EPL448: Data Mining on the Web – Lab 1

University of Cyprus
Department of
Computer Science

Παύλος Αντωνίου

Γραφείο: Β109, ΘΕΕ01

Αξιολόγηση και Κανόνες

- Ατομικές εργασίες: **5%**
  - 2-3 μικρές προγραμματιστικές ασκήσεις που θα βοηθούν στην εμπέδωση της γνώσης που αποκτάται στα εργαστήρια – δε θα αξιολογούνται ως προς την ορθότητά τους
    - θα υποβάλλονται στο Moodle
- Ομαδική (2-3 φοιτητών) Εργασία Εξαμήνου: **25%**
  - Περισσότερες πληροφορίες στο επόμενο εργαστήριο

Lab material

- Introduction to Apache Hadoop for Big Data storage and processing using the Map Reduce Programming model

- Data Manipulation and Visualization
- Data Preparation I: Cleaning, encoding, re-scaling, re-sampling
- Data Preparation II: Feature selection and extraction
- Machine Leaning: Regression and Classification
- Apache Spark for Big Data processing

Εισαγωγή - Map Reduce

- **MapReduce**
  - προγραμματιστικό μοντέλο εμπνευσμένο από τις συναρτήσεις map() και reduce() που χρησιμοποιούνται στον συναρτησιακό προγραμματισμό
  - επεξεργασία μεγάλων συνόλων δεδομένων (large datasets)
  - δεδομένα σε συστοιχίες (clusters) που αποτελούνται από χιλιάδες κόμβους
  - παράλληλη λειτουργία
- Πατενταρισμένο [2004] από την Google.

Υλοποιήσεις του Map Reduce

- **Google Map Reduce**
  - Χρησιμοποιήθηκε από την Google για την ανάκτηση δεδομένων από το Google File System μέχρι το 2014
  - Δεν υπάρχουν πολλές λεπτομέρειες της υλοποίησης, μάλλον σε C++
- Το **Hadoop** είναι η open-source υλοποίηση του **MapReduce**.

  - Υλοποίηση σε Java
  - Η υλοποίηση αυτή βασίστηκε κυρίως σε δύο άρθρα που δημοσίευσαν οι εμπνευστές του **MapReduce** και περιέγραφαν το μοντέλο.
  - Δημιουργία της Apache Foundation με οικονομική βοήθεια κυρίως από την Yahoo!

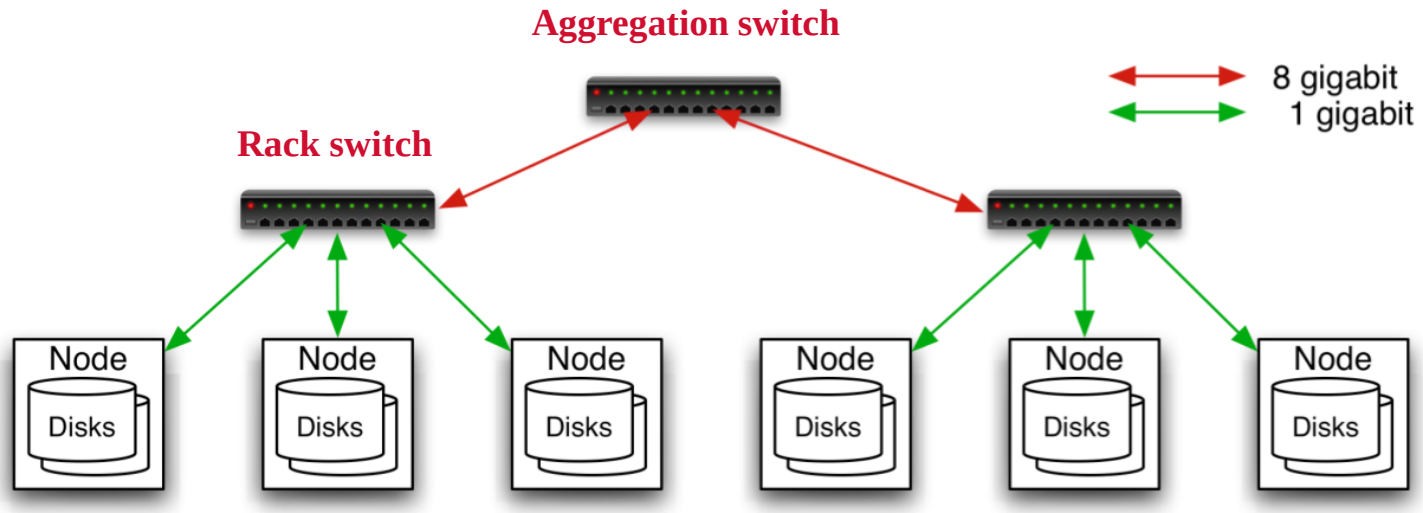
Εισαγωγή - Hadoop

- Το Hadoop επιτρέπει την επεξεργασία και ανάλυση τεράστιου όγκου δεδομένων σε κλίμακα **Petabyte (1 PB = 1024 TB)**.
- Ο σκοπός της ανάλυσης είναι η εξόρυξη χρήσιμών πληροφοριών και τάσεων σε *μη-πραγματικό χρόνο*
- Μεγάλα Hadoop clusters κατέχουν οι LinkedIn (~10000 nodes), Yahoo (4500 nodes*), Facebook (1400 nodes*)
- Χρησιμοποιείται επίσης και από άλλους οργανισμούς όπως eBay (532 nodes*), AOL (150 nodes*), Alibaba (15 nodes*) κ.α.

