TRAINING

Using BigData for the analysis of historic context information

Francisco Romero Comos BigData

FIWARE

Open APIs for Open Minds

Using Big Data for the analysis of historic context information

Francisco Romero Bueno
Technological Specialist. FIWARE data engineer
francisco.romerobueno@telefonica.com

FIWARE

Big Data:

What is it and how much data is there

What is big data?

FIWARE

What is big data?

http://commons.wikimedia.org/wiki/File:Interior_view_of_Stockholm_Public_Library.jpg

FIWARE

Not a matter of thresholds

If both the data used by your app and the processing capabilities your app logic needs fit the available infrastructure, then ***you are not dealing with a big data problem***

If either the data used by your app either the processing capabilities your app logic needs don't fit the available infrastructure, ***then you are facing a big data problem****, and you need specialized services*

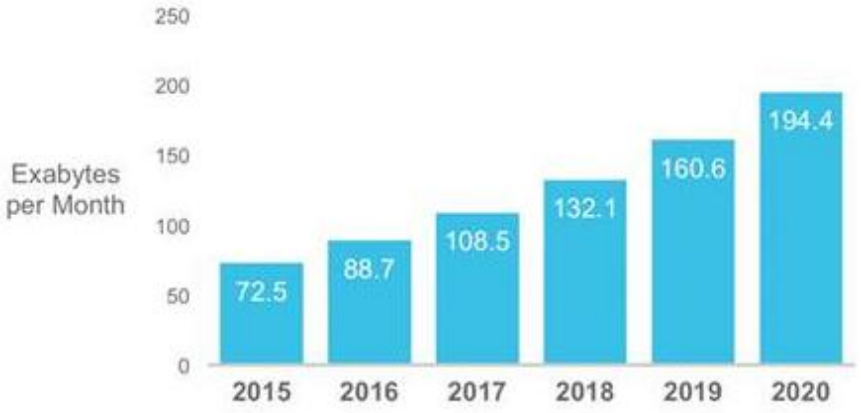
FIWARE

How much data is there?

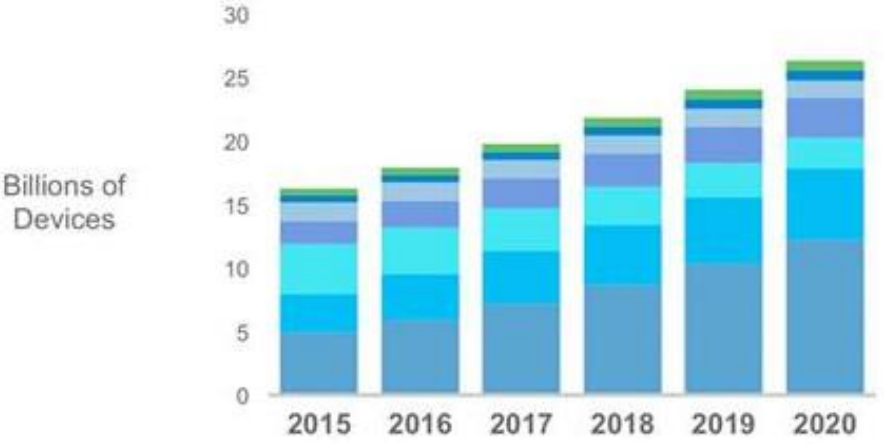
FIWARE

Data growing forecast

http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

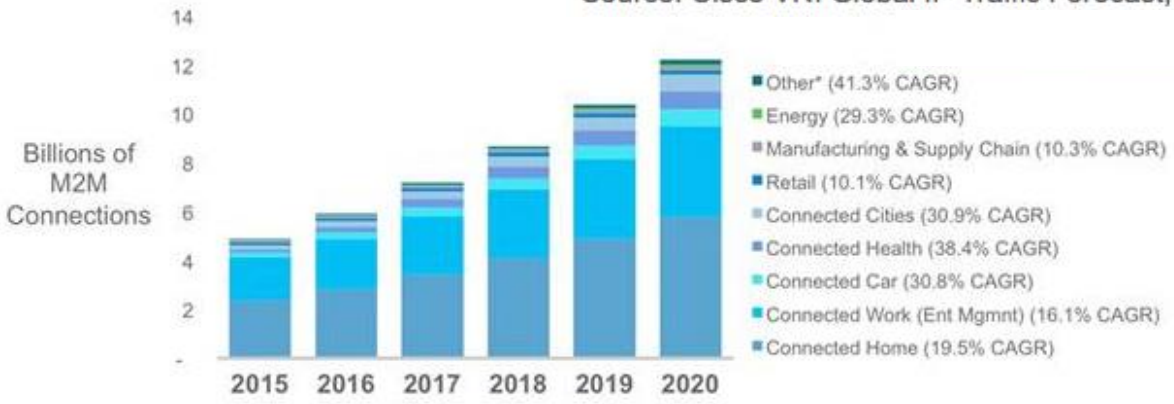


Source: Cisco VNI Global IP Traffic Forecast, 2015–2020



Figures (n) refer to 2015, 2020 device share.

Source: Cisco VNI Global IP Traffic Forecast, 2015–2020



* Other includes Agriculture, Construction, and Emergency Services.

Source: Cisco VNI Global IP Traffic Forecast, 2015–2020

Two (three) approaches for dealing with Big Data:

Batch and stream processing (and Lambda architectures)

Batch processing

- It is about joining a lot of data (batching)
  - A *lot* may mean Terabytes or more…
  - Most probably, data cannot be stored in a single server
- Once joined, it is analyzed
  - Most probably, aata cannot be analyzed using a single process
- Time is not a problem
  - Batching can last for days or even months
  - Processing can last for hours or even days
- Analysis can be reproduced

Stream processing

- It is about not storing the data and analyzing it on the fly
  - Most probably, data cannot be analyzed by a single process
- Time is important
  - Since the data is not stored, it must be analyzed as it is received
  - The results are expected to be available in near real-time
- Analysis cannot be reproduced

Lambda architectures

- A Big Data architecture is Lambda compliant if it produces near-real time data insights based on the last data only while large batches are accumulated and processed for robust insights
  - Data must feed both batch-based and stream-based sub-systems
  - Real-time insights are cached
  - Batch insights are cached
  - Queries to the whole system combine both kinds of insights
- http://lambda-architecture.net/

Distributed storage:

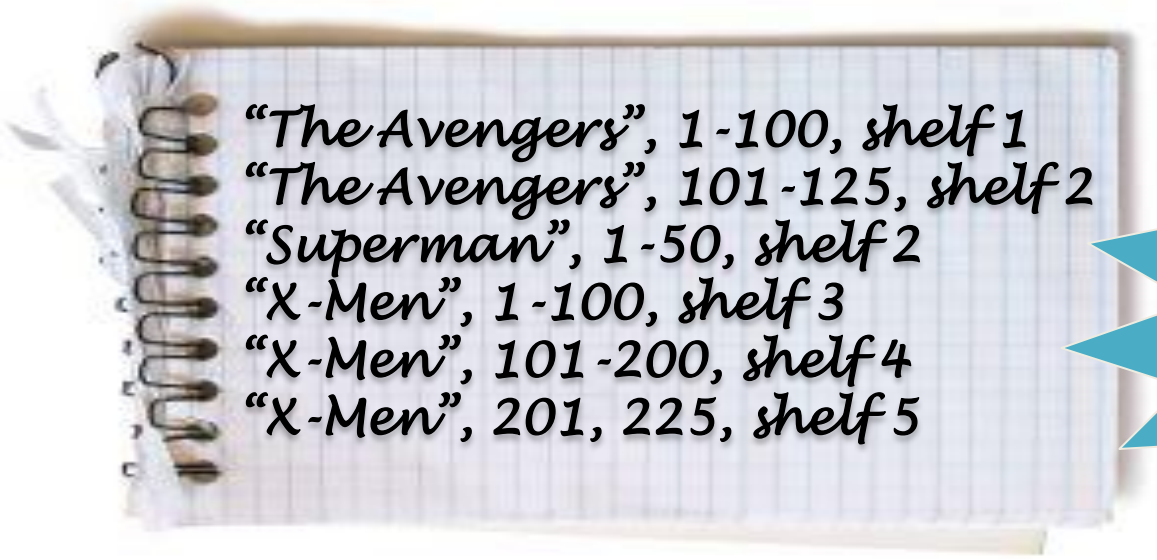
The Hadoop reference (HDFS)

