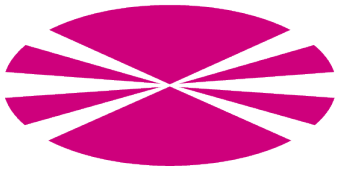

University of A Coruña

Computer Architecture Group

MapReduce Evaluator 1.1: User Guide

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

August 10, 2015

Contents

1 Overview

The MapReduce Evaluator 1.1 (MREv) is an evaluation tool 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 of 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		Command: workload command
Temporary directory		Command: data generation command
SSD optimization		Benchmark timeout
GbE interface		
IPoIB interface		
Cores per node		
Memory per node		

Hadoop specific parameters	Mahout specific parameters
io.sort.factor	Mahout heap size
io.sort.mb	
shuffle.parallelcopies	

MREv parameters
Enable plots
Enable stat records
Default timeout
MREv out directory

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-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)
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)[a]
Hadoop-2.7.1-GbE	06/07/2015	Gigabit Ethernet
Hadoop-2.7.1-IPoIB	06/07/2015	InfiniBand (IPoIB)
Hadoop-2.7.1-UDA	06/07/2015	InfiniBand (RDMA & IPoIB)[a]
RDMA-Hadoop-0.9.9-GbE	31/03/2014	InfiniBand (RDMA & Gigabit Ethernet)
RDMA-Hadoop-0.9.9-IPoIB	31/03/2014	InfiniBand (RDMA & IPoIB)
RDMA-Hadoop-2-0.9.7-GbE	26/05/2015	InfiniBand (RDMA & Gigabit Ethernet)
RDMA-Hadoop-2-0.9.7-IPoIB	26/05/2015	InfiniBand (RDMA & IPoIB)
DataMPI-0.6.0-HDFS-GbE[b]	16/04/2014	InfiniBand (RDMA) & Gigabit Ethernet
DataMPI-0.6.0-HDFS-IPoIB[b]	16/04/2014	InfiniBand (RDMA & IPoIB)
Spark-1.4.1-YARN-GbE	26/07/2015	Gigabit Ethernet
Spark-1.4.1-YARN-IPoIB	26/07/2015	InfiniBand (IPoIB)

[a] Mellanox UDA uses RDMA communications in the shuffle phase and IPoIB for HDFS
[b] DataMPI uses HDFS (Hadoop 1.2.1) as distributed filesystem configured with IPoIB

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	Executes a user-defined action, which can be set in the MREv configuration. If it is not specified, MREv opens an interactive shell, which is useful for debugging and interacting 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:

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

The environment variables and the configuration files are explained below, including the default values of the parameters.

Environment variables There are two environment variables that MREv uses to know where to find the configuration of the experiments. First, the `EXP_DIR` variable determines the directory that contains the configuration files mentioned above. If this variable is not set, the value taken by default is `$MREv_HOME/experiment`. Second, the `HOSTFILE` variable contains the file that lists the compute nodes. The first line of the file will be the master, and the remaining lines will be the slaves. If this variable is not set, the value taken by default is `$EXP_DIR/hostfile`.

```
export EXP_DIR=$MREv_HOME/experiment
export HOSTFILE=$EXP_DIR/hostfile
```

mrev-con.sh This file contains some parameters that configure the behaviour of MREv along the experiments. First, the `ENABLE_PLOT` parameter determines if MREv will generate performance graphs of the execution of the workloads. Similarly, `ENABLE_STAT` is set to perform the stat recording when executing the workloads, and generating the graphs afterwards. The `DEFAULT_TIMEOUT` parameter define the maximum execution time for a workload. Those which run above this limit will be killed, and the execution of the solution will be finished. Of course, this MREv feature can be disabled by setting `DEFAULT_TIMEOUT` to 0. Finally, 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`.

```
#!/bin/sh

export ENABLE_PLOT="true"
export ENABLE_STAT="true"
export DEFAULT_TIMEOUT="8h"
export OUT_DIR=$PWD/${METHOD_NAME}_OUT
```

