

Parte 6b: Hadoop Experience

Map Reduce

Corso di: Sistemi Distribuiti

Lauree in: Ingegneria Informatica,
delle Telecomunicazioni ed Informatica di Scienze

Gianni Pantaleo, Paolo Nesi, Imad Zaza

Department of Systems and Informatics, University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-2758515, fax: +39-055-2758570

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

<http://www.disit.dinfo.unifi.it/>

paolo.nesi@unifi.it

<http://www.disit.dinfo.unifi.it/nesi>



Agenda

- Prologue
- Apache Hadoop
- Monitoring
- Apache HBASE
- Case studio



Agenda

- Prologue
- Apache Hadoop
- Monitoring
- Apache HBASE
- Case studio



Big Data Problem: Twitter Data Analytics

- Twitter is an example big data source
- BI on Twitter & social data is growing in demand
- Possibile problems:
 - Count the number of tweets containing occurrence of one or more search string (e.g. «pippo pluto», <<pippo OR pluto>>) per day in a given time interval
 - NLP (Keywords, Keyphrase extraction and grammatical analysis on natural language text)

What we need ?

- A system which crawls twitter for tweets matching our queries -> we did it but is out of scope today
- A system storing the collected tweets
- Metric processing procedures

Single Host

- I. **Develop a data model**
- II. Use an RDBMS as data backend
- III. Use SQL as query language
wrappred in java or php ...
application

Single Host

- I. Develop a data model
- II. Use an RDBMS as data backend**
- III. Use SQL as query language
wrappred in java or php ...
application

Single Host

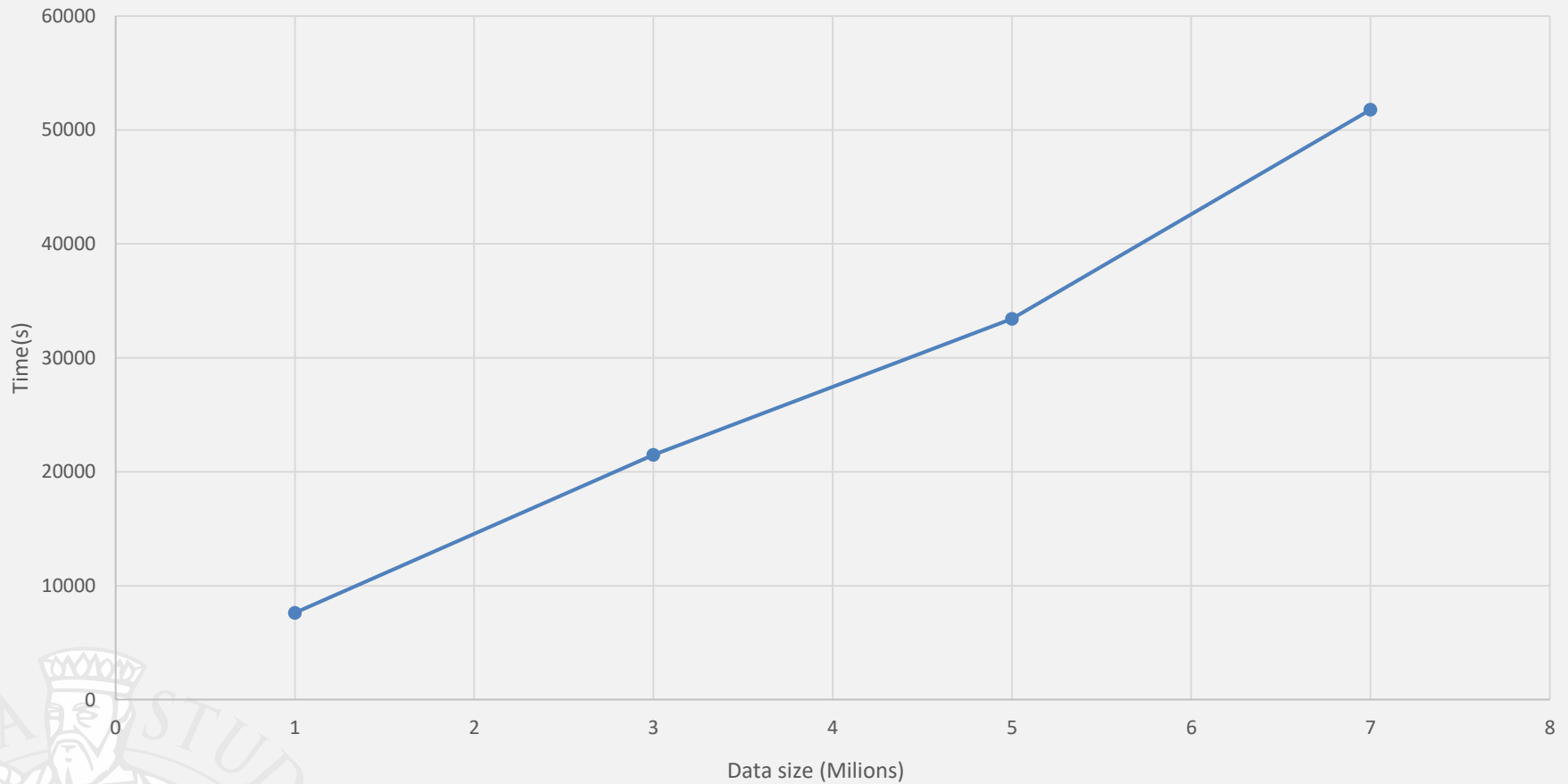
- I. Develop a data model
- II. Use an RDBMS as data backend
- III. Use SQL as query language
wrapped in java or php ...
application**

Problem

- Tweets collected grows fast
 - > Computation time degrade
 - > Reliability and Availability depends merely on hardware

Single Host

Twitter Metric processing time



Agenda

- Prologue
- Apache Hadoop
- Monitoring
- Apache HBASE
- Case studio



Cluster

- A **computer cluster** is a group of linked computers, working together closely so that in many respects they form a single computer.
- The components of a cluster are commonly, but not always, connected to each other through fast local area networks.
- Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.



Cluster (cont.)

Cluster consists of:

- Nodes (master+computing)
- Network
- OS
- Cluster middleware which permits compute



Pigsty

- 5 «pigs»
 - 3 pigs
 - 4GB Ram , dual core, 1TB disk
 - 1 pig
 - 6GB Ram, dual core, 1TB disk
 - 1 pig
 - 8GB ram, dual core, 1TB disk
- 5 Virtual Servers
 - 1 vm
 - 3GB Ram, dual core, 1TB disk
 - 2 vms
 - 8GB Ram, Dual core, 700GB
 - 2vms
 - 6GB Ram, Dual core, 700GB



HDFS

Storage

MapReduce

Processing

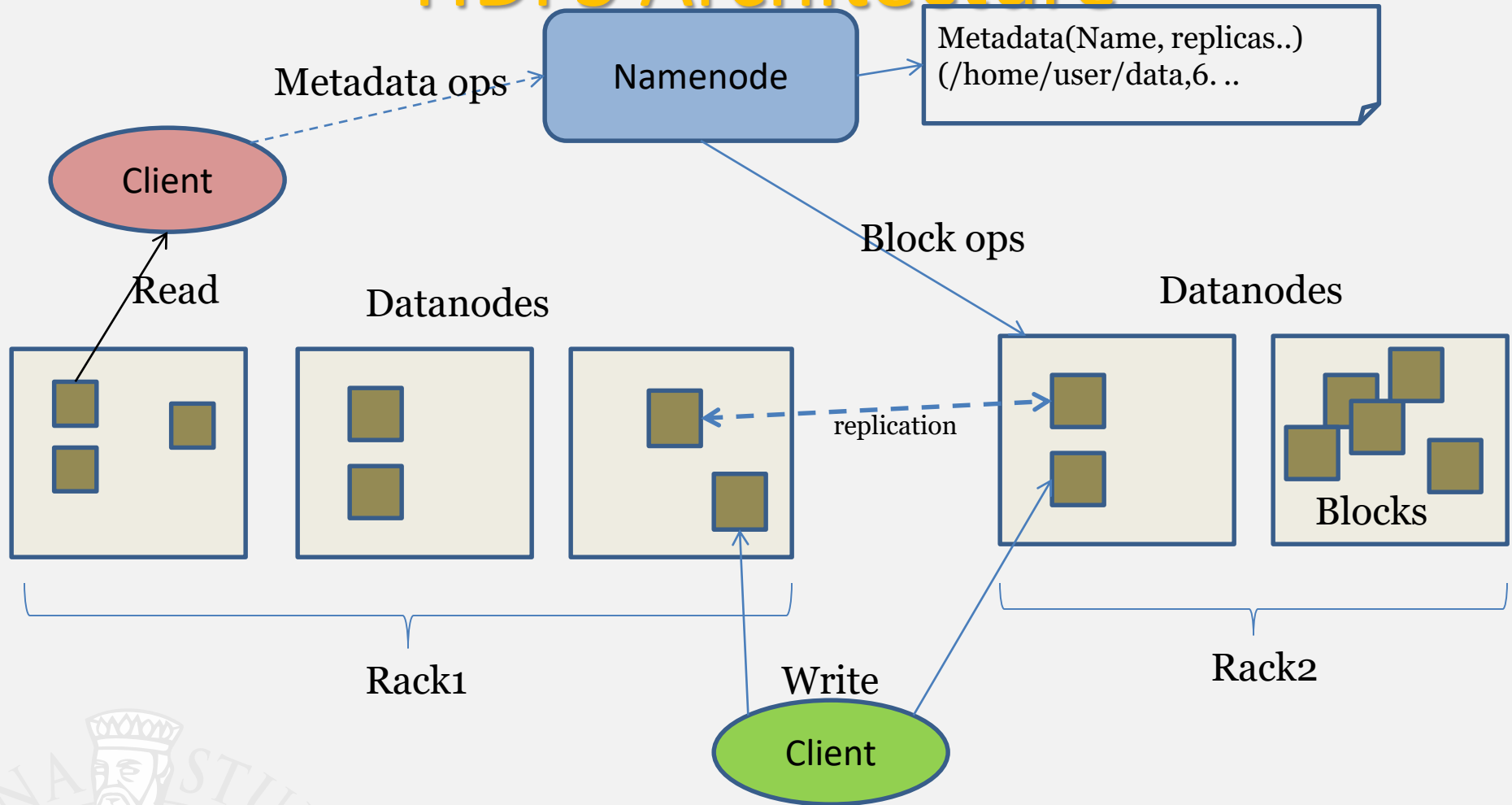
The Apache™ Hadoop® project develops open-source Software for reliable, scalable, distributed computing



Apache Hadoop

Storing data @hadoop

HDFS Architecture



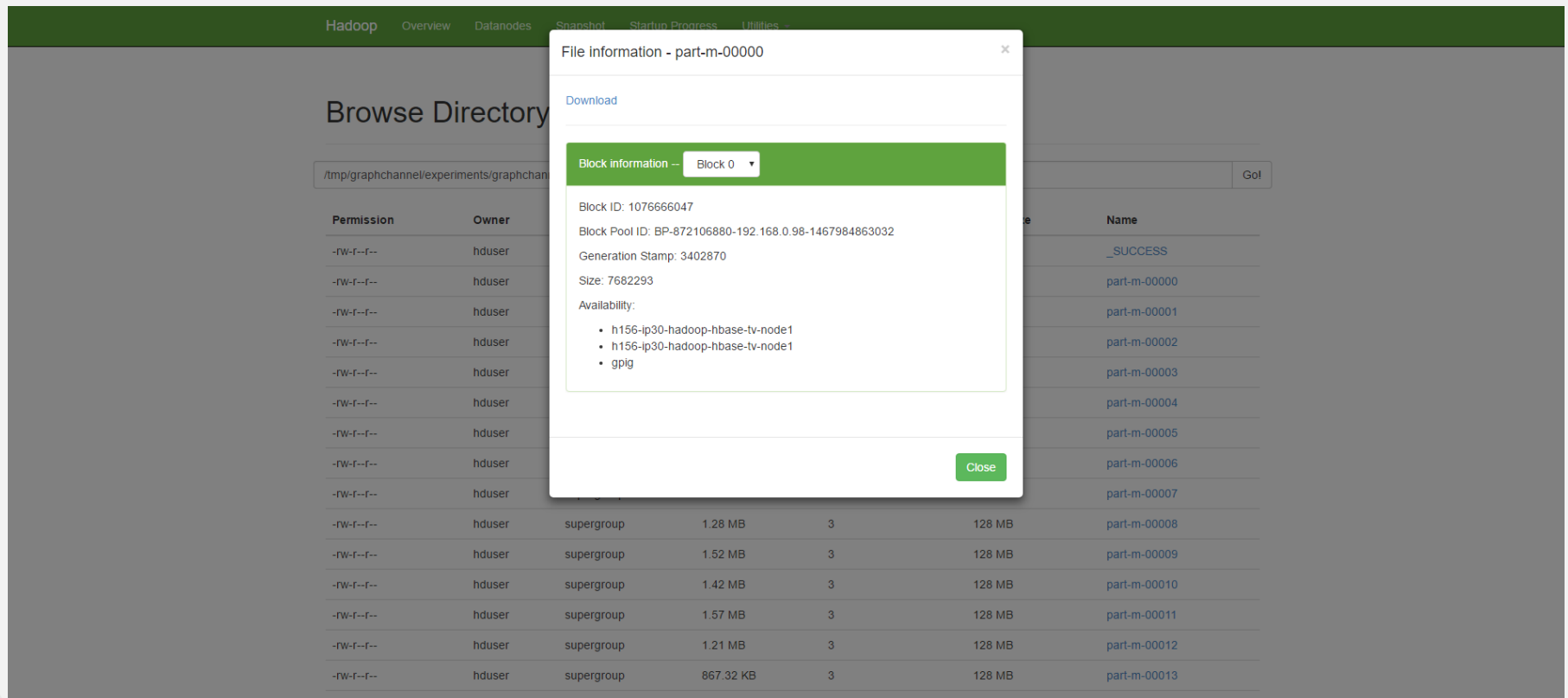
Namenode and Datanodes

- Master/slave architecture
- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of **DataNodes** usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

