available at here*/
```

Το παραπάνω παράδειγμα κάνει ένα `Put` ένα `Get` και ένα `Scan` σε έναν πίνακα της *HBase*. Μπορούμε να φτιάξουμε το αντίστοιχο *jar* χρησιμοποιώντας τον *Eclipse*, όπως είδαμε παραπάνω. Το *jar* πρέπει να μεταφερθεί στον master με χρήση του *scp* και μπορούμε να το εκτελέσουμε με την εντολή `bin/hadoop jar` από το *installation folder* του Hadoop.

3.8 Υλοποίηση του Hadoop στο Hortonworks Sandbox

3.8.1 Περιγραφή παραδείγματος

Στο συγκεκριμένο παράδειγμα θα δούμε πως μια πλατφόρμα ανοιχτού κώδικα, το Hortonworks Sandbox, μπορεί να εκμεταλλευθεί τις δυνατότητες του Hadoop για να επεξεργαστεί δεδομένα μεγάλου όγκου.

Χρησιμοποιείται απο εταιρείες όπως Accenture, Cisco, και Microsoft για τα επιχειρησιακά τους δεδομένα, και είναι ένα πρωτοπωριακό εργαλείο σε θέματα υπολογιστικής νέφους.

3.8.2 Πρακτική προσέγγιση

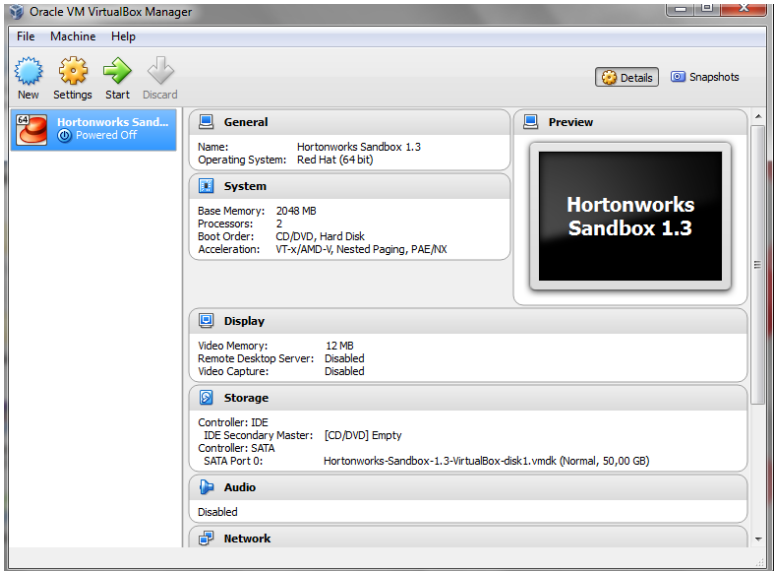
Το κατανεμημένο σύστημα αρχείων του Hadoop

Παρακάτω θα παρουσιάσουμε κάποια κύρια χαρακτηριστικά του κατανεμημένου συστήματος αρχείων του Hadoop (HDFS). Αυτό το σύστημα αρχείων διαχειρίζεται τα δεδομένα που αποθηκεύονται στους κόμβους της συστοιχίας. Είναι υπεύθυνο για το διαμοιρασμό των δεδομένων, την αντιγραφή τους και για λειτουργίες διαχειριστή όπως πρόσθεση, διαγραφή και επανάκτηση των κόμβων.

Εγκατάσταση

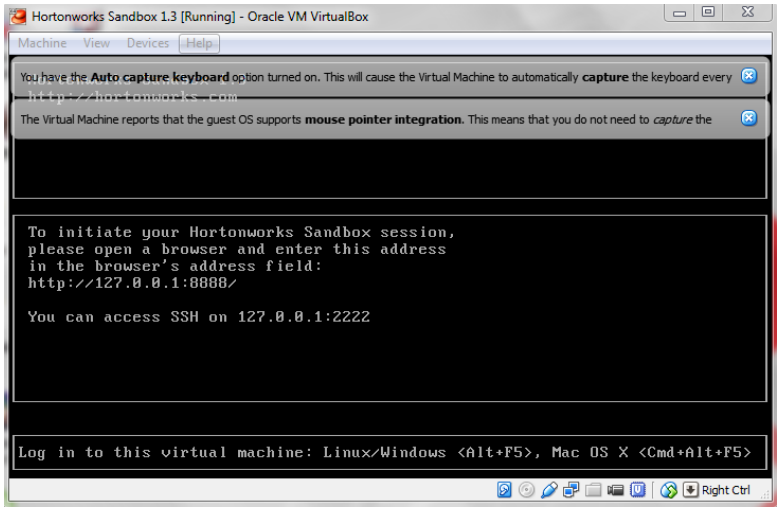
Αρχικά θα κατεβάσουμε απο το site του Hortonworks το VirtualBox VMs, που περιέχει το Oracle VM VirtualBox και το Hortonworks Sandbox.