(*) List of companies powered by Apache Hadoop:
https://cwiki.apache.org/confluence/display/HADOOP2/PoweredBy

Εισαγωγή - Hadoop

- Τυπικό Hadoop Cluster



- Node specs: 8-16 cores, 16-32 GB RAM, 4-8 × 1.5 TB disks
- Nodes are installed on racks
- 1 Gbps in rack, 8 Gbps out of rack

Εισαγωγή - Hadoop

- Το Hadoop Project αποτελείται από 4 modules:
  - Αποθήκευση δεδομένων (storage)
    - κατανεμημένο σύστημα αρχείων Hadoop Distributed File System (HDFS)
  - Παράλληλη Επεξεργασία μεγάλων δεδομένων (processing)
    - Προγραμματιστικό μοντέλο Map-Reduce
  - Χρονοπρογραμματισμός εργασιών και Διαχείριση πόρων (job scheduling and resource management)
    - Hadoop YARN
  - Common utilities για υποστήριξη των άλλων modules
    - Hadoop Common

Εισαγωγή - Hadoop

- Περισσότερες Πληροφορίες:
  - Apache Hadoop Documentation
  - HDFS Users Guide
  - HDFS Architecture
  - HDFS Default Configuration

HDFS

- Το Hadoop βασίζεται σε ένα κατανεμημένο σύστημα αρχείων το Hadoop Distributed File System (HDFS)
  - Εμπνευσμένο από το Google File System (GFS) το οποίο είναι φυσικά πατενταρισμένο.
- Σχεδιασμένο για την αποθήκευση μεγάλου όγκου δεδομένων.
- Το HDFS έχει data block size συνήθως 64MB ή 128MB. Τα filesystems NTFS (Microsoft) και ext4 (Linux) έχουν συνήθως μόνο 4 KB.
  - Μικρά αρχεία (με size σημαντικά μικρότερο από 64MB) αποτελούν μεγάλο πρόβλημα για το HDFS που είναι σχεδιασμένο να διαχειρίζεται τεράστια αρχεία αποδοτικά

HDFS

- Το HDFS παρέχει αξιοπιστία μέσω αντιγραφής των δεδομένων σε περισσότερους από 1 κόμβους/υπολογιστές (nodes).

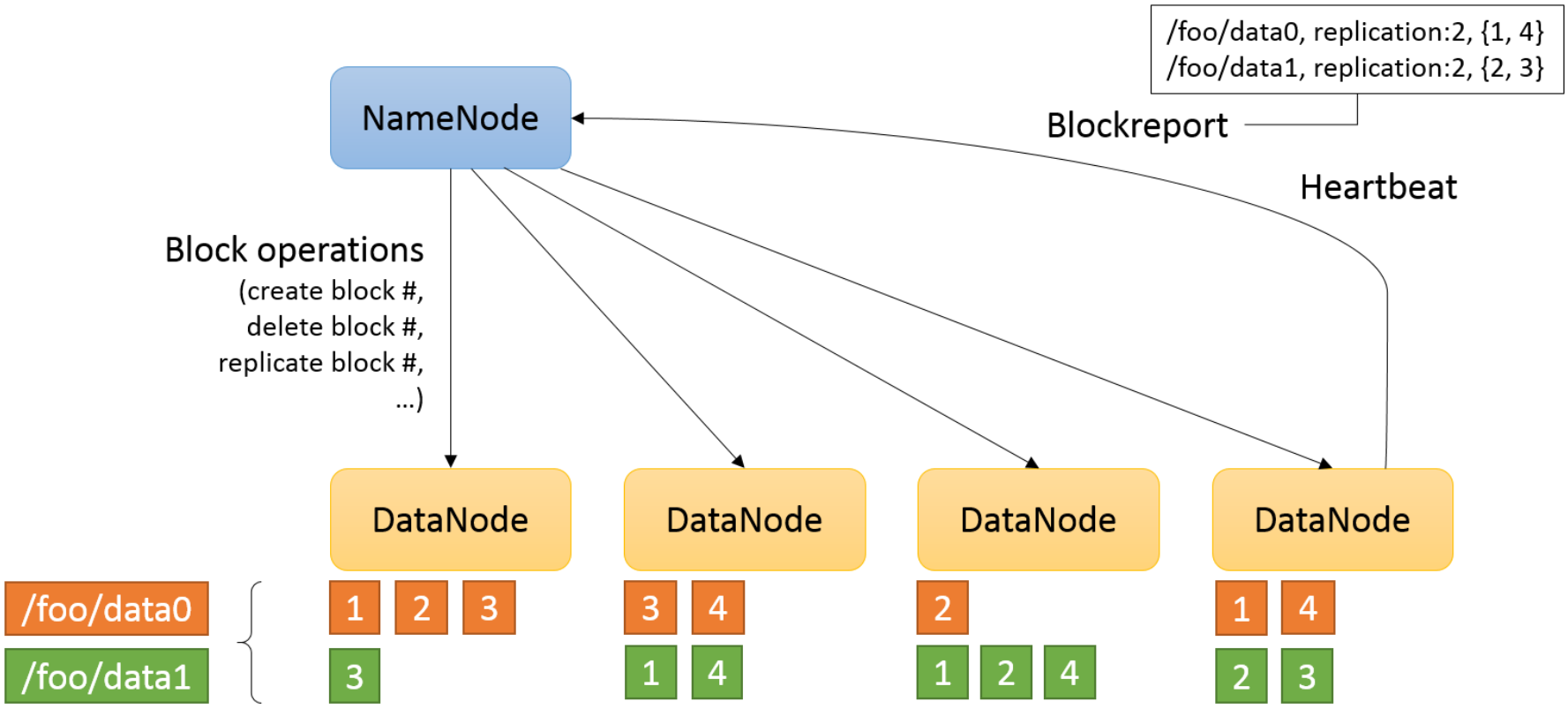
- Όταν ένας κόμβος δεν είναι διαθέσιμος, τα δεδομένα μπορούν να ανακτηθούν από άλλους κόμβους (διότι υπάρχουν αντίγραφα).

