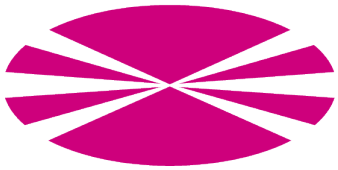

University of A Coruña

Computer Architecture Group

MapReduce Evaluator: User Guide

Authors:
Jorge Veiga, Roberto R. Expósito, Guillermo L. Taboada and Juan Touriño

December 9, 2014

Contents

1 Overview **3**

2 Features **3**
2.1 User-defined parameters 3
2.2 Solutions and benchmarks 3

3 Example **4**
3.1 Configuration of the experiments 5
3.2 Execution 6
3.3 Results 6
3.3.1 Performance 7
3.3.2 Resource utilization 7

A About Open Grid Scheduler/Grid Engine **9**

B About Environment Modules **9**

C System Requirements **9**

1 Overview

The MapReduce Evaluator (MREv) is an evaluation tool which can be used to extract valuable information about the performance, scalability and resource efficiency of HPC-oriented MapReduce solutions. It allows to compare several frameworks by means of different workloads, including micro-benchmarks and real-world applications.

MREv uses multiple user-defined parameters to unify the configuration the solutions, ensuring a fair comparison between them. In each experiment, the user can select the workloads and the solutions to be run. Several cluster sizes can be used in order to check the scalability of the frameworks and ensure an optimal use of the nodes of the system. The user can also determine the number of times each workload is executed in order to obtain statistical information.

2 Features

This section discusses the user-defined parameters, solutions and benchmarks currently available in MREv. It also describes the main behaviour of the execution of the experiments.

2.1 User-defined parameters

Table 1 shows the configuration parameters currently available in MREv. They are classified depending on whether they affect the configuration of the solutions, the experiments to run or the input data sets of the workloads.

Table 1: Available MREv configuration parameters

System dependent	Experiment dependent	Workload dependent
Mappers per node	Benchmarks	RandomWriter/Teragen data size
Reducers per node	Cluster sizes	TestDFSIO: number of files
Java heap size (slave)	Solutions	TestDFSIO: file size
Java heap size (master)	Number of executions	PageRank: number of iterations
HDFS block size		Bayes: input directory
HDFS replication factor		
Temporary directory		
SSD optimization		
GbE interface		
IPoIB interface		

Hadoop specific parameters
io.sort.factor
io.sort.mb
shuffle.parallelcopies

Mahout specific parameters
Mahout heap size

2.2 Solutions and benchmarks

Table 2 and 3 show the frameworks, and the benchmarks, respectively, currently supported by MREv. All benchmark are available for Hadoop-based solutions, although DataMPI only

supports WordCount, Sort and TeraSort, provided with the distribution. Benchmarks for Spark are still under development.

Table 2: Supported frameworks

Solution	Release Date	Interconnect
Hadoop-2.5.1-GBE	12/09/2014	Gigabit Ethernet
Hadoop-2.5.1-IPoIB	12/09/2014	InfiniBand (IPoIB)
Hadoop-2.5.1-UDA	03/09/2014	InfiniBand (RDMA & IPoIB)
Hadoop-2.4.1-GBE	30/06/2014	Gigabit Ethernet
Hadoop-2.4.1-IPoIB	30/06/2014	InfiniBand (IPoIB)
Hadoop-2.4.1-UDA	03/09/2014	InfiniBand (RDMA & IPoIB)
Hadoop-1.2.1-GBE	01/08/2013	Gigabit Ethernet
Hadoop-1.2.1-IPoIB	01/08/2013	InfiniBand (IPoIB)
Hadoop-1.2.1-UDA	07/06/2013	InfiniBand (RDMA & IPoIB)
RDMA-Hadoop-0.9.9	31/03/2014	InfiniBand (RDMA)
RDMA-Hadoop-2-0.9.5	26/11/2014	InfiniBand (RDMA)
DataMPI-0.6.0-HDFS-GBE[a]	16/04/2014	InfiniBand (RDMA) & Gigabit Ethernet
DataMPI-0.6.0-HDFS-IPoIB[a]	16/04/2014	InfiniBand (RDMA & IPoIB)
Spark-1.1.0-YARN-GBE	11/09/2014	Gigabit Ethernet
Spark-1.1.0-YARN-IPoIB	11/09/2014	InfiniBand (IPoIB)