Αφού εγκαταστήσουμε το Oracle VM VirtualBox, το ανοίγουμε και μας δείχνει την εικόνα:



Εικόνα 3.8.2.1: Αρχική οθόνη VirtualBox

Αν επιλέξουμε *Start* θα ξεκινήσει η διαδικασία δημιουργίας σύνδεσης και θα μας δείξει το παράθυρο:

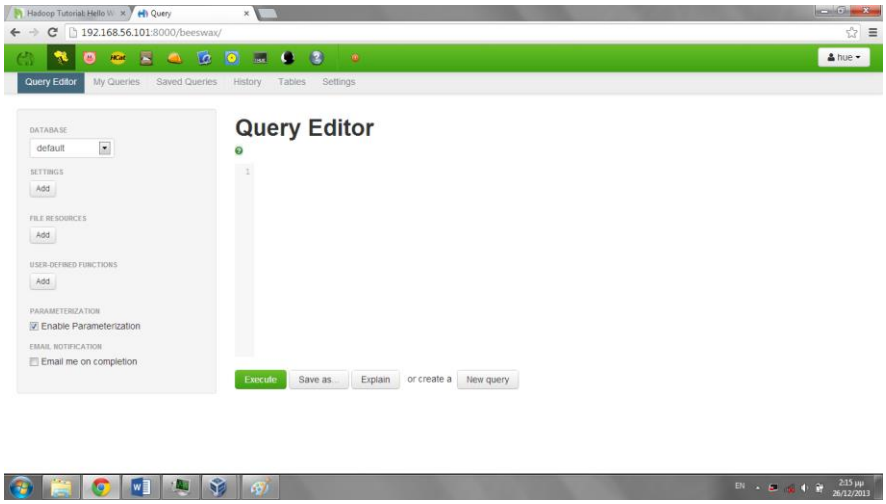


Εικόνα 3.8.2.2: Η διεύθυνση IP στην οποία θα συνδεθούμε

Το οποίο μας δείχνει σε ποια διεύθυνση IP θα συνδεθούμε.

Apache Hive

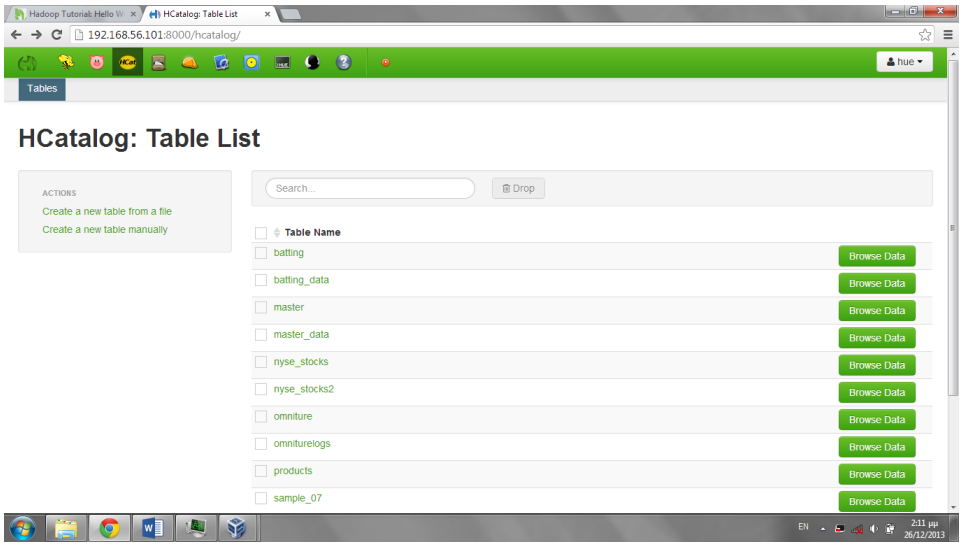
Αυτό το project παρέχει στο χρήστη τη δυνατότητα να δει τα δεδομένα. Χρησιμοποιεί μια γλώσσα SQL, δίνει τη δυνατότητα στο χρήστη να κάνει ανάλυση των δεδομένων και να θέτει ερωτήματα. Η γενική προσέγγιση που παρέχει το Hive είναι να παρέχει μια δομή πίνακα για τα δεδομένα, και να διαχειρίζονται με μια γλώσσα SQL.



Εικόνα 3.8.2.3: Ο editor της Hive

Apache HCatalog

Η λειτουργία του HCatalog είναι να κρατά την τοποθεσία και να δίνει πληροφορίες για τα δεδομένα που υπάρχουν στη συστοιχία. Αυτό δίνει τη δυνατότητα στις διεργασίες να διαχωρίζονται απο την τοποθεσία των δεδομένων και τις πληροφορίες που τα συνοδεύουν. Επιπλέον υποστηρίζει εργαλεία όπως τα Hive και Pig.