- Το χαρακτηριστικό αυτό ονομάζεται replication.
  - default replication factor in HDFS is 3
    - one original block and two replicas
    - can be set in hdfs-site.xml

HDFS

- Ο κεντρικός node (master node) στο HDFS έχει το ρόλο του **NameNode**.
- Ο **NameNode** διατηρεί διάφορες μετα-πληροφορίες (metadata) για το σύστημα αρχείων όπως τον πίνακα (index) που περιγράφει **πού** βρίσκεται το κάθε αρχείο ή κομμάτι αρχείου (chunk), δηλ. σε ποιο node και ρυθμίζει την πρόσβαση των χρηστών στα αρχεία.
- Τα υπόλοιπα nodes έχουν τον ρόλο του **DataNode** δηλ. αποθηκεύουν τα δεδομένα και εκτελούν πράξεις σε αυτά (create, delete data block) έπειτα από οδηγίες του NameNode.

HDFS: NameNode/DataNode



Each Datanode frequently (default, every 3 mins) sends a heartbeat and block report (list of available blocks) to the Namenode. If a Datanode does not send a heartbeat before dfs.namenode.stale.datanode.interval is reached, the Datanode is marked dead.

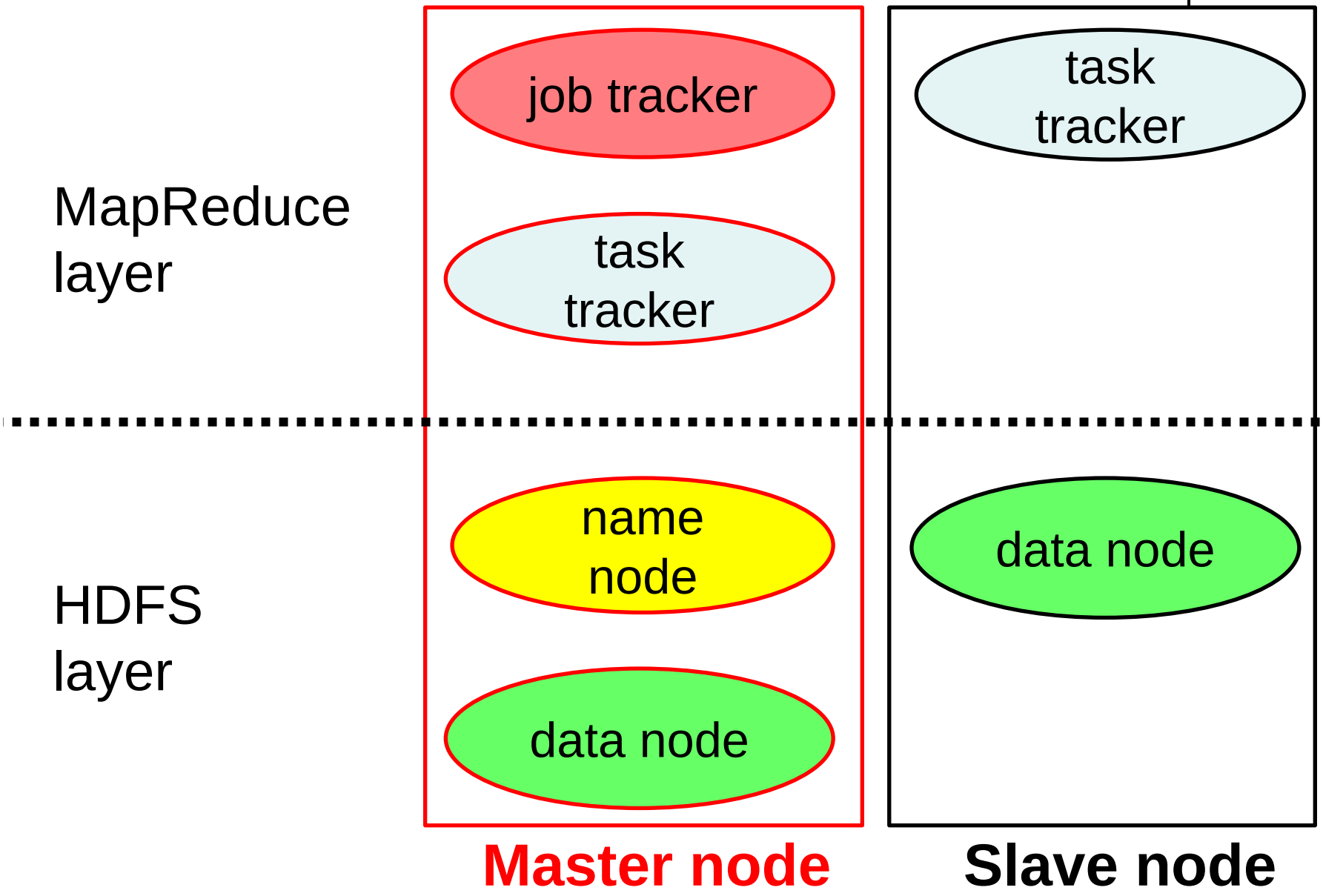
HDFS

- Αλληλεπιδρούμε με το HDFS μέσω shell-like commands της μορφής **hadoop fs <args>**
- Παραδείγματα (εκτέλεση από unix terminal)
  - `hadoop fs -ls /`
    - Παρουσίαση περιεχομένου ριζικού φακέλου (/) μέσα στο HDFS.
  - `hadoop fs -copyFromLocal /home/myfolder /myfolder`
    - Αντιγραφή του καταλόγου /home/myfolder (μαζί με τα περιεχόμενά του) που είναι στον τοπικό (local) δίσκο, προς το HDFS, σε κατάλογο που ονομάζεται myfolder (ο φάκελος αυτός βρίσκεται στον ριζικό κατάλογο του HDFS)
  - `hadoop fs -copyToLocal /hdfsfolder /localfolder`
  - `hadoop fs -cat /user/csdeptucy/myfolder/test.java`
    - Τυπώνει στην οθόνη το περιεχόμενο του αρχείου test.java το οποίο είναι αποθηκευμένο μέσα στο HDFS
  - `hadoop fs –rm -r /hdfsfolder`
    - Διαγραφή καταλόγου που είναι αποθηκευμένος μέσα στο HDFS

