Perugia

INFN Istituto Nazionale di Fisica Nucleare

Using Hadoop File System and MapReduce in a Small/Medium Grid Site

Hassen Riahi[1], Giacinto Donvito[2], Livio Fano'[1], Massimiliano Fasi'[1], Giovanni Marzulli[2], Andrea Valentini[1]

1) INFN Perugia, Via A. Pascoli 2) INFN Bari, Via E. Orbona

06123 Perugia, Italy 70125 Bari, Italy

Hadoop[1] is data processing system that follows the MapReduce paradigm[2] for scalable data analysis. It includes a fault-tolerant and scalable execution environment, named MapReduce, and a distributed file system, named Hadoop Distribted File Sytem(HDFS).

- The e largest Hadoop-based Cluster is installed at Facebook to manage nearly 31 PB of online disk data.
- Other companies, such as Yahoo and Last.Fm, are also making use of this technology.

\

Data storage and access represent the key of CPU-intensive and data-intensive high performance Grid computing. However, the small/medium size Grid sites are often constrained to use commodity Hardware which exposes them to Hardware failure.

The goal is the deployment of Hadoop-based solution for data storage and processing in High Energy Physics (HEP) Grid sites.

Advantages of Hadoop-based solution for data processing and storage

The deployment of Hadoop-based solution for data storage and processing represents many advantages. The most important characteristics of this solution are:

- Reliability:
  - HDFS allows the deployment of commodity Hardware. To deal with unreliable storage/servers:
    - Use replication across servers
    - Handle task resubmission on failure
- Scalability:
  - Hadoop MapReduce splits the data computation into fine grained Map and Reduce tasks, which results in:
    - Improved load balancing inside the Cluster
    - Faster recovery from failed tasks
  - This solution allows to benefit of data locality optimization:
    - Tasks are scheduled close to the closest replica of the input

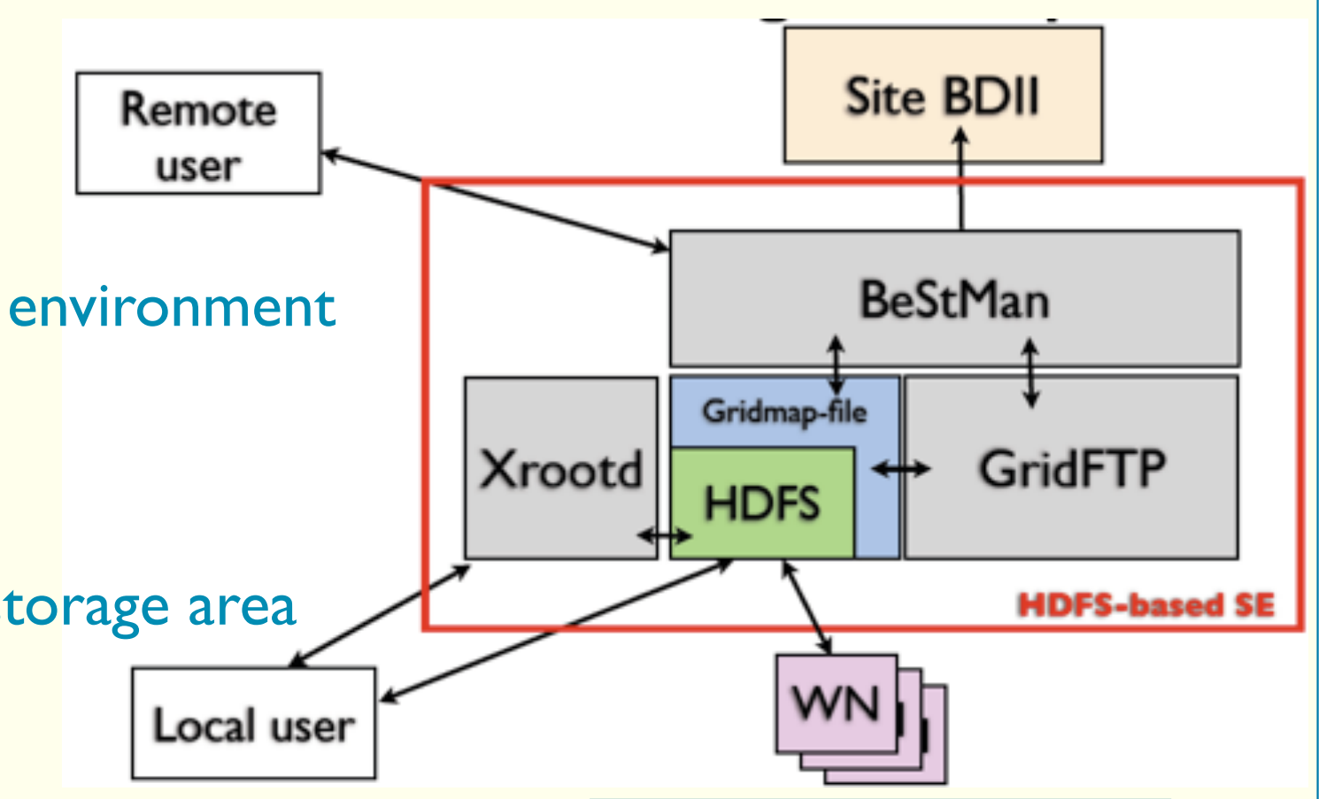
Hadoop-based Storage Element in gLite environment