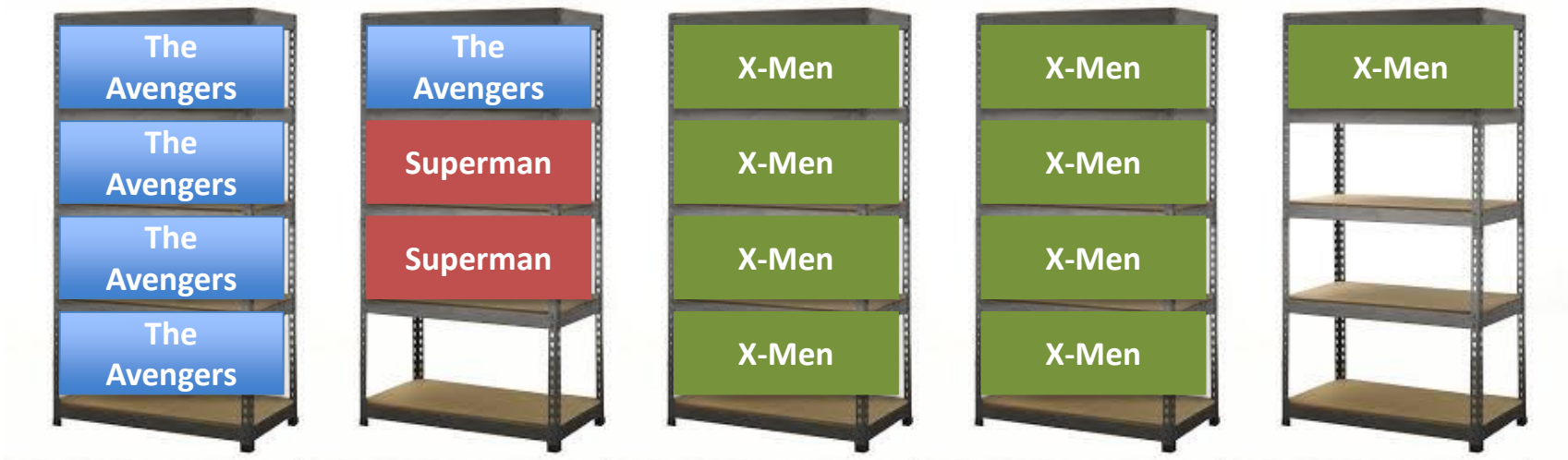
What happens if one shelving is not enough?

You buy more shelves…

FIWARE

… then you create an index



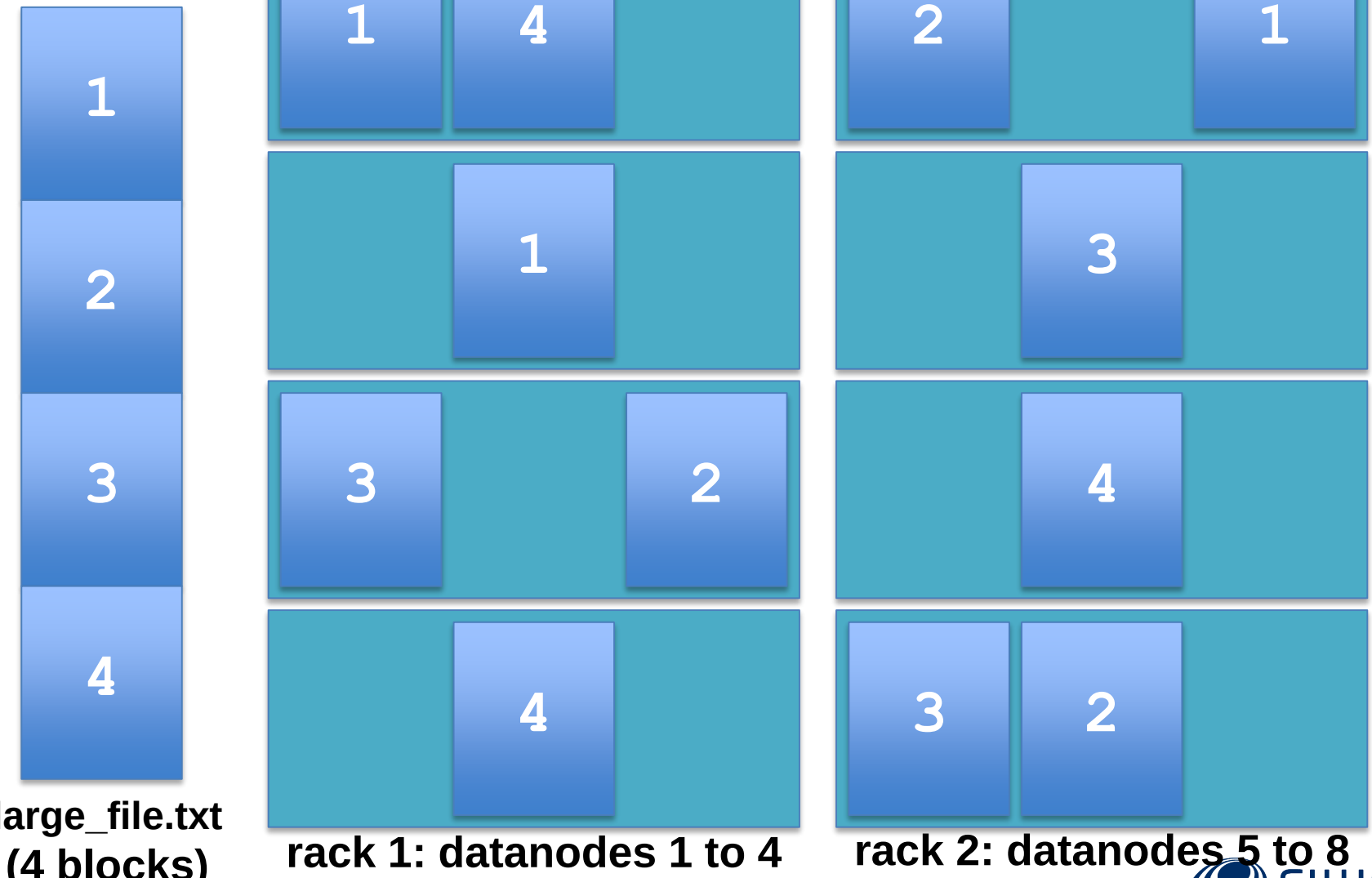


FIWARE

Hadoop Distributed File System (HDFS)

