UNIVERSIDADE DA CORUÑA

Big Data Evaluator: User Guide

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

Contents

1 Overview

The Big Data Evaluator (BDEv)[1] is a benchmarking tool that extracts valuable information about the performance, resource utilization, energy efficiency and microarchitectural behaviour of several Big Data processing frameworks. It supports different workloads to perform the comparisons, including micro-benchmarks and real-world applications.

BDEv uses a wide range of user-defined parameters that unify the configuration of the frameworks, ensuring fair comparisons between them. In each experiment, the user can select the frameworks to launch and the workloads to execute, testing their scalability using several cluster sizes. The user can also determine the number of times each workload is executed in order to obtain statistical information.

This tool is distributed as free software under the MIT license and is publicly available to download from GitHub at https://github.com/UDC-GAC/bdev. Further information about BDEv is also available at https://bdev.des.udc.es.

2 Citation

If you use BDEv in your research, please cite our work using the following reference:

- Jorge Veiga, Jonatan Enes, Roberto R. Expósito and Juan Touriño. BDEv 3.0: Energy efficiency and microarchitectural characterization of Big Data processing frameworks. Future Generation Computer Systems, vol. 86, pages 565-581. September 2018

3 Download

In order to download a specific version of BDEv, which is the recommended approach, you must clone a specific tag from the GitHub repository instead of cloning the master branch. For example, you can download version 3.9 by executing the following command:

```
[user@host ~]$ git clone --branch v3.9 https://github.com/UDC-GAC/bdev
```

If you want to use the BDWatchdog[2] framework for monitoring resource usage, which is integrated within BDEv as a git submodule, you should execute the following command instead:

```
[user@host ~]$ git clone --branch v3.9 --recurse-submodules https://github.com/UDC-GAC/
    bdev
```

3.1 Framework Download

By default, Big Data processing frameworks must be stored in the `$BDEv_HOME/solutions/dist` directory. From version 3.3 onwards, BDEv does not include any framework within its distribution package in order to keep it into a reasonable size. To use any supported frameworks with BDEv, the user must download and store it in the following directory:

```
$BDEv_HOME/solutions/dist/$FRAMEWORK_NAME/$FRAMEWORK_VERSION
```

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

[2] https://bdwatchdog.dec.udc.es

Where `$FRAMEWORK_NAME` and `$FRAMEWORK_VERSION` must be the name and version of the framework, respectively, as they appear in the `$BDEv_HOME/experiment/solutions.lst` file. An example for Hadoop-YARN version 2.10.2 and Spark version 3.1.3 would be as follows:

```
$BDEv_HOME/solutions/dist/Hadoop-YARN/2.10.2
$BDEv_HOME/solutions/dist/Spark/3.1.3
```

4 Configuration

The configuration of an specific 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

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`.

```
[user@host ~]$ export EXP_DIR=$BDEv_HOME/experiment
[user@host ~]$ export HOSTFILE=$EXP_DIR/hostfile
```

hostfile This file lists the cluster nodes that will be used in the experiments. The first line of the file will be configured as the master node, and the remaining lines will be the slaves or workers. NOTE: the "localhost" hostname can only be used if the evaluation is going to run on localhost, otherwise the user must specify the hostname that corresponds to each node.

bdev-conf.sh This file contains some parameters that configure the behaviour of BDEv along the experiments. Most importantly, the user can indicate which monitoring tools to activate:

`ENABLE_PLOT`: generates performance graphs with the execution time of the workloads

`ENABLE_STAT`: monitors resource usage (e.g., CPU, memory) using built-in dool tool

ENABLE_ILO: records system power consumption via the HPE iLO interface of the nodes

ENABLE_RAPL: measures power consumption of Intel CPUs using a built-in RAPL-based tool implemented upon the PAPI[3] library

ENABLE_OPROFILE: counts microarchitecture-level events using Oprofile

ENABLE_BDWATCHDOG: monitors resource usage using the BDWatchdog framework

Along with the enablement of these tools and its related configuration parameters, further properties are also contained in this file. 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 framework will be finished. Of course, this BDEv feature can be disabled by setting DEFAULT_TIMEOUT to 0. 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. MONITOR_DELAY_SECONDS allows to set a delay time after/before starting/stopping all previous monitoring tools. Finally, when ENABLE_HOSTNAMES is set to false, BDEv will use IP addresses instead of hostnames for the cluster nodes.