File system Namespace

- Hierarchical file system with directories and files
- Create, remove, move, rename etc.
- Namenode maintains the file system
- Any meta information changes to the file system recorded by the Namenode.
- An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.

Example



The screenshot shows the Hadoop web interface with a modal window open for file information. The background shows a 'Browse Directory' page with a table of files. The modal window, titled 'File information - part-m-00000', displays the following details:

- Block information**: Block 0
- Block ID: 107666047
- Block Pool ID: BP-872106880-192.168.0.98-1467984863032
- Generation Stamp: 3402870
- Size: 7682293
- Availability:
 - h156-ip30-hadoop-hbase-tv-node1
 - h156-ip30-hadoop-hbase-tv-node1
 - gplg

The background table lists files with columns for Permission, Owner, Name, and other details. The file 'part-m-00000' is highlighted in blue.

Permission	Owner	Name
-rw-r--r--	hduser	_SUCCESS
-rw-r--r--	hduser	part-m-00000
-rw-r--r--	hduser	part-m-00001
-rw-r--r--	hduser	part-m-00002
-rw-r--r--	hduser	part-m-00003
-rw-r--r--	hduser	part-m-00004
-rw-r--r--	hduser	part-m-00005
-rw-r--r--	hduser	part-m-00006
-rw-r--r--	hduser	part-m-00007
-rw-r--r--	hduser	part-m-00008
-rw-r--r--	hduser	part-m-00009
-rw-r--r--	hduser	part-m-00010
-rw-r--r--	hduser	part-m-00011
-rw-r--r--	hduser	part-m-00012
-rw-r--r--	hduser	part-m-00013

Interaction with hdfs

- Web interface
 - Web application bundled with hadoop
 - Hue
- Console Interface



Basic Interface

Hadoop | Overview | Datanodes | Snapshot | Startup Progress | Utilities

Overview 'hadoop-namenode-9000' (active)

Started:	Thu Dec 22 09:52:46 CET 2016
Version:	2.7.2 (build 20160924) (revision 20160924)
Compiled:	2016-10-20T00:10:27 by jenkins from (detached from 0cf4050)
Cluster ID:	CID-3db4b3ad-5fca-4c29-9928-f0647bd44c68
Block Pool ID:	BP-872106880-192.168.0.98-1467984863032

Summary

Security is off.
Safemode is off.

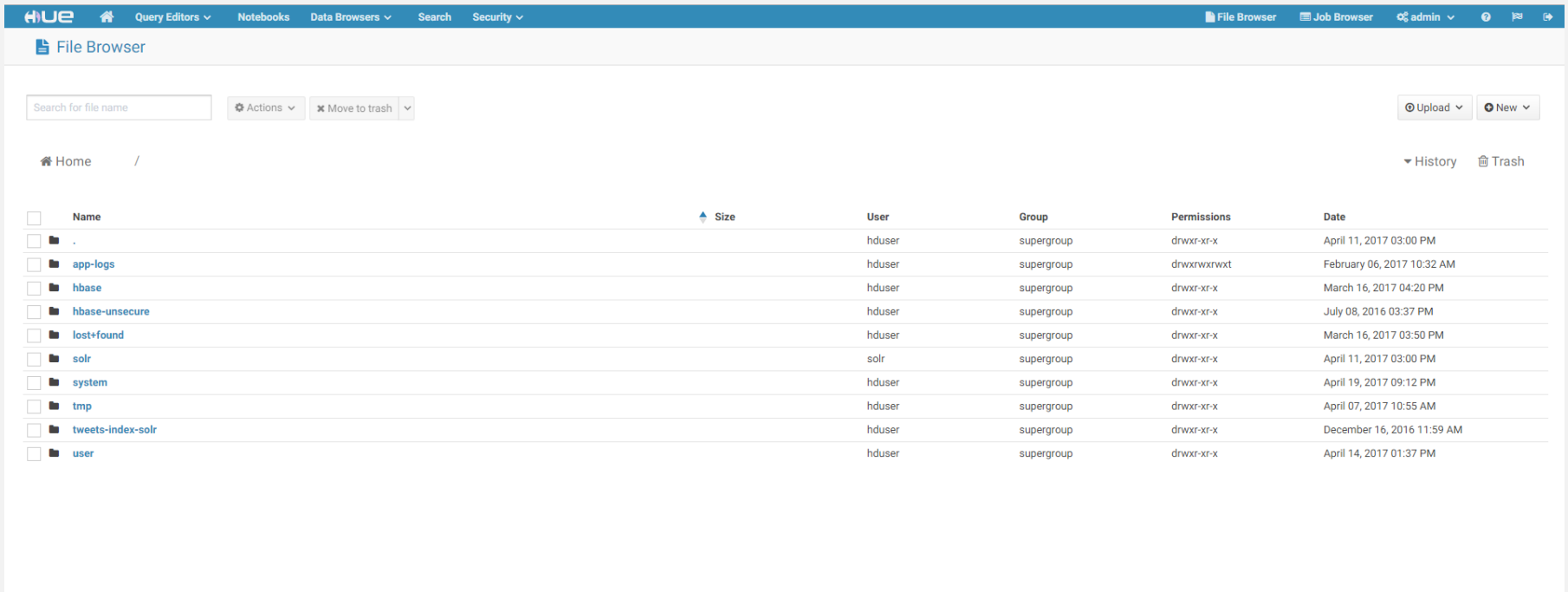
345630 files and directories, 307885 blocks = 653515 total filesystem object(s).

Heap Memory used 564.82 MB of 771.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 94.58 MB of 96.25 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	10.17 TB
DFS Used:	4.78 TB
Non DFS Used:	559.62 GB
DFS Remaining:	4.85 TB
DFS Used%:	46.98%
DFS Remaining%:	47.65%
Block Pool Used:	4.78 TB
Block Pool Used%:	46.98%
DataNodes usages% (Min/Median/Max/stdDev):	0.02% / 44.70% / 55.78% / 14.55%
Live Nodes	11 (Decommissioned: 1)

Advanced Interface



The screenshot shows the HUE File Browser interface. At the top, there is a navigation bar with 'HUE' and various menu items like 'Query Editors', 'Notebooks', 'Data Browsers', 'Search', and 'Security'. Below the navigation bar, there is a search bar and several action buttons: 'Actions', 'Move to trash', 'Upload', and 'New'. The main content area displays a list of files and folders in a table format. The table has columns for 'Name', 'Size', 'User', 'Group', 'Permissions', and 'Date'. The files listed include '.', 'app-logs', 'hbase', 'hbase-unsafe', 'lost+found', 'solr', 'system', 'tmp', 'tweets-index-solr', and 'user'.

Name	Size	User	Group	Permissions	Date
.		hduser	supergroup	drwxr-xr-x	April 11, 2017 03:00 PM
app-logs		hduser	supergroup	drwxrwxrwt	February 06, 2017 10:32 AM
hbase		hduser	supergroup	drwxr-xr-x	March 16, 2017 04:20 PM
hbase-unsafe		hduser	supergroup	drwxr-xr-x	July 08, 2016 03:37 PM
lost+found		hduser	supergroup	drwxr-xr-x	March 16, 2017 03:50 PM
solr		solr	supergroup	drwxr-xr-x	April 11, 2017 03:00 PM
system		hduser	supergroup	drwxr-xr-x	April 19, 2017 09:12 PM
tmp		hduser	supergroup	drwxr-xr-x	April 07, 2017 10:55 AM
tweets-index-solr		hduser	supergroup	drwxr-xr-x	December 16, 2016 11:59 AM
user		hduser	supergroup	drwxr-xr-x	April 14, 2017 01:37 PM



Console Interface: some commands

- Create directory
hdfs dfs -mkdir <hdfs_path>
- List directory
hdfs dfs -ls <hdfs_path>
- Delete file
**hdfs dfs -rm <hdfs_path>
/file**
- Delete directory
hdfs dfs -rm -r -f <hdfs_path>
- Upload file to hdfs
hdfs dfs -put file.txt <hdfs_path>
- Download file from hdfs
hdfs dfs -get <hdfs_path>/file.txt



Let's try it !

Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
- Each file is a sequence of blocks.
- All blocks in the file except the last are of the same size.
- Blocks are replicated for fault tolerance.
- Block size and replicas are configurable per file.
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.

Filesystem Metadata

- The HDFS namespace is stored by Namenode.
- Namenode uses a transaction log called the EditLog to record every change that occurs to the filesystem meta data.
 - For example, creating a new file.
 - Change replication factor of a file
 - EditLog is stored in the Namenode's local filesystem
- Entire filesystem namespace including mapping of blocks to files and file system properties is stored in a file FsImage. Stored in Namenode's local filesystem.

Namenode

- Keeps image of entire file system namespace and file Blockmap in memory.
- 4GB of local RAM is sufficient to support the above data structures that represent the huge number of files and directories.
- When the Namenode starts up it gets the FsImage and Editlog from its local file system, update FsImage with EditLog information and then stores a copy of the FsImage on the filesystem as a checkpoint.
- Periodic checkpointing is done. So that the system can recover back to the last checkpointed state in case of a crash.

Datanode

- A Datanode stores data in files in its local file system.
- Datanode has no knowledge about HDFS filesystem
- It stores each block of HDFS data in a separate file.
- Datanode does not create all files in the same directory.
- When the filesystem starts up it generates a list of all HDFS blocks and send this report to Namenode:
Blockreport.

The Communication Protocol

- All HDFS communication protocols are layered on top of the TCP/IP protocol
- A client establishes a connection to a configurable TCP port on the Namenode machine. It talks ClientProtocol with the Namenode.
- The Datanodes talk to the Namenode using Datanode protocol.
- RPC abstraction wraps both ClientProtocol and Datanode protocol.
- Namenode is simply a server and never initiates a request; it only responds to RPC requests issued by DataNodes or clients.

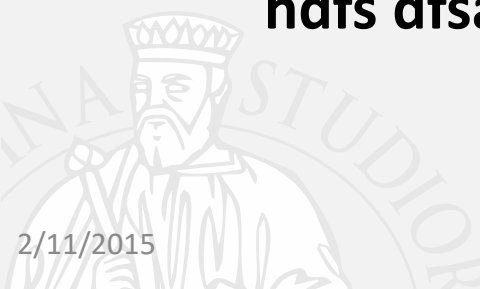
HDFS maintance (Cli)

- Reports basic filesystem information and statistics. Optional flags may be used to filter the list of displayed DataNodes.

hdfs dfsadmin -report

- Re-read the hosts and exclude files to update the set of Datanodes that are allowed to connect to the Namenode and those that should be decommissioned or recommissioned.

hdfs dfsadmin -refreshNodes



HDFS maintance (Cli)

- HDFS data might not always be placed uniformly across the DataNode. One common reason is addition of new DataNodes to an existing cluster. While placing new blocks (data for a file is stored as a series of blocks), NameNode considers various parameters before choosing the DataNodes to receive these blocks. Some of the considerations are:
 - Policy to keep one of the replicas of a block on the same node as the node that is writing the block.
 - Need to spread different replicas of a block across the racks so that cluster can survive loss of whole rack.
 - One of the replicas is usually placed on the same rack as the node writing to the file so that cross-rack network I/O is reduced.
 - Spread HDFS data uniformly across the DataNodes in the cluster.

hdfs balancer -policy blockpool





Let's try it !