[a] DataMPI uses HDFS (Hadoop 1.2.1) as distributed filesystem

Table 3: Supported benchmarks

Micro-benchmarks	
TestDFSIO	Tests the read and write throughput of HDFS by generating a large number of tasks performing reads and writes simultaneously
Wordcount	Counts the number of times each word appears in the input text data set, which is set up using the RandomTextWriter data generator
Sort	Sorts the input text data set, also generated by RandomTextWriter
TeraSort	Sorts 100B-sized $< key, value >$ tuples, generated by the TeraGen data generator. Each key is 10B-sized and each value is 90B-sized
Real-world Applications	
PageRank	Ranks websites by counting the number and quality of the links to each one. Developed by Google, it is used to obtain Google search results
Bayes	Performs a classification algorithm, based on Bayes' Theorem
Other	
Command	Opens a shell where the user can type actions, which is useful to perform user-defined workloads and interact with other Hadoop components like HDFS.

3 Example

This section describes a practical use case of MREv, including the configuration of user-defined parameters, the execution of the experiments and some comments about the provided results.

3.1 Configuration of the experiments

The configuration of a experiment affects the following files:

- system-conf.sh
- experiment-conf.sh
- solutions.lst
- benchmarks.lst
- cluster_sizes.lst

Environment variables The directory where these files are available can be set by the `EXP_DIR` variable. If this variable is not set, the value taken by default is `$MREv_HOME/experiment`. The directory where MREv will write the results of the experiment can also be configured by using the `OUT_DIR` variable. If this variable is not set, the value taken by default is `$PWD/MREv_OUT`. Finally, the `HOSTFILE` variable must be set, which contains the compute nodes. The first line of the file will be the master, and the remaining lines will be the slaves. If this variable is not set, the value taken by default is `$EXP_DIR/hostfile`.

```
export EXP_DIR=$HOME/terasort-experiment
export OUT_DIR=$HOME/terasort-out
export HOSTFILE=$HOME/hostfile
```

system-conf.sh This file contains the parameters related to the system where MREv is being run.

```
#!/bin/sh
export TMP_DIR=/scratch/$USER # Directory used to store local data in each node
export SSD="false" # Enable SSD optimization (only applies to RDMA-Hadoop)
export GBE_INTERFACE="eth1" # GbE interface to use in the nodes
export IPOIB_INTERFACE="ib0" # IPoIB interface to use in the nodes
export MAPPERS_PER_NODE=7 # Maximum number of map tasks per node
export REDUCERS_PER_NODE=7 # Maximum number of reduce tasks per node
export HEAPSIZE=$((3*1024)) # Heap volume size per map/reduce task (MB)
export MASTER_HEAPSIZE=$((16*1024)) # Heap volume size per master daemon (MB)
export BLOCKSIZE=$((64*1024*1024)) # HDFS block size (Bytes)
#Hadoop specific parameters
export IO_SORT_FACTOR=100 # Number of streams to merge at once while sorting files
export IO_SORT_MB=$(( $HEAPSIZE / 3 )) # Total amount of buffer memory to use while
    sorting files (MB)
export SHUFFLE_PARALLELCOPIES=20 # Default number of parallel transfers run by reduce
    during the copy(shuffle) phase
#MAHOUT
export MAHOUT_HEAPSIZE=$((32*1024)) # Heap volume size for Mahout master process (MB)
```

experiment-conf.sh This file sets the problem size of the benchmark and the number of times each one is executed.

```
#!/bin/sh
export NUM_EXECUTIONS=3 # Number of times each benchmark is executed
#Wordcount, sort & terasort
export DATASIZE=$((128 * 1024 * 1024 * 1024)) # Size of each input dataset (Bytes)
#TestDFSIO
export DFSIO_N_FILES=100 # Number of files to generate
```

```
export DFSIO_FILE_SIZE=100 # Size of each file (MB)
#PageRank
export PAGERANK_ITERATIONS=10 # Number of iterations to obtain the results
#Bayes
export BAYES_INPUT=$PWD/wikixml # Input data set (optional)
```