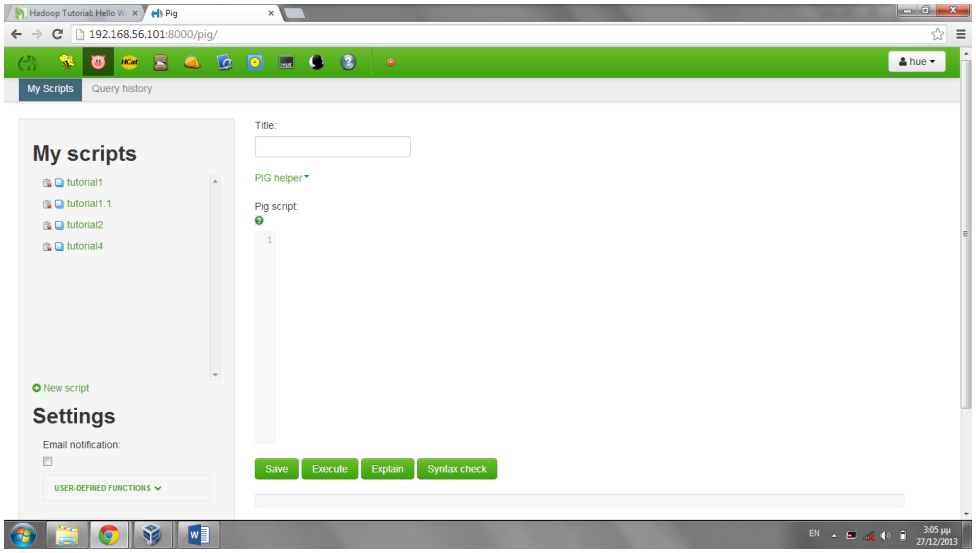
Εικόνα 3.8.2.4: Ο Hcatalog έχει όλους τους πίνακες

Apache Pig

Η Pig είναι μια γλώσσα που χρησιμοποιείται για την ανάλυση των δεδομένων και τις εσωτερικές διαδικασίες. Μεταφράζεται σε μια σειρά διεργασιών του MapReduce που τρέχουν σε μια συστοιχία του Hadoop. Μπορεί να γραφεί και σε Java. Τα script του Pig είναι γλώσσα υψηλού επιπέδου που δημιουργούν διεργασίες

MapReduce για την εκτέλεση διαδικασιών στους κόμβους. Μπορεί να χρησιμοποιηθεί σαν συστατικό για τη δημιουργία εφαρμογών που διαχειρίζονται πραγματικά επιχειρησιακά θέματα.

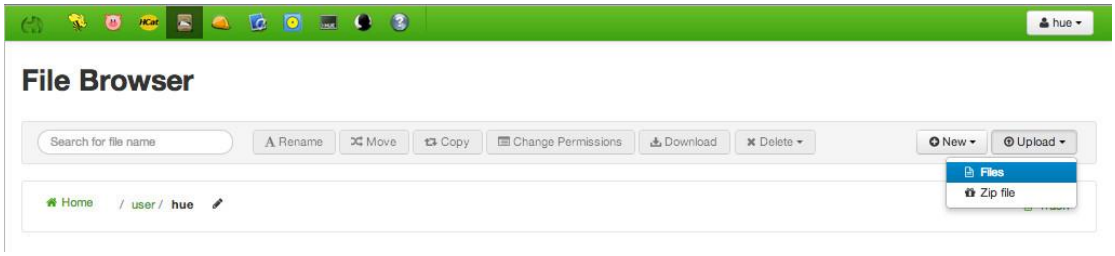
Ένα παράδειγμα εφαρμογής της Pig είναι το μοντέλο ETL(Extract-Transform-Load), το οποίο περιγράφει πως μια διαδικασία μπορεί να αποσπάσει δεδομένα απο κάποια πηγή, να τα αλλάξει σύμφωνα με ένα σύνολο κανόνων, και να τα αποθηκεύσει σε μια αποθήκη δεδομένων. Μπορεί να πάρει δεδομένα απο αρχεία ή άλλες πηγές με τις User Defined Functions(UDF). Αφού διαβάσει τα δεδομένα μπορεί να τελέσει συναρτήσεις όπως select, και να τα αποθηκεύσει με το σύστημα αρχείων του Hadoop(HDFS).