- Based on **Google File System**
- Large files are stored across multiple machines (**Datanodes**) by spliting them into blocks that are distributed
- Metadata is managed by the **Namenode**
- Scalable by simply adding more Datanodes
- Fault-tolerant since HDFS replicates each block (default to 3)
- Security based on authentication (Kerberos) and authorization (permissions, HACLs)
- It is managed like a Unix-like file system

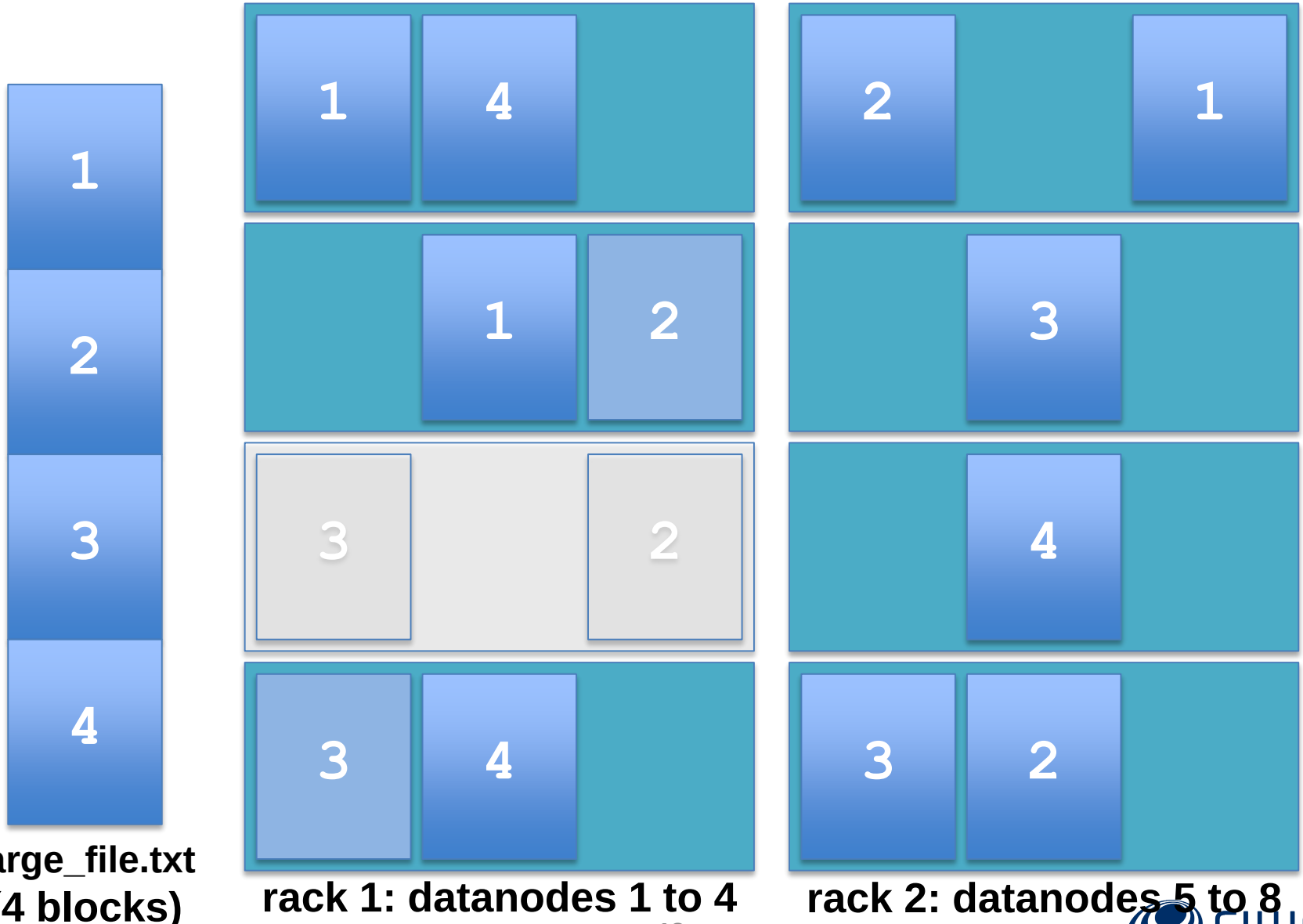
Spliting, replication and distribution



Namenode metadata

Path	Replicas	Block IDs
/user/user1/data/large_file.txt	3	1 → {dn1,dn2,dn5} 2 → {dn3,dn5,dn8} 3 → {dn3,dn6,dn8} 4 → {dn1,dn4,dn7}
/user/user1/data/other_file.txt	2	5 → {…} 6 → {…} 7 → {…}
…	…	…

Datanodes failure recovering



large_file.txt
(4 blocks)