solutions.lst This file contains the solutions to be used in the experiment.

```
#Hadoop-1.2.1-GBE
#Hadoop-1.2.1-IPoIB
#Hadoop-1.2.1-UDA
#Hadoop-2.4.1-GBE
#Hadoop-2.4.1-IPoIB
#Hadoop-2.4.1-UDA
Hadoop-2.5.1-GBE
Hadoop-2.5.1-IPoIB
Hadoop-2.5.1-UDA
RDMA-Hadoop-0.9.9
#RDMA-Hadoop-2-0.9.5
DataMPI-0.6.0-HDFS-GBE
#DataMPI-0.6.0-HDFS-IPoIB
#Spark-1.1.0-YARN-GBE
#Spark-1.1.0-YARN-IPoIB
```

benchmarks.lst This file contains the benchmarks to be used in the experiment.

```
#testdfsio # Tests the read and write throughput of HDFS
#wordcount # Counts the number of times each word appears in the input data set
#sort # Sorts the input data set
terasort # Sorts 100B-sized < key, value > tuples
#pagerank # Ranks websites by counting the number and quality of the links to each one
#bayes # Performs a classification algorithm, based on Bayes' Theorem
#command # Opens a shell where the user can type actions
```

cluster_sizes.lst This file contains the cluster sizes with which the user wants to run the experiments. Additionally, the cluster size can be set to the maximum number of nodes available.

```
#3
5
9
13
#MAX
```

3.2 Execution

The following command starts the experiments:

```
bash MREv/bin/run.sh
```

3.3 Results

The results from the execution will be found in the `$OUT_DIR` directory, having the structure shown in Figure 1.

3.3.1 Performance

The performance results in terms of time are available in the `graphs` subdirectory. For example, for the TeraSort benchmark, they can be found in the `terasort-out/graphs/terasort.eps` file. For each cluster size, the graph includes the average, maximum and minimum execution times taken by each framework to perform the workload.

3.3.2 Resource utilization

The resource utilization results from the execution of a benchmark can be found at the `{cluster_size}/{framework}/{benchmark}_{num_execution}/stat_records` subdirectory. For example, the values of the first execution of TeraSort using Hadoop-2.5.1-IPoIB on 13 nodes are in `terasort-out/13/Hadoop-2.5.1-IPoIB/terasort_1/stat_records`. This directory contains one subdirectory for the values of each cluster node, plus another one for the average values among the slave nodes. The resource utilization graphs include CPU utilization values (`cpu_stat.eps`), CPU load values (`cpu_load_stat.eps`) during the last minute, memory usage values (`mem_stat.eps`), disk read/write values (`dsk_sda_rw_stat.eps`), disk utilization values (`dsk_sda_util_stat.eps`) and network send/recv values (`net_eth1_stat.eps`, `net_ib0_stat.eps`). Disks (sda) and network interfaces (eth1, ib0) are automatically detected by MREv. For some resources, like CPU utilization, there are different visualization modes that allow to see the results individually (with lines, `cpu_stat.eps`) or as a whole (with stacked values, `cpu_stat_stacked.eps`).