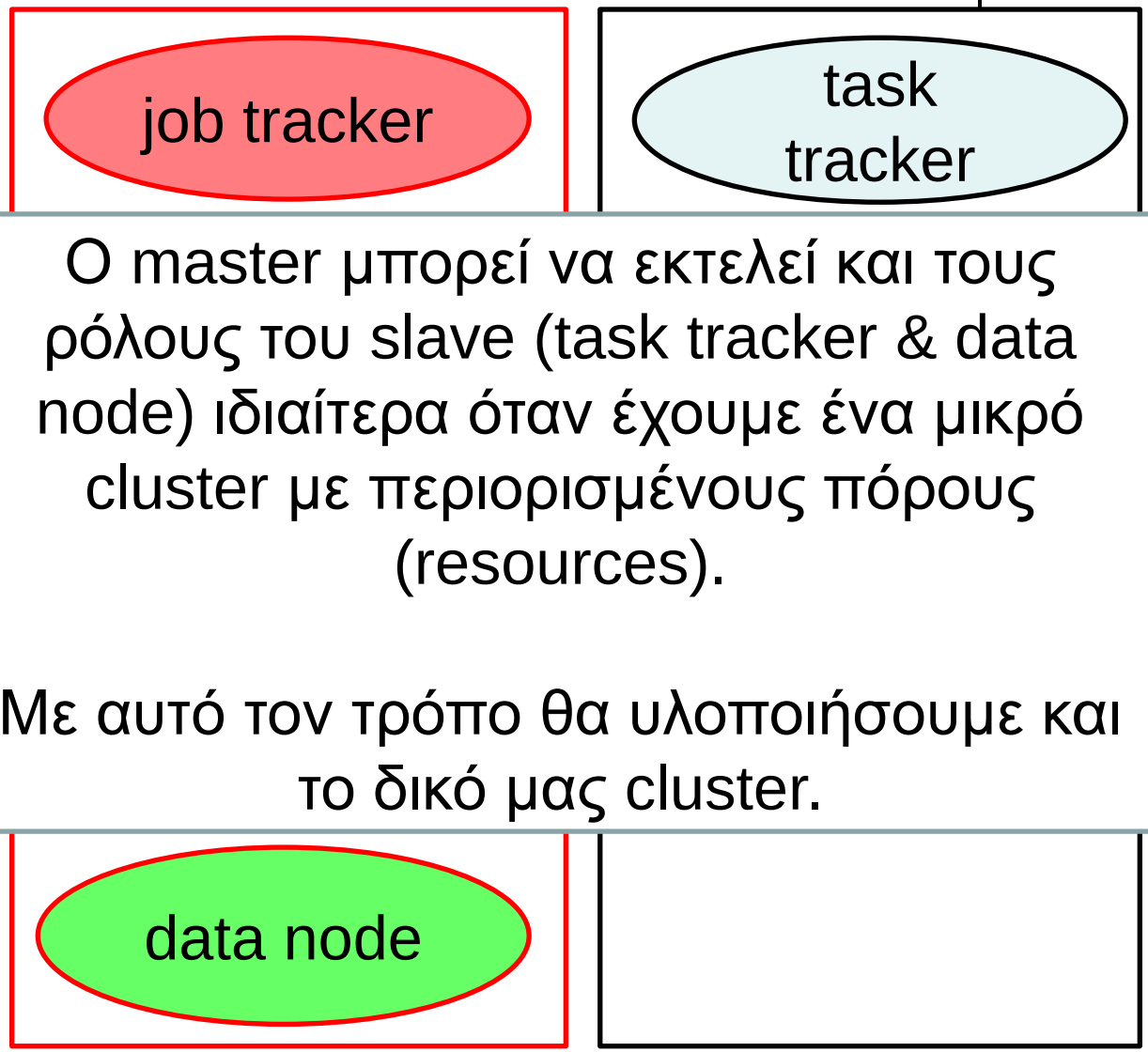
Αρχιτεκτονική Hadoop

- Η αρχιτεκτονική του Hadoop ακολουθεί την ιδεολογία/μοντέλο master-slave.
- Ο master node (κύριος κόμβος) είναι υπεύθυνος για το καταμερισμό των εργασιών.
- Οι slave nodes (υπηρέτες) εκτελούν τις εργασίες και επιστρέφουν το αποτέλεσμα στον master.
- Ο master node στο Hadoop έχει το ρόλο του **job tracker** (δηλ. την εποπτεία και τον καταμερισμό των εργασίων).
- Οι slave nodes έχουν το ρόλο **task tracker** (εκτέλεση της εργασίας που ανατέθηκε από τον job tracker).

Αρχιτεκτονική Hadoop



Αρχιτεκτονική Hadoop



Ο master μπορεί να εκτελεί και τους ρόλους του slave (task tracker & data node) ιδιαίτερα όταν έχουμε ένα μικρό cluster με περιορισμένους πόρους (resources).

Με αυτό τον τρόπο θα υλοποιήσουμε και το δικό μας cluster.

Μοντέλο Map Reduce

- Είσοδος: σύνολο από ζευγάρια
  < κλειδί εισόδου – τιμή >
- Έξοδος: σύνολο από ζευγάρια
  < κλειδί εξόδου – αποτέλεσμα >
- Δύο λειτουργίες: **map** & **reduce**
- Η λειτουργία map δέχεται σαν είσοδο μια λίστα με ζεύγη κλειδί-τιμή (ένα κάθε φορά) και για κάθε ζεύγος εισόδου παράγει σαν έξοδο ένα άλλο ζεύγος < κλειδί – ενδιάμεση τιμή >.
- Η λειτουργία reduce, μειώνει (reduces) το σύνολο ενδιάμεσων τιμών που έχουν το ίδιο κλειδί σε ένα μικρότερο σύνολο από τιμές.