On the WLCG, the protocol used for WAN data transfers is GridFTP[3] and the protocol used for metadata operations is SRM; both are necessary for interoperability between site storage and the Grid.

For SRM access, the first choice was to use Storm since it is widely used in gLite. But it was noticed that this solution cannot work with filesystem not supporting Access Control List (ACL). Since HDFS does not support the ACL, the Berkeley Storage Manager (BeStMan) SRM server[4] was used.

The following steps were required to setup HDFS as a Grid SE in gLite environment:

- Setup GridFTP server:
  - Get the HDFS-GridFTP library developed for OSG sites and recompile it in gLite environment
  - Start gLite GridFTP server: globusgridftp-server -p 2811 -dsi hdfs
- Setup SRM server:
  - Mount HDFS using Fuse
  - Install BeStMan and configure it correctly to be able to manage the experiments storage area
- Setup Xrootd service:
  - Install the xrootd-hdfs rpm used in OSG site with --nodeps option to bypass the credential check required for OSG sites
- Information Service:
  - A provider script is developed to publish dynamic information in gLite Information Service. SRM-PING is called to get required information

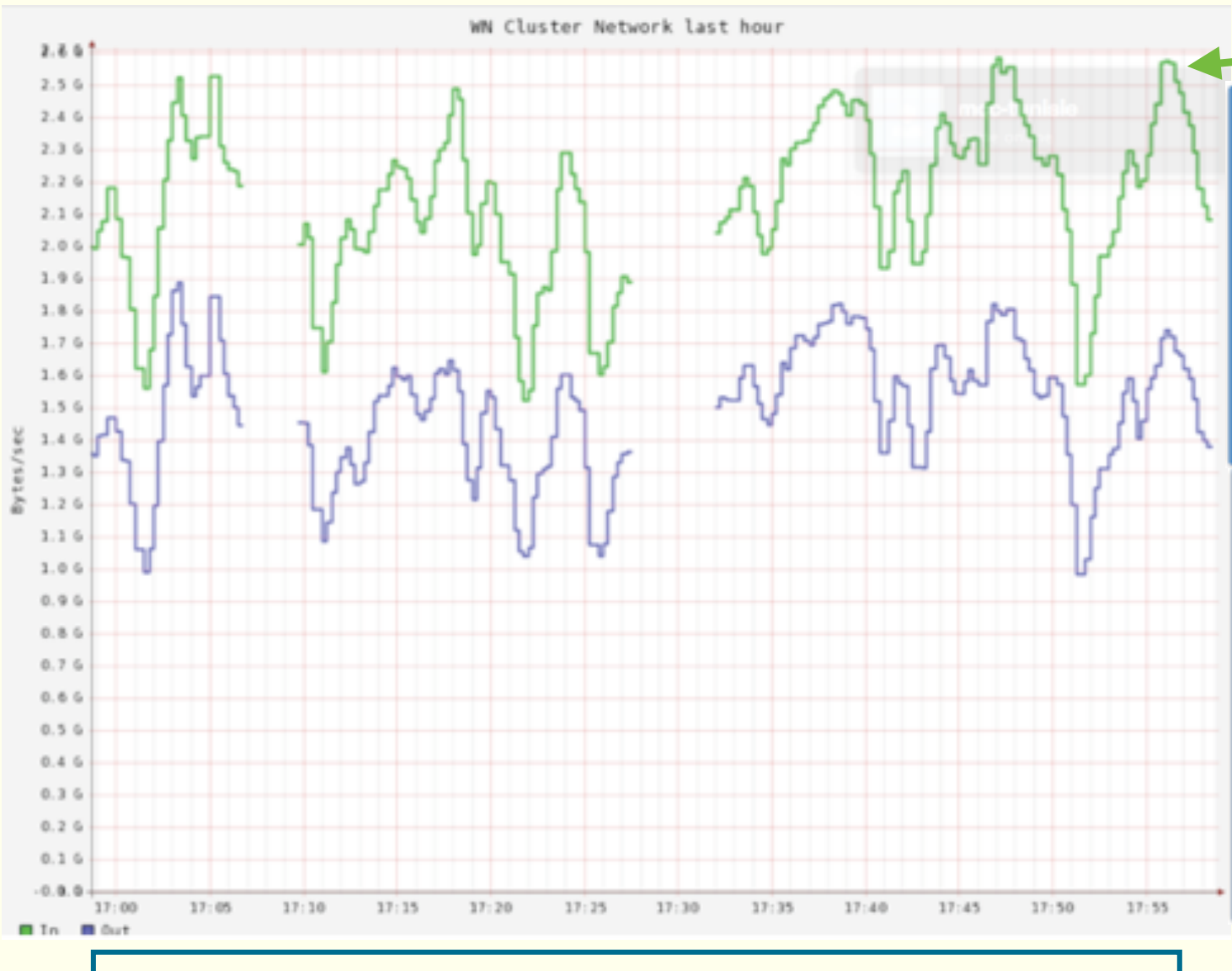


Hadoop-based SE in gLite environment

Fault tolerance and scalability test for small/medium sized Grid site

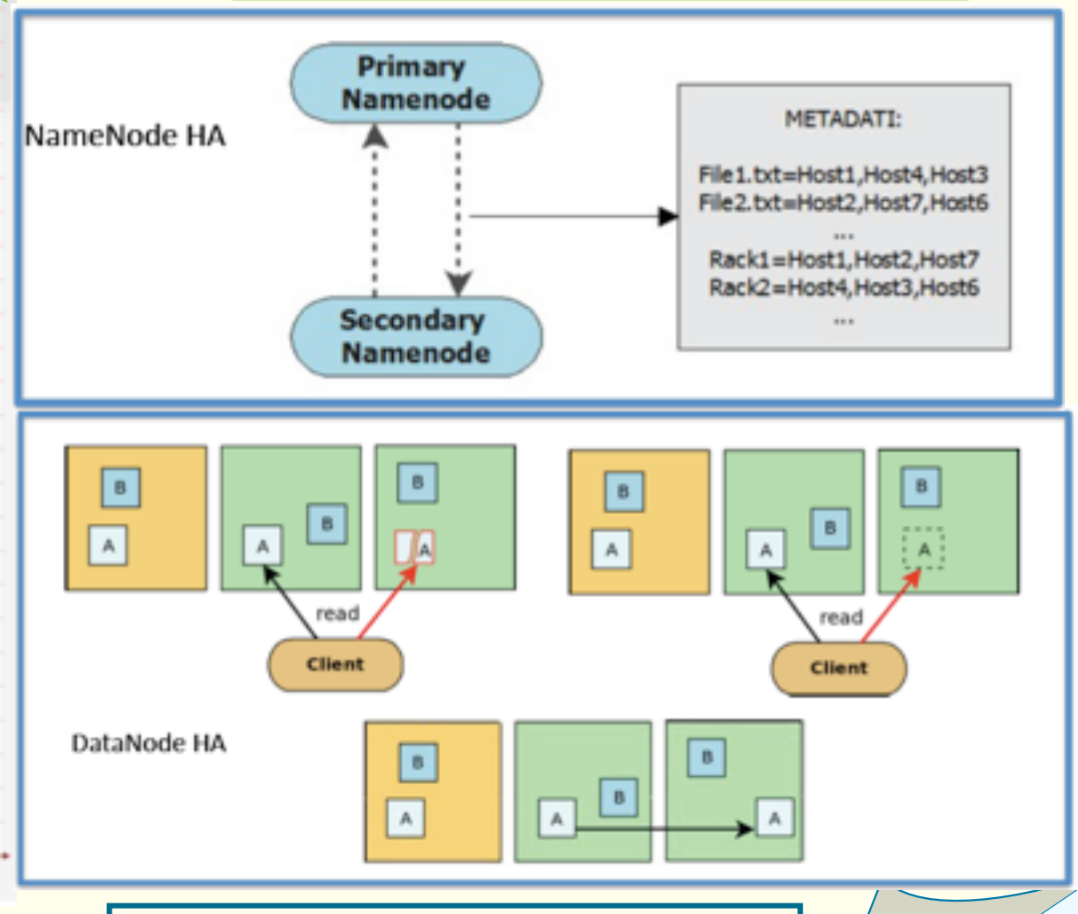
The work carried out is focused on:

