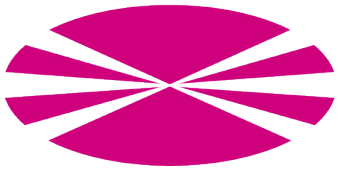

University of A Coruña

Computer Architecture Group

Big Data Evaluator 2.0: User Guide

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

March 2, 2016

Contents

1 Overview

The Big Data Evaluator 2.0 (BDEv) [1] is an evaluation tool to extract valuable information about the performance, scalability and resource efficiency of several Big Data frameworks. It allows to compare several solutions by means of different workloads, including micro-benchmarks and real-world applications.

BDEv 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.

This user guide aims to provide a clear explanation of the features available in BDEv, as long as to explain how to configure and run the evaluations.

2 Configuration of the experiments

The configuration of a experiment affects the following files:

- hostfile
- bdev-conf.sh
- system-conf.sh
- experiment-conf.sh
- core-conf.sh
- hdfs-conf.sh
- mapred-conf.sh
- yarn-conf.sh
- solutions-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 BDEv 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. This feature enables to set up different evaluations by means of several experiment directories. If this variable is not set, the value taken by default is `$BDEv_HOME/experiment`. Second, the `HOSTFILE` variable contains the path to the hostfile. If this variable is not set, the value taken by default is `$EXP_DIR/hostfile`.

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

[1] BDEv has evolved from MREv, a MapReduce Evaluation tool aimed to compare the performance of HPC-oriented MapReduce solutions. Apart from that kind of solutions, BDEv also evaluates new types of Big Data frameworks like Spark and Flink

hostfile This file lists the computing nodes that will be used in the experiments. The first line of the file will be the master, and the remaining lines will be the slaves. NOTE: the "localhost" host name can only be used if the evaluation is going to run on local, instead the user must specify the hostname that corresponds to the host.

```
localhost
localhost
```

bdev-con.sh This file contains some parameters that configure the behaviour of BDEv along the experiments. First, the `ENABLE_PLOT` parameter determines if BDEv 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. `ENABLE_ILO` can be set in order to record power consumption measurements using the HP-iLO technology (requires the previous instalation of this technology). `DEFAULT_TIMEOUT` defines the maximum execution time of a workload used by default. Those which run above this limit will be killed, and the execution of the solution will be finished. Of course, this BDEv feature can be disabled by setting `DEFAULT_TIMEOUT` to 0. Finally, the directory where BDEv 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/BDEv_OUT`. This file also contains some configuration parameters used when enabling the record of resource stat or HP-iLO measurements.

```
#!/bin/sh

## Configuration parameters regarding BDEv behaviour

export ENABLE_PLOT="true" # Enable plot generation
export ENABLE_STAT="false" # Enable resource stats generation
export ENABLE_ILO="false" # Enable HP-iLO power measurements
export DEFAULT_TIMEOUT="28800" # Default workload timeout
export OUT_DIR=$PWD/${METHOD_NAME}_OUT # Default report output directory

# Resource stats
export STAT_SECONDS_INTERVAL=5 # Interval (seconds) for each sample
export STAT_WAIT_SECONDS=20 # Waiting time (seconds) before each benchmark execution

# HP-iLO
export ILO_SECONDS_INTERVAL=5  # Interval (seconds) for each sample
export ILO_WAIT_SECONDS=60 # Waiting time (seconds) before each benchmark execution
export ILO_USERNAME="ilo_user" # User name for ILO interface
export ILO_PASSWD="..ilo_user.." # Password for ILO user
export ILO_BASE_IP="192.168.255" # Base IP for ILO interfaces
export ILO_MASTER="localhost" # Node which can connect to the ILO interface for all the
     slaves (localhost means to use the master node)
```

system-conf.sh This file contains the parameters related to the system where BDEv 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