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

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 COMMAND variable, which contains the action to run (e.g. executable script) in batch mode during the command benchmark. From version 3.6 onwards, this variable can be an executable script/command or a directory containing multiple scripts to be executed. Additionally, PREPARE_COMMAND is called to set up the input datasets needed for 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.

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

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

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

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

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

[3] https://icl.utk.edu/papi

solutions.lst This file contains the frameworks to be used in the experiment, specifying the version and the network interface to be configured. From version 3.3 onwards, BDEv automatically switches to command mode if no framework is selected in this file. This feature can be useful to run any other application or command in the cluster while taking advantage of all the monitoring capabilities provided by BDEv.

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

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.

5 Execution

The following command starts the experiments:

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

6 Evaluation Outcomes

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

6.1 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-2.9.2-IPoIB using 5 nodes is `5/Hadoop-\hadoopversion-IPoIB/etc/hadoop` and its log directory is `5/Hadoop-\hadoopversion-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.

6.2 Performance

The performance results in terms of runtime are available in the `graphs` subdirectory. For example, for the Wordcount benchmark, they can be found in the `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.

6.3 Resource Utilization

When using dool for monitoring resource usage (i.e., ENABLE_STAT is true), the metrics 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-2.9.2-IPoIB on 5 nodes are stored in `5/Hadoop-\hadoopversion-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 creation of a large number of unnecessary files. The user can generate them by running the script `gen_graphs.eps` manually, which contains the commands 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 (e.g., sda) and network interfaces (e.g., 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`).

When using BDWatchdog for monitoring resource usage (i.e., ENABLE_BDWATCHDOG is true), the metrics are stored in the OpenTSDB database that is managed by BDWatchdog. So, the appropriate REST endpoints for OpenTSDB must be configured accordingly in `bdev-conf.sh`. The time series data stored in OpenTSDB can be visualized using the BDWatchdog's web interface. Furthermore, the time stamping feature provided by BDWatchdog can also be used together with BDEv to manage automatically the generation of "start" and "end" UNIX-like timestamps for each benchmark execution. To do so, the MongoDB database managed by BDWatchdog must be configured properly in `bdev-conf.sh` by setting its hostname/IP and port number together with the appropriate REST endpoints. Further information about BDWatchdog configuration can be obtained at https://bdwatchdog.dec.udc.es/BDWatchdog/docs_web.

6.4 Energy Efficiency

As with resource utilization, energy-related metrics can be found at `{cluster_size}/{framework}/{benchmark}_{num_execution}/rapl_records`. This directory contains one subdirectory with the values of each cluster node, plus another one for the summary values among the slave nodes. Inside each subdirectory, CSV and graph files are provided for each energy consumption counter detected by BDEv, in the form of time series. These counters typically include energy and power values for each CPU socket, labeled as package, as well as for PP0 (Power Plane 0), PP1 and UNCORE. Separated values for the memory components of each socket are also provided (if available). The summary directory contains the average power values among the slave nodes and their total energy consumption values. The `gen_graphs.eps` script allows to generate the graphs once the evaluation has finished.

Apart from the time series graphs, further energy consumption information is provided to the user in `graphs/rapl`, containing a comparison of the energy consumed by each framework during the execution of each workload. Moreover, BDEv also obtains energy-performance efficiency ratios that correlate the execution times and the energy consumption of the frameworks, calculated as $ED2P = Energy\ Consumed \times Execution\ Time^2$.

6.5 Microarchitecture-level metrics

The results from the microarchitectural events are contained in `{cluster_size}/{framework}/{benchmark}_{num_execution}/oprofile_records`. As with the previous tools, one subdirectory is provided for each node in the cluster, containing the output of the monitorization. For each hardware counter specified in the configuration parameters, the output shows the total number of events occurred in the system during the execution of the workload. The `sum.csv` file contains the total number of events occurred among the slave nodes. These values are used to generate summary graphs for each event, stored in the `graphs/oprofile` directory.