- Testing the resilience to the Hardware and Software failures.
- The measurement of the horizontal scalability of an Hadoop filesystem.



Easy to go up to 2.6 GByte/s

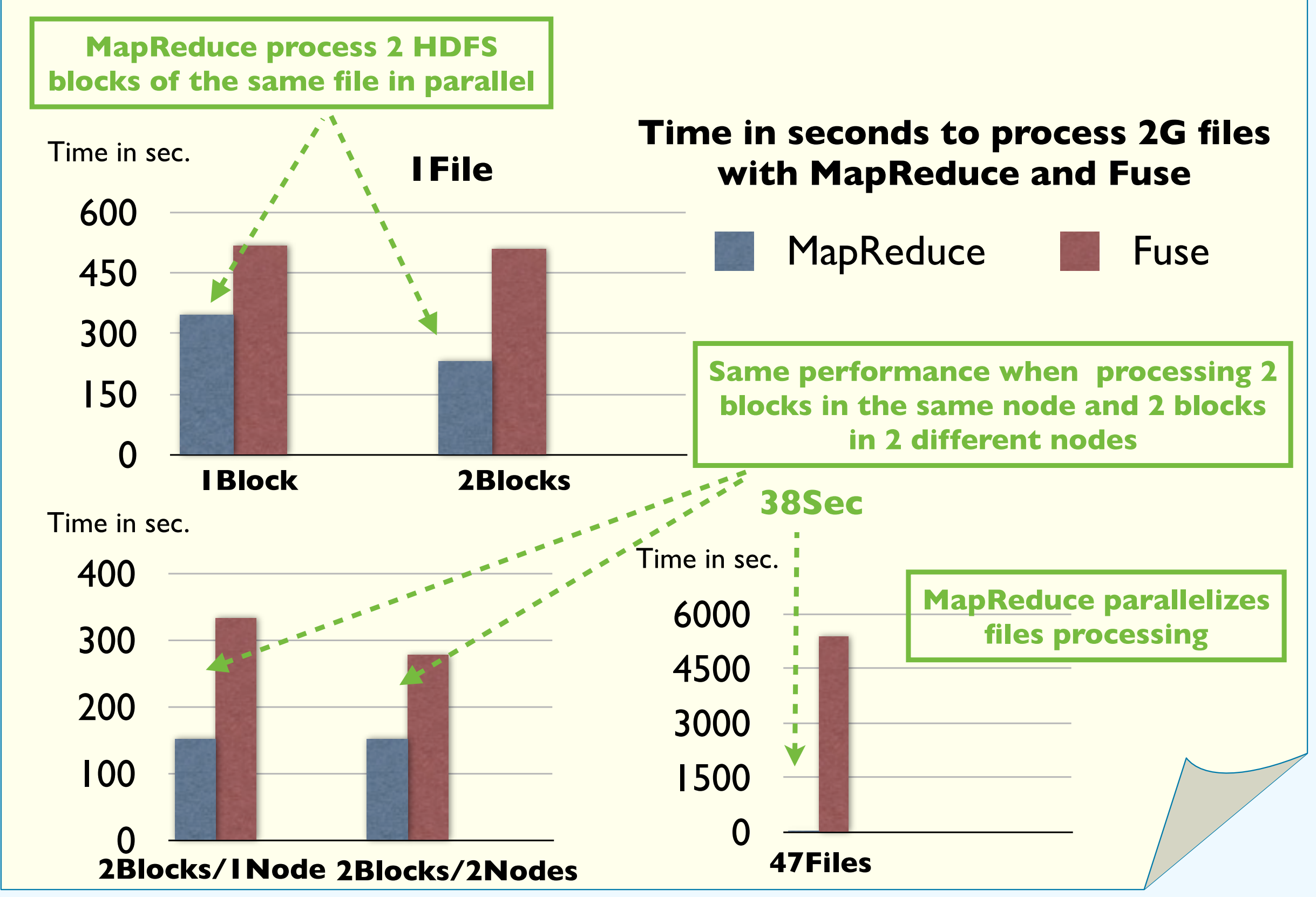
Sequential (concurrent) read and write using Hadoop on 130 WN + 1 disk server