## Configuration parameters regarding the host system characteristics

export TMP_DIR=/scratch/$USER/hadoop  # Directory used to store tmp files in each node
export LOCAL_DIRS=""   # List of directories 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‘ # CPUs per
    node
export CORES_PER_CPU=‘grep "^core id" /proc/cpuinfo | sort -u | wc -l‘ # Cores per CPU
export CORES_PER_NODE=$(( $CPUS_PER_NODE * $CORES_PER_CPU )) # Cores per node
export MEMORY_PER_NODE=$((‘grep MemTotal /proc/meminfo | awk ’{print $2}’‘/1024)) #
    Memory per node
export MASTER_HEAPSIZE=‘op_int "$MEMORY_PER_NODE / 4"‘  # Heap size per master daemon (
    MB), e.g. NameNode, ResourceManager
export SLAVE_HEAPSIZE=1024  # Heap size per slave daemon (MB), e.g. DataNode,
    NodeManager
export ENABLE_MODULES="false"  # Enable use of modules environment
export MODULE_JAVA="java"  # Java module
export MODULE_MPI="mvapich2"  # MPI module
```

experiment-conf.sh This file sets the configuration of the benchmarks, including the problem size and additional parameters, as well as 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

## Configuration of the workloads

export NUM_EXECUTIONS=1 # Number of times each benchmark is executed

#Datasize for Wordcount
export WORDCOUNT_DATASIZE=$((1 * 1024 * 1024)) # Size of the input data set (Bytes)

#Sort
export SORT_DATASIZE=$((1 * 1024 * 1024)) # Size of the input data set (Bytes)

#Terasort
export TERASORT_DATASIZE=$((1 * 1024 * 1024)) # Size of the input data set (Bytes)

#Grep
export GREP_DATASIZE=$((1 * 1024 * 1024)) # Size of the input data set (Bytes)
export GREP_REGEX=".*toxoplasmosis.*" # Regular expression

#TestDFSIO
export DFSIO_N_FILES=32  # Number of files to generate
export DFSIO_FILE_SIZE=10 # Size of each file (MBytes)

#PageRank
export PAGERANK_PAGES=50 # Number of pages in the data set
export PAGERANK_MAX_ITERATIONS=1 # Maximum number of iterations

#Connected Components
export CC_PAGES=50 # Number of pages in the data set

#K-Means
```

```
export KMEANS_NUM_OF_CLUSTERS=5 # Number of clusters in the data set
export KMEANS_DIMENSIONS=3 # Number of dimensions of the data set
export KMEANS_NUM_OF_SAMPLES=30000 # Total number of samples
export KMEANS_SAMPLES_PER_INPUTFILE=6000 # Number of samples per input file
export KMEANS_K=10 # K constant
export KMEANS_MAX_ITERATIONS=1 # Maximum number of iterations

#Bayes
export BAYES_PAGES=25000 # Number of pages in the data set
export BAYES_CLASSES=10 # Number of classes in the data set
export BAYES_NGRAMS=1 # Number of NGrams in the data set

#Aggregation
export AGGREGATION_PAGES=120 # Number of pages in the data set (nodes of the graph)
export AGGREGATION_USERVISITS=1000 # Number of user visits in the data set (edges of
    the graph)

#Join
export JOIN_PAGES=120 # Number of pages in the data set (nodes of the graph)
export JOIN_USERVISITS=1000 # Number of user visits in the data set (edges of the graph
    )

#Scan
export SCAN_PAGES=120 # Number of pages in the data set (nodes of the graph)
export SCAN_USERVISITS=1000 # Number of user visits in the data set (edges of the graph
    )

#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 GREP_TIMEOUT="0"
#export PAGERANK_TIMEOUT="0"
#export CC_TIMEOUT="0"
#export KMEANS_TIMEOUT="0"
#export BAYES_TIMEOUT="0"
#export AGGREGATION_TIMEOUT="0"
#export JOIN_TIMEOUT="0"
#export SCAN_TIMEOUT="0"
#export COMMAND_TIMEOUT="0"
```

core-conf.sh This file contains configuration parameters which are related to the core-site.xml file of Hadoop configuration.

```
#!/bin/sh

## Configuration parameters corresponding with the core-site.xml file of Hadoop
    configuration

export FS_PORT=8020 # Filesystem port number
```

hdfs-conf.sh This file contains configuration parameters which are related to the hdfs-site.xml file of Hadoop configuration.

```
#!/bin/sh

## Configuration parameters corresponding with the hdfs-site.xml file of Hadoop
    configuration

export BLOCKSIZE=$((128*1024*1024))  # HDFS block size (Bytes)
export REPLICATION_FACTOR=3  # Number of block replications
```