Εικόνα 3.8.2.5: Η Pig δίνει τη δυνατότητα να γράψουμε κώδικα για να διαχειριστούμε δεδομένα

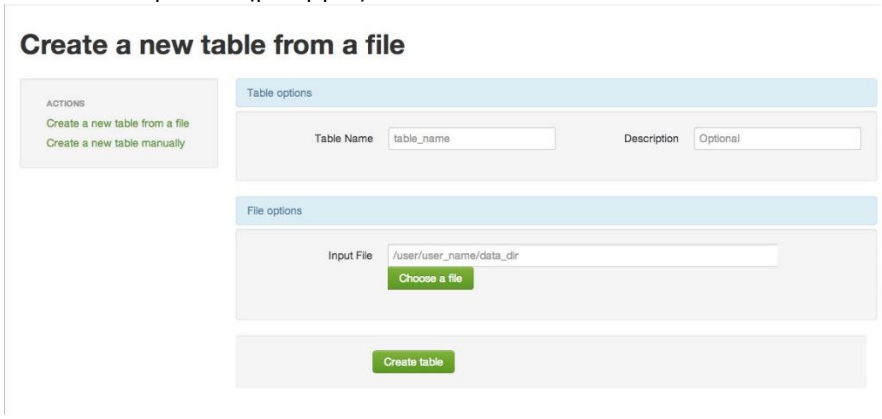
File Browser

Ο File Browser είναι εύκολος στη χρήση του και επιτρέπει την διαχείρηση των αρχείων,ενώ υπάρχει η δυνατότητα για ανέβασμα αρχείων για διαχείρηση της πληροφορίας που έχουν:



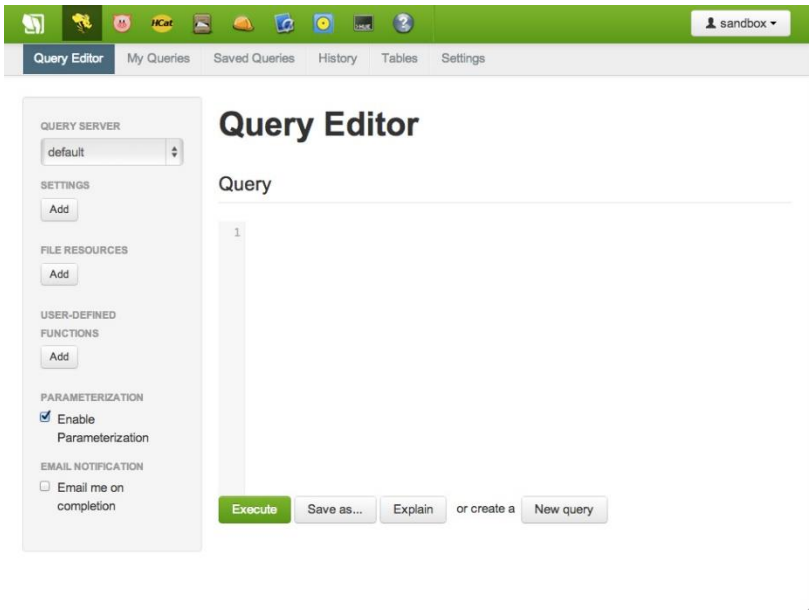
Εικόνα 3.8.2.6: Φορτώνουμε αρχεία

Αυτό είναι το μενού δημιουργίας πίνακα:



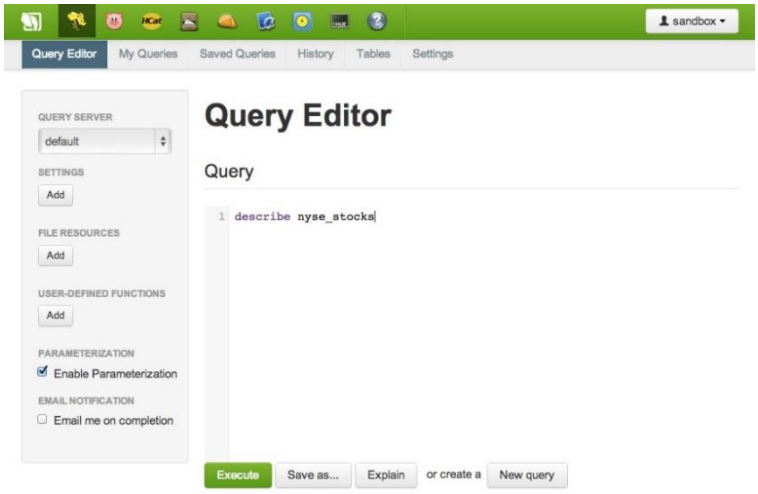
Εικόνα 3.8.2.7: Δημιουργία πίνακα

Με τον Query editor, την επιλογή Beeswax μπορούμε να θέσουμε SQL ερωτήματα:



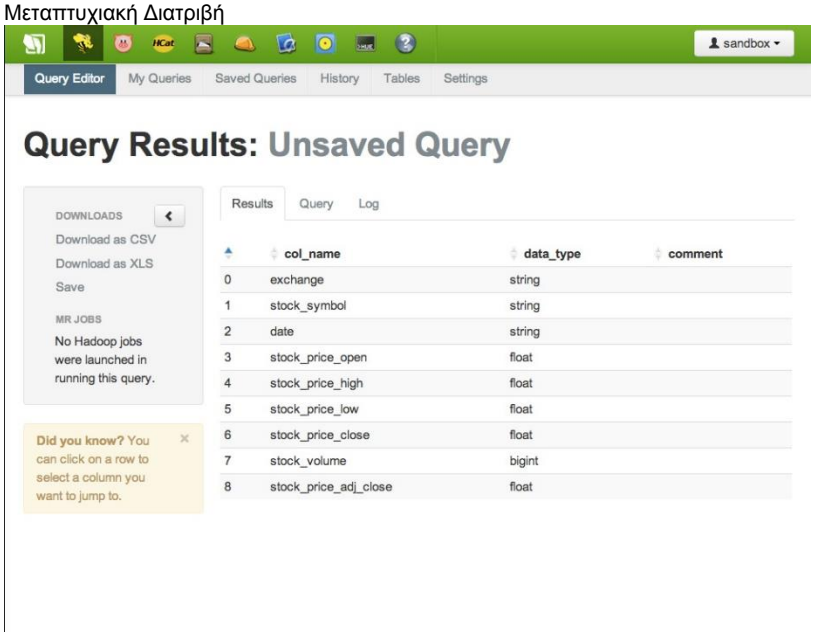
Εικόνα 3.8.2.8: Διαχείρηση αρχείων με γλώσσα SQL