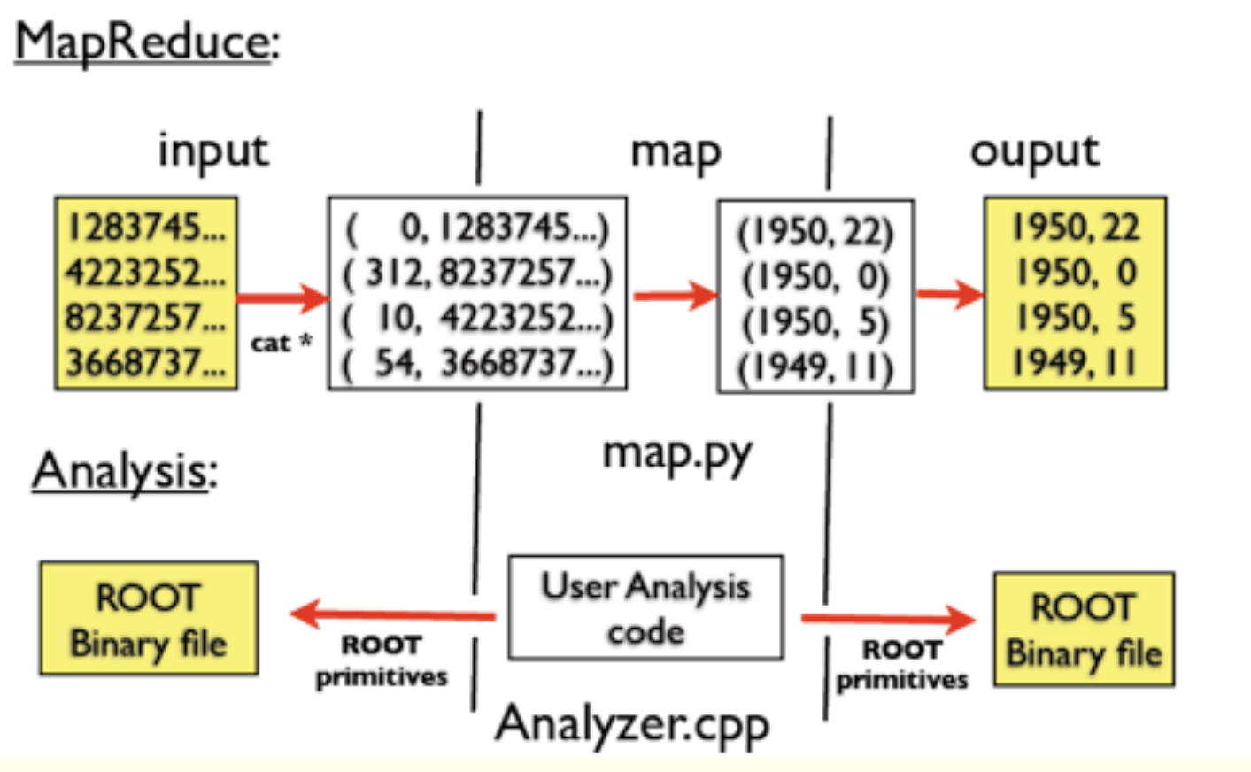
Highly available NameNode and DataNode

Performance test

MapReduce vs Fuse



Mapping the Analysis data flow into MapReduce



MapReduce and Analysis data flows

MapReduce workflow

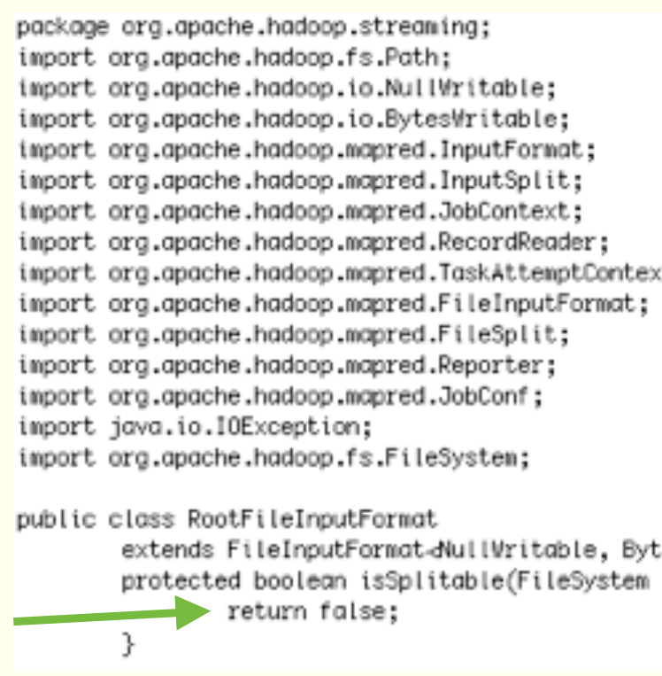
- Input: default TXT files
  - Split per line
- Map
  - Process TXT input streams
- Output: default TXT files

Analysis workflow

- Input: ROOT files
  - Split per file
- Analyzer
  - Open and process ROOT files
- Output: ROOT files

Execution of the Analysis workflow using MapReduce

1. Use the dedicated InputFormat developed to support the splitting and read of ROOT files
   - To achieve this, RootFileInputFormat.java and RootFileRecordReader.java were developed
2. Use map.input.file environment to locate the path of the input ROOT file in HDFS
3. Open the ROOT file using the HDFS Plugin of ROOT