rack 1: datanodes 1 to 4

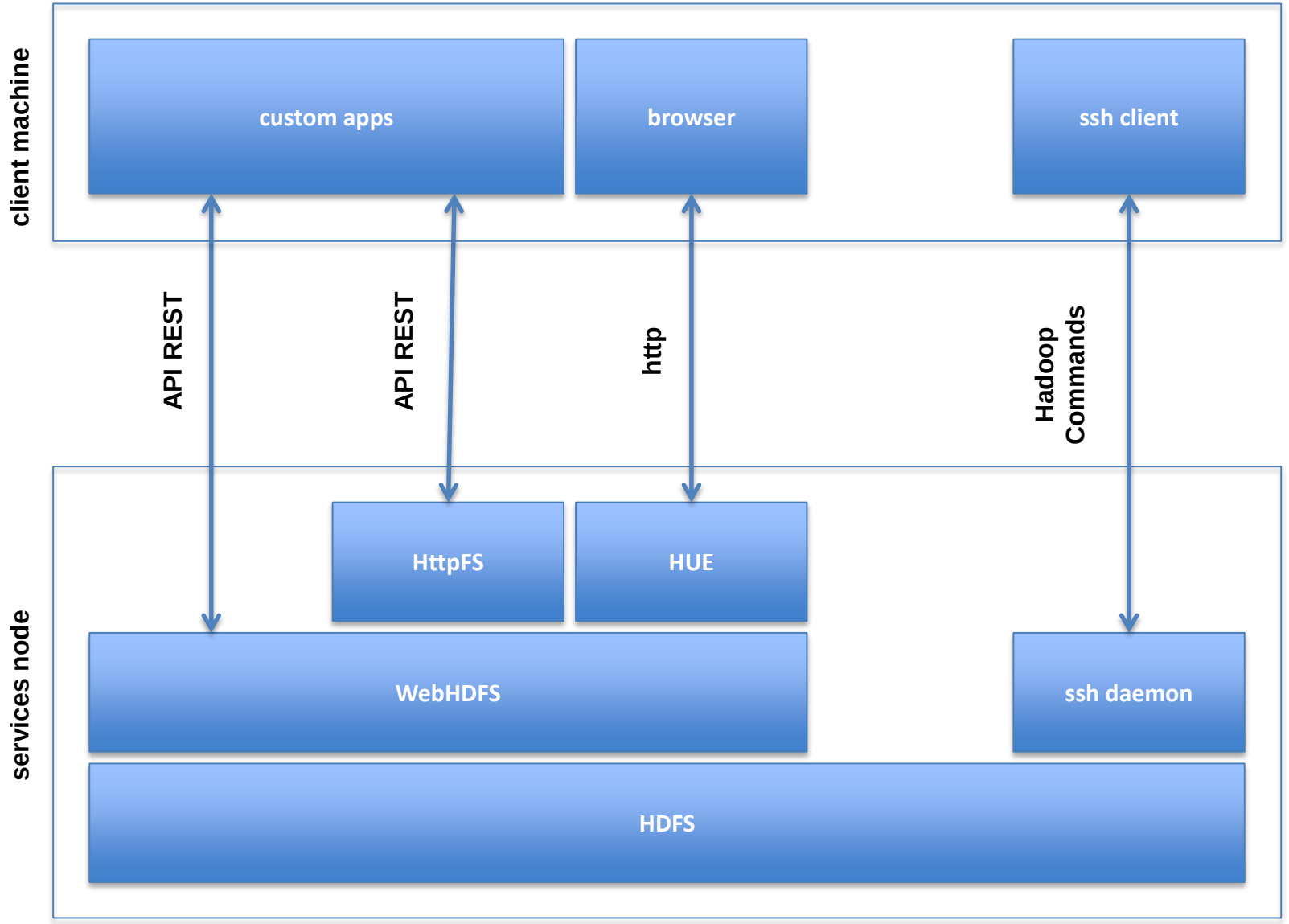
rack 2: datanodes 5 to 8

FIWARE

Namenode failure recovering

Path	Replicas	Block IDs
/user/user1/data/large_file.txt	3	1 → {dn1,dn2,dn5} 2 → {dn2,dn5,dn8} 3 → {dn4,dn6,dn8} 4 → {dn1,dn4,dn7}
/user/user1/data/other_file.txt	2	5 → {…} 6 → {…} 7 → {…}
…	…	…

Managing HDFS



FIWARE

Managing HDFS: HTTP REST API

- The HTTP REST API supports the **complete File System** interface for HDFS
  - Other Hadoop commands are not available through a REST API
- It relies on the `webhdfs` schema for URIs

```
webhdfs://<HOST>:<HTTP_PORT>/<PATH>
```

- **HTTP URLs** are built as:

```
http://<HOST>:<HTTP_PORT>/webhdfs/v1/<PATH>?op=…
```

- Full API specification
  - http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/WebHDFS.html

Managing HDFS: HTTP REST API

```
$ curl -X GET "http://cosmos.lab.fi-ware.org:14000/webhdfs/v1/user/frb/webinar/abriefhistoryoftime_page1?op=open&user.name=frb"
CHAPTER 1
OUR PICTURE OF THE UNIVERSE
A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast
$ curl -X PUT "http://cosmos.lab.fi-ware.org:14000/webhdfs/v1/user/frb/webinar/afolder?op=mkdirs&user.name=frb"
{"boolean":true}
$ curl -X GET "http://cosmos.lab.fi-ware.org:14000/webhdfs/v1/user/frb/webinar?op=liststatus&user.name=frb"
{"FileStatuses":{"FileStatus":[{"pathSuffix":"abriefhistoryoftime_page1","type":"FILE","length":3431,"owner":"frb","group":"cosmos","permission":"644","accessTime":1425995831489,"modificationTime":1418216412441,"blockSize":67108864,"replication":3},{"pathSuffix":"abriefhistoryoftime_page2","type":"FILE","length":1604,"owner":"frb","group":"cosmos","permission":"644","accessTime":1418216412460,"modificationTime":1418216412500,"blockSize":67108864,"replication":3},{"pathSuffix":"abriefhistoryoftime_page3","type":"FILE","length":5257,"owner":"frb","group":"cosmos","permission":"644","accessTime":1418216412515,"modificationTime":1418216412551,"blockSize":67108864,"replication":3},{"pathSuffix":"afolder","type":"DIRECTORY","length":0,"owner":"frb","group":"cosmos","permission":"755","accessTime":0,"modificationTime":1425995941361,"blockSize":0,"replication":0}]}}
$ curl -X DELETE "http://cosmos.lab.fi-ware.org:14000/webhdfs/v1/user/frb/webinar/afolder?op=delete&user.name=frb"
{"boolean":true}
```


FIWARE

Distributed batch computing:

The Hadoop reference

(MapReduce)

What happens if you cannot read all your books?

FIWARE

Hadoop was created by Doug Cutting at Yahoo!...

… based on the MapReduce patent by Google

Google

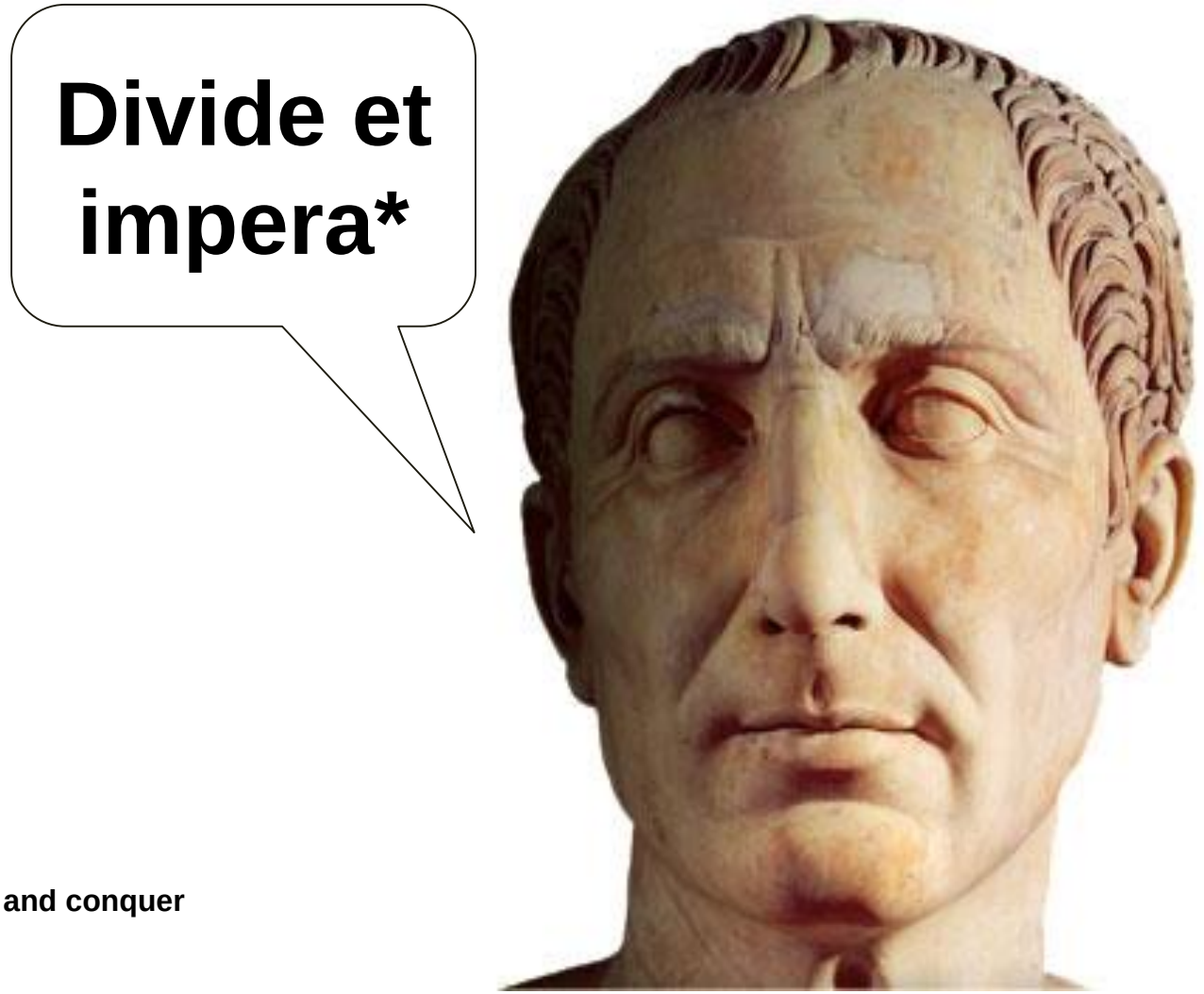
System and method for efficient large-scale data processing