Robustness

- Primary objective of HDFS is to store data reliably in the presence of failures.
- Three common failures are: Namenode failure, Datanode failure and network partition.

DataNode failure and heartbeat

- A network partition can cause a subset of Datanodes to lose connectivity with the Namenode.
- Namenode detects this condition by the absence of a Heartbeat message.
- Namenode marks Datanodes without Heartbeat and does not send any IO requests to them.
- Any data registered to the failed Datanode is not available to the HDFS.
- Also the death of a Datanode may cause replication factor of some of the blocks to fall below their specified value.

Data Integrity

- Consider a situation: a block of data fetched from Datanode arrives corrupted.
- This corruption may occur because of faults in a storage device, network faults, or buggy software.
- A HDFS client creates the checksum of every block of its file and stores it in hidden files in the HDFS namespace.
- When a clients retrieves the contents of file, it verifies that the corresponding checksums match.
- If does not match, the client can retrieve the block from a replica.

Metadata Disk Failure

- FsImage and EditLog are central data structures of HDFS.
- A corruption of these files can cause a HDFS instance to be non-functional.
- For this reason, a Namenode can be configured to maintain multiple copies of the FsImage and EditLog.
- Multiple copies of the FsImage and EditLog files are updated synchronously.
- Meta-data is not data-intensive.
- Prior Hadoop 2.x -> The Namenode could be single point failure: automatic failover is NOT supported!
- Hadoop bigger > 2.x has HA Feature (basically using nfs share or Zookeeper)

Data Blocks

- HDFS support write-once-read-many with reads at streaming speeds.
- A typical block size is 64MB (or even 128 MB).
- A file is chopped into 64MB chunks and stored.

Staging

- A client request to create a file does not reach Namenode immediately.
- HDFS client caches the data into a temporary file. When the data reached a HDFS block size the client contacts the Namenode.
- Namenode inserts the filename into its hierarchy and allocates a data block for it.
- The Namenode responds to the client with the identity of the Datanode and the destination of the replicas (Datanodes) for the block.
- Then the client flushes it from its local memory.

Staging (contd.)

- The client sends a message that the file is closed.
- Namenode proceeds to commit the file for creation operation into the persistent store.
- If the Namenode dies before file is closed, the file is lost.
- This client side caching is required to avoid network congestion;
- Not new: it has precedence is AFS (Andrew file system).

Replication Pipelining

- When the client receives response from Namenode, it flushes its block in small pieces (4K) to the first replica, that in turn copies it to the next replica and so on.
- Thus data is pipelined from Datanode to the next.

Application Programming Interface

- HDFS provides [Java API](#) for application to use.
- [Python](#) access is also used in many applications.
- A C language wrapper for Java API is also available.
- A HTTP browser can be used to browse the files of a HDFS instance.

An example of read write java program to hdfs

Prerequisite

- I. The classpath contains the Hadoop JAR files and its client-side dependencies.
- II. The hadoop configuration files on the classpath
- III. Log4J on the classpath along with a **log4.properties** resource, or commons-logging preconfigured to use a different logging framework.



First step

Create a **FileSystem** instance by passing a new Configuration object.

```
Configuration conf = new Configuration();  
FileSystem fs = FileSystem.get(conf);
```



Next ?

Given an input/output file name as string, we construct `inFile/outFile Path` objects. Most of the [FileSystem](#) APIs accepts [Path](#) objects.

```
Path inFile = new Path(argv[0]);  
Path outFile = new Path(argv[1]);
```

Some sanitizing (Validate the input/output paths before reading/writing.)

```
if (!fs.exists(inFile))  
    printAndExit("Input file not found");  
if (!fs.isFile(inFile))  
    printAndExit("Input should be a file");  
if (fs.exists(outFile))  
    printAndExit("Output already exists");
```



Final step

- i. Open inFile for reading.

```
FSDataInputStream in = fs.open(inFile);
```

- ii. Open outFile for writing.

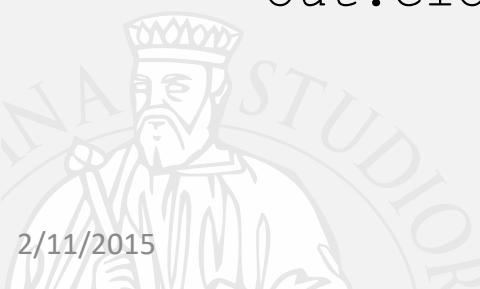
```
FSDataOutputStream out = fs.create(outFile);
```

- iii. Read from input stream and write to output stream until EOF.

```
while ((bytesRead = in.read(buffer)) > 0) {  
    out.write(buffer, 0, bytesRead);  
}
```

- Close the streams when done.

```
in.close();  
out.close();
```



Compile

- `Mkdir <DIR_for_jar>`
- `Javac -cp $(hadoop classpath) -d <DIR_for_jar>
<ClassNameFile.java>`
- `Jar cvfe <Dest>.jar org/disit/ClassName -C
<DIR_for_jar> .`



launch

- `Hadoop jar <Dest>.jar`





Apache Hadoop

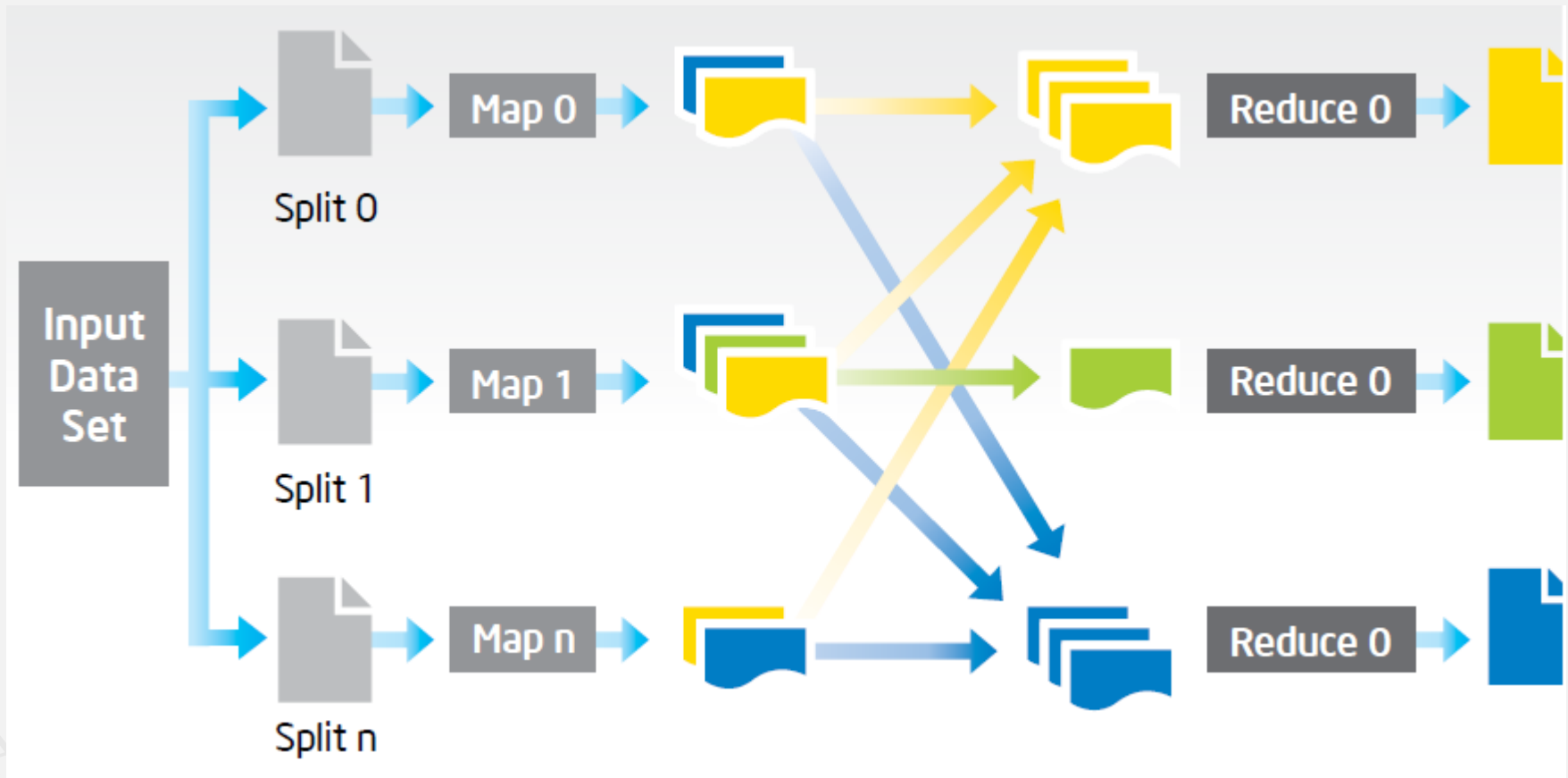
Parallel Computing@hadoop

Issue

- Past lesson we saw the fundamental of map reduce
- Working example bundle with hadoop documentation: wordcount



Review Map/Reduce Flow



Real Word Count Example

- Simple word counting program

- Document1.txt

Queste sono Le slide di Mario Rossi

- Document2.txt

Le slide del corso di Sistemi Distribuiti

- Expected Output:

Queste	1
sono	1
le	1
slide	2
di	2
Mario	1
Rossi	1
Le	1
del	1
corso	1
Sistemi	1
Distribuiti	1

Real Word Count Example

- WordCount.java → simple word counting Java program to be executed through MapReduce in a Hadoop Cluster.
- SampleDoc.txt → Input text file (programma del corso di Sistemi Distribuiti):

Programma del corso

dettagli e slide possono essere ottenuti da social network, smart city.

Overview parte 0, ver:0.6: una vista generale al corso

Introduzione (Parte 1, ver:2.0): (versione 2.4) Cosa sono i sistemi distribuiti, Tecnologie dei sistemi distribuiti, Internet e sua Evoluzione, Intranet, Penetrazione di internet, Crescita, Sistemi Mobili, Condivisione delle risorse, Web Server and Web Services, Caratteristiche: Eterogenei, aperti, sicuri, trasparenti, architetture, n-tier.

XML (parte 1b): fondamenti di XML, uso avanzato dell'XML

PHP e Drupal: Parte 1cI, Parte 1cII, architetture web server, programmazione in PHP, costrutti dell linguaggio, operatori, get/post, esempi; Parte II: Content Management Systems, CMS, moduli, call back, ruoli, etc. WEB services e REST remote invocation via Web Services and REST architectures, strumenti per i WEB services, verifica, SOAP.

[...]