system-conf.sh This file contains the parameters related to the system where MREv is being run. Some of them are automatically detected from the system, but can also be tuned by the user in order to maximize the leveraging of the system resources.

```
#!/bin/sh

export TMP_DIR=/tmp/$USER  # Directory used to store local data in each node
export GBE_INTERFACE="eth1"  # GbE interface to use in the nodes
export IPOIB_INTERFACE="ib0"  # IPoIB interface to use in the nodes
export CPUS_PER_NODE=‘grep "^physical id" /proc/cpuinfo | sort -u | wc -l‘  # Available
     CPUs per node
export CORES_PER_CPU=‘grep "^core id" /proc/cpuinfo | sort -u | wc -l‘  # Available
    cores per CPU
export CORES_PER_NODE=$(( $CPUS_PER_NODE * $CORES_PER_CPU ))  # Available cores per
    node
export MEMORY_PER_NODE=$((‘grep MemTotal /proc/meminfo | awk ’{print $2}’‘/1024))  #
    Available memory per node
export MAPPERS_PER_NODE=$(( $CORES_PER_NODE / 2 - 1 ))  # Maximum number of map tasks
    per node
export REDUCERS_PER_NODE=$(( $CORES_PER_NODE / 2 - 1 ))  # Maximum number of reduce
    tasks per node
export HEAPSIZE=‘op_int "$MEMORY_PER_NODE * 0.75 / ( $MAPPERS_PER_NODE + $REDUCERS_PER_
    NODE )"‘  # Heap volume size per map/reduce task (MB)
export MASTER_HEAPSIZE=‘op_int "$MEMORY_PER_NODE / 4"‘  # Heap volume size per master
    daemon (MB)
export BLOCKSIZE=$((128*1024*1024))  # HDFS block size (Bytes)
export REPLICATION_FACTOR=3  # Number of block replications
export SSD="false"  # Enable SSD optimization (only applies to RDMA-Hadoop)

#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=$MASTER_HEAPSIZE  # Heap volume size for Mahout master process (
    MB)
```

experiment-conf.sh This file sets the problem size of the benchmarks and the number of times each one is executed. Moreover, it also contains the `METHOD_COMMAND` variable, which contains the action to run in batch mode during the command benchmark. Additionally, `METHOD_PREPARE_COMMAND` is called to set up the input datasets needed for `METHOD_COMMAND`. This enables to perform accurate performance and resource utilization monitoring, without taking into account the data generation or the copy to HDFS. This file also allows to configure specific timeouts for each benchmark.

```
#!/bin/sh

export NUM_EXECUTIONS=1 # Number of times each benchmark is executed
#Wordcount, sort & terasort
export DATASIZE=$(( 1024 * 1024 * 1024)) # Size of each input dataset (Bytes)
#TestDFSIO
export DFSIO_N_FILES=100 # Number of files to generate
export DFSIO_FILE_SIZE=100 # Size of each file (MB)
#PageRank
export PAGERANK_ITERATIONS=10 # Number of iterations to obtain the results
#Bayes
export BAYES_INPUT=$PWD/wikixml # Input data set (optional)

## Command
# export METHOD_COMMAND= # Command to run in batch mode
# export METHOD_PREPARE_COMMAND= # Command to run to set up input datasets

## TIMEOUT
# export TESTDFSIO_TIMEOUT="0"
# export WORDCOUNT_TIMEOUT="0"
# export SORT_TIMEOUT="0"
# export TERASORT_TIMEOUT="0"
# export PAGERANK_TIMEOUT="0"
# export BAYES_TIMEOUT="0"
# export COMMAND_TIMEOUT="0"
```