US 7650331 B1

A large-scale data processing system and method includes one or more application-independent map modules configured to read input data and to apply at least one application-specific map operation to the input data to produce intermediate data values, wherein the map operation is automatically parallelized across multiple processors in the parallel processing environment. A plurality of

FIWARE

Well, MapReduce was really invented by Julius Caesar



* Divide and conquer

FIWARE

An example

How much pages are written in latin among the books in the Ancient Library of Alexandria?

GREEK REF7 P20 | LATIN REF4 P73 | LATIN REF1 P45

LATIN pages 45 ✓

LATIN REF5 P34 | GREEK REF2 P128

still reading…

Reducer

GREEK REF8 P230 | EGYPT REF6 P10 | EGYPT REF3 P12

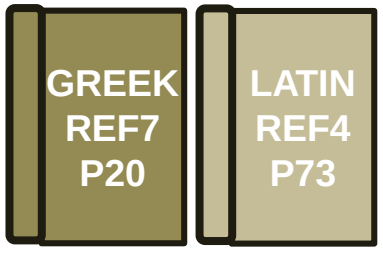
EGYPTIAN ✗

Mappers

FIWARE

An example

How much pages are written in latin among the books in the Ancient Library of Alexandria?



still reading…

GREEK

Reducer

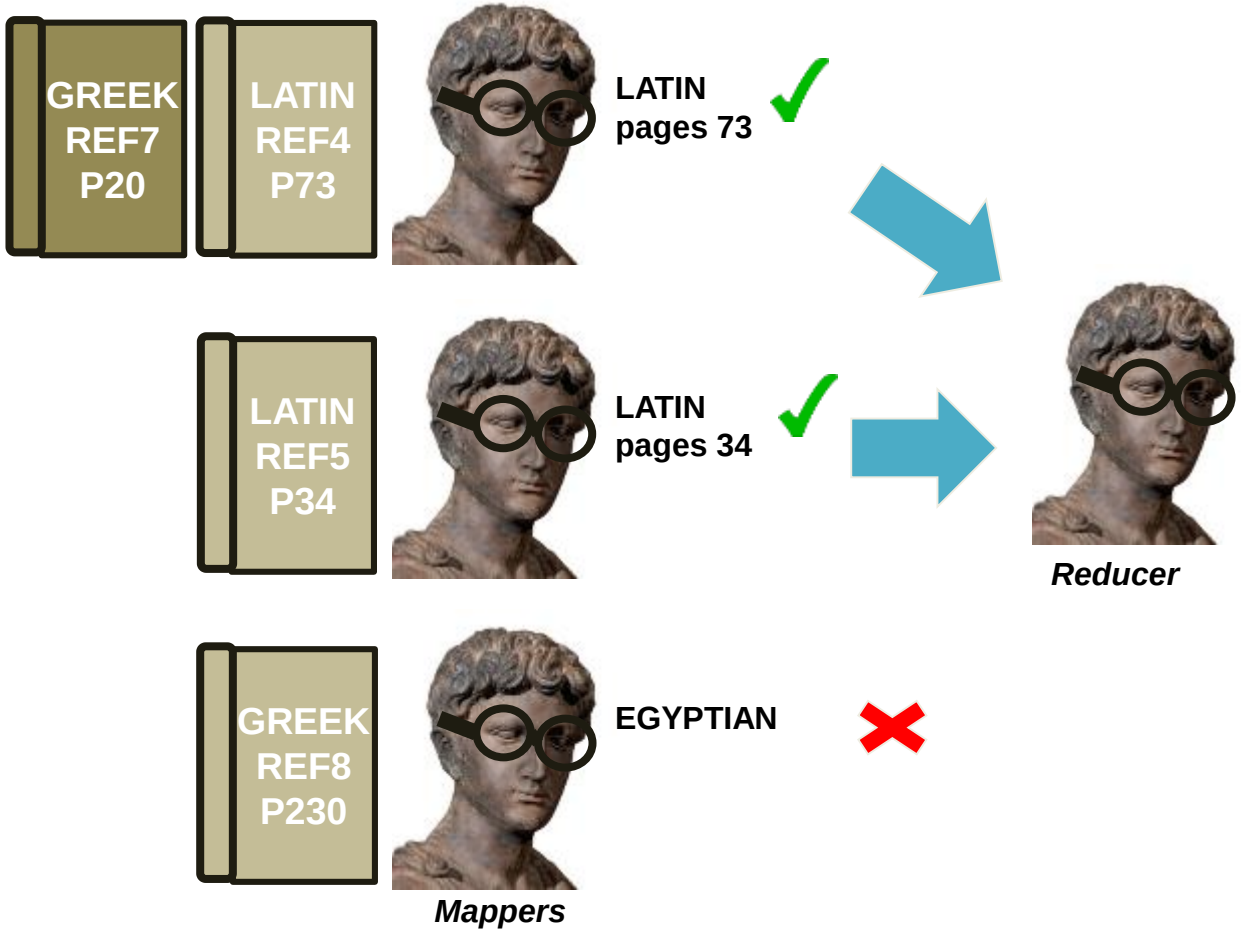
Mappers

EGYPTIAN

FIWARE

An example

How much pages are written in latin among the books in the Ancient Library of Alexandria?



An example

How much pages are written in latin among the books in the Ancient Library of Alexandria?

An example

How much pages are written in latin among the books in the Ancient Library of Alexandria?

idle…

idle…

idle…

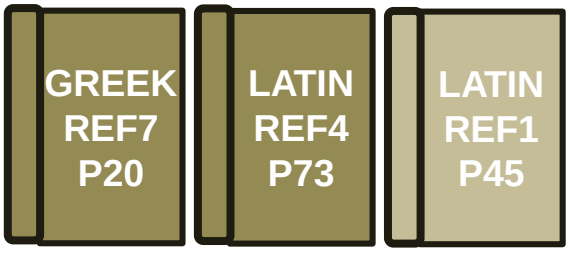
Mappers

Reducer

FIWARE

Another example

How much pages are written in all the languages among the books in the Ancient Library of Alexandria?