mapred-conf.sh This file contains configuration parameters which are related to the mapred-site.xml file of Hadoop configuration.

```
#!/bin/sh

## Configuration parameters corresponding with the mapred-site.xml file of Hadoop
    configuration

if [[ $CORES_PER_NODE == 1 ]]
then
       export MAPPERS_PER_NODE=1  # Maximum number of map tasks per node
       export REDUCERS_PER_NODE=1  # Maximum number of reduce tasks per node
else
       export MAPPERS_PER_NODE=$(( $CORES_PER_NODE / 2 ))  # Maximum number of map
           tasks per node
       export REDUCERS_PER_NODE=$(( $CORES_PER_NODE / 2 ))  # Maximum number of reduce
           tasks per node
fi

export MAP_MEMORY=$CONTAINER_MEMORY # The amount of memory to request from YARN per map
     task (MB)
export REDUCE_MEMORY=$CONTAINER_MEMORY # The amount of memory to request from YARN per
    reduce task (MB)
export MAP_HEAPSIZE_FACTOR=0.90 # Percentage of the mapper memory allocated to heap
export REDUCE_HEAPSIZE_FACTOR=0.90 # Percentage of the reducer memory allocated to heap
export MAP_HEAPSIZE=‘op_int "$MAP_MEMORY * $MAP_HEAPSIZE_FACTOR"‘  # Heap size per map
    task (MB)
export REDUCE_HEAPSIZE=‘op_int "$REDUCE_MEMORY * $REDUCE_HEAPSIZE_FACTOR"‘ # Heap size
    per reduce task (MB)
export IO_SORT_MB=$(( $MAP_HEAPSIZE / 4 )) # Total amount of buffer memory to use while
     sorting files (MB)
export IO_SORT_FACTOR=$(( $IO_SORT_MB / 10 )) # Number of streams to merge at once
    while sorting files
export IO_SORT_RECORD_PERCENT=0.05 # The percentage of io.sort.mb dedicated to tracking
     record boundaries
export IO_SORT_SPILL_PERCENT=0.80 # The soft limit in either the buffer or record
    collection buffers
export SHUFFLE_PARALLELCOPIES=20  # Default number of parallel transfers run by reduce
    during the copy(shuffle) phase
export MR_JOBHISTORY_SERVER="false" # Start the MapReduce JobHistoryServer
```

yarn-conf.sh This file contains configuration parameters which are related to the yarn-site.xml file of Hadoop configuration.

```
#!/bin/sh

## Configuration parameters corresponding with the yarn-site.xml file of Hadoop
    configuration

export APP_MASTER_HEAPSIZE=1024 # Application Master heapsize
export APP_MASTER_MEMORY=‘op_int "$APP_MASTER_HEAPSIZE * 1.5"‘ # Application Master
    memory
export NODEMANAGER_MEMORY_FACTOR=0.95 # Percentage of the node memory available for
    allocation
export NODEMANAGER_MEMORY=‘op_int "$MEMORY_PER_NODE * $NODEMANAGER_MEMORY_FACTOR"‘ #
    Memory available for allocation
export NODEMANAGER_VCORES=$CORES_PER_NODE # Number of cores per NodeManager
export NODEMANAGER_MIN_ALLOCATION=256 # Minimum memory allocation for containers in
    NodeManager
export NODEMANAGER_INCREMENT_ALLOCATION=256 # Container memory allocations are rounded
    up to the nearest multiple of this number
export CONTAINER_MEMORY=‘op_int "($NODEMANAGER_MEMORY - $APP_MASTER_MEMORY) / $CORES_
    PER_NODE"‘ # Memory size for containers
```