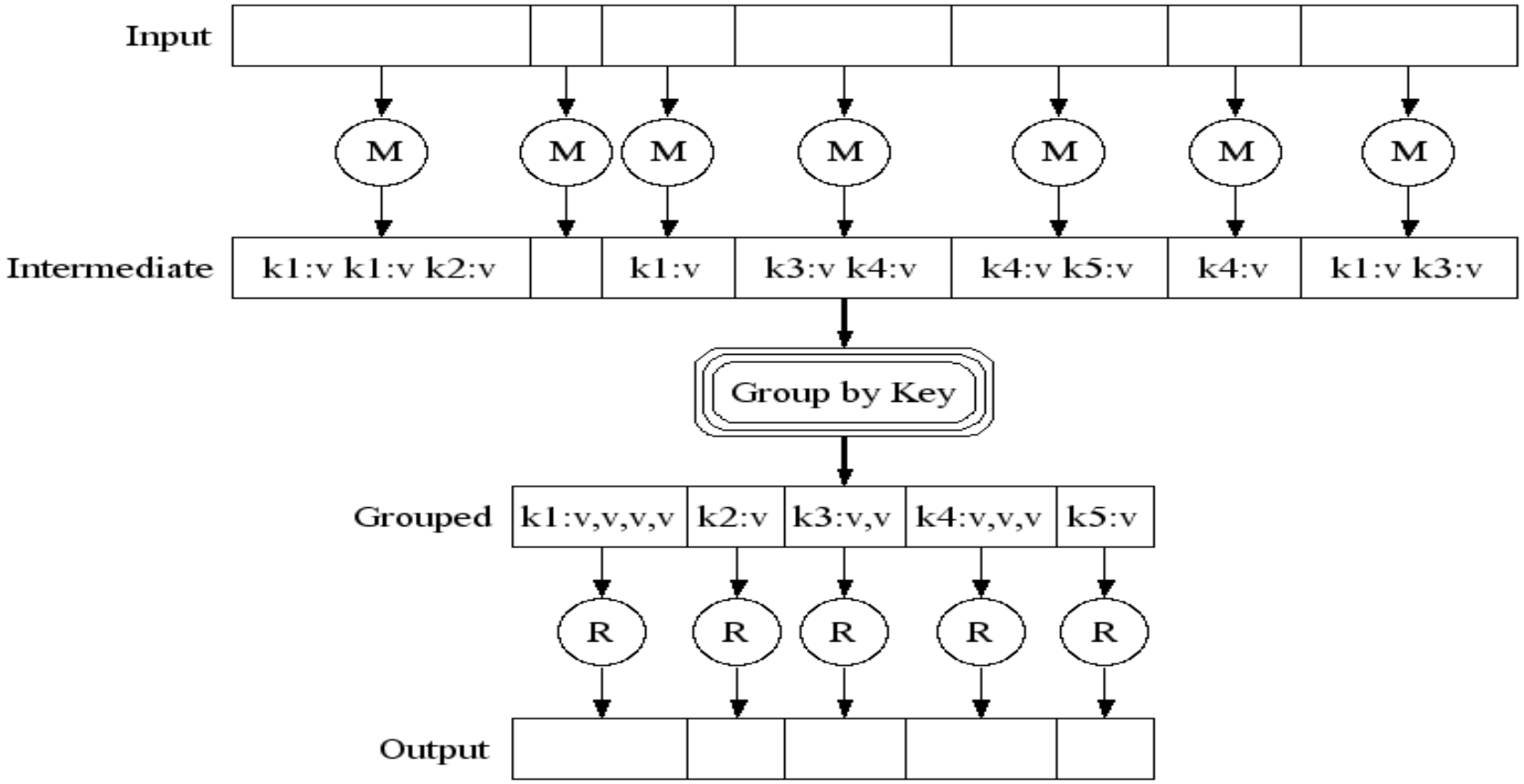
Λειτουργία Map

- Έστω ότι δίνεται ένα αρχείο κειμένου:
  - Ένα αρχείο μπορεί να διαχωριστεί σε πολλαπλά records (εγγραφές)
    - Μια εγγραφή μπορεί να ναι μια γραμμή του αρχείου
- **Η μέθοδος map:**
  - Υλοποιείται από το χρήστη,
  - Σε κάθε κλήση, παίρνει σαν είσοδο ένα record στη μορφή ενός ζεύγους <κλειδί,τιμή> και παράγει ένα ζεύγος <κλειδί,ενδιάμεση τιμή>.

Λειτουργία Reduce

- Το σύνολο ζευγών < κλειδί, ενδιάμεση τιμή > παραχθέντων υπό των κλήσεων της μεθόδου map ομαδοποιούνται ως προς το κλειδί
  - Όλες οι ενδιάμεσες τιμές (values) για ένα συγκεκριμένο κλειδί (key) συνενώνονται σε μια λίστα, και δίδονται σε στον reducer.
    - < key, <list of values with the same key>>
  - Ο reducer εκτελεί συνάθροιση των τιμών της λίστας παράγοντας στην έξοδο ένα τελικό ζεύγος κλειδί/τιμή
    - < key, new_value>
- Ο Reducer περιλαμβάνει 3 κύριες φάσεις:
  - Shuffle, Sort, Reduce

Map & Reduce Tasks μαζί



Παράδειγμα – WordCount

- Κλασσικό παράδειγμα για το πώς λειτουργεί το Map Reduce.
- Μετρά πόσες φορές εμφανίζεται η κάθε λέξη σε ένα ή περισσότερα αρχεία.
- Η επεξεργασία γίνεται μέσω Hadoop και άρα είναι κατανεμημένη (distributed).

Παράδειγμα – WordCount

- Το map παίρνει σαν input ζευγάρια της μορφής <key, value> όπου:
  - το **key** είναι το offset της γραμμής στο αρχείο (η θέση της γραμμής στο αρχείο σε bytes).
  - το **value** είναι μια ολόκληρη γραμμή από ένα από τα αρχεία
- Η θέση που βρίσκεται η γραμμή στο αρχείο δεν μας ενδιαφέρει γι' αυτό και αγνοούμε το key.
- Το map κάνει tokenize («σπάει» τη γραμμή σε λέξεις) και δίνει σαν έξοδο ζευγάρια <key, value> :
  - το **key** είναι κάθε μια από τις λέξεις
  - το **value** είναι πάντα ο αριθμός 1
- Τα πιο πάνω ζευγάρια εισάγονται στο reduce

Παράδειγμα – WordCount

- Το reduce παίρνει σαν input ένα ζευγάρι < **key**, <**list of values**> > όπου:
  - **key** είναι μια από τις λέξεις
  - το **value** είναι μια λίστα που περιέχει τον αριθμό 1 τόσες φόρες όσες εμφανίζεται η λέξη που βρίσκεται στο key στα αρχεία μας.
  - Π.χ < the, <1, 1, 1, 1> >
- Το reduce δίνει σαν output ένα ζευγάρι <**key**, **value**> όπου:
  - **key** είναι η λέξη που πήραμε σαν input
  - **value** είναι το άθροισμα των άσσων στη λίστα των values.

Παράδειγμα – WordCount

Line 1: the government said yesterday it was on track to cover the financing needs for the next three months but for the next three years there was no choice to cover

MAP

Input to map: key = 0, value = the government said yesterday […]

Output: <the,1>, <government,1>, <said,1>, <yesterday,1>, <it,1>, <was,1>, <on,1> … … <choice,1> , <to,1> <cover,1>