(lat,45)

Reducer

still reading…

(egy,12)

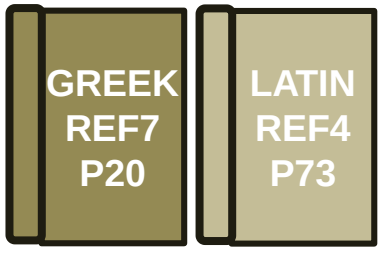
Reducer

Mappers

FIWARE

Another example

How much pages are written in all the languages among the books in the Ancient Library of Alexandria?



still reading…

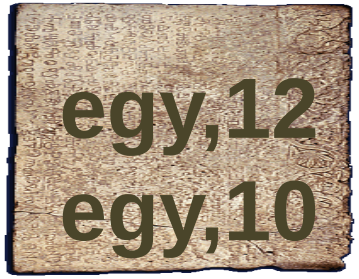
(gre,128)

(egy,10)

Mappers

Reducer

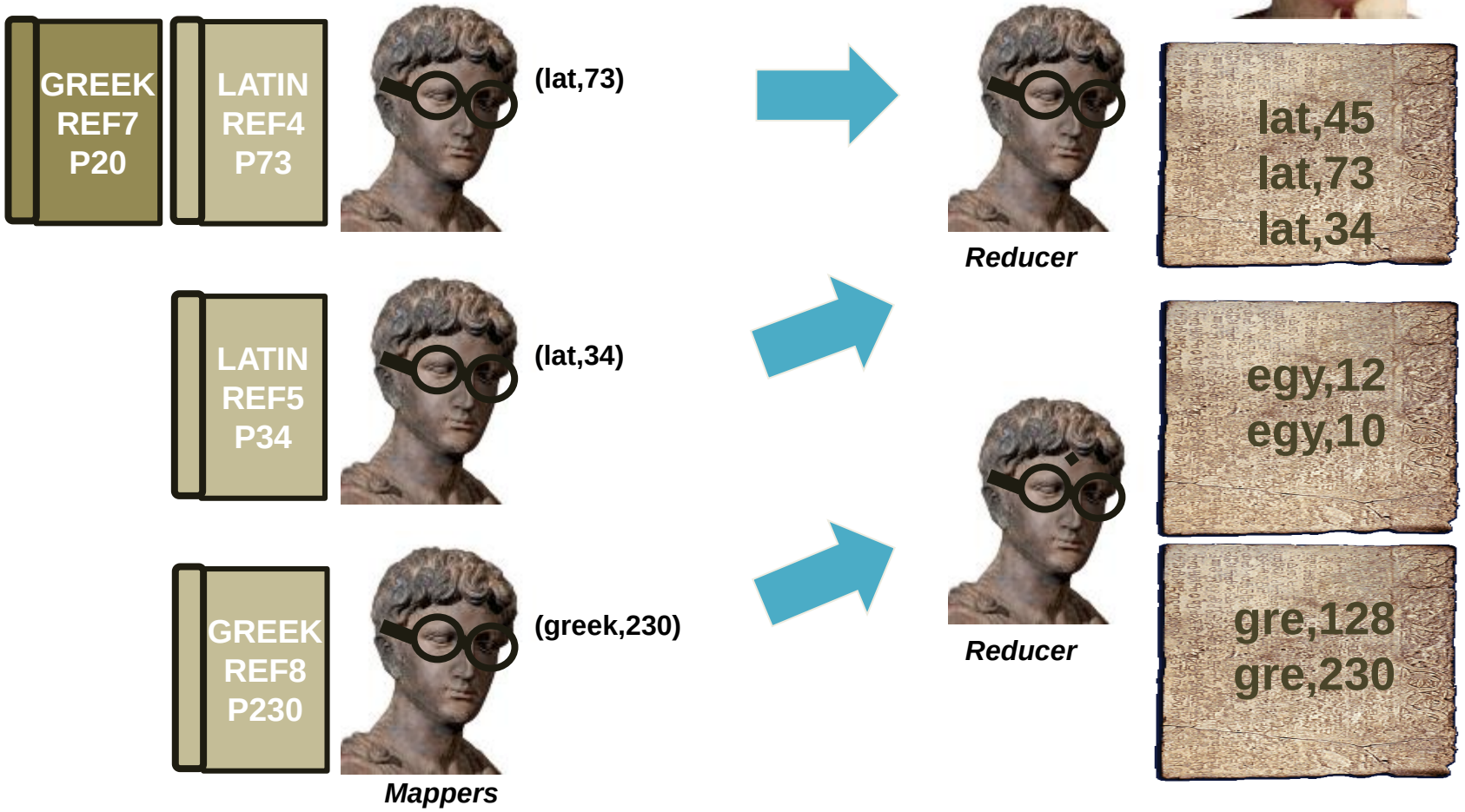
Reducer



FIWARE

Another example

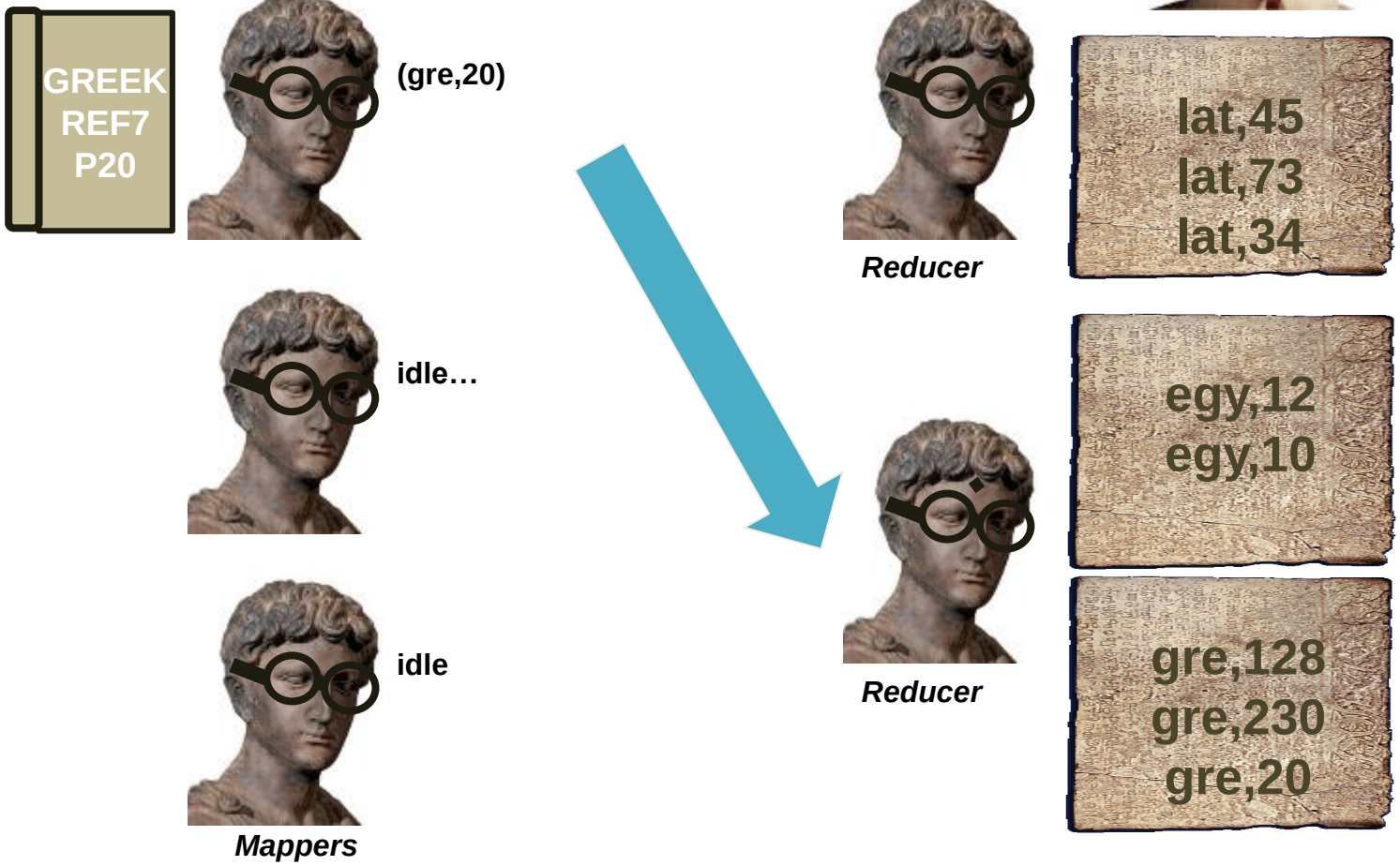
How much pages are written in all the languages among the books in the Ancient Library of Alexandria?