Για παράδειγμα άν φορτώσουμε το κατάλληλο αρχείο nyse_stocks, με την εντολή describe nyse_stocks:



Εικόνα 3.8.2.9: Φορτώνει το αρχείο nyse_stocks

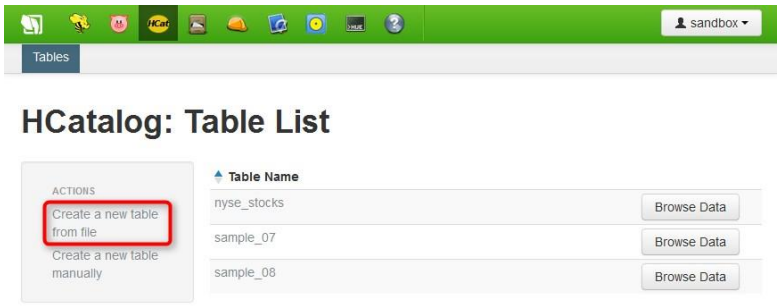
Παίρνουμε το αποτέλεσμα:



Εικόνα 3.8.2.10: Οι στήλες είναι συναρτήσεις του nyse_stocks

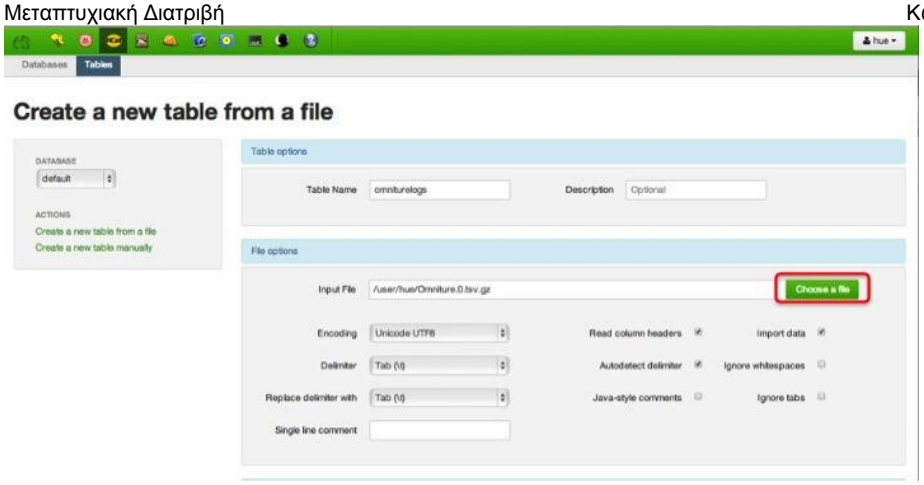
Ένα άλλο εκτενέστερο παράδειγμα:

Πηγαίνουμε στο Hcat, και επιλέγουμε create a new table from a file:



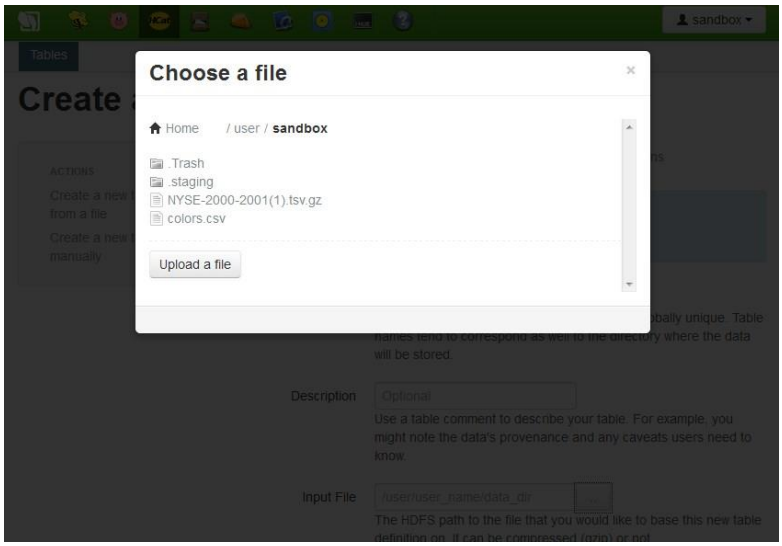
Εικόνα 3.8.2.11: Επιλογή υπάρχοντος πίνακα

Στη συνέχεια επιλέγουμε choose a file:



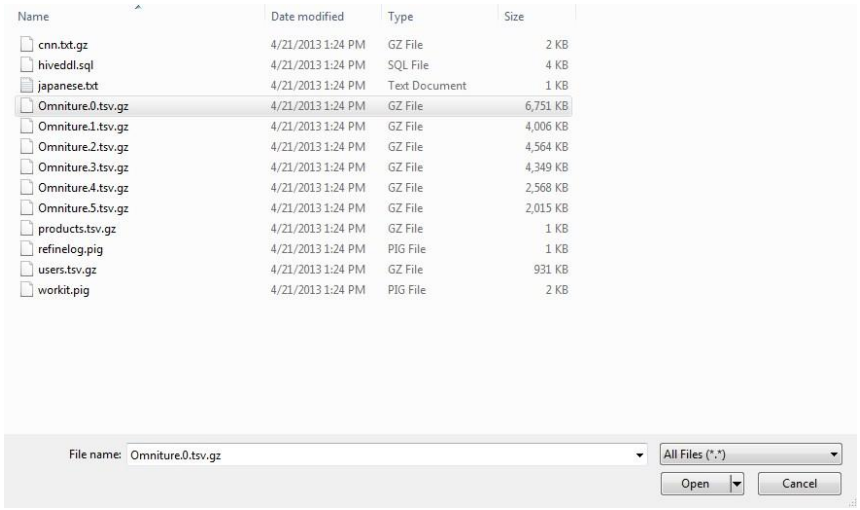
Εικόνα 3.8.2.12: Επιλογή συγκεκριμένου αρχείου