   - Patch submitted to export the CLASSPATH environment variable in the Plugin
4. Process the ROOT file content
5. Write the output into HDFS using libhdfs



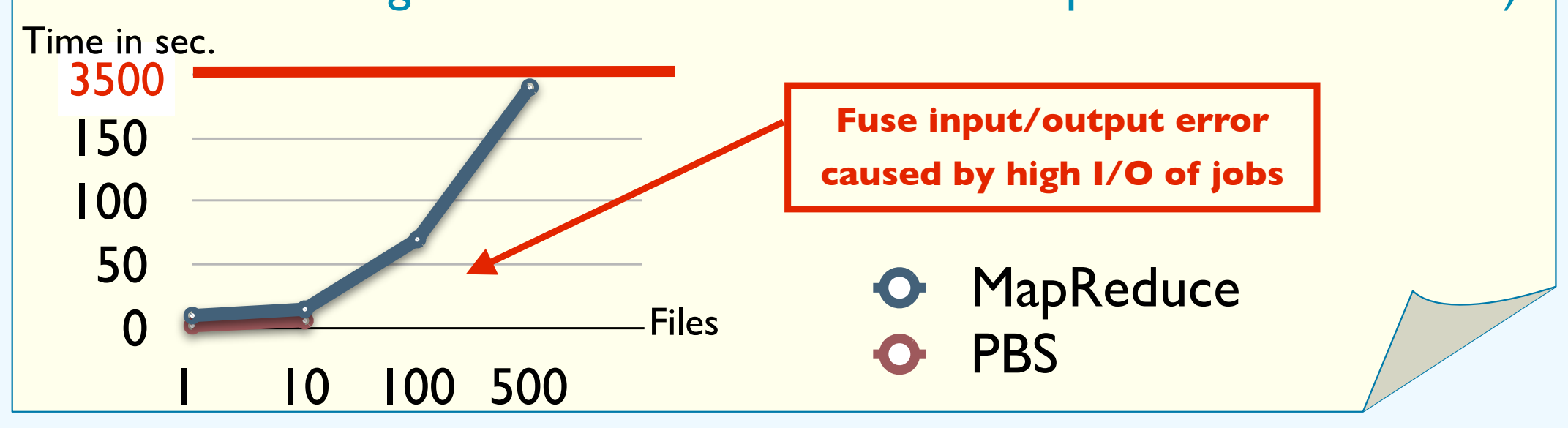
```
package org.apache.hadoop.streaming;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.mapred.InputFormat;
import org.apache.hadoop.mapred.InputSplit;
import org.apache.hadoop.mapred.JobContext;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.TaskAttemptContext;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileSplit;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.JobConf;
import java.io.IOException;
import org.apache.hadoop.fs.FileSystem;

public class RootFileInputFormat
        extends FileInputFormat<NullWritable, BytesWritable> {
        protected boolean isSplitable(FileSystem fs, Path filename) {
                return false;
        }
```