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-2.5.1-GbE
#Hadoop-2.5.1-IPoIB
#Hadoop-2.5.1-UDA
Hadoop-2.7.1-GbE
#Hadoop-2.7.1-IPoIB
#Hadoop-2.7.1-UDA
#RDMA-Hadoop-0.9.9-GbE
#RDMA-Hadoop-0.9.9-IPoIB
#RDMA-Hadoop-2-0.9.7-GbE
#RDMA-Hadoop-2-0.9.7-IPoIB
#DataMPI-0.6.0-HDFS-GbE
#DataMPI-0.6.0-HDFS-IPoIB
#Spark-1.4.1-YARN-GbE
#Spark-1.4.1-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 # Executes user-defined actions (interactive or batch)
```

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 Log & configuration

MREv creates separate log and configuration directories for each framework and stores them at `{cluster_size}/{framework}`. For example, the configuration directory of Hadoop-2.7.1-IPoIB with 5 nodes is `report_MREv_08_10_12-00-00/5/Hadoop-2.7.1-IPoIB/etc/hadoop` and its log directory is `report_MREv_08_10_12-00-00/5/Hadoop-2.7.1-IPoIB/log`. Both directories can be used to check the configuration generated by MREv and the execution of the workloads. Moreover, this feature enables to run simultaneous evaluations of the same framework using different configurations.

3.3.2 Performance

The performance results in terms of time are available in the `graphs` subdirectory. For example, for the Wordcount benchmark, they can be found in the `report_MREv_08_10_12-00-00/graphs/wordcount.eps` file. For each cluster size, the graph depicts the average, maximum and minimum execution times taken by each framework to perform the workload.

3.3.3 Resource Utilization

The resource utilization results from the execution of a benchmark can be found at `{cluster_size}/{framework}/{benchmark}_{num_execution}/stat_records`. For example, the values of the first execution of Wordcount using Hadoop-2.7.1-IPoIB on 5 nodes are at `report_MREv_08_10_12-00-00/5/Hadoop-2.7.1-IPoIB/wordcount_1/stat_records`. This directory contains one subdirectory for the values of each cluster node, plus another one for the average

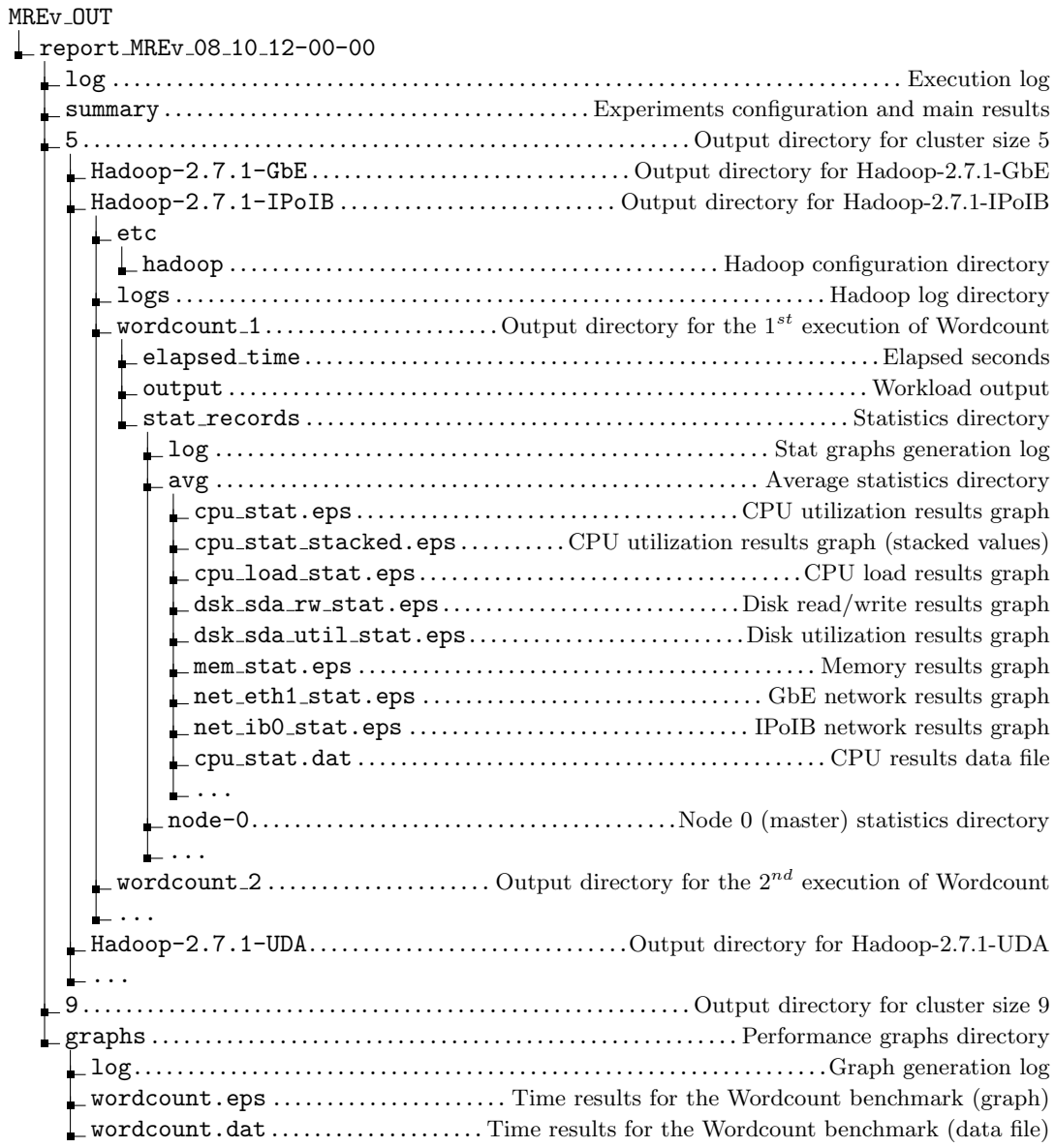

Figure 1: MREv output directory structure

values among the slave nodes. The resource utilization graphs include CPU utilization (`cpu_stat.eps`), CPU load (`cpu_load_stat.eps`), memory usage (`mem_stat.eps`), disk read/write (`dsk_sda_rw_stat.eps`), disk utilization (`dsk_sda_util_stat.eps`) and network send/recv (`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`).

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 performs a light 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 1.7
- MPI [1]

[1] DataMPI has been tested using Mvapich2