Στη συνέχεια ανεβάζουμε το αρχείο που θέλουμε με την επιλογή upload a file:



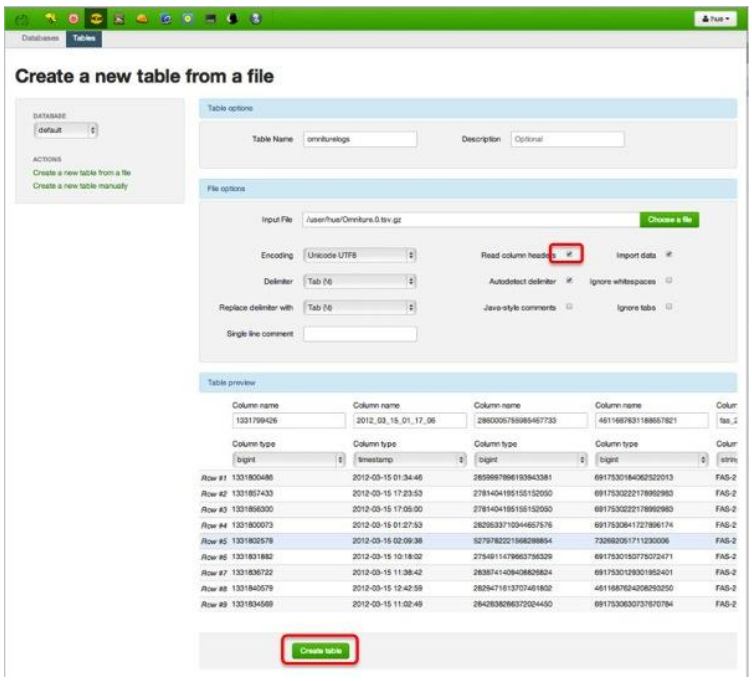
Εικόνα 3.8.2.13: Παράθυρο που ανεβάζουμε ένα αρχείο

Επιλέγουμε το αρχείο μας:



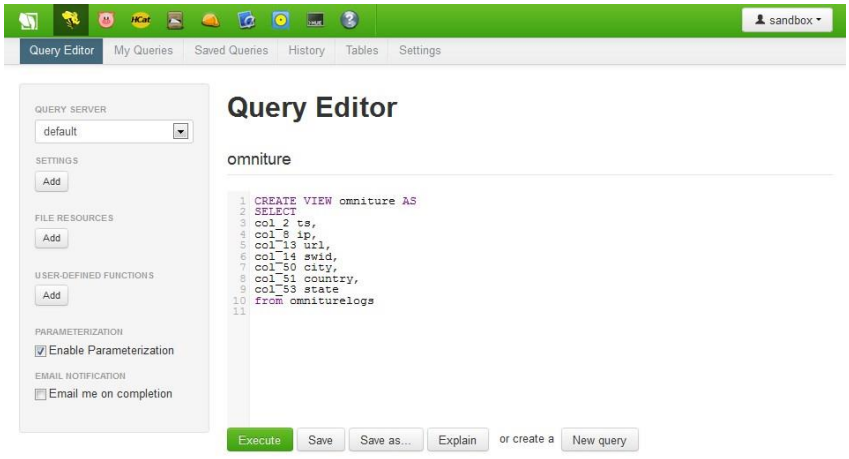
Εικόνα 3.8.2.14: Επιλογή υπάρχοντος πίνακα

Και τελικά δημιουργούμε τον πίνακα:



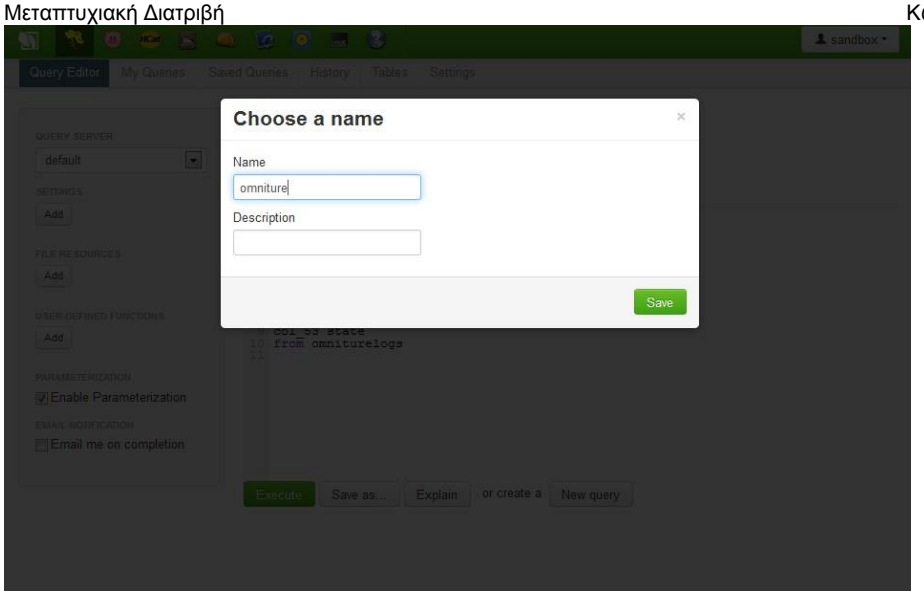
Εικόνα 4.8.2.15: Βλέπουμε τα στοιχεία και τις τιμές τους

Μπορούμε να δημιουργήσουμε ένα omniture view του πίνακα:



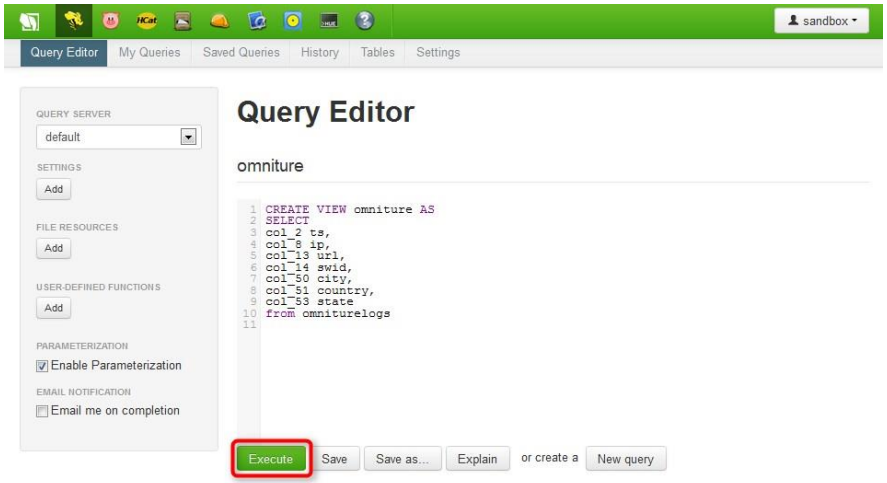
Εικόνα 4.8.2.16: Βλέπουμε τον κώδικα για τη δημιουργία πίνακα

Επιλέγουμε ένα όνομα:



Εικόνα 4.8.2.17: Εισαγωγή ονόματος στον πίνακα

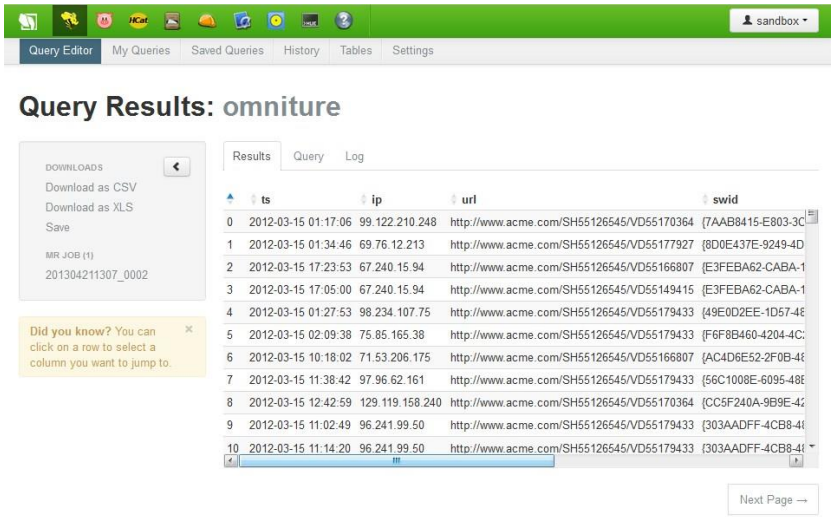
Στη συνέχεια τρέχουμε το script:



Εικόνα 4.8.2.18: Εκτέλεση του κώδικα

Μπορούμε να δούμε τα δεδομένα:

Εικόνα 4.8.2.19: Μπορούμε να δούμε τα αποτελέσματα



Εικόνα 4.8.2.20: Τα στατιστικά στοιχεία των αποτελεσμάτων

3.9 Συμπεράσματα

Από όσα είδαμε παραπάνω, το Hadoop είναι ένα πρωτοποριακό εργαλείο. Μπορεί να εγκατασταθεί σε διάφορα περιβάλλοντα (Linux, Windows), με διαφορετικούς τρόπους (Command Line ή IDE) και προσφέρει εύκολη και γρήγορη επεξεργασία δεδομένων. Υπάρχουν υλοποιήσεις οι οποίες προσφέρουν επεξεργασία και πρόσβαση δεδομένων από απόσταση, κάτι που διευκολύνει την αντιμετώπιση προβλημάτων στις σύνθετες και συνεχώς αυξανόμενες απαιτήσεις της σύγχρονης εποχής.