RootFileInputFormat.java

Whole file per split

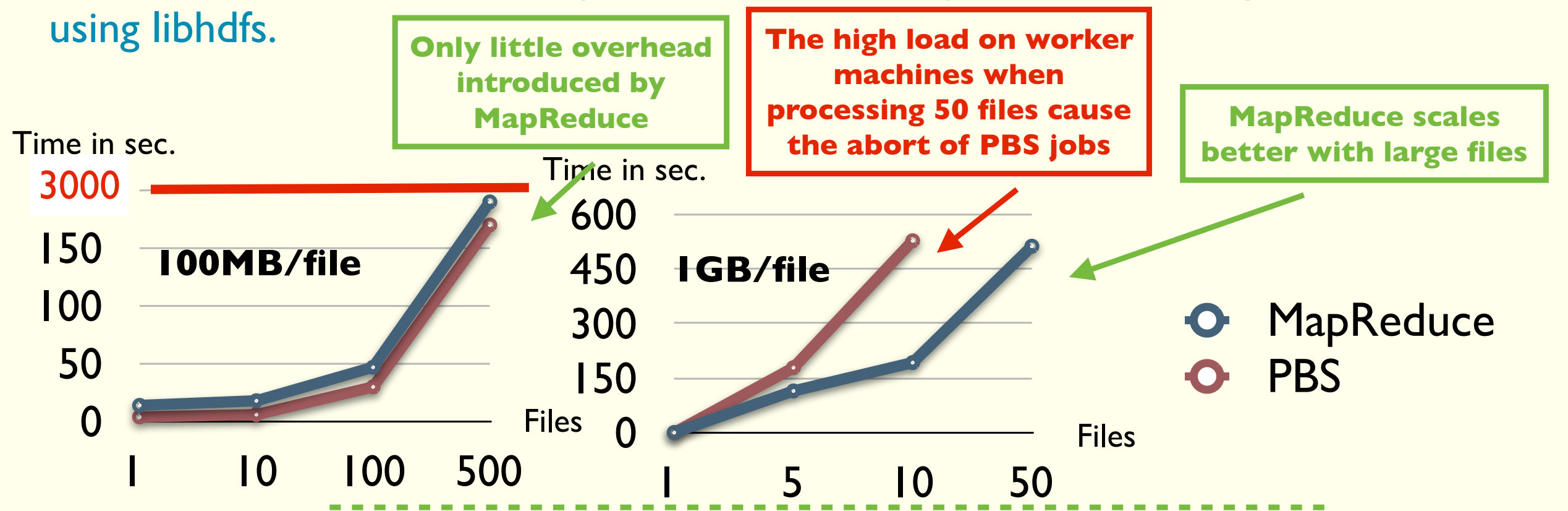
Preliminary test

- A testbed composed of 3 machines was setup to test the Analysis data flow implemented. 1 admin node and 2 Worker machines. Each machine has 24 cores, 24 G RAM, and 1 disk of 500G.
- The used Analyzer reads a Tree from ROOT file and writes TH1 histogram from one of its variables in a new output ROOT.
- The size of the file ROOT read is **100 MB**.
- The ROOT files are read/write into HDFS through Fuse (Fuse is mounted using the default value of rdbuffer option which is 10MB).



Results

- In these tests, it is used the same Analyzer of the preliminary test.
- Files are read from HDFS using ROOT's HDFS Plugin while the outputs are write using libhdfs.



~ 95 % MapReduce jobs has benefited from the data locality

Conclusions

HDFS has been deployed successfully in gLite environment and has shown satisfactory performance in term of scalability and fault tolerance while dealing with small/medium site environment constraints.

ROOT files Analysis workflow has been deployed using Hadoop and has shown promising performance.

Future works:

- Extend the Analysis workflow to include the "reduce" (such as for merging the ROOT output files).
- Perform scale tests of the Analysis workflow implemented using Hadoop.
- Implement the required Plugins to deploy a Hadoop-based Grid Computing Element.

References:

[1] Apache hadoop, 2009. http://hadoop.apache.org/.
[2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107{113, 2008.
[3] I Mandrichenko, W Allcock, and T Perelmutov. Gridftp v2 protocol description, 2005. http://www.ggf.org/documents/GFD.47.pdf.
[4] A Shoshani et al. Storage resource managers: Recent international experience on requirements and multiple co-operating implementations. In 24th IEEE Conference on Mass Storage Systems and Technologies. IEEE Computer Society, September 2007.