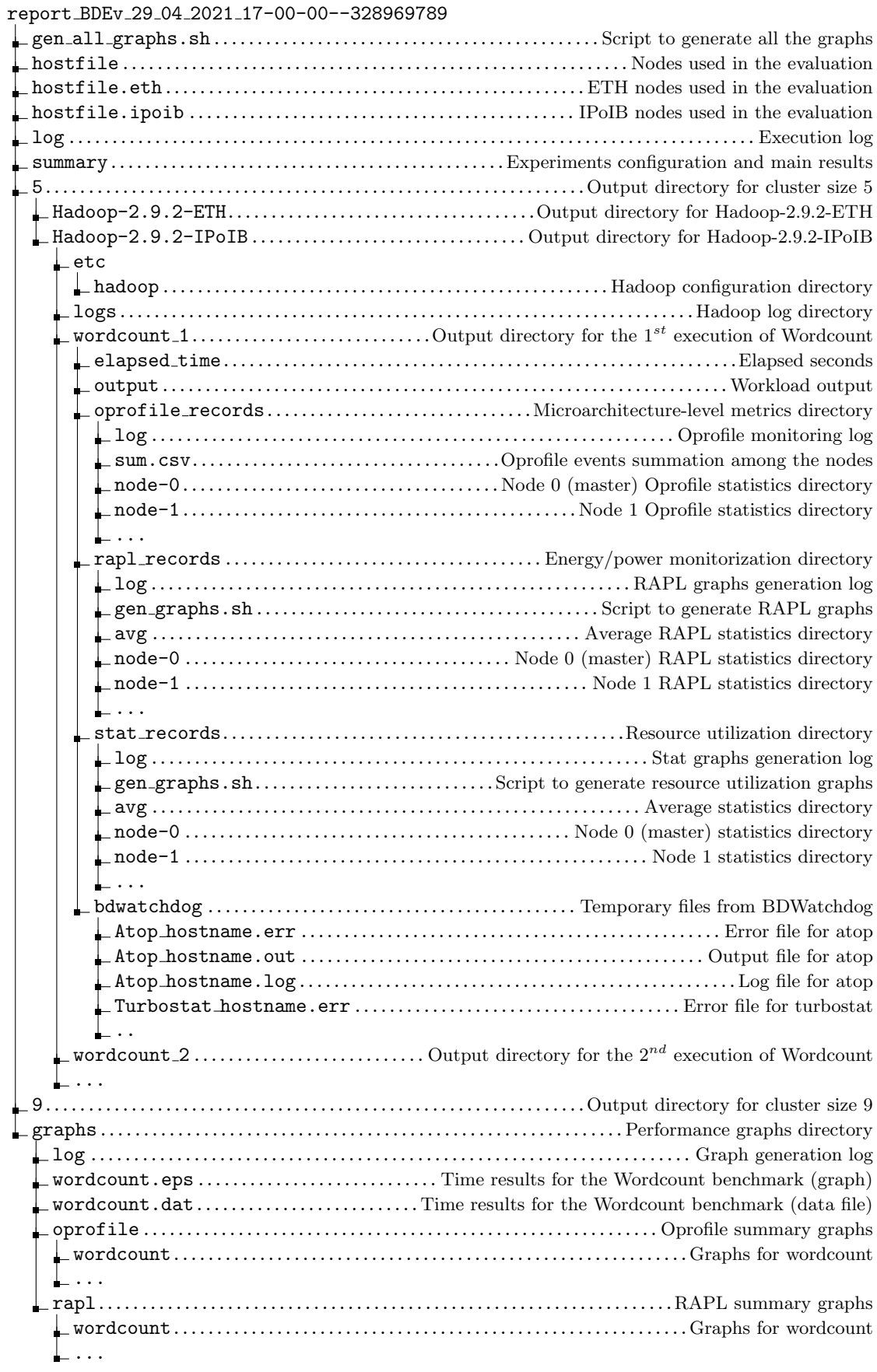

Figure 1: BDEv output directory structure

A About Flink

Flink does not include the Hadoop compatibility package by default. So, the user must configure it manually. Once the desired Flink version has been downloaded and stored in the `$BDEv_HOME/solutions/dist/Flink/$flink_version` folder, the flink-hadoop-compatibility[4] jar package must be placed within the `lib` folder of the Flink distribution.