NLP in Hadoop – A Real Case

```
package org.disit;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class WordCount {
```

```
    public static class TokenizerMapper extends Mapper<Object, Text,
Text, IntWritable>{
        // [ . . . ]
    }
```

```
    public static class IntSumReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {
        // [ . . . ]
    }
```

```
public static void main(String[] args) throws Exception {

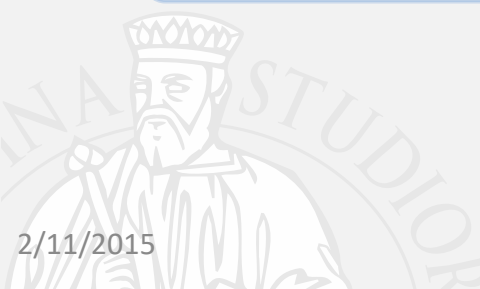
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);

    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```



NLP in Hadoop – A Real Case

Map Class

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        StringTokenizer itr = new StringTokenizer(value.toString());    // Tokenizzazione del file di testo

        while (itr.hasMoreTokens()) {

            word.set(itr.nextToken());
            context.write(word, one);

        }

    }

}
```



NLP in Hadoop – A Real Case

Reduce Class

```
public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
    private IntWritable result = new IntWritable();  
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();           /* Somma i valori unitary del conteggio di ogni singola parola  
                                       iterando su tutte le parole                */  
        }  
        result.set(sum);  
        context.write(key, result);  
    }  
}
```


Real Word Count Example

- Make executable jar:

```
$ javac -cp $(hadoop classpath) -d ./jar WordCount.java  
$ jar cvfe wordCount.jar org/disit.WordCount -C ./jar .
```

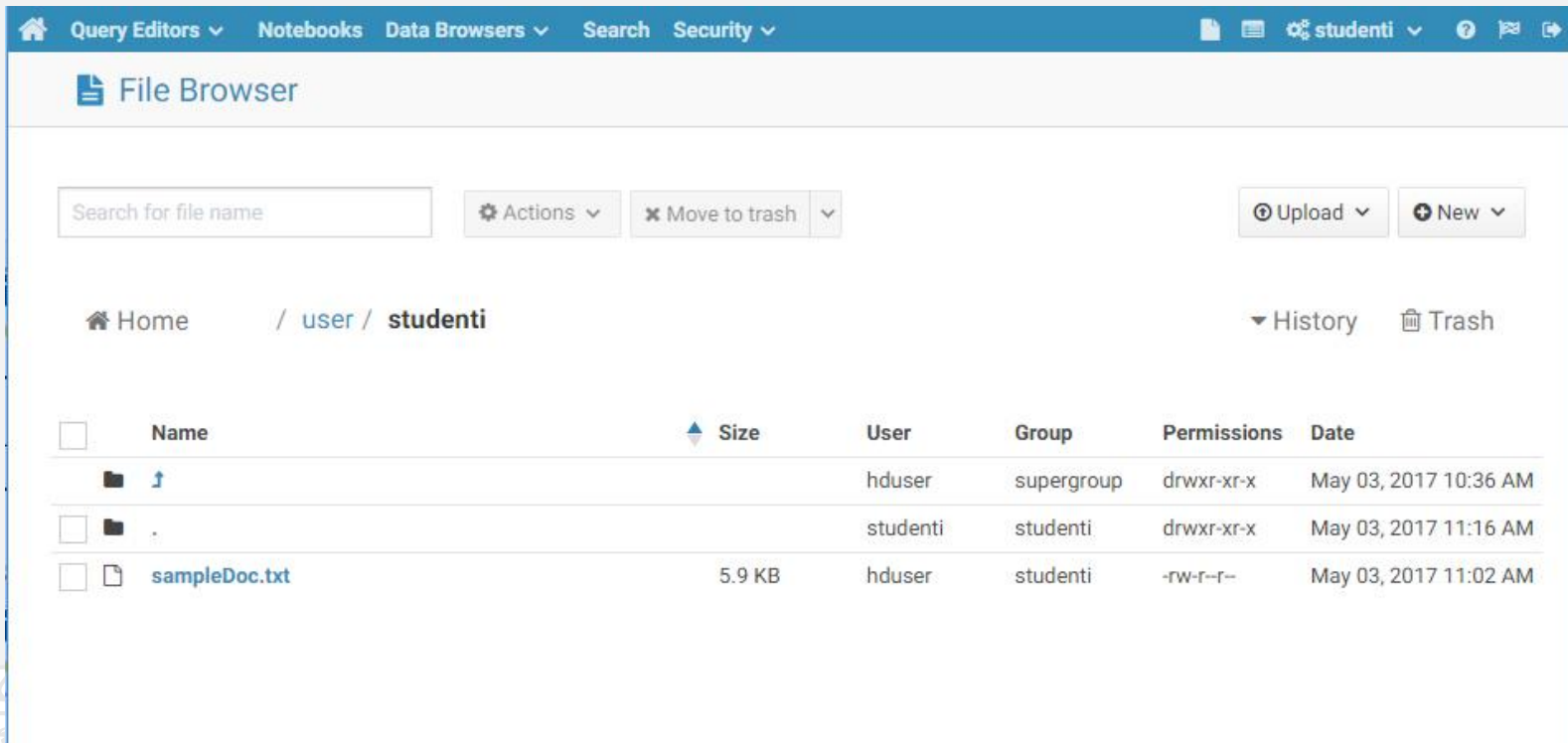
- Copy input text file `sampleDoc.txt` in HDFS: `hadoop fs -copyFromLocal <file_to_be_copied> <HDFS_Folder_Path>`

```
$ hadoop fs -copyFromLocal sampleDoc.txt /users/studenti/
```



Real Word Count Example

- Browsing HDFS Filesystem → <http://<dedicatedHueHostIP>:8000>



The screenshot shows the Hue File Browser interface. The breadcrumb path is `Home / user / studenti`. The file listing table is as follows:

<input type="checkbox"/>	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	<code>↑</code>		hduser	supergroup	drwxr-xr-x	May 03, 2017 10:36 AM
<input type="checkbox"/>	<code>.</code>		studenti	studenti	drwxr-xr-x	May 03, 2017 11:16 AM
<input type="checkbox"/>	<code>sampleDoc.txt</code>	5.9 KB	hduser	studenti	-rw-r--r--	May 03, 2017 11:02 AM

Real Word Count Example

- Execute the Word Count program (wc.jar) in HDFS:

```
hadoop jar <jarFile.jar> <input_File_HDFS_Path> <output_HDFS_Folder>
```

```
$ hadoop jar wordCount.jar /user/studenti/sampleDoc.txt /user/studenti/output
```

```
SSH Secure Shell 3.2.9 (Build 283)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.

Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-49-generic x86_64)

 * Documentation: https://help.ubuntu.com/

412 packages can be updated.
285 updates are security updates.

New release '16.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

You have new mail.
Last login: Wed May  3 12:50:42 2017 from 192.168.0.242
hduser@hadoop-pigpen:~$ cd /
hduser@hadoop-pigpen:/$ cd srv/hadoop/share/hadoop/mapreduce/
hduser@hadoop-pigpen:/srv/hadoop/share/hadoop/mapreduce$ hadoop jar wordCount.jar /user/studenti/sampleDoc.txt /user/studenti/output
```

Real Word Count Example

```


hduser@hadoop-pigpen:/srv/hadoop/share/hadoop/mapreduce$ hadoop jar wordCount.jar /user/studenti/sampleDoc.txt /user/studenti/output
17/05/03 13:36:49 INFO client.RMProxy: Connecting to ResourceManager at /192.168.0.98:8050
17/05/03 13:36:49 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
17/05/03 13:36:50 INFO input.FileInputFormat: Total input paths to process : 1
17/05/03 13:36:51 INFO mapreduce.JobSubmitter: number of splits:1
17/05/03 13:36:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1493383630745_0039
17/05/03 13:36:52 INFO impl.YarnClientImpl: Submitted application application_1493383630745_0039
17/05/03 13:36:52 INFO mapreduce.Job: The url to track the job: http://hadoop-pigpen:8080/proxy/application_1493383630745_0039/
17/05/03 13:36:52 INFO mapreduce.Job: Running job: job_1493383630745_0039
17/05/03 13:37:01 INFO mapreduce.Job: Job job_1493383630745_0039 running in uber mode : false
17/05/03 13:37:01 INFO mapreduce.Job: map 0% reduce 0%
17/05/03 13:37:47 INFO mapreduce.Job: map 100% reduce 0%
17/05/03 13:37:55 INFO mapreduce.Job: map 100% reduce 33%
17/05/03 13:37:56 INFO mapreduce.Job: map 100% reduce 100%
17/05/03 13:37:58 INFO mapreduce.Job: Job job_1493383630745_0039 completed successfully
17/05/03 13:37:58 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=5605
    FILE: Number of bytes written=442041
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=6164
    HDFS: Number of bytes written=5319
    HDFS: Number of read operations=12
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=6
  Job Counters
    Launched map tasks=1
    Launched reduce tasks=3
    Rack-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=42617
    Total time spent by all reduces in occupied slots (ms)=34252
    Total time spent by all map tasks (ms)=42617
    Total time spent by all reduce tasks (ms)=17126
    Total vcore-seconds taken by all map tasks=42617
    Total vcore-seconds taken by all reduce tasks=17126
    Total megabyte-seconds taken by all map tasks=87279616
    Total megabyte-seconds taken by all reduce tasks=70148096
  Map-Reduce Framework
    Map input records=15
    Map output records=806
    Map output bytes=9272
    Map output materialized bytes=5593
    Input split bytes=117
    Combine input records=806
    Combine output records=513
    Reduce input groups=513
    Reduce shuffle bytes=5593
    Reduce input records=513
    Reduce output records=513
    Spilled Records=1026
    Shuffled Maps =3
    Failed Shuffles=0
    Merged Map outputs=3
    GC time elapsed (ms)=1894

```



Real Word Count Example

- Monitoring Running Apps and Resources → <http://<masterNodeHostIP>:8080>



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
36	0	2	34	3	6 GB	40 GB	0 B	3	24	0	10	1	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	2	34	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1493383630745_0036	hduser	word count	MAPREDUCE	root.hduser	Wed, 03 May 2017 09:16:16 GMT	N/A	ACCEPTED	UNDEFINED	<input type="text"/>	ApplicationMaster	0
application_1493383630745_0035	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Wed, 03 May 2017 03:39:59 GMT	Wed, 03 May 2017 03:41:36 GMT	FINISHED	SUCCEEDED	<input type="text"/>	History	N/A
application_1493383630745_0034	hduser	Graphchannel	MAPREDUCE	root.Data Analytics.graphchannel	Tue, 02 May 2017 22:30:25 GMT	N/A	RUNNING	UNDEFINED	<input type="text"/>	ApplicationMaster	0
application_1493383630745_0033	hduser	GraphchannelRetweet	MAPREDUCE	root.Data Analytics.graphchannel	Tue, 02 May 2017 22:30:25 GMT	Wed, 03 May 2017 03:39:56 GMT	FINISHED	SUCCEEDED	<input type="text"/>	History	N/A
application_1493383630745_0032	hduser	GraphKeywords	MAPREDUCE	root.Data Analytics.graphkeywords	Tue, 02 May 2017 22:30:24 GMT	Wed, 03 May 2017 00:27:04 GMT	FINISHED	SUCCEEDED	<input type="text"/>	History	N/A
application_1493383630745_0031	hduser	GraphKeywordsRetweet	MAPREDUCE	root.Data Analytics.graphkeywords	Tue, 02 May 2017 22:30:24 GMT	Wed, 03 May 2017 00:28:40 GMT	FINISHED	SUCCEEDED	<input type="text"/>	History	N/A

Real Word Count Example

- Monitoring Running Apps and Resources → <http://<dedicatedHueHostIP>:8000/jobbrowser>

Logs	ID	Name	Application Type	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted
	1493383630745_0039	word count	MAPREDUCE	RUNNING	hduser	5%	5%	root.hduser	N/A	25s	05/03/17 13:36:52

JOB ID word count
1493383630745...