REDUCE

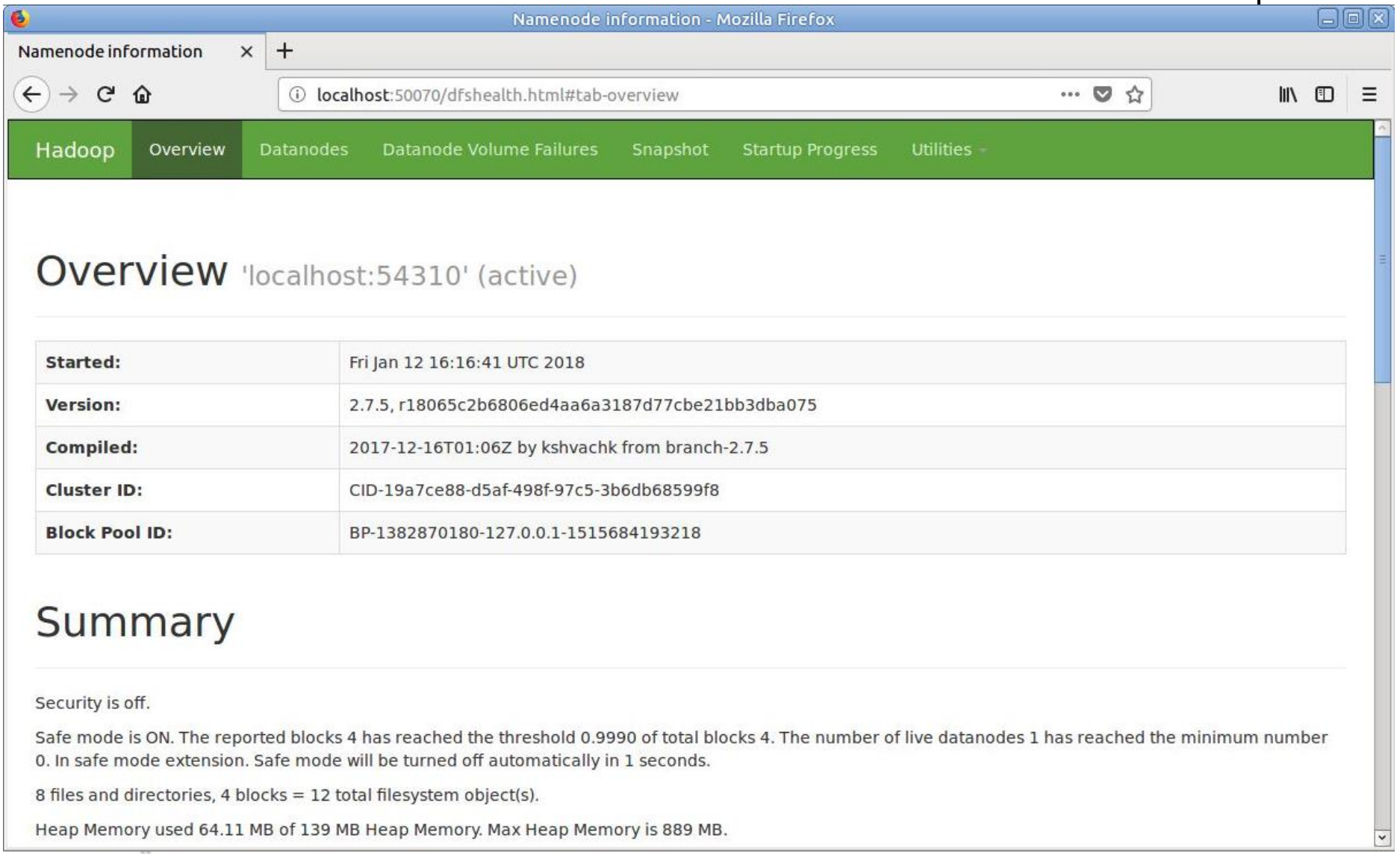
Input to reduce:
key = the, value = <1,1,1,1>
key = government, value = <1> … …