B About batch-queuing HPC schedulers

As most HPC clusters and supercomputers use a batch-queuing job scheduler for distributed resource management, BDEv is aware of the environment variables and resource constraints that are typically present in these systems. The behaviour of BDEv under this kind of schedulers has been tested with Open Grid Scheduler/Grid Engine (OGS/GE), Son of Grid Engine (SGE) and Slurm.

BDEv automatically infers from the environment the appropriate cluster nodes to use within an interactive/batch job allocation. In this case, no `HOSTFILE` variable is needed, although it can also be set if desired.

C About Environment Modules

BDEv also supports the Environment Modules[5] tool for dynamically modifying the user's environment, which is typically available in HPC clusters. If enabled in the configuration (i.e., ENABLE_MODULES is set to true in `system-conf.sh`), BDEv will use it for loading the Java and MPI modules specified in MODULE_JAVA and MODULE_MPI variables, respectively.

D About Hardware Counters

When monitoring power consumption, either using the built-in RAPL-based tool included in BDEv or using the turbostat tool through the BDWatchdog framework, the cluster nodes must be properly configured in order to allow regular users to access the hardware performance counters. Moreover, the microarchitecture-level monitoring using the Oprofile tool also requires this specific configuration in the cluster nodes.

As a representative example, the following set of commands show how to configure the access to hardware counters in each node of the cluster on a system running RHEL/CentOS.

```
[root@host ~]# modprobe msr # Load MSR kernel module
[root@host ~]# chmod 666 /dev/cpu/*/msr # Enable read permisions on MSR files
[root@host ~]# echo 0 > /proc/sys/kernel/perf_event_paranoid # Enable access to hardware
    counters for regular users
[root@host ~]# echo '$USER ALL=NOPASSWD:/sbin/setcap' >> /etc/sudoers # Allow executing
    setcap for the user running BDEv
```

Further information about this topic can be obtained at https://github.com/icl-utk-edu/papi/tree/master/src/components/rapl.

[4] https://mvnrepository.com/artifact/org.apache.flink/flink-hadoop-compatibility
[5] https://github.com/cea-hpc/modules

E About BDWatchdog

When monitoring power consumption using the turbostat tool with BDWatchdog, regular users must also have access to the hardware performance counters as explained in the previous section. Moreover, the ability of regular users to execute setcap is also needed when monitoring the network traffic using the nethogs tool through BDWatchdog.

F System Requirements

The following packages and/or tools are needed by BDEv:

- Expect 1.1 or higher (required when timeouts are enabled)
- Gnuplot 4.4 or higher (required for generating graphs)
- MPI[6] library (required by DataMPI)
- Oprofile 1.2.0 or higher (required for microarchitecture-level event counting)
- Python 3 (required for monitoring resource usage with dool or BDWatchdog)
  - Required Python modules when using BDWatchdog: python-daemon, requests
- turbostat (required by BDWatchdog when enabling this specific feature)

G Contact

BDEv has been developed in the Computer Architecture Group[7] at the University of A Coruña[8] by the following authors:

- Jorge Veiga: https:gac.udc.es/~jveiga
- Roberto R. Expósito: https:gac.udc.es/~rober
- Jonatan Enes: https:gac.udc.es/~jonatan
- Guillermo L. Taboada: https:gac.udc.es/~gltaboada
- Juan Touriño: https:gac.udc.es/~juan

To report any question, bug, requirement or information about BDEv, feel free to contact us at https://bdev.des.udc.es.

[6] DataMPI has been tested using MVAPICH2
[7] https://gac.udc.es/?page_id=770&lang=en
[8] https://udc.es/en