solutions-conf.sh This file contains configuration parameters which are specific to each solution, as well as some variables for Apache Mahout and Apache Hive.

```
#!/bin/sh

## Configuration parameters of the different frameworks

# RDMA-Hadoop/RDMA-Hadoop-2
export RDMA_HADOOP_IB_ENABLED="true" # Enable RDMA connections through InfiniBand (IB)
export RDMA_HADOOP_ROCE_ENABLED="false" # Enable RDMA connections through RDMA over
    Converged Ethernet (RoCE)
export RDMA_HADOOP_DFS_REPLICATION_PARALLEL="false" # Enable parallel replication
export RDMA_HADOOP_DFS_SSD_USED="false" # Enable SSD-oriented optimizations for HDFS
export RDMA_HADOOP_DISK_SHUFFLE_ENABLED="true" # Enable disk-based shuffle

# Spark (common)
export SPARK_HADOOP_HOME=${SOLUTIONS_DIST_DIR}/Hadoop-YARN/2.7.2
export SPARK_DRIVER_HEAPSIZE=$MASTER_HEAPSIZE # Driver heapsize
export SPARK_DRIVER_CORES=1 # Number of cores of the driver
export SPARK_EXECUTORS_PER_NODE=1 # Number of executors (workers) per node
export SPARK_CORES_PER_EXECUTOR=$CORES_PER_NODE # Number of cores per executor
export SPARK_LOCAL_DIRS=$LOCAL_DIRS # Executor temporary directories

# Spark standalone
export SPARK_EXECUTOR_HEAPSIZE_FACTOR=0.90 # Percentage of the executor heapsize
    allocated to heap
export SPARK_EXECUTOR_HEAPSIZE=‘op_int "($CONTAINER_MEMORY * $SPARK_CORES_PER_EXECUTOR)
     * $SPARK_EXECUTOR_HEAPSIZE_FACTOR"‘ # Executor heapsize

# Spark on YARN (client mode)
export SPARK_AM_HEAPSIZE=1024 # YARN Application Master heapsize
export SPARK_YARN_EXECUTOR_MEMORY_OVERHEAD=0.10 # Percentage of the executor heapsize
    not allocated to heap in YARN
export SPARK_YARN_EXECUTOR_MEMORY=‘op_int "($CONTAINER_MEMORY * $SPARK_CORES_PER_
    EXECUTOR)"‘ # Amount of memory allocated to the executor in YARN
```

```
export SPARK_YARN_EXECUTOR_HEAPSIZE_FACTOR=‘op "1 - $SPARK_YARN_EXECUTOR_MEMORY_
    OVERHEAD"‘ # Percentage of the executor heapsize allocated to heap in YARN
export SPARK_YARN_EXECUTOR_HEAPSIZE=‘op_int "$SPARK_YARN_EXECUTOR_MEMORY * $SPARK_YARN_
    EXECUTOR_HEAPSIZE_FACTOR"‘ # Executor heapsize in YARN

# RDMA-Spark
export RDMA_SPARK_IB_ENABLED="true" # Enable RDMA connections through InfiniBand (IB)
export RDMA_SPARK_ROCE_ENABLED="false" # Enable RDMA connections through RDMA over
    Converged Ethernet (RoCE)
export RDMA_SPARK_SHUFFLE_CHUNK_SIZE=524288 # Chunk size for shuffle

# Flink (common)
export FLINK_HADOOP_HOME=${SOLUTIONS_DIST_DIR}/Hadoop-YARN/2.7.2
export FLINK_TASKMANAGER_SLOTS=$CORES_PER_NODE # Number of slots per TaskManager
export FLINK_TASKMANAGER_TMP_DIRS=$LOCAL_DIRS # TaskManager temporary directories
export FLINK_TASKMANAGER_NETWORK_NUMBEROFBUFFERS_PER_SLAVE=‘op_int " $FLINK_TASKMANAGER
    _SLOTS ^ 2 * 6 "‘ # Number of network buffers per slave

# Flink standalone
export FLINK_JOBMANAGER_HEAPSIZE=$MASTER_HEAPSIZE # JobManager heapsize
export FLINK_TASKMANAGER_HEAPSIZE_FACTOR=0.85 # Percentage of the TaskManager heapsize
    allocated to heap
export FLINK_TASKMANAGER_HEAPSIZE=‘op_int "($CONTAINER_MEMORY * $FLINK_TASKMANAGER_
    SLOTS) * $FLINK_TASKMANAGER_HEAPSIZE_FACTOR"‘ # TaskManager heapsize

# Flink on YARN
export FLINK_YARN_JOBMANAGER_MEMORY=2048 # JobManager memory in YARN
export FLINK_YARN_TASKMANAGER_MEMORY=‘op_int "($CONTAINER_MEMORY * $FLINK_TASKMANAGER_
    SLOTS) - $FLINK_YARN_JOBMANAGER_MEMORY"‘ # TaskManager memory in YARN

# DataMPI
export DATAMPI_HADOOP_HOME=${SOLUTIONS_DIST_DIR}/Hadoop/1.2.1
export DATAMPI_TASK_HEAPSIZE_FACTOR=0.90 # Percentage of the task memory allocated to
    heap
export DATAMPI_TASK_HEAPSIZE=‘op_int "$CONTAINER_MEMORY * $DATAMPI_TASK_HEAPSIZE_FACTOR
    "‘ # Task heapsize

# Mellanox UDA library
export UDA_VERSION=3.3.2 # UDA library version
export UDA_LIB_DIR=$SOLUTIONS_LIB_DIR/uda-$UDA_VERSION # UDA library directory

# Apache Mahout
export MAHOUT_HEAPSIZE=$MASTER_HEAPSIZE # Heap size for Mahout master process
export HADOOP_1_MAHOUT_VERSION=0.11.1 # Mahout version for Hadoop 1
export HADOOP_2_MAHOUT_VERSION=0.11.1 # Mahout version for Hadoop 2 (YARN)

# Apache Hive
export HADOOP_1_HIVE_VERSION=1.2.1 # Hive version for Hadoop 1
export HADOOP_2_HIVE_VERSION=1.2.1 # Hive version for Hadoop 2 (YARN)
```