TYPE Attempts Tasks Metadata Counters

MR2

USER hduser

STATUS SUCCEEDED

LOGS

MAPS 100%

REDUCES 100%

DURATION 56s

Recent Tasks

Logs	Tasks	Type
	task_1493383630745_0039_m_000000	MAP
	task_1493383630745_0039_r_000000	REDUCE
	task_1493383630745_0039_r_000001	REDUCE
	task_1493383630745_0039_r_000002	REDUCE



Real Word Count Example

- Browsing the Output in HDFS:

File Browser

Search for file name

Actions Move to trash Upload New

Home / user / studenti

Name	Size	User	Group	Permissions	Date
↑		hduser	supergroup	drwxr-xr-x	May 03, 2017 10:36 AM
.		studenti	studenti	drwxr-xr-x	May 03, 2017 12:10 PM
output		hduser	studenti	drwxr-xr-x	May 03, 2017 12:10 PM
sampleDoc.txt	5.9 KB	hduser	studenti	-rw-r--	May 03, 2017 11:02 AM

File Browser

Search for file name

Actions Move to trash Upload New

Home / user / studenti / output

Name	Size	User	Group	Permissions	Date
↑		studenti	studenti	drwxr-xr-x	May 03, 2017 12:10 PM
.		hduser	studenti	drwxr-xr-x	May 03, 2017 12:10 PM
._SUCCESS	0 bytes	hduser	studenti	-rw-r--	May 03, 2017 12:10 PM
part-r-00000	1.6 KB	hduser	studenti	-rw-r--	May 03, 2017 12:10 PM
part-r-00001	2.0 KB	hduser	studenti	-rw-r--	May 03, 2017 12:10 PM
part-r-00002	1.6 KB	hduser	studenti	-rw-r--	May 03, 2017 12:10 PM



Real Word Count Example

The screenshot shows a web-based file browser interface. The breadcrumb path is `/ user / studenti / output / part-r-00000`. The file content is displayed as a list of words and their counts:

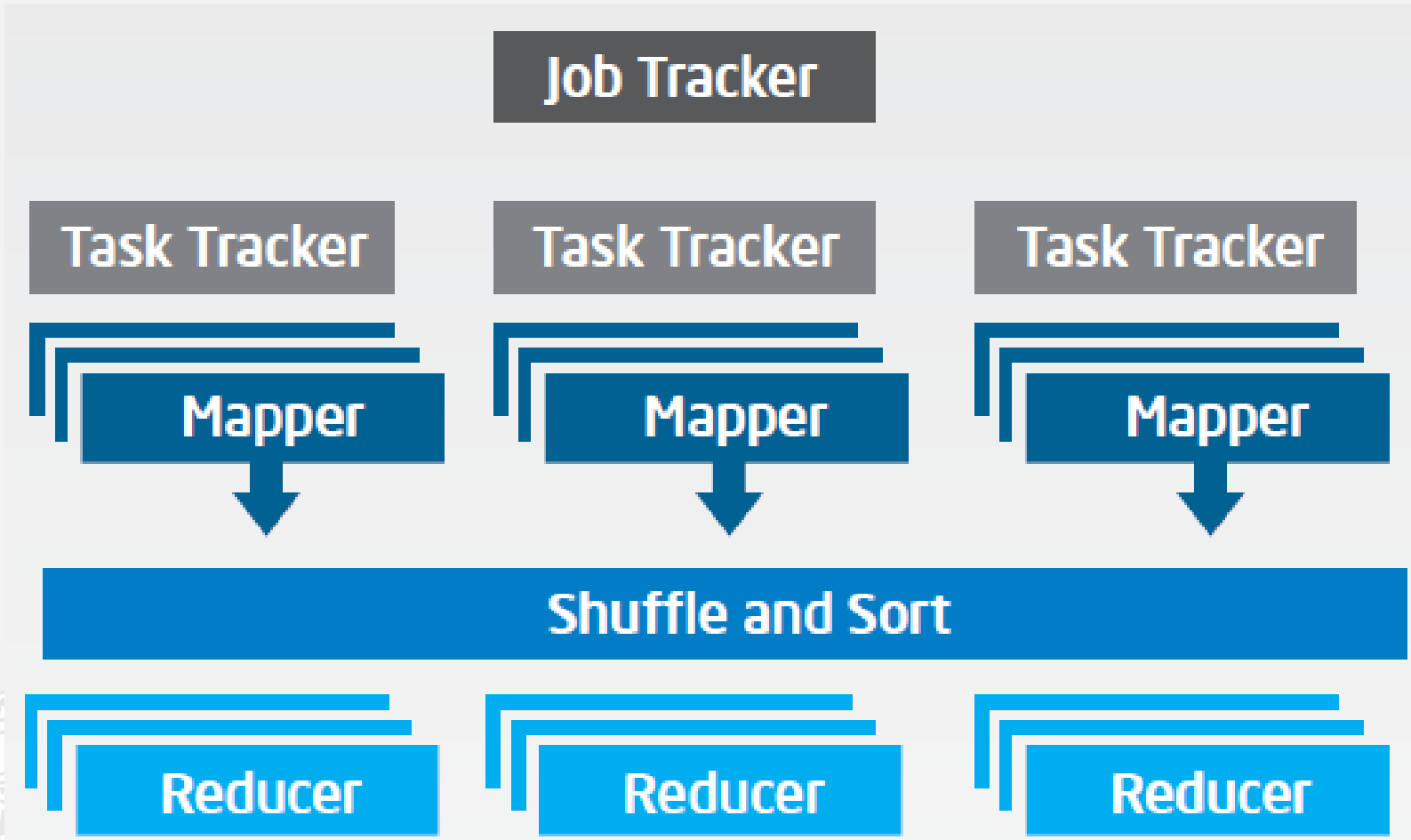
```
(Parte 9
(requisiti, 1
(versione 6
0, 1
1cII, 1
2.0) 1
2.9) 1
3, 1
3.8) 2
4b, 1
AXMEDIS 1
Adapter; 2
Applicazioni 1
Architetture 1
Asincrona; 1
CORBA 3
CORBA; 6
Call 2
Caratteristiche: 1
Client 2
Cloud 1
Cloud; 1
Comunicazione; 1
Confronto 1
```

On the left side of the browser, there is a metadata table for the file:

Last modified	User	Group	Size	Mode
03/05/2017 10:10	hduser	studenti	1.58 KB	100644

Logical Architecture

Processing: MapReduce (Hadoop v1)



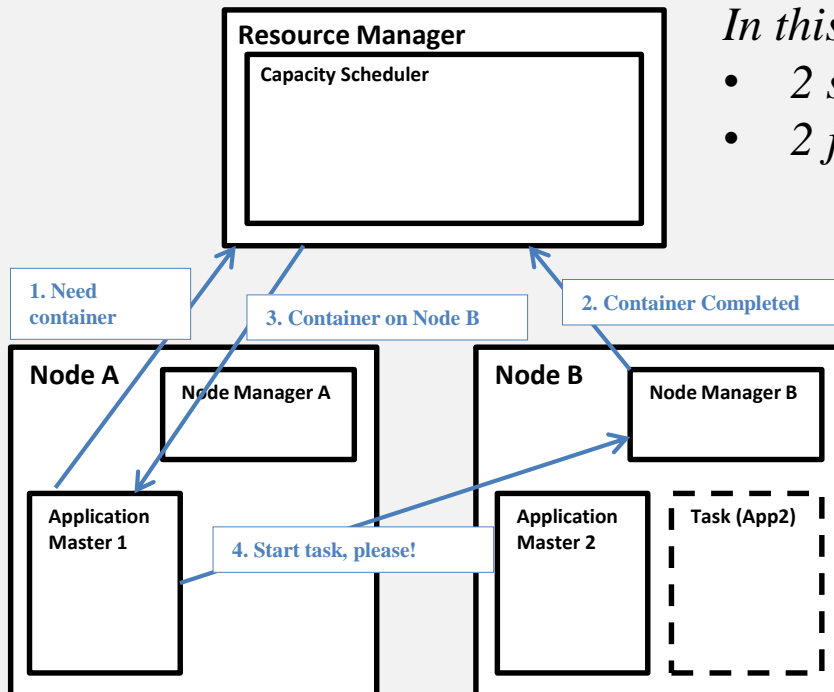
YARN Scheduler

- Used in Hadoop 2.x +
- YARN = Yet Another Resource Negotiator
- Treats each server as a collection of *containers*
 - Container = fixed CPU + fixed memory
- Has 3 main components
 - Global *Resource Manager (RM)*
 - Scheduling
 - Per-server *Node Manager (NM)*
 - Daemon and server-specific functions
 - Per-application (job) *Application Master (AM)*
 - Container negotiation with RM and NMs
 - Detecting task failures of that job

YARN: How a job gets a container

In this figure

- 2 servers (A, B)
- 2 jobs (1, 2)



What really going on !?

- How many maps are issued?
- Where are computed the maps?
- Does my job executing normally or something is going wrong !?
- Does my cluster being underutilized or overutilized !?



Basic web interface



Logged in as: dr.who

All Applications

- Cluster
- About Nodes
- Applications
 - NEW
 - NEW SAVING
 - SUBMITTED
 - ACCEPTED
 - RUNNING
 - FINISHED
 - FAILED
 - KILLED
- Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
28	0	0	28	0	0 B	40 GB	0 B	0	24	0	10	1	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	28	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1493383630745_0028	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Tue, 02 May 2017 09:45:43 GMT	Tue, 02 May 2017 09:46:56 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0027	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Tue, 02 May 2017 05:55:16 GMT	Tue, 02 May 2017 05:57:21 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0026	hduser	GraphchannelRetweet	MAPREDUCE	root.Data Analytics.graphchannel	Mon, 01 May 2017 22:30:43 GMT	Tue, 02 May 2017 09:45:40 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0025	hduser	Graphchannel	MAPREDUCE	root.Data Analytics.graphchannel	Mon, 01 May 2017 22:30:42 GMT	Tue, 02 May 2017 05:55:01 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0024	hduser	GraphKeywords	MAPREDUCE	root.Data Analytics.graphkeywords	Mon, 01 May 2017 22:30:43 GMT	Mon, 01 May 2017 23:50:11 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0023	hduser	GraphKeywordsRetweet	MAPREDUCE	root.Data Analytics.graphkeywords	Mon, 01 May 2017 22:30:42 GMT	Mon, 01 May 2017 23:34:41 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0022	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Mon, 01 May 2017 03:53:31 GMT	Mon, 01 May 2017 03:54:56 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0021	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Mon, 01 May 2017 00:38:08 GMT	Mon, 01 May 2017 00:38:54 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0020	hduser	GraphchannelRetweet	MAPREDUCE	root.Data Analytics.graphchannel	Sun, 30 Apr 2017 22:30:54 GMT	Mon, 01 May 2017 00:38:05 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0019	hduser	Graphchannel	MAPREDUCE	root.Data Analytics.graphchannel	Sun, 30 Apr 2017 22:30:54 GMT	Mon, 01 May 2017 03:53:26 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0018	hduser	GraphKeywordsRetweet	MAPREDUCE	root.Data Analytics.graphkeywords	Sun, 30 Apr 2017 22:30:54 GMT	Sun, 30 Apr 2017 23:17:24 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0017	hduser	GraphKeywords	MAPREDUCE	root.Data Analytics.graphkeywords	Sun, 30 Apr 2017 22:30:54 GMT	Sun, 30 Apr 2017 23:28:13 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0016	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Sun, 30 Apr 2017 07:28:57 GMT	Sun, 30 Apr 2017 07:30:00 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1493383630745_0015	hduser	GraphChannel Phase 2	MAPREDUCE	root.hduser	Sun, 30 Apr 2017 06:25:35 GMT	Sun, 30 Apr 2017 06:26:22 GMT	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A

Advanced web interface

HUE Query Editors ▾ Notebooks Data Browsers ▾ Search Security ▾ File Browser Job Browser admin ▾

Job Browser

1493383630745_0019	Graphchannel	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphchannel	N/A	5h:22m:32s	05/01/17 00:30:54
1493383630745_0020	GraphchannelRetweet	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphchannel	N/A	2h:7m:11s	05/01/17 00:30:54
1493383630745_0017	GraphKeywords	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphkeywords	N/A	57m:19s	05/01/17 00:30:54
1493383630745_0018	GraphKeywordsRetweet	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphkeywords	N/A	46m:30s	05/01/17 00:30:54
1493383630745_0016	GraphChannel Phase 2	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.hduser	N/A	1m:2s	04/30/17 09:28:57
1493383630745_0015	GraphChannel Phase 2	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.hduser	N/A	47s	04/30/17 08:25:35
1493383630745_0014	GraphchannelRetweet	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphchannel	N/A	8h:58m:28s	04/30/17 00:30:26
1493383630745_0013	Graphchannel	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphchannel	N/A	7h:55m:6s	04/30/17 00:30:25
1493383630745_0011	GraphKeywords	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphkeywords	N/A	1h:51m:43s	04/30/17 00:30:25
1493383630745_0012	GraphKeywordsRetweet	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.Data Analytics.graphkeywords	N/A	1h:49m:15s	04/30/17 00:30:25
1493383630745_0010	GraphChannel Phase 2	MAPREDUCE	SUCCEEDED	hduser	100%	100%	root.hduser	N/A	1m:14s	04/29/17 09:44:18