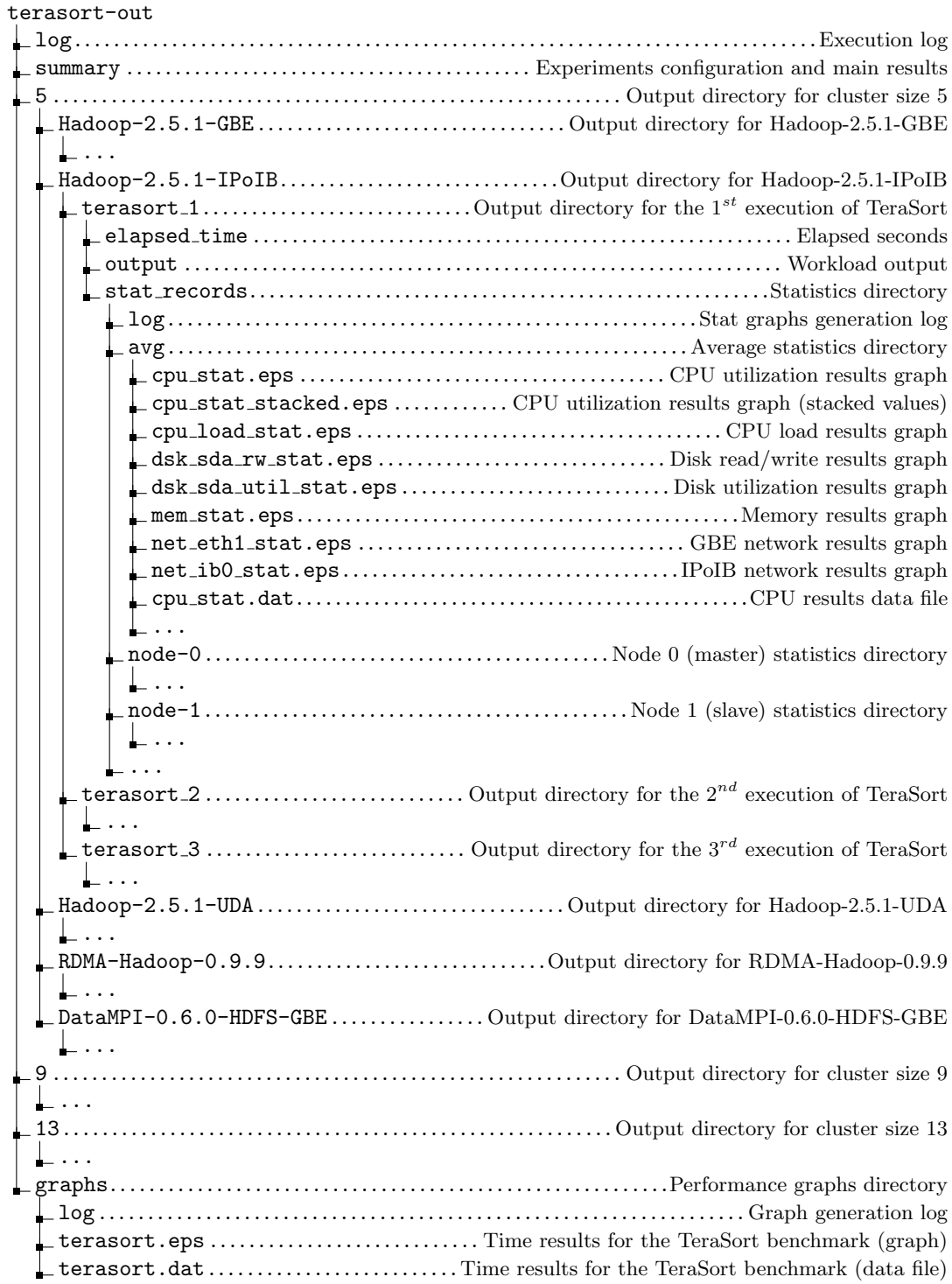

Figure 1: MREv output directory structure

A About Open Grid Scheduler/Grid Engine

As most supercomputers use a batch-queuing system for distributed resource management, MREv is aware of the environment variables and connection constraints that are typically present in these cases. The correct behaviour of MREv under this kind of systems has been tested with the Open Grid Scheduler/GE (OGS/GE).

MREv will detect the `PE_HOSTFILE` environment and use it to read the compute nodes. In this case, no `HOSTFILE` variable will be needed, although it can also be set. Moreover, the ssh connections used to launch DataMPI and Hadoop daemons do not work properly under OGS/GE, so MREv modifies them to enable their execution.

B About Environment Modules

MREv is aware of the use of Modules for dynamically modifying the user's environment. If available, MREv will use it for loading the Java and MPI environment variables.

C System Requirements

The following packages need to be installed:

- Gnuplot 4.4
- Java JRE
- MPI [1]

[1] DataMPI has been tested using Mvapich2