solutions.lst This file contains the solutions to be used in the experiment, specifying the framework, its version and the network interface to be configured.

```
#Framework   #Version #Network interface
#Hadoop    1.2.1  GbE
#Hadoop    1.2.1  IPoIB
```

```
#Hadoop-YARN  2.7.1  GbE
#Hadoop-YARN  2.7.1  IPoIB
Hadoop-YARN   2.7.2  GbE
#Hadoop-YARN  2.7.2  IPoIB
#Hadoop-UDA   1.2.1  IPoIB
#Hadoop-UDA-YARN 2.7.2  IPoIB
#RDMA-Hadoop  0.9.9  GbE
#RDMA-Hadoop  0.9.9  IPoIB
#RDMA-Hadoop-2  0.9.8  GbE
#RDMA-Hadoop-2  0.9.8  IPoIB
#DataMPI   0.6.0  GbE
#DataMPI   0.6.0  IPoIB
#Spark    1.5.2  GbE
#Spark    1.5.2  IPoIB
#Spark-YARN   1.5.2  GbE
#Spark-YARN   1.5.2  IPoIB
#Spark    1.6.0  GbE
#Spark    1.6.0  IPoIB
#Spark-YARN   1.6.0  GbE
#Spark-YARN   1.6.0  IPoIB
#RDMA-Spark   0.9.1  GbE
#RDMA-Spark   0.9.1  IPoIB
#RDMA-Spark-YARN 0.9.1  GbE
#RDMA-Spark-YARN 0.9.1  IPoIB
#Flink    0.10.2  GbE
#Flink    0.10.2  IPoIB
#Flink-YARN   0.10.2  GbE
#Flink-YARN   0.10.2  IPoIB
```

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

```
## Benchmarks to run in the evaluation

#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
#grep     # Counts the matches of a regular expression in the input data set
#terasort    # Sorts 100B-sized <key,value> tuples
#pagerank    # Graph algorithm which ranks elements by counting the number and quality
    of the links to each one
#connected_components # Graph algorithm which finds the connected components of a graph
#bayes     # Performs the Naive Bayesian classification algorithm
#kmeans     # Clustering algorithm which partitions N observations into K clusters
#aggregation   # SQL-based query which sums the values from a column
#join     # SQL-based query which joins two tables
#scan     # SQL-based query which extracts the rows that match a certain pattern
#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 Execution

The following command starts the experiments:

```
bash BDEv/bin/run.sh
```

4 Evaluation outcomes

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

4.1 Log & configuration

BDEv 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_BDEv_03_02_12-00-00/5/Hadoop-2.7.1-IPoIB/etc/hadoop` and its log directory is `report_BDEv_03_02_12-00-00/5/Hadoop-2.7.1-IPoIB/log`. Both directories can be used to check the configuration generated by BDEv and the execution of the workloads. Moreover, this feature enables to run simultaneous evaluations of the same framework using different configurations.

4.2 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_BDEv_03_02_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.

4.3 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_BDEv_03_02_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 values among the slave nodes.

The resource utilization graphs are not automatically generated by BDEv in order to prevent the apparition of an excessively number of unnecessary files. The user can generate them by running the script `gen_graphs.eps`, which contains the command lines needed for generating the resource utilization graphs contained in that directory. These 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 BDEv. 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`).

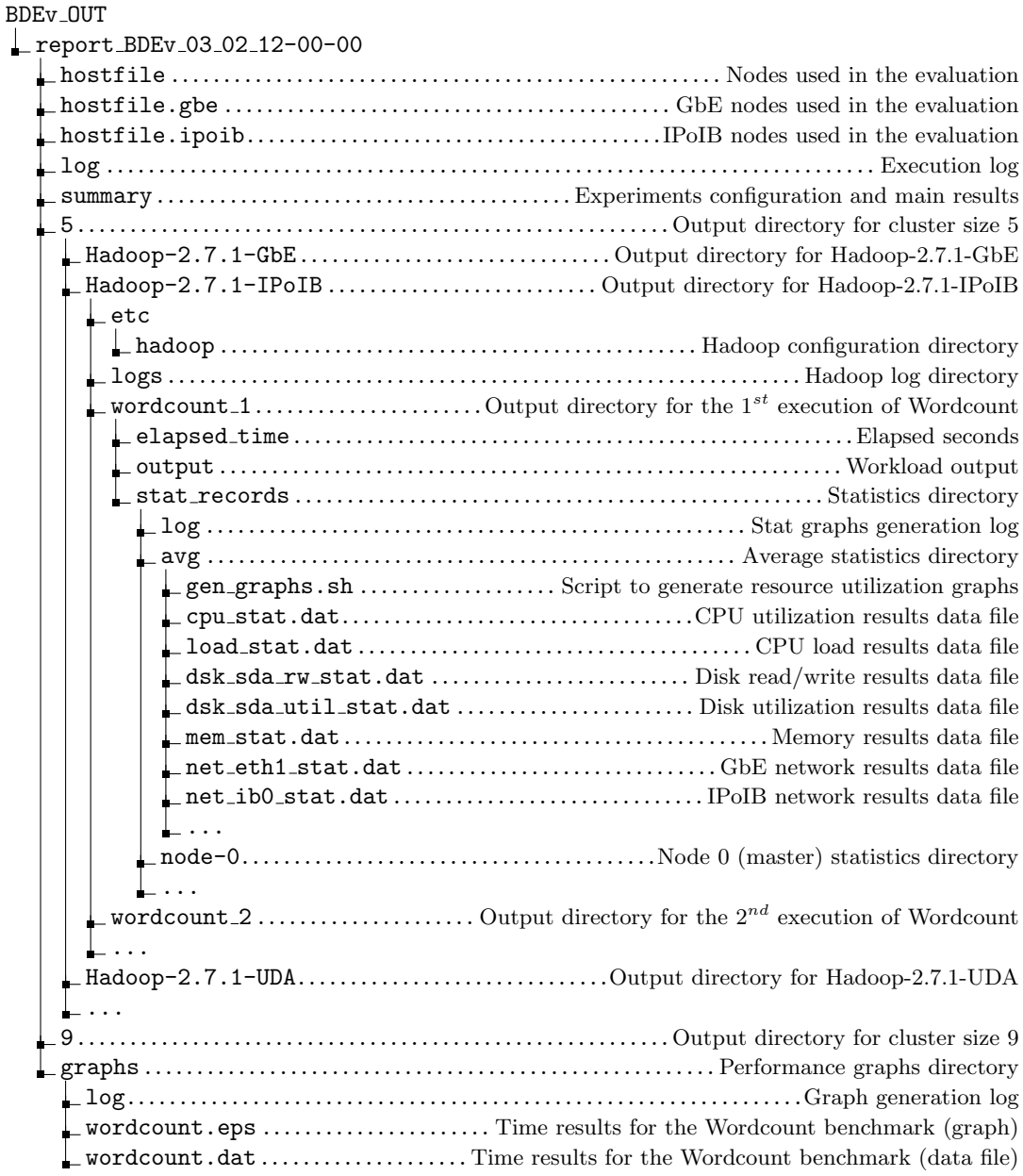

Figure 1: BDEv output directory structure

A About Open Grid Scheduler/Grid Engine

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

BDEv 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 the different framework daemons do not work properly under OGS/GE, so BDEv performs a light modification to enable their execution.

B About Environment Modules

BDEv is aware of the use of Modules for dynamically modifying the user's environment. If enabled in the configuration, BDEv will use it for loading the Java and MPI environment variables.

C System Requirements

The following packages need to be installed:

- Java JRE 1.7 (always needed)
- Gnuplot 4.4 (needed for generating the graphs)
- MPI [2] (needed for DataMPI)

[2] DataMPI has been tested using Mvapich2