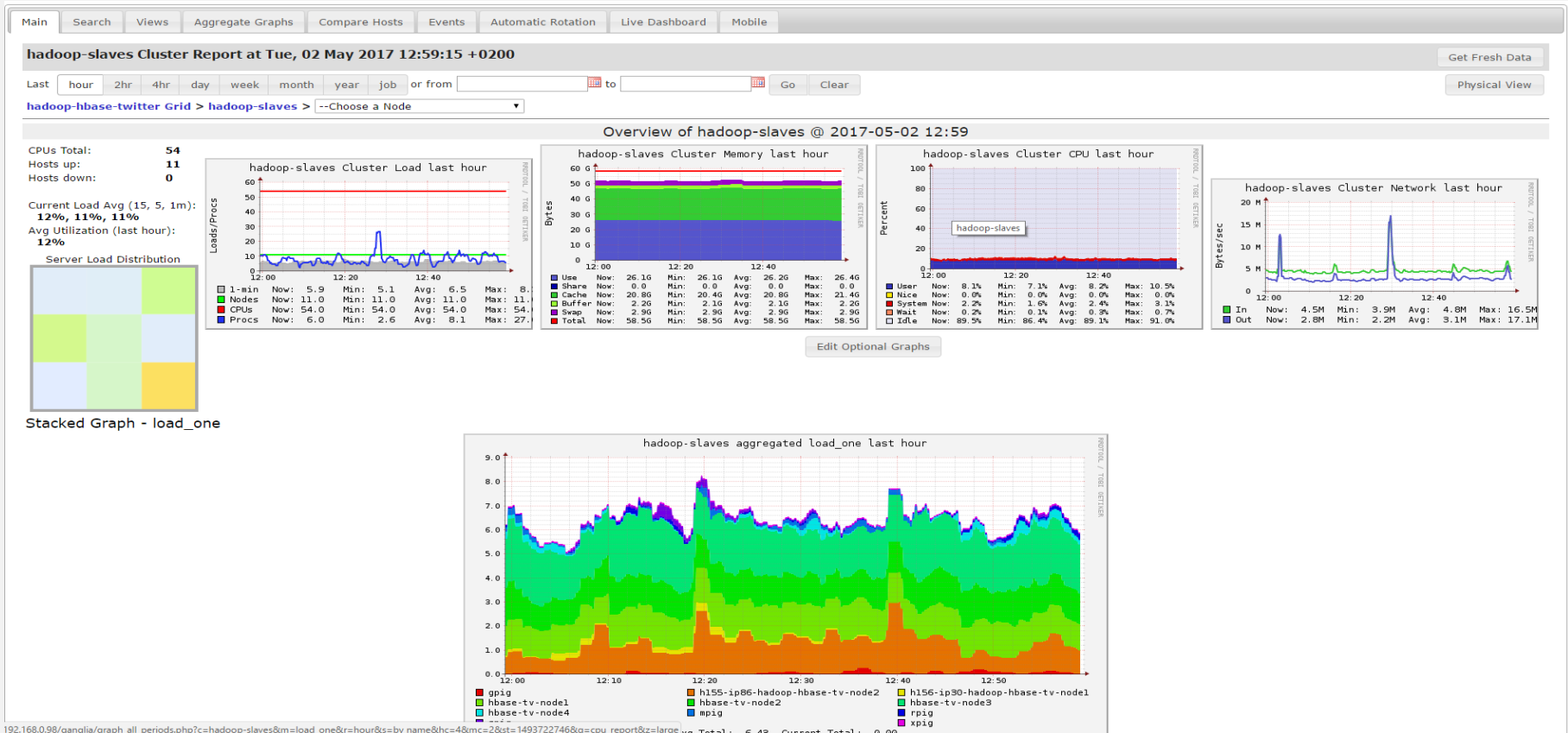


What really really going on !?

- The aforementioned application doesn't give specific information ...
- Unix tools
- Batch System tools
- Really need something that can provide a quick visual overview of the health and load on your cluster



Ganglia



Host level detail

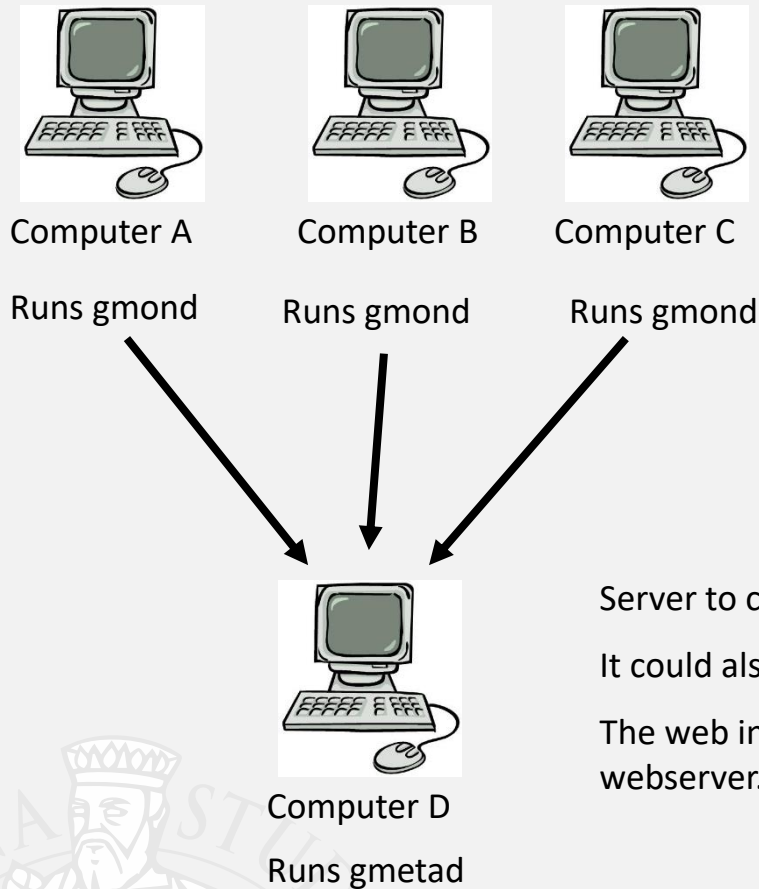


How does Ganglia work?

- Ganglia works through a small agent, *gmond*, on each node or machine to be monitored. You can distribute a single *gmond* instance to lots of machines at once. *Gmonds* communicate the state of their local node to a machine running a Master *gmetad* instance.
- The server uses RRDtool to store the data over time
- The Ganglia framework can be extended to monitor many parameters.

Setup

The software can be downloaded from <http://ganglia.sourceforge.net/>



Clients just have to run gmond,
which is configured by
`/etc/gmond.conf`

Server to collect the data runs gmetad.
It could also run gmond to monitor itself.
The web interface needs to run on a
webserver.



Computer D
gmetad
gmond
httpd

Client Setup



Computer A

Runs gmond



Computer B

Runs gmond



Computer C

Runs gmond

Apt-get install ganglia-gmond

edit config file

service gmond start

chkconfig gmond on

/etc/gmond.conf extracts

```
cluster {
  name = "LCG Workers"
}
/* Feel free to specify as many udp_send_channels as you like. Gmond
  used to only support having a single channel */
udp_send_channel {
  mcast_join = 239.2.11.95
  port = 8649
}
udp_send_channel {
  port = 8649
  host = pplxconfig
}
/* You can specify as many udp_rcv_channels as you like as well. */
udp_rcv_channel {
  mcast_join = 239.2.11.95
  port = 8649
  bind = 239.2.11.95
}
/* You can specify as many tcp_accept_channels as you like to share
  an xml description of the state of the cluster */
tcp_accept_channel {
  port = 8649
}
```

Server Setup



Computer D

gmetad

gmond

httpd

Extracts from `/etc/gmetad.conf`

```
data_source "LCG Workers" computerA.physics.ox.ac.uk ComputerB.physics.ox.ac.uk  
computerC.physics.ox.ac.uk
```

```
data_source "LCG Servers" t2se01.physics.ox.ac.uk:8656  
t2ce02.physics.ox.ac.uk:8656 gridlogger.physics.ox.ac.uk:8656
```

Aptitude install ganglia-gmond ganglia-gmetad ganglia-web

edit `/etc/gmond.conf`

edit `/etc/gmetad.conf`



HBASE

Introduction

- Hadoop is a framework that supports operations on a large amount of data.
- Hadoop includes the Hadoop Distributed File System (HDFS)
- HDFS does a good job of storing large amounts of data, but lacks quick random read/write capability.

Introduction (cont.)

- We need a tabular form of storing our data
- The storing system must preserve the advantages of hadoop
 - Fault tolerance
 - High performance on large amount of data



Introduction (Cont.)

- HBase is an open source, sparse, consistent distributed, sorted map modeled after Google's BigTable.
- Developed as part of Apache's Hadoop project and runs on top of Hadoop Distributed File System.

Conceptual View

- A data row has a sortable row key and an arbitrary number of columns.
- A Time Stamp is designated automatically if not artificially.
- <family>:<label>

Hbase table

HBase Browser

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: col2 to col3] {Filter1()} AND Filter

`<family>:<label>`

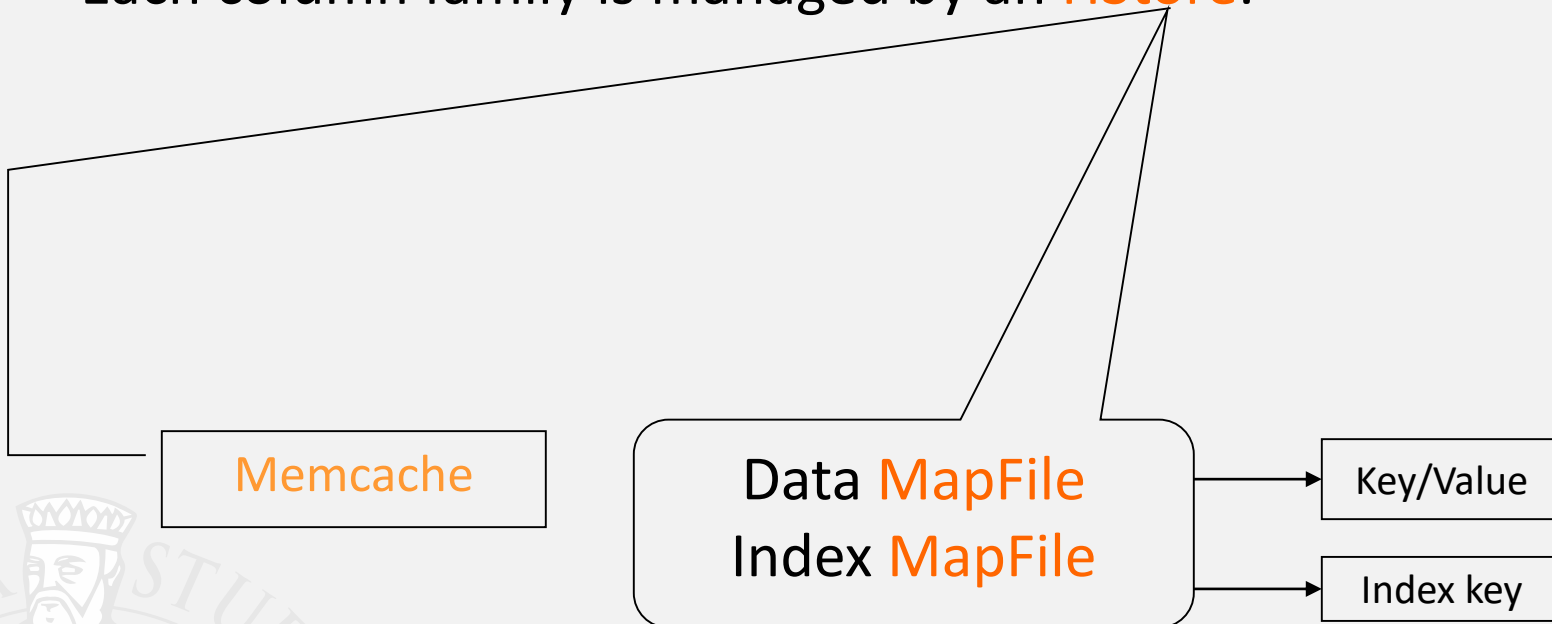
twitter_info: retweetCount	twitter_info: sqLid	twitter_info: mentions	twitter_info: lang	twitter_info: place	twitter_info: publicationTime	twitter_info: time_zone	twitter_info: hashtagsOnTwitter	twitter_info: message	twitter_info: geo_long	twitter_info: locationUser	twitter_info: favoriteCount	twitter_info: ...
pt	#Lurdinhasex1 http://t.co/vn9dgtL91Q6	0		0	607565246186799184	1	carlos	caldo	BR	Greenland		2015
pt	(x) pergunta no grupo da universidade	0		2	609854356561205160	3	felipeanchieta_	caldo	BR	Brasilia		2015
es	#Framando	0		0	600847996499206144	0	TINCHOROMEROOK	#empire	UY	Buenos Aires	@foxlife_ar	2015
in	#Empire	0		0	601702468565643264	0	rayn_christ	#empire	ID	Jakarta		2015
			en	e				Guadalajara				
it	sei troppo arguto	0		0	615195778613669888	3	claudiorossit	caldo				2015



Let's try it !