FIWARE

Another example

How much pages are written in all the languages among the books in the Ancient Library of Alexandria?



FIWARE

Another example

How much pages are written in all the languages among the books in the Ancient Library of Alexandria?

idle…

idle…

idle…

Mappers

Reducer

Reducer

FIWARE

Writing MapReduce applications

- MapReduce applications are commonly **written in Java** language**:**
  - Can be written in other languages through **Hadoop Streaming**
- A MapReduce job consists of:
  - A **driver**, a piece of software where to define inputs, outputs, formats, etc. and the entry point for launching the job
  - A set of **Mappers**, given by a piece of software defining its behaviour
  - A set of **Reducers**, given by a piece of software defining its behaviour
- https://hadoop.apache.org/docs/current/api/ (MapReduce section)

Implementing the example

- The input will be a single big file containing:

```
symbolae botanicae,latin,230
mathematica,greek,95
physica,greek,109
ptolomaics,egyptian,120
terra,latin,541
iustitia est vincit,latin,134
```

- The mappers will receive pieces of the above file, which will be read line by line
  - Each line will be represented by a (key,value) pair, i.e. the offset on the file and the real data within the line, respectively
  - For each input pair a (key,value) pair will be output, i.e. a common “num_pages” key and the third field in the line
- The reducers will receive arrays of pairs produced by the mappers, all having the same key (“num_pages”)
  - For each array of pairs, the sum of the values will be output as a (key,value) pair, in this case a “total_pages” key and the sum as value

Implementing the example: JCMapper.class

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

   private final Text globalKey = new Text("num_pages");
   private final IntWritable bookPages = new IntWritable();

   @Override
   public void map(Object key, Text value, Context context)
   throws Exception {
      String[] fields = value.toString().split(",");
      system.out.println("Processing " + fields[0]);

      if (fields[1].equals("latin")) {
         bookPages.set(fields[2]);
         context.write(globalKey, bookPages);
      } // if
   } // map

} // JCMapper
```

Implementing the example: JCReducer.class

```
public static class JCReducer extends
Reducer<Text, IntWritable, Text, IntWritable> {

   private final IntWritable totalPages= new IntWritable();

   @Override
   public void reduce(Text globalKey, Iterable<IntWritable>
   bookPages, Context context) throws Exception {
      int sum = 0;

      for (IntWritable val: bookPages) {
         sum += val.get();
      } // for

      totalPages.set(sum);
      context.write(globalKey, totalPages);
   } // reduce

} // JCReducer
```

Implementing the example: JC.class

```
public static void main(String[] args) throws Exception {
   int res = ToolRunner.run(
      new Configuration(), new CKANMapReduceExample(), args);
   System.exit(res);
} // main

@Override
public int run(String[] args) throws Exception {
   Configuration conf = this.getConf();
   Job job = Job.getInstance(conf, "julius caesar");
   job.setJarByClass(JC.class);
   job.setMapperClass(JCMapper.class);
   job.setCombinerClass(JCReducer.class);
   job.setReducerClass(JCReducer.class);
   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(IntWritable.class);
   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));
   return job.waitForCompletion(true) ? 0 : 1;
} // run
```

Simplifying the batch analysis:

Querying tools

Querying tools

- MapReduce paradigm may be hard to understand and, the worst, to use
- Indeed, many data analyzers just need to **query** for the data
  - If possible, by using already well-known languages
- Regarding that, some **querying tools** appeared in the Hadoop ecosystem
  - **Hive** and its **HiveQL** language → quite similar to SQL
  - **Pig** and its **Pig Latin** language → a new language

Hive and HiveQL

- HiveQL reference
  - https://cwiki.apache.org/confluence/display/Hive/LanguageManual
- All the **data is loaded** into Hive tables
  - Not real tables (they don't contain the real data) but metadata pointing to the real data at HDFS
- The best thing is Hive uses **pre-defined MapReduce jobs** behind the scenes!
  - Column selection
  - Fields grouping
  - Table joining
  - Values filtering
  - …
- **Important remark**: since MapReduce is used by Hive, the queries make take some time to produce a result

Hive CLI

```
$ hive
hive history
file=/tmp/myuser/hive_job_log_opendata_XXX_XXX.txt
hive>select column1,column2,otherColumns from mytable where
column1='whatever' and columns2 like '%whatever%';
Total MapReduce jobs = 1
Launching Job 1 out of 1
Starting Job = job_201308280930_0953, Tracking URL =
http://cosmosmaster-
gi:50030/jobdetails.jsp?jobid=job_201308280930_0953
Kill Command = /usr/lib/hadoop/bin/hadoop job  -
Dmapred.job.tracker=cosmosmaster-gi:8021 -kill
job_201308280930_0953
2013-10-03 09:15:34,519 Stage-1 map = 0%,  reduce = 0%
2013-10-03 09:15:36,545 Stage-1 map = 67%,  reduce = 0%
2013-10-03 09:15:37,554 Stage-1 map = 100%,  reduce = 0%
2013-10-03 09:15:44,609 Stage-1 map = 100%,  reduce = 33%
```


FIWARE

Hive Java API

- Hive CLI and Hue are OK for human-driven testing purposes
  - But it **is not usable by remote applications**
- Hive has **no REST API**
- Hive has several **drivers and libraries**
  - JDBC for Java
  - Python
  - PHP
  - ODBC for C/C++
  - Thrift for Java and C++
  - https://cwiki.apache.org/confluence/display/Hive/HiveClient
- A remote Hive client usually performs:
  - A **connection** to the Hive server (TCP/10000)
  - The **query execution**

Hive Java API: get a connection

```
private static Connection getConnection(String ip, String port,
      String user, String password) {
   try {
      Class.forName("org.apache.hadoop.hive.jdbc.HiveDriver");
   } catch (ClassNotFoundException e) {
      System.out.println(e.getMessage());
      return null;
   } // try catch

   try {
      return DriverManager.getConnection("jdbc:hive://" + ip
         + ":" + port + "/default?user=" + user + "&password="
         + password);
   } catch (SQLException e) {
      System.out.println(e.getMessage());
      return null;
   } // try catch
} // getConnection
```

Hive Java API: do the query

```
private static void doQuery() {
    try {
        Statement stmt = con.createStatement();
        ResultSet res = stmt.executeQuery(
            "select column1,column2,"
            + "otherColumns from mytable where "
            + "column1='whatever' and "
            + "columns2 like '%whatever%'");

        while (res.next()) {
            String column1 = res.getString(1);
            Integer column2 = res.getInteger(2);
        } // while

        res.close(); stmt.close(); con.close();
    } catch (SQLException e) {
        System.exit(0);
    } // try catch
} // doQuery
```

Hive tables creation

- Both locally using the CLI, or remotely using the Java API, use this command:

```
create [external] table...
```

- **CSV**-like HDFS files

```
create external table <table_name> (<field1_name>
<field1_type>, ..., <fieldN_name> <fieldN_type>) row format
delimited field terminated by '<separator>' location
'/user/<username>/<path>/<to>/<the>/<data>';
```

- **Json**-like HDFS files

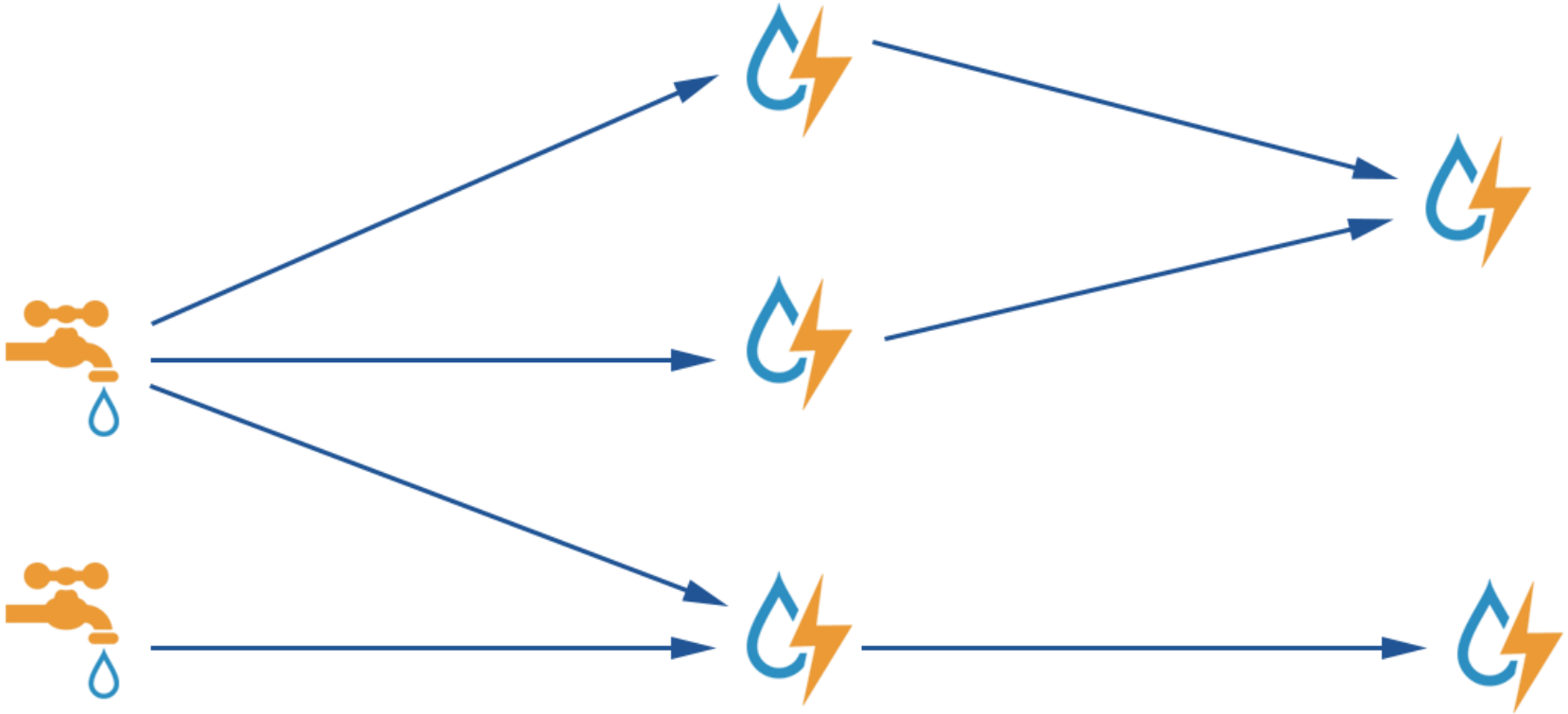
```
create external table <table_name> (<field1_name>
<field1_type>, ..., <fieldN_name> <fieldN_type>) row format
serde 'org.openx.data.jsonserde.JsonSerDe' location
'/user/<username>/<path>/<to>/<the>/<data>';
```

Distributed streaming computing:

The Storm reference

Storm project

- Created by Natham Marz at BackType/Twitter
- Distributed realtime computation system



FIWARE

Storm basics

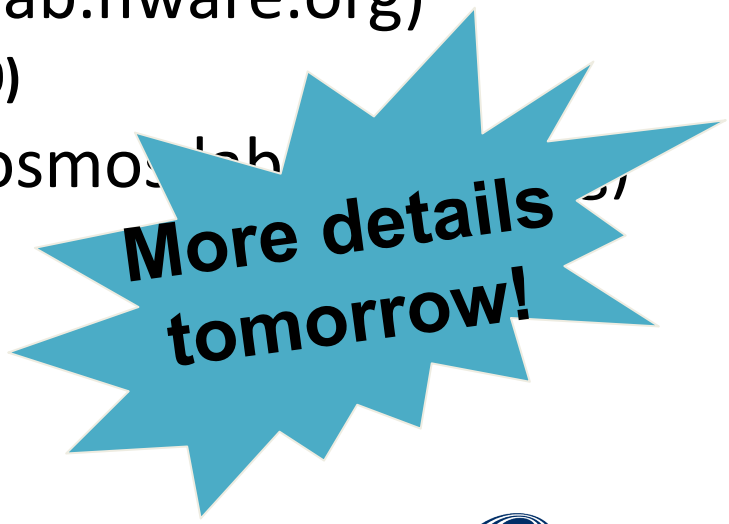
- Based on processing building blocks that can be composed in a topology
  - **Spouts**: blocks in charge of polling for data streams, producing data tuples
  - **Bolts**: blocks in charge of processing data tuples, performing basic operations
    - 1:1 operations: arithmetics, transformations…
    - N:1 operations: filtering, joining…
    - 1:N operations: spliting, replication…
- It is **scalable** and **fault-tolerant**
  - A basic operation can be replicated many times in a layer of bolts
  - If a bolt fails, there are serveral other bolts performing the same basic operation in the layer
- Guarantees the data will be processed
  - Storm perform an **ACK** mechanism for data tuples

Big Data in FIWARE Lab:

Cosmos and Sinfonier

Cosmos

- Cosmos is the name of the Hadoop-based global instance in FIWARE Lab
- Nothing has to be installed!
- There are two clusters exposing some services:
  - **Storage** (storage.cosmos.lab.fiware.org)
    - **WebHDFS REST API (TCP/14000)**
  - **Computing** (computing.cosmos.lab.fiware.org)
    - **Tidoop REST API (TCP/12000)**
    - **Auth proxy (TCP/13000)**
    - **HiveServer2 (TCP/10000)**



FIWARE

Feeding Cosmos with context data

- Cygnus tool
  - Apache Flume-based
- Standard NGSI connector for FIWARE
- Provides connectors for a wide variety of persistence backends
  - **HDFS**
  - MySQL
  - CKAN
  - MongoDB
  - STH Comet
  - PostgreSQL
  - Kafka
  - DynamoDB
  - Carto

FIWARE

Sinfonier

- Sinfonier will be the name of the Storm-based global instance in FIWARE Lab
- Nothing will have to be installed!
- There will be one cluster exposing streaming analysis services through an IDE
- Will be fed using Cygnus and Kafka queues
- **Coming soon!**

Thank you!

http://fiware.org
Follow @FIWARE on Twitter