Output: <the, 4>, <government, 1>, <said, 1> … <to, 2>, <cover, 2>

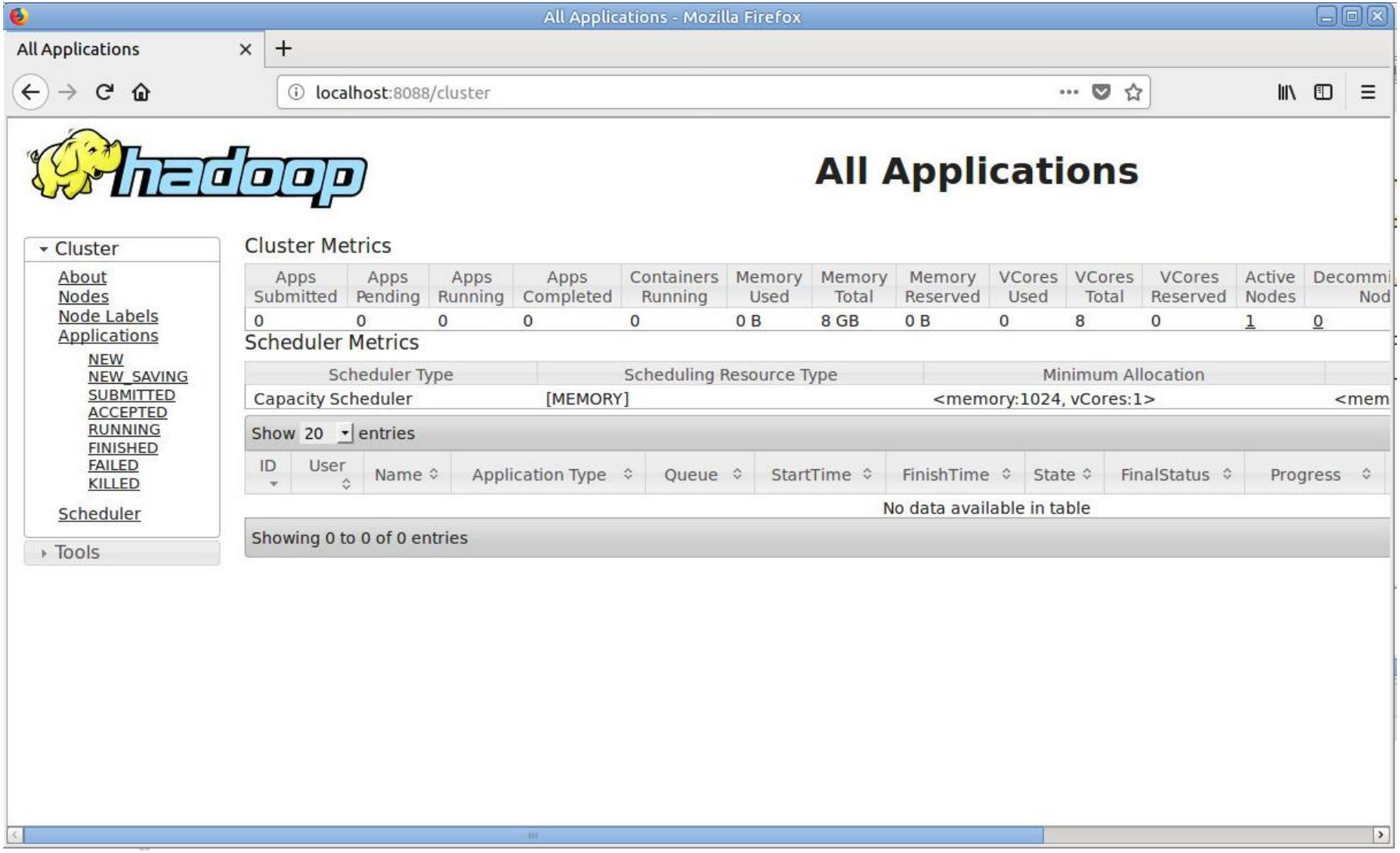
Πρακτική – WordCount

- Θα χρησιμοποιήσουμε ένα Virtual Machine που έχει ήδη εγκατεστημένα το Hadoop, Java, Eclipse και ό,τι άλλο χρειάζεται για το παράδειγμα word count.
- Χρήσιμες πληροφορίες:
  - Hadoop API:
    - https://hadoop.apache.org/docs/current/api/
  - HDFS NameNode web interface:
    - localhost:50070
  - Resource Manager web interface:
    - localhost:8088

HDFS Namenode



Resource Manager (YARN)



Πρακτική – WordCount

- Ξεκινήστε το Hadoop ☺ [Admin Commands]
  - Πλοηγηθείτε στον κατάλογο /usr/local/hadoop/sbin και εκτελέστε την εντολή ./start-all.sh
  - Βεβαιωθείτε με την εντολή jps ότι ξεκίνησαν NameNode, SecondaryNameNode, DataNode, NodeManager, ResourceManager (The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The NodeManager is the per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager. See more here)
- Ανοίξτε το eclipse, και στο Project Hadoop_1 συμπληρώστε τον κώδικα του WordCount.java
  - /user/ubuntu/input (ο φάκελος μέσα στο HDFS όπου βρίσκονται τα αρχεία μας)
  - /user/ubuntu/output (ο φάκελος μέσα στο HDFS όπου θέλουμε να αποθηκευτούν τα αποτελέσματα του MapReduce)

Πρακτική – WordCount

- Τρέξτε τον κώδικα
  - Αν ο φάκελος output υπάρχει ήδη μέσα στο HDFS, δεν θα τρέξει το πρόγραμμα. Σβήστε τον με την εντολή: **hadoop fs -rm -r /user/csdeptucy/output**
- Αντιγράψτε το αρχείο εξόδου (στο φάκελο output) από το HDFS στο local file system και δείτε το περιεχόμενό του [εναλλακτικά μπορείτε να δείτε το αρχείο μέσω του browser]
  - **hadoop fs -copyToLocal /user/csdeptucy/output/part-r-00000**
  - The output files are by default named part-x-yyyyy where:
    - x is either 'm' or 'r', depending on whether the job was a map only job, or reduce
    - yyyyy is the mapper or reducer task number (starting from 0)
      - » So a job which has 32 reducers will have files named part-r-00000 to part-r-00031, one for each reducer task

Πρακτική – WordCount

- Μπορείτε να δείτε κάποια από τα περιεχόμενα του αρχείου μέσω της εντολής
  - **hadoop fs -cat output/part-r-00000 | head**
- Λεπτομερή παρουσίαση του παραδείγματος μπορείτε να βρείτε εδώ: http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Πρακτική – WordCount

- Mapper and reducer can output different types for key – value pairs
- If mapper & reducer output the same types (for key and value) you can use `setOutputKeyClass()` and `setOutputValueClass()` in main
  - as in WordCount example
- If they output different types, you can set the types emitted **by the mapper** with the `JobConf`'s `setMapOutputKeyClass()` and `setMapOutputValueClass()` methods while the previous methods `setOutputKeyClass()` and `setOutputValueClass()` will define the output types of the Reducer

Hadoop Admin Commands

- The /usr/local/hadoop/bin directory contains some scripts used to launch Hadoop DFS and Hadoop Map/Reduce daemons. These are:
- **start-all.sh** - Starts all Hadoop daemons, the namenode, datanodes, the jobtracker and tasktrackers.
- **stop-all.sh** - Stops all Hadoop daemons.
- **start-mapred.sh -** Starts the Hadoop Map/Reduce daemons, the jobtracker and tasktrackers.
- **stop-mapred.sh** - Stops the Hadoop Map/Reduce daemons.
- **start-dfs.sh** - Starts the Hadoop DFS daemons, the namenode and datanodes.
- **stop-dfs.sh** - Stops the Hadoop DFS daemons.