Physical Storage View

- Physically, tables are stored on a per-column family basis.
- Empty cells are not stored in a column-oriented storage format.
- Each column family is managed by an **HStore**.



Hbase table

HStore

<family>:<label>

HBase Browser

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: col2 to col3] {Filter1()} AND Filter 2

twitter_info: retweetCount	twitter_info: sqLid	twitter_info: mentions	twitter_info: lang	twitter_info: place	twitter_info: publicationTime	twitter_info: time_zone	twitter_info: hashtagsOnTwitter	twitter_info: message	twitter_info: geo_long	twitter_info: locationUser	twitter_info: favoriteCount	twitter_info: retweetCount
pt	#Lundinhasex1 htt p://t.co/vn9dgtL91Qe	0		0	607565246186799104	1	lcfcarlos	caldo	BR	Greenland		2015
pt	(x) pergunta no gru po da universidade	0		2	609854956561756160	3	felipeanchieta_	caldo	BR	Brasilia		2015
es	#Tramando	0		0	600847996499206144	0	TINCHROMEROOK	#empire	UY	Buenos Aires	@foxlife_ar	2015
in	#Empire	0		0	60178246855643264	0	rayn_christ	#empire	ID	Jakarta		2015
			en	0				Guadalajara				
it	- sei troppo arguto	0		0	615195778613669888	3	claudiorossit	caldo				2015

Row Ranges: Regions

- Row key/ Column ascending, Timestamp descending
- Physically, tables are broken into row ranges contain rows from start-key to end-key

What we can do ?

- Mysql commands
 - Create table
 - Drop table
 - Insert data in table
 - Delete data
 - Query data
- Mysql interface (cli, API)



Hbase commands

- Table manipulations
 - create 't1', 'f1', 'f2', 'f3'
 - drop 't1' (but t1 must be disabled)
 - Disable/enable 't1'



More commands

- put 't1', 'r1', 'c1', 'value', ts1
- delete 't1', 'r1', 'c1'
- delete 't1', 'r1', 'c1', 'ts1'
- get 't1', 'r1'
- get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
- get 't1', 'r1', {COLUMN => 'c1'}
- Scan 't1'

HBASE Client

- Cli : hbase
- Java API



Trace

- create 'student', cf=['a','b']
- put 'student', 'pippo', 'a:name', 'pippo'
- put 'student', 'pippo', 'a:age', 20
- put 'student', 'pluto', 'a:name', 'pluto'
- scan 'student'

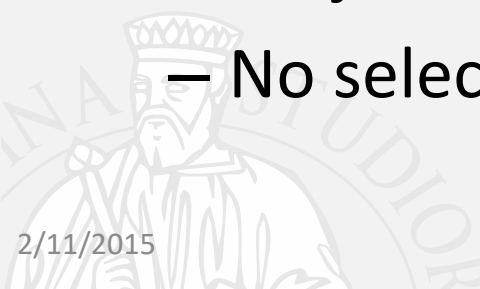




Let's try it !

Missing !?

- Mysql is sql complaint
 - To search data we issue commands like
 - «select columns where conditions >>
 - Generally we take advantage of joining table
- HBASE is essentially an hashmap
 - No join of tables
 - No select



Filters

- Filters are Java classes restricting matches;
- Filter list: combines multiple filters with AND and OR
- Compare values of one or multiple columns
 - Smaller, equal, greater, substring, prefix,
- Compare metadata: column family and qualifier
 - Qualifier prefix filter: Return (first few) matching columns
 - Column range filter: return a slice of columns (e.g. bb-bz)
- Compare names of rows Note: it is preferable to use scan options

Scan advanced

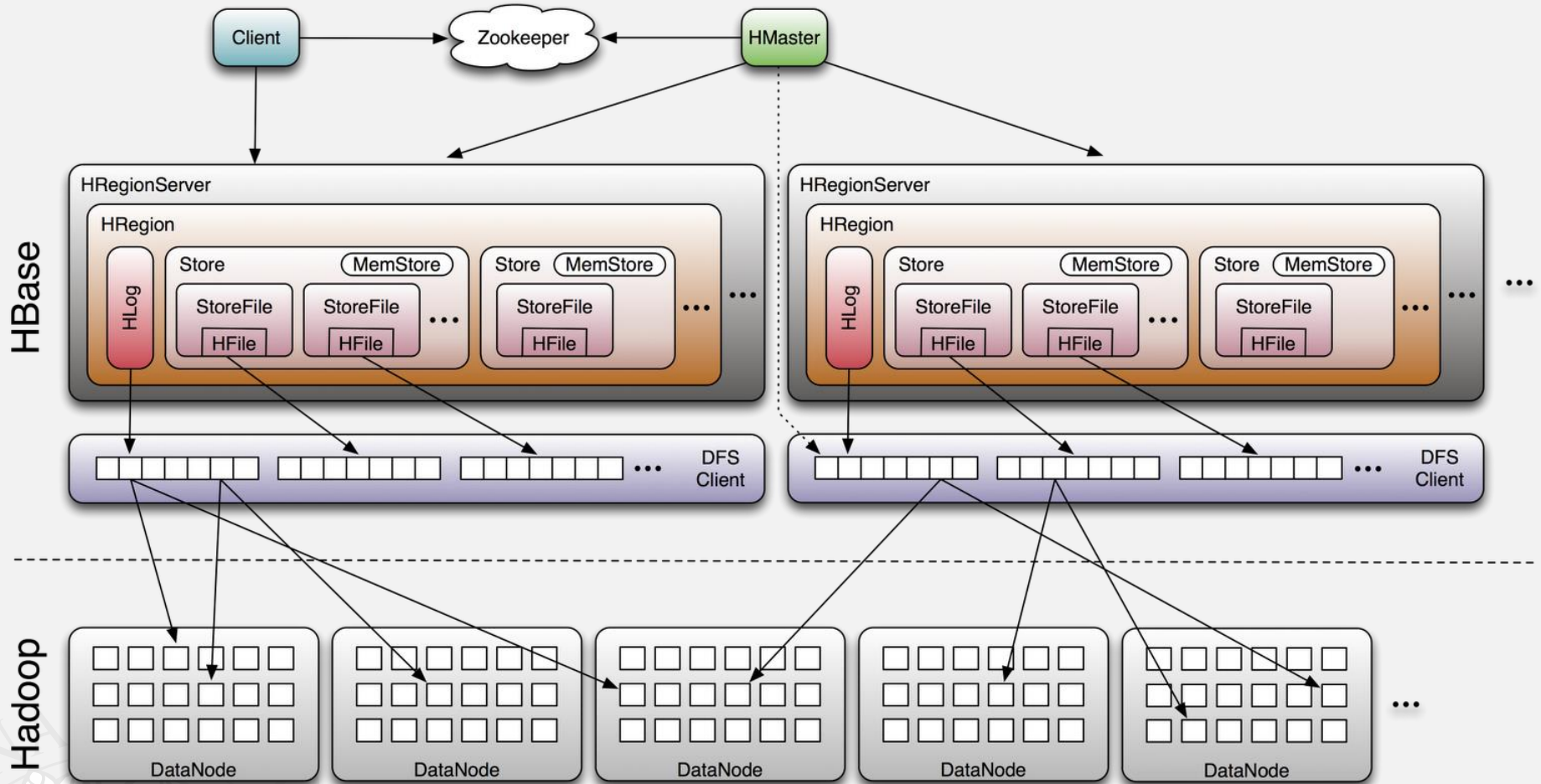
- scan 'student',{ FILTER => "KeyOnlyFilter()"}
- scan 'student',{ FILTER => "(PrefixFilter ('pi')) AND MultipleColumnPrefixFilter('a','b') AND ColumnCountGetFilter(2)" }
- scan 'student',{ COLUMNS=>['a:age'], FILTER => "SingleColumnValueFilter('a','age',>,'binary:19')"}



HBase Components

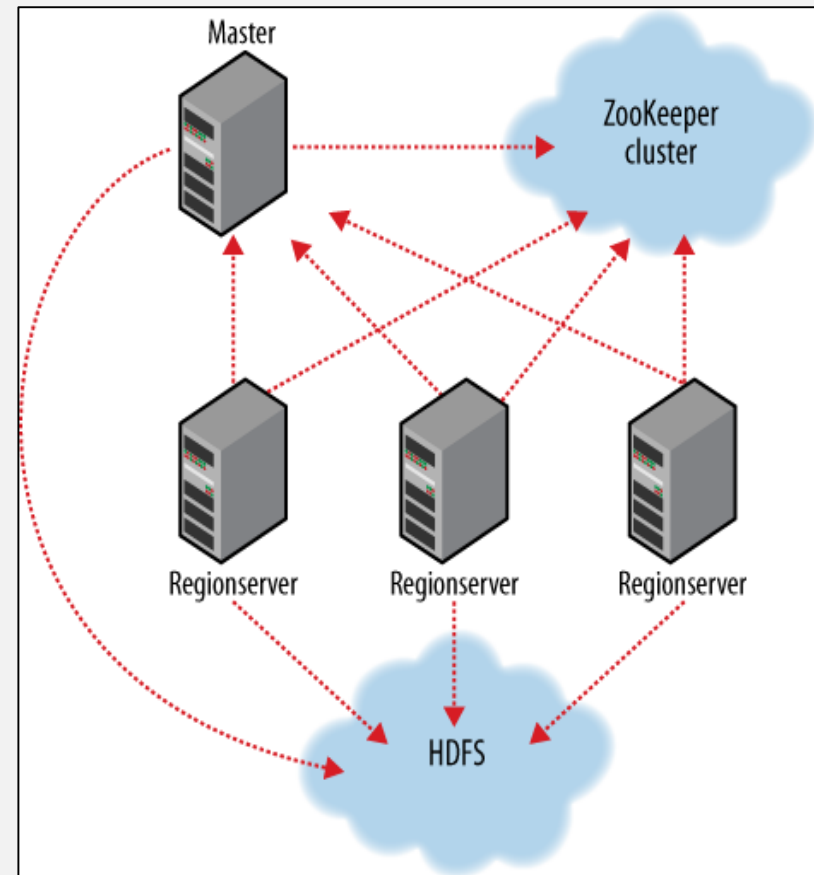
- **Region**
 - A subset of a table's rows, like horizontal range partitioning
 - Automatically done
- **RegionServer (many slaves)**
 - Manages data regions
 - Serves data for reads and writes (*using a log*)
- **Master**
 - Responsible for coordinating the slaves
 - Assigns regions, detects failures
 - Admin functions

Big Picture



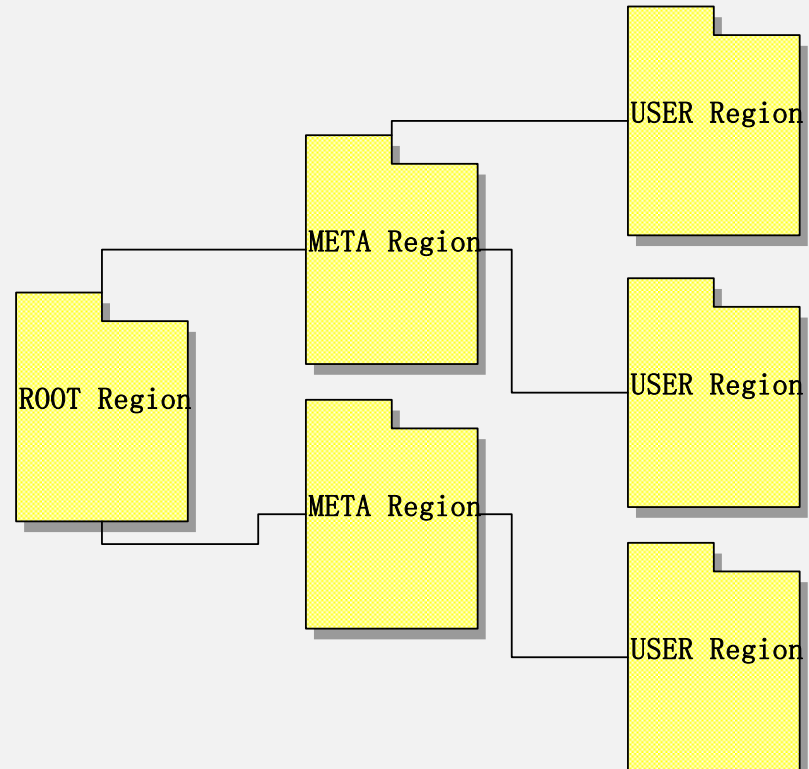
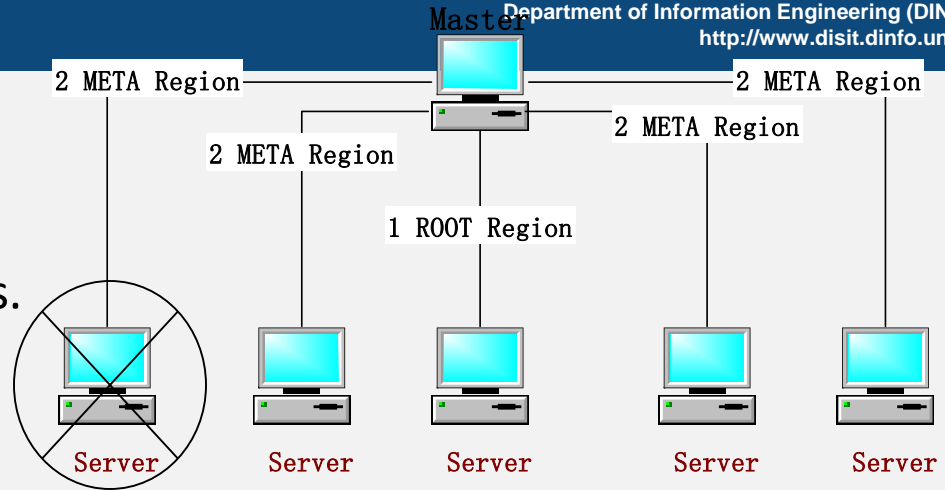
ZooKeeper