4. ΑΝΑΦΟΡΕΣ

[1] Ιστότοπος εταιρείας λογισμικού Cloudera (2008). *Πηγές*. Διαθέσιμο: http://www.cloudera.com/content/dam/cloudera/Resources/PDF/Olson_IQT_Quarterly_Spring_2010.pdf. Προσπελάστηκε: 18 Νοε. 2013.

[2] Εργαστήριο Διαχείρησης Δεδομένων (2012/13). *Διδακτικό Υλικό*. Διαθέσιμο: http://qid3.mmci.uni-saarland.de/teaching/ss13/ddm/slides/5NoSQL_1on1.pdf. Προσπελάστηκε: 9 Φεβ. 2014

[3] Campus wiki. *Confluence*. Διαθέσιμο: https://wiki.rice.edu/confluence/download/attachments/4435861/comp322-s13-lab12.pdf?version=2&modificationDate=1366151320473&api=v2. Προσπελάστηκε: 15 Νοε. 2013

[4] Εργ. Τμήματος Πληροφορικής (2009). *FAQs*. Διαθέσιμο: http://www.cs.tau.ac.il/system/faq/services/files/files-1/Hadoop.ppt. Προσπελάστηκε: 23 Φεβ. 2014

[5] Engineering wiki, University of Illinois (2011). *Dashboard*. Διαθέσιμο: https://wiki.engr.illinois.edu/download/attachments/202932615/Lecture+4,+Hadoop+lab.pptx?version=1&modificationDate=1343123551000. Προσπελάστηκε: 1 Μαϊ. 2014

[6] Ιστότοπος μαθήματος Προχωρημένα Θέματα Βάσεων Δεδομένων (2005-2006). *Διαλέξεις*. Διαθέσιμο: http://www.cslab.ntua.gr/courses/atds/files/fall2013_14/odigos_egkatastasis_hadoop_hbase.pdf. Προσπελάστηκε: 30 Ιαν. 2014

[7] Ιστότοπος Σχολής Επιστήμης Υπολογιστών Andrei Clubisco (2011). *Μαθήματα*. Διαθέσιμο: http://andrei.clubcisco.ro/cursuri/f/f-sym/5master/aac-cc/CC%20Hadoop%20Lab.pdf. Προσπελάστηκε: 5 Φεβ. 2014

[8] Εργ. Μαθημάτων Πανεπιστημίου Δικτύων και Πληροφοριακών Συστημάτων Επιστήμης και Τεχνολογίας, Πανεπιστήμιο του Πεκίνου (2002). *Μαθήματα*. Διαθέσιμο: http://net.pku.edu.cn/~course/cs402/2010/codelab/Codelab1.pdf. Προσπελάστηκε: 5 Φεβ. 2014

[9] Εργ. Big Data University. *Courses*. Διαθέσιμο: http://db2university.db2oncampus.com/BD001EN/Transcripts/Lab1_HDFS_Instructions.pdf. Προσπελάστηκε: 6 Φεβ 2014

[10] Encycl. Wikipedia. *Hadoop, MapReduce, HDFS*. Διαθέσιμο: http://en.wikipedia.org/. Προσπελάστηκε: 18 Νοε. 2013.

[11] Forum grokbase (2013). *Hadoop Services.* Διαθέσιμο: http://grokbase.com/t/cloudera/scm-users/1387rtcfp9/cant-start-cloudera-manager-or-hadoop-services-after-hard-restart. Προσπελάστηκε: 27 Δεκ. 2013

[12] Ιστότοπος oreillynet. *Other programming*. Διαθέσιμο: http://www.oreillynet.com/pub/a/other-programming/excerpts/hadoop-tdg/installing-apache-hadoop.html. Προσπελάστηκε: 9 Μαρ. 2014

[13] Forum Stackoverflow. *Questions*. Διαθέσιμο: http://stackoverflow.com/. Προσπελάστηκε: 15 Μαρ. 2014

[14] Ιστότοπος Cygwin (2000). Διαθέσιμο: http://cygwin.com/. Προσπελάστηκε: 26 Νοε. 2013

[15] Ιστότοπος Apache Hbase (2010). Διαθέσιμο: http://hbase.apache.org/. Προσπελάστηκε: 30 Ιαν. 2014

[16] Εργ. Okeanos (2011). *About*. Διαθέσιμο: https://okeanos.grnet.gr/about/what/. Προσπελάστηκε: 30 Ιαν. 2014

[17]Ιστοχώρος Apache. Hadoop2OnWindows. Διαθέσιμο: https://wiki.apache.org/hadoop/Hadoop2OnWindows. Προσπελάστηκε: 24 Ιουν. 2014