- HBase depends on ZooKeeper
- By default HBase manages the ZooKeeper instance
 - E.g., starts and stops ZooKeeper
- HMaster and HRegionServers register themselves with ZooKeeper



HBaseMaster

- Assign regions to HRegionServers.
1. ROOT region locates all the META regions.
 2. META region maps a number of user regions.
 3. Assign user regions to the HRegionServers.
- Enable/Disable table and change table schema
 - Monitor the health of each Server



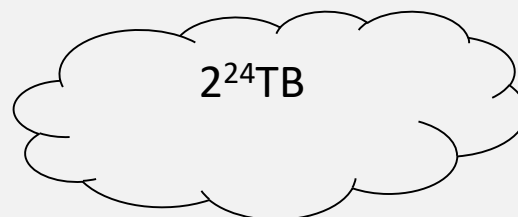
ROOT/META Table

- Each row in the ROOT and META tables is approximately 1KB in size. At the default size of 256MB.

$$1ROOTtable = 2^{18} METAregions$$

$$= 2^{18} \times 2^{18} USERregions$$

$$= 2^{54} KB = 2^{64} bytes$$



HRegionServer

write



- Write Requests
- Read Requests
- Cache Flushes
- Compactions
- Region Splits

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t9		"anchor:cnn.com"	"CNN"
	t8		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

Mapfile1.1
Mapfile1.2

Memcache1

Memcache2

Hstore1

Hstore2

HRegionServer



- Write Requests
- **Read Requests**
- Cache Flushes
- Compactions
- Region Splits

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t9		"anchor:cnn.com"	"CNN"
	t8		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

Mapfile1.1
Mapfile1.2

Memcache1

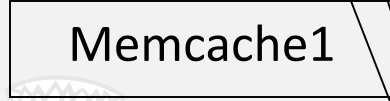
Hstore1

HRegionServer

Cache



- Write Requests
- Read Requests
- **Cache Flushes**
- Compactions
- **Region Splits**



Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t9		"anchor:cnn.com"	"CNN"
	t8		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

HRegionServer

Compactions

- Write Requests
- Read Requests
- Cache Flushes
- **Compactions**
- Region Splits

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t9		"anchor:cnn.com"	"CNN"
	t8		"anchor:my.look.cnn.com"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

Mapfile1
Mapfile1.2

Memcache1

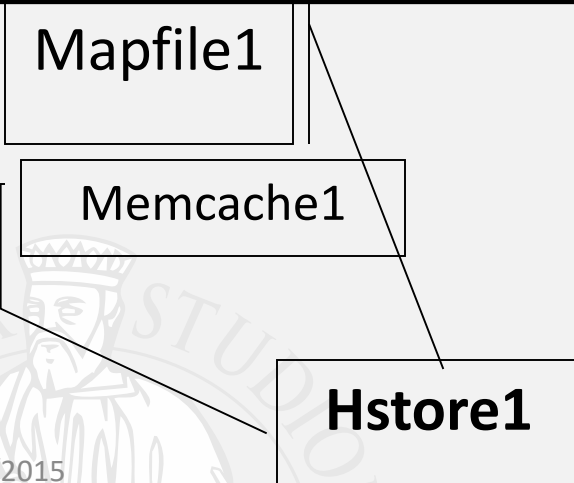
Hstore1

HRegionServer

Region Splits

- Write Requests
- Read Requests
- Cache Flushes
- Compactions
- **Region Splits**

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t9		"anchor:cnn.com"	"CNN"
	t8		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		





Case Studio

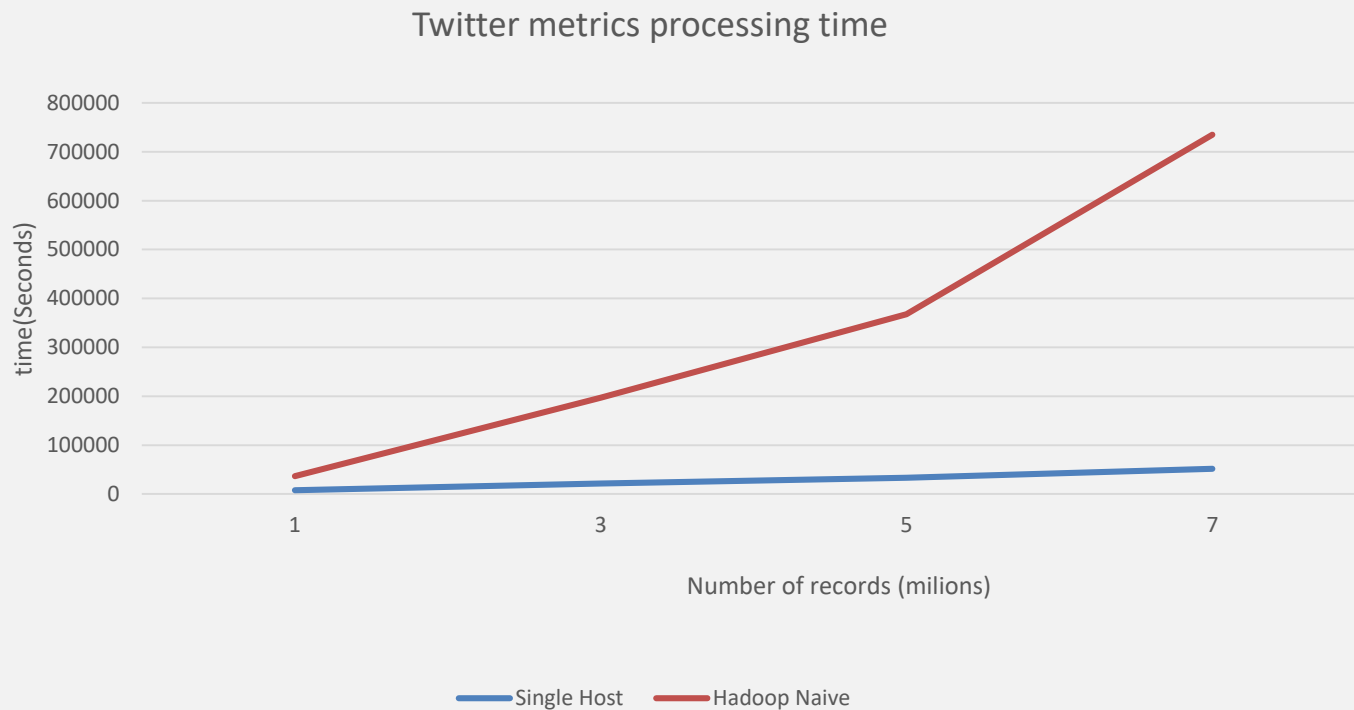
Solution

- Migrating to a distributed architecture
 - We have to rethink the problem in parallel way

Sequential minded approach

- For each search execute a MR job
 - If a MR job takes 1 minute (ideally) time and i have to compute for 1000 -> i have to wait 1000 minute !

Single Host vs Hadoop

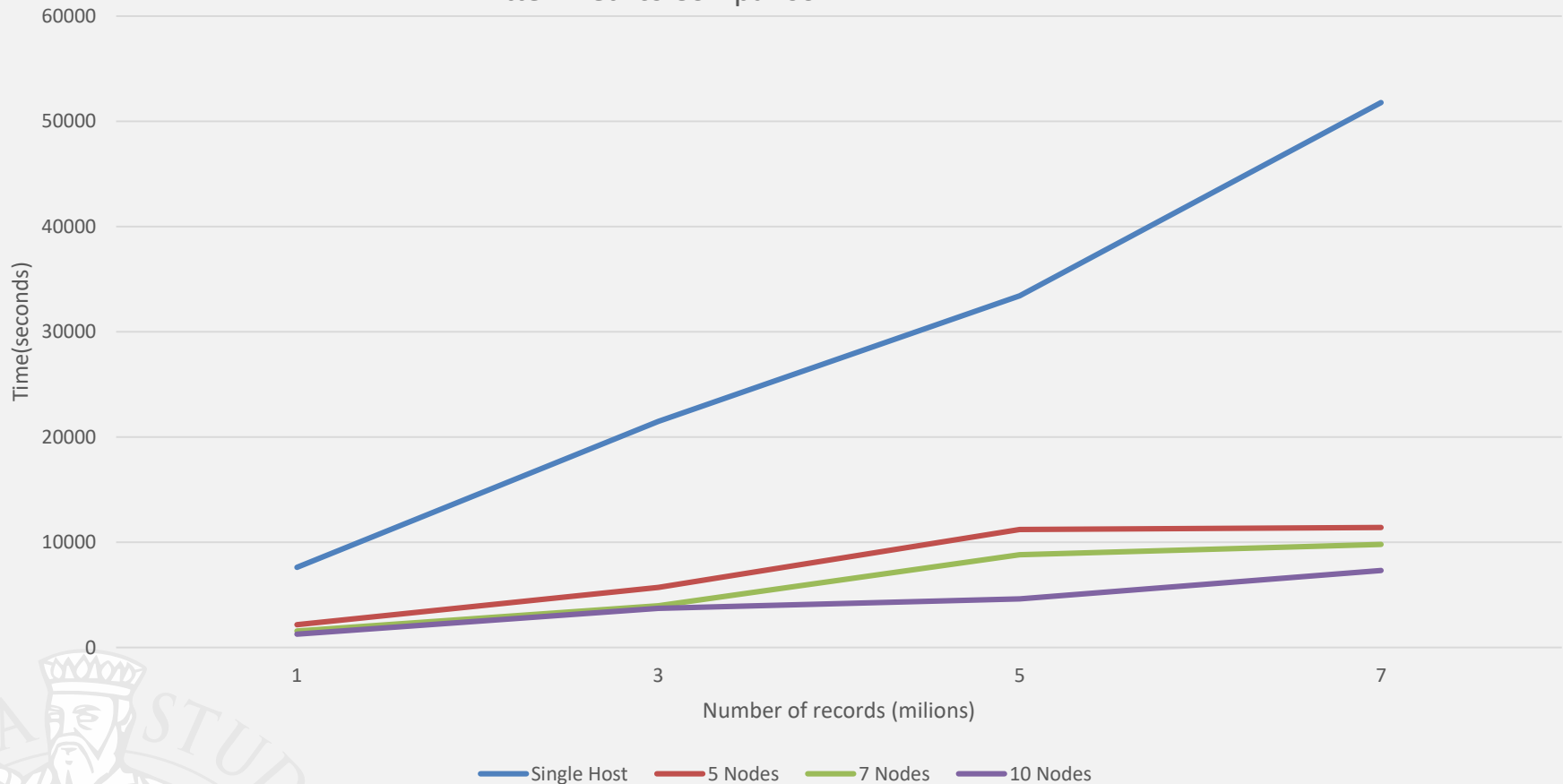


Parallel-computing minded approach

- Scan all records and counts each word sorting per day
- As final step consider only the words involved in our search

Single Host vs Hadoop

Twitter Metrics Comparison



All Quiet on the Western Front ...

Google calls it:	Hadoop equivalent:
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase
Chubby	Zookeeper

